Proceedings OLA'2021 International Conference on Optimization and Learning

ola 2021

21-23 June 2021, Catania, Italia

Organizers

UNIVERSITÀ degli STUDI di CATANIA

Université de Lille

Table of contents

FFCNN: Fast FPGA based Acceleration for Convolution neural network inference

F. Keddous,[1,2], H-N. Nguyen[2] and A. Nakib[1]

[1] Université Paris Est Créteil, Laboratoire LISSI, 122 Rue P. Armangot, Vitry Sur Seine, France
nakib@u-pec.fr
[2]

Atos-BULL, Rue Jean Jaures,
78340 Les Clayes-sous-Bois, France

1 Introduction

Recent work on the neural networks have shown great improvements over traditional machine learning algorithms. Especially in computer vision where a high adaptive capacity for a wide range of pattern recognition problems was demonstrated. The convolutional neuron network (AlexNet)[11] improved the classification accuracy of TOP-5 images in ImageNet [12] datasets from 73.8% to 84.7% and helped to improve the performance of different computer vision problems [13] with its ability to extract features. However, the complexity of its calculation and storage is high. According to current research, the size of the RN model continues to increase. In Table 1, we list the number of operations (additions or multiplications), the number of parameters and the top-1 precision on the ImageNet dataset [12] of the Convolutional Neural Networks (CNN) models found in the literature for image classification, object detection, and image segmentation.

For instance, one of the largest and widely used CNN requires 39 billion floating point (FLOP) operations with an image size of 224×224 and has a model parameter of 500 MB (VGG[14]). The complexity of the calculations is proportional to the size of the input, then, the calculation of high resolution images will require more than 100 billion operations.

Therefore, it is important to select a computing architecture for any CNN based solution. A typical CPU runs 10 to 100 GFLOP per second. Energy efficiency is often less than 1 GOP per day. The CPUs are difficult to apply to cloud applications that require high performance in terms FLOP and mobile applications that require low power consumption. On the other hand, GPUs offer high performance up to 10 TOP per second.

Usually, hardware accelerators are based on ASIC [12] or FPGA [13, 14]. ASIC-based accelerators offer the highest performance and energy efficiency, but must withstand considerable development costs. Because of their reconfigurable nature, FPGA-based accelerators are more economical given development costs.

For years, FPGA developers have been struggling with difficult-to-use Register Transfer Level (RTL) programming languages such as VHDL and Verilog HDL. This makes programming a major issue for the FPGA. Thus, FPGA providers are beginning to provide high-level synthesis tools such as the OpenCL framework [15] to enable FPGA programming using high-level languages. Although developers can easily port codes originally designed for CPUs / GPUs to FPGAs with the OpenCL framework, it is still difficult to make OpenCL codes run efficiently on FPGAs. The same code may have different performance on different platforms because of the different execution methods related to the architecture. Therefore, developers must consider the FPGA architecture when optimizing OpenCL code.

The main contributions of this work are as follows: (1) an OpenCL based FPGA accelerator with an efficient pipelined kernel structure is proposed for large scale network (CNN) implementation; (2) the design space of the proposed architecture was fully explored on the Arria FPGA 10 and Stratix-10, two large-scale CNN models, were implemented and tested. The results show that the proposed scheme improves performance and resource utilization compared to previous work.

The rest of the paper is organized as follow: in the next section we recall CNN definition. In section 3, the proposed implementation is presented. The obtained results are shown in the section 4. The conclusion ends the paper.

2 Convolution Neural Network

In this section, we present the basic functions of a neural network and we focus only on the inference procedure, which means that the Neural Network model was already trained and validated to predict or classify new data.

The basic architectural ideas of a Convolution Neural Network (CNN) [5] consist of the local receptive fields via the convolution operation and the spatial sub-sampling via the pooling operation. The Convolution operation can be formally written as:

$$f_{x,y,h}^{C,l} = \mathbf{w}_h^{l\,T} f_{x,y}^{Op,l-1} + b_h^l \tag{1}$$

where $\mathbf{w}_h^l$ and b_h^l are the weights and bias of the h^{th} feature map, $f^{Op,l-1}$ and $f_{x,y,h}^{C,l}$ are the input and output feature maps, l denotes the layer and (x, y) is the spatial image coordinate. The superscript C denotes convolution and Op represents various operations, *e.g.*, input (when $l = 1$), convolution, pooling, activation, etc.

Pooling applies local operations, *e.g.*, computing the maximum within a local neighborhood has the following form:

$$f_{x,y,h}^{P_{max},l} = max_{(m,n)\in\mathcal{N}_{x,y}}(f_{m,n,h}^{Op,l-1}) \tag{2}$$

where $\mathcal{N}_{x,y}$ denotes the local spatial neighborhood and P_{max} denotes the max pooling. Often a spatial resolution reduction is applied after the max-pooling operation. Besides the two above-mentioned operations, there are several strategies applied within the CNN models, such as non-linear activation (*e.g.*, the Rectified Linear Unit (ReLU) [6]), dropout [7] and batch normalization [8]. A Fully Connected (FC) layer, can be added at the end of the concatenated layers. It takes all nodes (neurons) from the feature maps of the previous layer as input and connects it to every nodes (neurons) of the output feature map. At the last layer, called dense layer, of the CNN models (referred to as the prediction layer), it is the common to use the Softmax activation function defined as follows:

$$Softmax = \left(\frac{\exp(z_j)}{\sum_{g=1}^{K} \exp(z_g)} \right)_{j=1}^{K} \tag{3}$$

where K denotes the number of categories or classes, $z = (z_1, \ldots, z_K)$ is the output of the affine transformation in the dense layer.

Then, the convolution (CONV) layers and the dense layer of fully connected layer (FC) layers are two common types of layers most of architectures. CONV layers conduct two-dimensional (2D) convolutions on a set of input feature maps and add the results to get output feature maps. FC layers receive a feature vector as input and conduct matrix-vector multiplications.

Besides CONV and FC layers, NN layers also have pooling, ReLU, concat[9], elementwise[10], and other types of layers. But these layers contributes little to the computation and storage requirement of a neural network model. Figure1 shows the distribution of weights and operations in the VGG-11 model. In this model, CONV and FC layers together contribute more than 99% of the network's weights and operations, which is similar to most of the CNN models. It is obvious that most of the neural network acceleration systems must be focus on these two types of layers.

3 Proposed implementation

In this work, we used an Altera FPGA Development Kit to build our CNN accelerator. In particular, the overall memory controller is a DDR3/DDR4 controller, the link controller is a PCIe controller, and the host computer is a desktop PC based on an x86 architecture.

The figure2 illustrates the proposed architecture that consists of four kernels which are connected using Altera OpenCL extension channel/pipes.

The single threaded Convolution kernel is designed to implement both the 3D multiply-accumulate operation, defined by:

$$D_0(f_0, y, x) = \sum_{f_i=1}^{C_l} \sum_{k_y=0}^{K-1} \sum_{k_x=0}^{K-1} W_l(f_0, f_i, k_y, k_x) D_i(f_i, y + k_y, x + k_x) \tag{4}$$

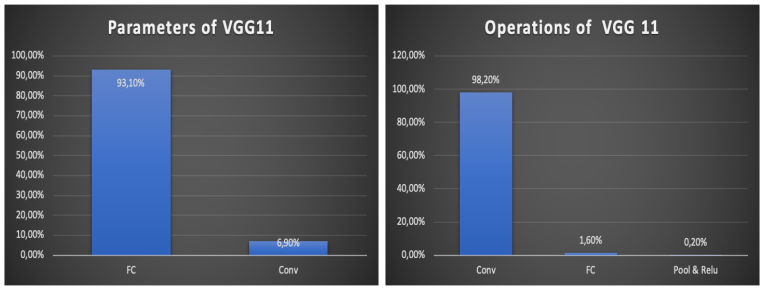


Fig. 1. Distribution of the parameters and the operations in chain based architecture. Example of VGG with 11 layers.

where $D_i(f_i, y, x)$ and $D_0(f_0, y, x)$ denote the neurons located at position (x, y) in the input feature map f_i, and the output feature map f_0, respectively. $W_l(f_0, f_i, y, x)$ represents the corresponding weights in the l^{th} layer which is convoluted with f_i. The size of the convolution filters is $K \times K$, while the total number of input feature maps is C_l. In this paper, we propose to implement 4 using a 1-D convolution structure that flattens 3-D convolution as follows:

$$D_0(f_0) = \sum_{x_i=1}^{C_l \times K \times K} W_l(f_0, x_i) D_i(x_i) \tag{5}$$

where x_i is the index of the parameters of the layer i. Local response normalization (LRN) layers that perform normalization operations on each inputv neuron value by a factor that depends on the neighboring neurons are also used following the pooling layer.

Therefore, we avoid nested 5-way loops levels and we get a 2-level nested loop structure, therefore, the multiplier-adder tree structure with a buffer can be efficiently pipelined by the OpenCL compiler.

Two DataIN and DataOut data transfer kernels inspired by the work of [2], two NDRange 3-D multi-mode transfer data of characteristics and weights from / to the global memory.

In addition to the most compute-intensive convolution kernel, we have designed new OpenCL kernels to speed-up layer operations widely used in CNNs, such as pooling, etc. Therefore, our proposed model can handle the CNN Forward compute stream with very small host CPU involvement, resulting in high throughput and low latency.

Cascading kernels form a deep compute pipeline able to implement a series of basic CNN operations without the need to store the interlayer data in global memory. It greatly reduces the bandwidth requirements.

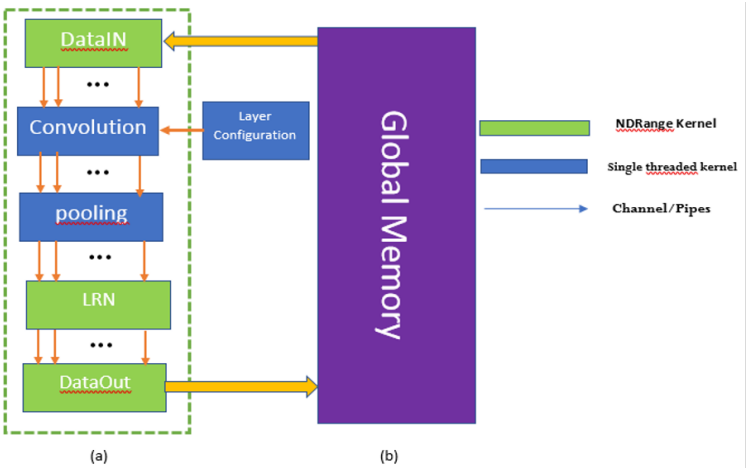


Fig. 2. Proposed CNN accelerator architecture. LRN: local response normalization

4 Results and discussions

In this section, we present the results of the implementation of the OpenCL model proposed on the Alaric board based on Altera Arria 10 GX FPGA and the NallaTech board based on statix - 10 GX 2800.

The Arria 10 FPGA includes 660K logical elements (LE) 1687 DSP blocks and 42MB M20K, while the stratix 10 FPGA includes 2753K logical elements (LE), 5760 DSP blocks and 229MB M20K memory.

It should be noted that the card has a 2 GB DDR3 DRAM connected to the FPGA which functions as global memory for Alaric and 32 GB of DDR4 for Nallatech. OpenCL kernel codes are compiled using Altera OpenCL SDK v16.0 (Alaric) and v18.0 (Nallatech).

The host computer is equipped with an Intel Core i5-4590 processor and is running Ubuntu Linux 14.04.3. We followed the same methodology described in [11].

and we implemented the basic design on the same Arria 10 platform. We also use the Caffe [6] convolutional learning framework as a baseline for our CPU. We extract the input image, pre-trained weights and output functions of Caffe. We compare the result of our implementation with the result of Caffe to verify functional correctness.

Two large-scale CNN models: AlexNet (8 layers) and ResNet-50 (50 layers) models were used as benchmarks to measure performance.

Since CNNs are intensive floating multiplications, the number of DSPs consumed is used as a metric for evaluating performance. As in [2] the proposed CNN design implements full-precision direct computation (32-bit float format), which also makes it favorable for implementing back-propagation flow in the learning phase of the model. To make fair comparison, we provided the normalized performance as "performance density" in the table. It can be noticed that the proposed implementation takes efficiently profit from the DSPs. The classification time is also better than all other implementations.

Table 1. Comparison with other works. 2016a is in [3], FPGA2015 is in [4], and FPGRA2016b is in [2]

	FPGA2016a	FPGA2015	FPGA2016b	This Work	
Device	Stratix-V GXA7	Virtex-7 VX485T	Stratix-V GXA7	Arria 10 GX	Startix 10 GX-2800
FPGA Capacity	622K LUTs 256DSP	485K LUTS 2800 DSP	622K LUTs 256 DSP	660K LUTs 1687 DSP	2.753K LUTs 5760 DSP
Design Scheme	OpenCL	Vivado HLS	OpenCL	OpenCL	
Frequency	120 MHz	100 MHz	181 MHz	167 MHz	275 MHz
Precision	Fixed(8-16b)	Float	Float	Float	
Classification Time	45.7 ms	21.6 ms	43 ms	50 ms	21.2 ms
Throughput	31.8 GOPS	61.6 GOPS	33.9 GOPS	58.45 GOPS	96.25 GOPS
DSP Consumed	246	2240	162	379	181
Performance Density	0.13 GOPS/DSP	0.027 GOPS/DSP	0.21 GOPS/DSP	0.15 GOPS/DSP	0.53 GOPS/DSP

References

1. GUO, Kaiyuan, ZENG, Shulin, YU, Jincheng, et al. [DL] A Survey of FPGA-based Neural Network Inference Accelerators. ACM Transactions on Reconfigurable Technology and Systems (TRETS), 2019, vol. 12, no 1, p.2:1-2:23
2. WANG, Dong, AN, Jianjing, et XU, Ke. PipeCNN: an OpenCL-based FPGA accelerator for large-scale convolution neuron networks. arXiv preprint arXiv:1611.02450, 2016.
3. C. Zhang, P. Li, G. Sun, Y. Guan, B. J. Xiao, and J. Cong, "Optimizing FPGA-based accelerator design for deep convolutional neural networks," in Proc. ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '15), 2015.

4. N. Suda, V. Chandra, G. Dasika, A. Mohanty, Y. F. Ma, S. Vrudhula, J. S. Seo, and Y. Cao, "Throughput-Optimized OpenCL-based FPGA accelerator for large-scale convolutional neural networks," in Proc. ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '16), 2016.
5. Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning ap- plied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324. doi:10.1109/5.726791.
6. K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, IEEE International Conference on Computer Vision (ICCV 2015) 1502.
7. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, Journal of Machine Learning Research 15 (2014) 1929–1958.
8. S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network train- ing by reducing internal covariate shift, in: Proceedings of the 32Nd In- ternational Conference on International Conference on Machine Learning - Volume 37, ICML'15, JMLR.org, 2015, pp. 448–456.
9. Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. 2015. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'15).
10. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 770–778
11. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems. 1097–1105.
12. Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) 115, 3 (2015), 211–252. https: //doi.org/10.1007/s11263-015-0816-y
13. Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition. 580–587.
14. Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014).
15. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and paˆern recognition. 770–778.

Bayesian Optimization with deceptive Gaussian Processes How to maintain efficiency in the optimization process ?

J. Sadet[1,2], F. Massa[4], T. Tison[1], E-G. Talbi[2], and I. Turpin[3]

[1] Polytechnic University of Hauts-de-France, LAMIH, CNRS, UMR 8201, F-59313 Valenciennes, France
{jeremy.sadet,thierry.tison}@uphf.fr
[2] University of Lille, CRIStAL UMR CNRS 9189, Inria-Lille Nord Europe
el-ghazali.talbi@univ-lille.fr
[3] Polytechnic University of Hauts-de-France, LMI, F-59313 Valenciennes
isabelle.turpin@uphf.fr
[4] Polytechnic University of Hauts-de-France, LAMIH, CNRS, UMR 8201, F-59313 Valenciennes, France
INSA Hauts-de-France, F-59313 Valenciennes, France
franck.massa@uphf.fr

This contribution deals with the optimization of highly expensive black-box functions, used in engineering applications. Thus, optimization of such systems becomes a challenging task which cannot be solved with classical optimization algorithms. One way to alleviate this cost is to use surrogate models and replace black-box functions by a cheaper and faster model. Based on a training set which contains prior knowledge about the function, surrogate modelling have already exhibited probant results [1–3]. From all surrogate models, Gaussian Process represents one of the most efficient surrogate model due to its ability to resist overfitting and quantify modelling uncertainty. Several authors have highlighted performances of Gaussian Process in brake squeal analysis [4, 5], identification of breathing cracks of rotors [6] and digital twins [7].

Albeit Gaussian Process is a practical surrogate model, problematical predictions may arise from a loss of correlation between the samples of the training set, due to a small lengthscale after optimization of the surrogate model [8, 7, 9]. This issue is all the more frequent when the function under study is highly non linear, non stationary and multimodal.

In this contribution, we investigate the impact of this phenomenon on Bayesian Optimization and suggest a method to handle it. Then, the performance of the suggested strategy is assessed with a one mathematical problem well known in the literature.

1 Theoretical aspects

Gaussian Processes (hereby denoted GP) are a probabilistic class of surrogate models where the output prediction $\widehat{\mathbf{Y}}$ is described as a Gaussian random vector. Its first two statistical moments (mean $\widehat{y}$ and variance $\widehat{s}$) are shown in (Eq. 1) for a given set of parameter values $\mathbf{x}_*$.

$$\begin{aligned}\widehat{y}(\mathbf{x}_*) &= \mathbf{C}(\mathbf{x}_*, \mathbf{X})\mathbf{C}(\mathbf{X}, \mathbf{X})^{-1}\mathbf{y} \\ \widehat{s}(\mathbf{x}_*) &= \mathbf{C}(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{C}(\mathbf{x}_*, \mathbf{X})\mathbf{C}(\mathbf{X}, \mathbf{X})^{-1}\mathbf{C}(\mathbf{x}_*, \mathbf{X})^T\end{aligned} \tag{1}$$

where $\mathbf{C}(\cdot, \cdot)$ is the covariance matrix, $\mathbf{X}$ the matrix of input parameter values and $\mathbf{y}$ the vector of solutions of the solver given $\mathbf{X}$.

The numerical workflow for Bayesian Optimization with GP is shown in Fig. 1, as long as the references to corresponding equations. The notations in Fig. 1 rely on the following data, namely σ_k is the signal variance (or nugget), θ_i is the lengthscale associated with the dimension i, n is the number of input parameter values of $\mathbf{X}$, k is an hyperparameter value which controls the constraint over the likelihood and ϵ is a precision parameter equal to $5 \cdot 10^{-3}$.

At first, a training set, initialized with a Latin Hypercube Sampling procedure, is computed to provide prior knowledge about the behavior of the considered function. Then, the inference is performed by selecting a covariance function (Eq. 2) and finding θ_i that maximize the likelihood (Eq. 3). In this communication, we mainly focus on Gaussian likelihood and Matern 3/2 covariance function. Finally, the prediction is carried out using (Eq. 1).

In Bayesian Optimization, the mean and variance of output prediction $\widehat{\mathbf{Y}}$ is used to determine the location of a solution which represents the best guess for the true global optimum of the

considered function. Via iterative incrementations of the training set and the use of an acquisition function to determine the new best candidate, the algorithm converges toward the global optimum.

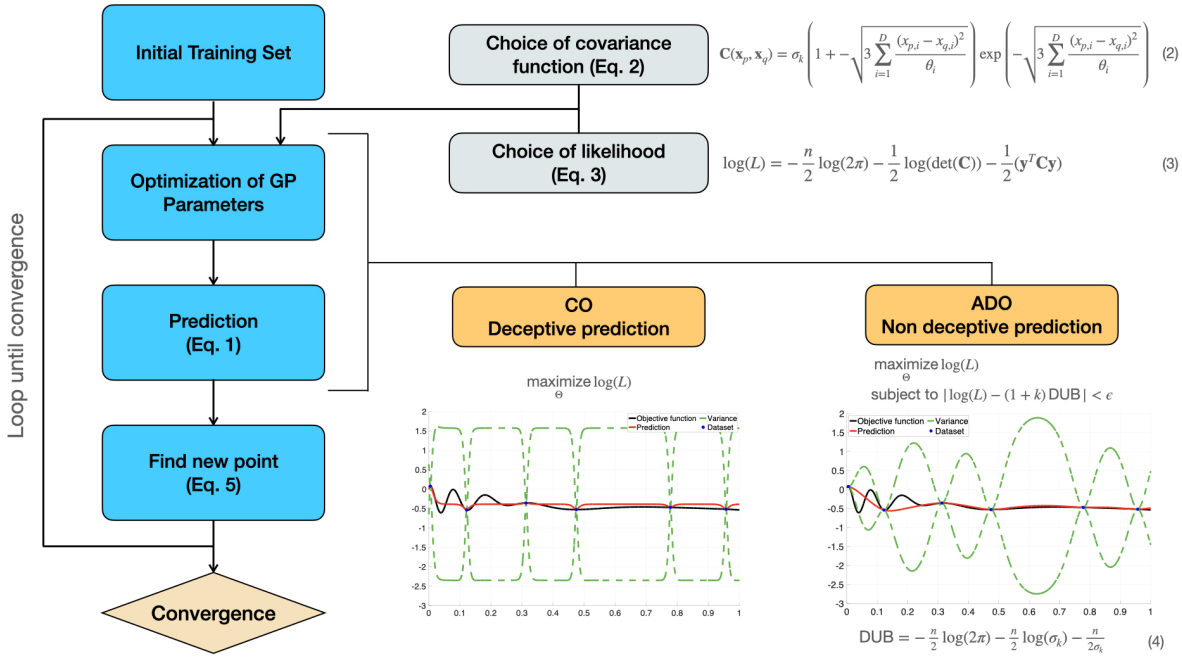


Fig. 1: Workflow of Bayesian Optimization procedure (blue rectangles) with a comparison between the Classical Optimization of GP parameters prediction (denoted CO procedure) and the suggested Anti-Deceptive Optimization prediction (denoted ADO procedure) for a given training set

A common choice for acquisition function is the Expected Improvement (Eq. 5), introduced by [10], which relies on a compromise between exploitation (first term of the sum, searching for the mininum) and exploration (second term of the sum, diversifying the search).

$$A_{EI}(x_c) = (y_{min} - \widehat{y}(x_c)) \left[\frac{1}{2} + \frac{1}{2}\text{erf}\left(\frac{y_{min} - \widehat{y}(x_c)}{\sqrt{2}\,\widehat{s}(x_c)}\right)\right] + \frac{\widehat{s}(x_c)}{\sqrt{2\pi}} \exp\left[-\left(\frac{y_{min} - \widehat{y}(x_c)}{(\sqrt{2}\,\widehat{s}(x_c))}\right)^2\right] \quad (5)$$

where x_c corresponds to the candidate point and y_{min}, the minimum of the training set output.

Nevertheless, as the maximization of the likelihood succeeds in providing good approximations most of the time, some training sets may induce poor predictions which annihilate the approximation power of GP. This phenomenon is as all the more so frequent that the considered function is highly non-stationary and non-linear and is characterized by an optimum of (Eq. 3) associated with a lengthscale θ_i going to 0^+. It is referred as deceptive prediction.

As emphasized by Fig. 1, a deceptive prediction, associated with the CO procedure, is defined by almost constant mean and variance prediction. To handle these peculiar scenarii, we suggested a method in [11] which consists in constraining the optimization of the likelihood by forcing the maximum of the likelihood to be smaller than a precision value. This method hinges on the definition of a criterion, the Deceptive Upper Bound (hereby denoted DUB), which allows to detect these bad predictions. This method works well to alleviate the deceptiveness behavior of the prediction as shown in Fig. 1.

From the aspect of the mean and variance when considering CO procedure, it is clear that the impact is important on the selection of new candidates in Bayesian Optimization algorithm. Thus, the following section is focusing on quantifying this impact over the process.

2 Numerical application

To illustrate the impact of deceptiveness over the convergence toward the global optimum, we consider the Xiong's function (Eq. 6) introduced in [12]. This function is highly non-stationary which causes the optimization to begin quite challenging. The global minimum value is -0.6093.

$$f(x) = -0.5\left(\sin\left(40(x-0.85)^4\right)\cos(2.5(x-0.95)) + 0.5(x-0.9) + 1\right) \tag{6}$$

Both procedures have been evaluated for 50 different training sets, whose sizes are all equal to 10 samples. At first, results show that both procedures perform almost equally. A very small speed up has been detected with ADO procedure (about 0.6 computations less than CO procedure for the 50 computations).

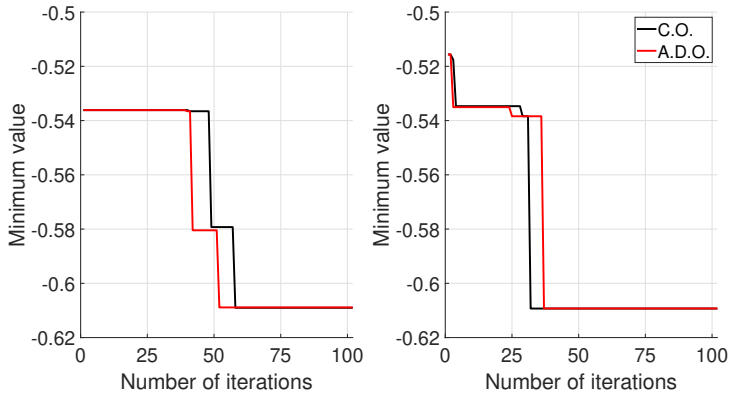


Fig. 2: Convergence of two different initial training sets for both optimization procedure

Fig. 2 exhibits two examples of convergence from the considered benchmark of 50 training sets and where great differences have been noticed. The left one shows a faster convergence of ADO procedure, which needs 51 computations against 57 for CO. On the other hand, the right one presents a faster convergence of CO which needs 29 computations against 34 for ADO.

The reason of this performance of deceptive prediction comes from a flaw of the Expected Improvement. Some areas of the design space might be overlooked because it only focuses on non-explored areas with a potential of holding the optimum. Thus, it induces a non-optimal diversification. This lack of diversification in the exploration is also noticeable for other acquisition functions.

As regards of deceptiveness, new samples are almost added "randomly" since the mean and variance provided to the Bayesian Optimization algorithm are flawed. The new sample may be added anywhere in the design space. Consequently, deceptiveness, which is primarily a defect of GP, becomes helpful by bringing diversification during the Bayesian Optimization algorithm.

3 Conclusions

In this communication, the performance of Bayesian Optimization was investigated while considering deceptive Gaussian Processes. The phenomenon of deceptiveness was highlighted and a method for detection and correction was described. Then, the impact of deceptiveness over Bayesian Optimization convergence has been emphasized with a non-stationary mathematical function.

This work has allowed to show that the deceptiveness is a phenomenon that penalizes the performance of Gaussian Process for prediction whereas, for Bayesian Optimization, this phenomenon may help the algorithm to converge faster. Indeed, it allows to introduce diversification in the search of new potential candidates for global optimum. Nevertheless, it remains a flaw of the Gaussian Process formalism and other methods to introduce diversification in the process have to be studied.

Acknowledgements

This work was carried out within the framework of the CNRS Research Federation on Ground Transports and Mobility, in articulation with the ELSAT2020 project supported by the European Community, the French Ministry of Higher Education and Research, the Hauts de France Regional Council.

References

1. Liping Wang, Don Beeson, Gene Wiggs, and Mahidhar Rayasam. *A Comparison Meta-Modeling Methods Using Practical Industry Requirements*.
2. M. J. Asher, B. F. W. Croke, A. J. Jakeman, and L. J. M. Peeters. A review of surrogate models and their application to groundwater modeling. *Water Resources Research*, 51(8):5957–5973, 2015.
3. Bruno Sudret. Meta-models for structural reliability and uncertainty quantification. In *"Asian-Pacific Symposium on Structural Reliability and its Applications"*, pages 1–24, Singapore, Singapore, May 2012.
4. L. Nechak, F. Gillot, S. Besset, and J.-J. Sinou. Sensitivity analysis and kriging based models for robust stability analysis of brake systems. *Mechanics Research Communications*, 69:136 – 145, 2015.
5. Amir Nobari, Huajiang Ouyang, and Paul Bannister. Uncertainty quantification of squeal instability via surrogate modelling. *Mechanical Systems and Signal Processing*, 60-61:887–908, 02 2015.
6. Zhiwen lu, Yong Lv, and Huajiang Ouyang. A super-harmonic feature based updating method for crack identification in rotors using a kriging surrogate model. *Applied Sciences*, 9:2428, 06 2019.
7. S. Chakraborty, S. Adhikari, and R. Ganguli. The role of surrogate models in the development of digital twins of dynamic systems. *Applied Mathematical Modelling*, 90:662 – 681, 2021.
8. Romain Benassi, Julien Bect, and Emmanuel Vazquez. Robust Gaussian process-based global optimization using a fully Bayesian expected improvement criterion. In *5th International Conference on Learning and Intelligent Optimization (LION 5)*, volume 6683 of *Lecture Notes in Computer Science*, pages 176–190, Rome, Italy, January 2011. Springer.
9. Ali Hebbal, Loïc Brevault, Mathieu Balesdent, El-Ghazali Talbi, and Nouredine Melab. Bayesian optimization using deep Gaussian processes with applications to aerospace system design. *Optimization and Engineering*, June 2020.
10. Donald R Jones, Matthias Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.
11. Jeremy Sadet, Franck Massa, El-Ghazali Talbi, Thierry Tison, and Isabelle Turpin. Bayesian Optimization of Gaussian Process and Deep Gaussian Process for the optimization of black-box functions. In *"International Conference on Optimization and Learning"*, 2020.
12. Ying Xiong, Wei Chen, Daniel Apley, and Xuru Ding. A nonstationary covariance based kriging method for metamodeling in engineering design. *International Journal for Numerical Methods in Engineering*, 71:733 – 756, 08 2007.

A multi-objective approach for an energy-efficient buffer allocation

Y. Alaouchiche[1] Y. Ouazene[1] and F. Yalaoui[1]

Université de Technologie de Troyes
yasmine.alaouchiche@utt.fr
yassine.ouazene@utt.fr
farouk.yalaoui@utt.fr

Abstract. The problem of finding the optimal allocation of storage capacities in a production line, known as the buffer allocation problem (BAP), is one of the most researched problems in manufacturing systems design. However, the current context requires substantial efforts for including sustainability concerns to the issue. Therefore, considering the energetic dimension becomes crucial. In this study, we present a novel variant of the BAP for throughput maximization and energy consumption minimization. The problem is solved using a multi-objective approach.

Keywords : Buffer allocation problem, Production lines, Multi-objective optimization, Energy efficiency

1 Introduction

One of the major concerns that face researchers and industrials in the design of production systems is the buffer allocation problem (BAP). Buffers are used to compensate the negative effects of machines unreliability. However, larger storage capacities result in important costs for both investment and work-in-process inventory. Therefore, the issue of finding the optimal buffering is intensively studied since decades.
A recent paper of [9] reviews the literature related to the BAP. Although various versions of the BAP are presented, the dual and the primal BAPs are the most studied. These models were first introduced by [2]. In the dual BAP, the objective is to maximize the throughput under total buffer space constraint, whereas in the primal BAP, the objective is to minimize the total storage space under a minimal required throughput constraint.
Nevertheless, with the current context of ecological awareness, limited energy sources and, increasing energy costs, including the energetic dimension in the BAP becomes crucial. Therefore, in this study, we formulate a novel variant of the BAP that optimizes energy consumption along with throughput of unreliable production lines.

2 Problem formulation

In this study, a serial production line composed of K unreliable workstations and $K-1$ buffer areas with finite capacities is considered. This system is presented in figure 1. The processing, failure

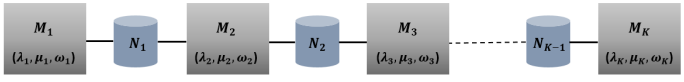


Fig. 1: Serial production line

and, repair rates of the machines, ω_i, λ_i and, μ_i respectively, are assumed to be exponentially

distributed. It is also assumed that there is unlimited supply before the first machine and unlimited storage capacity after the last machine. Therefore, the first machine cannot be starved and the last machine cannot be blocked. In addition, operation dependent failure is assumed, no setup time is considered and, transitions times between machines and buffers are assumed equal to zero.

In the BAP, buffer capacities are finite variables to be determined. Based on our literature review, it is clear that this problem has been widely studied. However, research efforts focus mainly on the issue of productivity improvement. Nevertheless, the current context requires a crucial focus on energy efficiency in the design of manufacturing systems. The novel variant of the BAP proposed considers a multi-objective optimization: energy consumption minimization along with throughput maximization under total buffer space constraint. The mathematical model of this problem is given as follows:

$$\text{Find } N = (N_1, N_2, ..., N_{K-1}) \text{ so as to :}$$

$$\begin{cases} (Max\ \psi\ and\ Min\ E) & \\ \text{s.t.} & \\ \sum_{j=1}^{K-1} N_j \leq N_{total} & j \in 1...K-1, \\ N_j \in \mathbb{N}^* & j \in 1...K-1. \end{cases} \tag{1}$$

K is the number of machines in the production system, ψ the throughout of the line and, E the total energy consumption. Moreover, N is the buffer size vector and N_{total} the total buffer space available to be allocated among the $K-1$ buffer areas. $N_j, \forall j = 1...K-1$ are non-negative integers denoting the capacity allocated for each buffer B_j.

3 Performance evaluation approach

The evaluation approach of the two crucial performances considered in the problem, i.e. throughput and energy consumption of the line, are obtained using the performance evaluation method developed in [1]. This recently developed method is, according to our literature review, the unique study that considers the integrated evaluation of throughput and energy consumption of unreliable production lines.

In this method, the throughput is evaluated using birth death Markov processes (the Equivalent Machine Method [4]). In this analytical formulation, the different states of each buffer are analyzed using birth-death Markov processes. Thereafter, each original machine is replaced by an equivalent one taking into account the probabilities of blockage and starvation. The throughput of the production line is defined as the bottleneck between the effective production rates of the equivalent machines. Due to its main approach that considers only full and empty buffer states, the state space cardinality of the Markov chain representation of the system is reduced. Results from numerical experiments demonstrate a high accuracy with extensively reduced computational time when compared to other methods from the literature, such as the decomposition and aggregation methods.

The second part of the method evaluates the energy consumption and efficiency of the production line. The energy consumption of the line is the sum of energy consumption of its K machines. For each machine, the energy consumption is evaluated per machine state. A Markov chain formulation is used to obtain transition and steady state probabilities for each state. These probabilities are obtained as a function of machine parameters as well as probabilities of empty and full buffer states derived from the throughput evaluation part. Thereafter, energy consumption and energy efficiency are formulated for each machine $M_i, \forall i = 1...K$ and consequently for the production line, using steady state probabilities and specific state energy consumption. Corresponding formulations for throughput and energy consumption evaluation as well as calculation details can be found in [1] and [4].

4 Multi-objective approach and numerical experiments

The buffer allocation problem formulated in equation 1 requires a multi-objective resolution approach. In order to find the closest possible set of solutions to Pareto optimal front, ϵ- constraint

method [3] is used. In this method, one of the objective functions is selected to be optimized while the other(s) are converted into additional constraints. In our case, the throughput is maximized and the energy consumption minimization is converted into a constraint. Therefore, the problem becomes a single-objective problem formulated as follows:

$$\text{Find } N = (N_1, N_2, ..., N_{K-1}) \text{ so as to :}$$

$$\begin{cases} Max\ \psi & \\ \text{s.t.} & \\ E \leq \epsilon & \\ \sum_{j=1}^{K-1} N_j \leq N_{total} & j \in 1...K-1, \\ N_j \in \mathbb{N}^* & j \in 1...K-1. \end{cases} \tag{2}$$

The model is formulated as a mixed integer non-linear program implemented on Lingo solver. Numerical experiments are conducted on literature instances. The proposed BAP with throughput and energy optimization is compared to the dual BAP that focuses on the throughput maximization. The aim is to highlight potential improvements and energy economics allowed by the model. Results for the ϵ-constraint method using the instance represented in table 1 (refer to [1] for energy parameters), are given in figure 2. Throughput loss and energy savings denote the gap between of the proposed BAP and the dual BAP for respectively the throughput and the energy consumption. Results show that the model can be used to develop a significant tool for decision making.

Table 1: Production line instance parameters [2, 6, 7, 8, 5]

Machine	M1	M2	M3	M4	M5
Failure rate	*0.3*	*0.4*	0.45	0.35	0.1
Repair rate	0.64	0.83	0.75	0.85	0.74
Processing rate	2.8	1.7	2.5	3.4	1.9

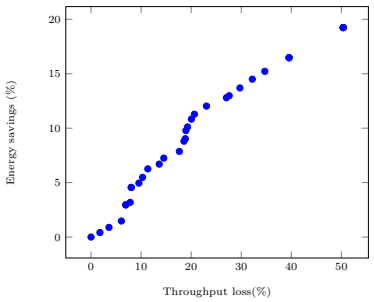


Fig. 2: Multi-objective optimization using ϵ- constraint method

5 Conclusion

In this paper, a novel variant of the BAP with energy minimization and throughput maximization was proposed. This problem, that requires a bi-objective optimization approach, is solved using ϵ-constraint method. Numerical experiments highlight the relevance of considering energy consumption along with throughput optimization in the design of storage spaces. Future work focuses on the multi-objective study of the energy-efficient BAP considering other approaches. Moreover, this novel BAP allows to explore a new field of research and tackle more interesting areas in manufacturing systems design.

References

[1] Alaouchiche, Y., Ouazene, Y., & Yalaoui, F. (2020). Economic and energetic performance evaluation of unreliable production lines: An integrated analytical approach. *IEEE Access*, *8*, 185330–185345.

[2] Gershwin, S. B., & Schor, J. E. (2000). Efficient algorithms for buffer space allocation. *Annals of Operations research*, *93*, 117–144.

[3] Haimes, Y., Lasdon, L., & Wismer, D. (1971). On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man, and Cybernetics - TSMC*, *1*, 296–297. doi:10.1109/TSMC.1971.4308298.

[4] Ouazene, Y., Chehade, H., Yalaoui, A., & Yalaoui, F. (2013). Equivalent machine method for approximate evaluation of buffered unreliable production lines. In *2013 IEEE Symposium on Computational Intelligence in Production and Logistics Systems (CIPLS)* (pp. 33–39). IEEE.

[5] Ouazene, Y., Yalaoui, A., Yalaoui, F., & Chehade, H. (2014). Non-linear programming method for buffer allocation in unreliable production lines. In *International Conference on Analytical and Stochastic Modeling Techniques and Applications* (pp. 80–94). Springer.

[6] Papadopoulos, H., & Vidalis, M. (2001). A heuristic algorithm for the buffer allocation in unreliable unbalanced production lines. *Computers & Industrial Engineering*, *41*, 261–277.

[7] Sabuncuoglu, I., Erel, E., & Gocgun, Y. (2006). Analysis of serial production lines: characterisation study and a new heuristic procedure for optimal buffer allocation. *International Journal of Production Research*, *44*, 2499–2523.

[8] Seong, D., Chang, S. Y., & Hong, Y. (1995). Heuristic algorithms for buffer allocation in a production line with unreliable machines. *International Journal of Production Research*, *33*, 1989–2005.

[9] Weiss, S., Schwarz, J. A., & Stolletz, R. (2019). The buffer allocation problem in production lines: Formulations, solution methods, and instances. *IISE Transactions*, *51*, 456–485.

Investigating Overlapped Strategies to Solve Overlapping Problems in a Cooperative Co-evolutionary Framework*

Julien Blanchard[1][0000−0002−3423−4521], Charlotte Beauthier[2][0000−0001−6413−0080], and Timoteo Carletti[1][0000−0003−2596−4503]

[1] Department of Mathematics and naXys institute, University of Namur, Namur, Belgium
{julien.blanchard,timoteo.carletti}@unamur.be
[2] Cenaero Research Center, Gosselies, Belgium
charlotte.beauthier@cenaero.be

Abstract. Cooperative co-evolution is recognized as an effective approach for solving large-scale optimization problems. It breaks down the problem dimensionality by splitting a large-scale problem into ones focusing on a smaller number of variables. This approach is successful when the studied problem is decomposable. However, many practical optimization problems can not be split into disjoint components. Most of them can be seen as interconnected components that share some variables with other ones. Such problems composed of parts that overlap each other are called overlapping problems. This paper proposes a modified cooperative co-evolutionary framework allowing to deal with non-disjoint subproblems in order to decompose and optimize overlapping problems efficiently. The proposed algorithm performs a new decomposition based on differential grouping to detect overlapping variables. A new cooperation strategy is also introduced to manage variables shared among several components. The performance of the new overlapped framework is assessed on large-scale overlapping benchmark problems derived from the CEC'2013 benchmark suite and compared with a state-of-the-art non-overlapped framework designed to tackle overlapping problems.

Keywords: large-scale global optimization · evolutionary algorithms · cooperative co-evolution · overlapping problem.

1 Introduction

Nowadays, many real-world optimization problems arising in engineering and sciences deal with a large number of variables [7]. They present challenging characteristics making them hard to efficiently optimize. They are commonly solved by means of metaheuristics such as evolutionary algorithms or swarm

* The present research benefited from computational resources made available on the Tier-1 supercomputer of the Fedration Wallonie-Bruxelles, infrastructure funded by the Walloon Region under the grant agreement n°1117545.

intelligence [3]. However, the standard metaheuristics are not suitable to solve such large-scale global optimization (LSGO) problems because they suffer from the curse of dimensionality, i.e. their performance deteriorates when increasing the number of variables [1]. In this context, new approaches relying on the "divide-and-conquer strategy" have been proposed. They divide the initial LSGO problem into smaller ones which focus on smaller groups of variables. The latter are optimized in a round-robin fashion with a standard metaheuristic with the aim of producing the solution of the initial problem. This framework has been introduced by Potter and De Jong [9]. They designed a cooperative coevolutionary (CC) approach to optimize LGSO problems by means of a genetic algorithm. Following this promising approach, the CC strategy have been embedded in many other metaheuristics such as evolutionary programming [6], particle swarm optimization [2] and differential evolution [11].

In any case, the efficiency of this approach is highly dependent on the performed decomposition. The latter depends on the characteristics of the objective function in terms of separability. A function is separable if the influence of any variable on the function value depends only on itself [18]. In this case, any decomposition that reduces the dimensionality is efficient in the CC framework. Other functions can be classified as additively separable [8] if they can be written as:

$$f(x) = \sum_{i=1}^{m} f_i(x_i), \tag{1}$$

where x_i $(i = 1, \ldots, m)$ are mutually exclusive k_i-dimensional decision vectors of f_i, x is the n-dimensional decision vector of the function f and m is the number of independent components such that $k_1 + \ldots + k_m = n$. In this way, the influence, of any variable in a component, on the function value depends only on other variables of the same component. Therefore, an ideal decomposition would divide the initial problem such that each subproblem focuses on one component given in Equation (1). The main challenge is thus to identify these components. It can be done by using the differential grouping strategy [8, 16].

However, separable and partially separable problems are not representative of most LSGO problems arising in real-world optimization applications. Most of them incorporate several components that usually interact with each other. For example, the supply chain design and optimization [4] involves several components such as suppliers, manufacturers and distributors that interact with each other through a variety of transportation and delivery methods. Such interconnected problems are often referred as overlapping problems [17] because they are composed of parts that overlap others. In other words, each component involves multiple variables and some of them are shared with one or several other components. This kind of function is very challenging and standard CC algorithms fail to optimize them efficiently. Indeed, most of them rely either on random grouping [18] or on intelligent decomposition methods based on interaction identification [8]. The former simply completes several random decompositions in order to try catching linked variables in a same component but does not explicitly consider the interaction structure. The latter assigns all the linked variables in a

single group and therefore does not reduce the dimensionality when dealing with overlapping problems. Two exceptions are the decomposition based on spectral clustering introduced in [5] and the decomposition specially designed for overlapping problems introduced in [15]. The latter breaks the linkage at shared variables between components in order to reduce the problem dimensionality, even for overlapping problems. It will be further discussed in Section 2.2.

In addition to the above methods, other CC strategies considering subsets that overlap each other have also received some attention. They raise some questions related to the exchange of information between components and related to the construction of the complete n-dimensional solution. In [14], non separable problems are decomposed into overlapping subproblems on the basis of a statistical variable interdependence learning scheme. The exchange of information is ensured by a periodically updated global solution (built on the basis of subproblem cores) used as shared memory. In [13], an overlapping decomposition covering the set of variables is predetermined. Compete and sharing strategies are implemented to choose the representative variables and share them among components. In [12], overlapping is not used to facilitate the decomposition but to overlap influential variables and evolves them in several components.

Some of these algorithms claim to tackle overlapping problems but do it with non-overlapped strategies [5, 15]. Others, although based on overlapped strategies, do not explicitly claim to be able to tackle overlapping problems [12–14]. One may obviously think that the best way to optimize them in a CC framework is to do it with overlapped strategies. Nevertheless, to the best of the authors' knowledge, there are no research studies in that way. This paper introduces such a strategy and compare it with the non-overlapped approach specially designed for overlapping problems in [15]. The paper is organized as follows: Section 2 briefly describes the CC framework and the recursive differential grouping. Section 3 introduces the new strategy to split LSGO problems into overlapping subproblems and the overlapped CC framework that manages the exchange of information between subproblems. Experimental settings and results analysis are given in Section 4. Finally, findings and perspectives are discussed in Section 5.

2 Related work

2.1 Cooperative Co-evolutionary algorithms

The first attempt to optimize a LSGO problem with an evolutionary algorithm by means of a divide-and-conquer strategy was presented in 1994 [9]. Since then, this new approach, called cooperative co-evolution, has been widely studied [7]. The classical structure of this framework is described as follows:

1. *Decomposition*: Split the n-dimensional decision vector into some smaller disjoint subcomponents;
2. *Optimization*: Optimize each subcomponent with a standard evolutionary algorithm for a fixed number of iterations in a round-robin strategy;

3. *Combination*: Merge the solutions from each subcomponent to build the n-dimensional solution.

Throughout the optimization stage, the individuals in each subcomponent need to be evaluated with the n-dimensional function. For this purpose, they are completed with the variables of the *context vector*. The latter is a n-dimensional vector that contains information from all the subcomponents. Typically, it is composed of the variables of the current best solutions in each subcomponent and it is updated each time a better solution is found in a subcomponent.

2.2 Recursive Differential Grouping

In a CC framework, the decomposition should ideally be performed in such a way that there is no interaction between variables from different subcomponents. For additively separable problems, it can be uncovered with the Differential Grouping (DG) strategy [8, 16]. In particular, the Recursive Differential Grouping (RDG) that benefits, as stated by its name, from recursive interaction detections between subsets of variables, relies on the following result [15, 16]:

Theorem 1. *Let $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ be an objective function; X_1 and X_2 be two mutually exclusive subsets of decision variables: $X_1 \cap X_2 = \emptyset$. If there exist a candidate solution $x^\star$ and sub-vectors a_1, a_2, b_1, b_2 such that*

$$f_{1,1}(x^\star) - f_{2,1}(x^\star) \neq f_{1,2}(x^\star) - f_{2,2}(x^\star) \tag{2}$$

where, $f_{i,j}(x^\star)$ is the function value obtained when replacing, in $x^\star$, the variables of X_1 with a_i and the variables of X_2 with b_j ($i, j = 1, 2$), then there is some interaction between the decision variables in X_1 and X_2.

In practice, all the variables of $x^\star$, a_1 and b_1 are set to the lower bounds l of the search space. The variables of a_2 are set to the upper bounds u and those of b_2 are set to the mean $\bar{m}$ of the lower bounds and the upper bounds. Furthermore, equation (2) is not directly employed since the inequality may be the results of computational round-off errors instead of interaction detection, as expected. Thus, the following quantities are computed

$$\Delta_1 = f_{1,1}(x^\star) - f_{2,1}(x^\star),\ \Delta_2 = f_{1,2}(x^\star) - f_{2,2}(x^\star),\ \lambda = |\Delta_1 - \Delta_2| \tag{3}$$

and some interaction is detected when λ is greater than a threshold ϵ (see [16] for further details). Eventually, the success of the RDG algorithm relies on the recursive use of Theorem 1 to identify variables in X_2 that interact with those of X_1. Indeed, if any interaction between X_1 and X_2 is detected using Equation (3), the set X_2 is divided into two nearly equally-sized groups G_1 and G_2. Then, the interaction between X_1 and G_1 and X_2 and G_2 is checked. The process is repeated until all single variables in X_2 that interact with X_1 are identified.

In brief, the complete RDG algorithm can be presented as follows: (1) determine all the variables that interact with a selected variable x_i using the above recursive strategy and put them in a set X_1; (2) identify variables that interact

with X_1 and add them to X_1, repeat the process until no more variable is added to X_1; (3) select another variable that is yet to be classified and return to step (1). Note that this approach would set all the variables of an overlapping problem into a single group. In [15], this issue was solved by slightly modifying the step (2) by imposing a condition on the size of X_1. In this new approach called RDG3, the step (2) is repeated: (a) until no more variable is added to X_1 or (b) until X_1 contains more than ϵ_n variables, where ϵ_n is fixed to a predetermined value.

3 Proposed algorithm

The newly proposed algorithm aims to tackle LSGO overlapping problems within an overlapped CC framework. The fact that it has to deal with subcomponents that share several variables raises new challenges. The first one is to perform an accurate decomposition that detects overlapping variables efficiently and share them among several subcomponents. It can be achieved by using the modified approach of the RDG strategy presented in Section 3.1. The second challenge concerns the management of overlapping variables during the optimization, in particular for function evaluations. It will be discussed in Section 3.2.

3.1 Overlapped Recursive Differential Grouping

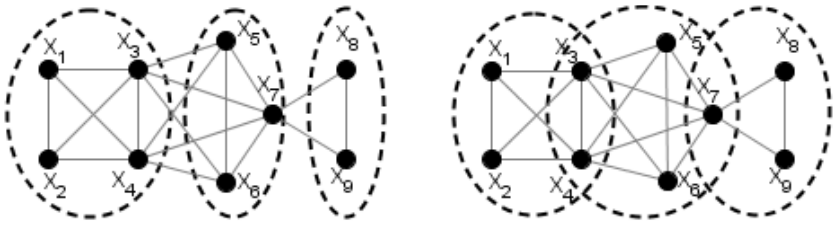
The main idea of the newly proposed decomposition strategy is to relax the grouping by identifying variables that make the link between several components in interconnected problems and share them among these components. For example, in the interaction graph presented in Figure 1, three components can be identified:

$$S_1 = \{x_1, x_2, x_3, x_4\}, S_2 = \{x_3, x_4, x_5, x_6, x_7\} \text{ and } S_3 = \{x_7, x_8, x_9\}. \quad (4)$$

In each of them, interaction between variables are plentiful while there is no direct interaction between variables from distinct components, i.e. $\forall\, i, j(i \neq j), k, l(k \neq l)$ such that $x_i \in S_k \backslash S_l$ and $x_j \in S_l \backslash S_k$, x_i does not interact with x_j. Using the RDG3 strategy to decompose such a problem will break the linkage at shared variables and will lead to the decomposition illustrated in Figure 1a. The latter might not be the optimal one since x_3 and x_4 (resp. x_7) are not optimized with x_5, x_6 and x_7 (resp. x_8 and x_9) while they are strongly connected. The new strategy, called *Overlapped RDG* (ORDG), is aimed to allow some overlapping between subcomponents to prevent from breaking these important linkages. It will produce the decomposition proposed in Figure 1b.

The ORDG strategy is presented in Algorithm 1. It is very closed to the RDG algorithm except for the instructions in the "else" statement at line 12. In particular, the instruction at line 5 recursively identifies variables in X_2 that interact with X_1. They are added to X_1 to constitute the set $X_1^\star$ (see Algorithm 2).

- If no interaction has been identified, i.e. if $|X_1^\star| = |X_1|$, the X_1 set is recognized as a nonseparable subset if it contains several variables, otherwise the



(a) RDG3 (b) Overlapped RDG

Fig. 1: The two obtained decompositions for an overlapping problem using RDG3 and Overlapped RDG strategies respectively.

Algorithm 1: Overlapped Recursive Differential Grouping

$seps = \{\}$, $nonseps = \{\}$;
Set all the variables of x_l to the lower bounds, compute $\underline{f} = f(x_l)$;
$X_1 = \{x_1\}$, $X_2 = \{x_2, \ldots, x_n\}$;
while $X_2 \neq \{\}$ **do**
  $X_1^\star$ =R_Inter$(X_1, X_2, \underline{f}, f)$;
  if $|X_1^\star| = |X_1|$
    `// For RDG3, the if would be: if` $|X_1^\star| = |X_1|$ `or` $|X_1^\star| > \epsilon_n$
  then
    if $|X_1| = 1$ **then** $seps = seps \cup X_1$;
    else $nonseps = nonseps \cup X_1$;
    $X_1 = \{x_j\}$ s.t. $j \leq i \; \forall x_i \in X_2$;
    $X_2 = X_2 \setminus \{x_j\}$;
  else
    `// For RDG3, the else statement would only contains the following instructions:` $X_1 = X_1^\star$, $X_2 = X_2 \setminus X_1$;
    if $|X_1| = 1$ **then**
      $X_1 = X_1^\star$, $X_2 = X_2 \setminus X_1$;
    else
      $X_1^{\star\star} = L\_inter(X_1, X_2, \underline{f}, f)$;
      $nonseps = nonseps \cup X_1$;
      $X_1 = X_1^\star \setminus X_1 \cup X_1^{\star\star}$;
      $X_2 = X_2 \setminus X_1^\star$;
if $|X_1| = 1$ **then** $seps = seps \cup X_1$;
else $nonseps = nonseps \cup X_1$;
return *seps and nonseps*;

only variable in X_1 is identified as a separable one (lines 8-9). The process moves on to the next variable that is yet to be classified (lines 10-11).
– Otherwise, some interaction has been identified between X_1 and X_2. The variables in X_2 responsible of the interaction have been identified during the recursive detection at line 5 but at this stage, the variables in X_1 responsible of the interaction have not yet been determined (and they should be to perform the overlapped decomposition). If X_1 contains only one variable, this is the one responsible of the interaction. In this case, the algorithm moves on to the next iteration while making the same update that for the RDG3 strategy (lines 13-14). Otherwise (i.e. if X_1 contains several variables), those interacting with X_2 are identified at line 16 using a recursive mechanism again (see Algorithm 3) and the update described in lines 17-19 produces the desired overlapped decomposition.

Algorithm 2: R_Inter($X_1, X_2, \underline{f}, f$)

```
1 if Interact(X1, X2, f, f) then
2     if |X2| = 1 then
3         X1 = X1 ∪ X2 ;
4     else
5         Split X2 into equally-sized
            groups G1, G2 ;
6         X1^1 =R_Inter(X1, G1, f, f);
7         X1^2 =R_Inter(X1, G2, f, f);
8         X1 = X1^1 ∪ X1^2 ;
9 return X1 ;
```

Algorithm 3: L_Inter($X_1, X_2, \underline{f}, f$)

```
1 if Interact(X1, X2, f, f) then
2     if |X1| = 1 then
3         return X1 ;
4     else
5         Split X1 into equally-sized
            groups G1, G2 ;
6         X1^1 =L_Inter(G1, X2, f, f);
7         X1^2 =L_Inter(G2, X2, f, f);
8         X1 = X1^1 ∪ X1^2 ;
9 return X1 ;
```

Note that the main difference between the `R_inter` and the `L_inter` functions lies in the fact that the former focuses on the set X_2 while the latter works on X_1. Furthermore, the two functions also differ in line 3 (see Algorithms 2 and 3) since the `R_inter` function adds variables from X_2 that interact with X_1 to X_1 while the `L_inter` function only returns variables from X_1 that interact with X_2. For both algorithms, the `Interact` function at line 1 relies on Theorem 1.

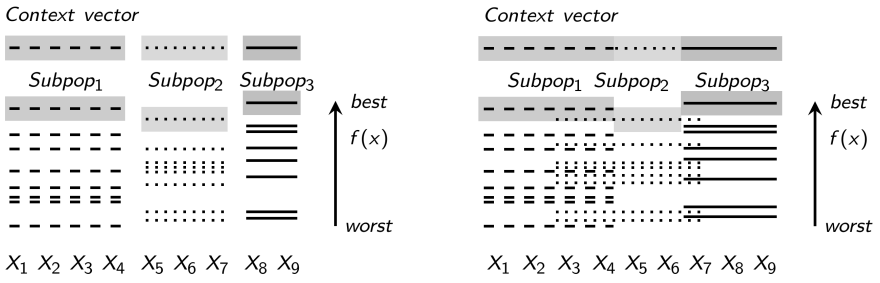
3.2 Overlapped CC framework

The main layout of the overlapped CC framework is similar to the standard CC presented in Section 2.1 with the major exception that it is designed to detect and manage overlapping variables efficiently. For this purpose, the decomposition step performs the ORDG algorithm presented in the previous section.

The optimization still consists in iteratively evolving each subcomponent in a round-robin strategy. However, in this step, the cooperation between subproblems through the sharing of best solutions in the context vector needs to be revised. Since subcomponents overlap, a variable x_i belonging to one component S_k may also appear in another component S_l. This introduces the issue of

which value of x_i has to be shared in the context vector. In a standard framework (i.e. without any overlapping), this value is the one of the variable x_i of the best individual in the (only) subpopulation containing x_i (see Figure 2a for an illustrative example). For the overlapped framework, this idea is extended: the value of x_i in the context vector is the one of the best individual among all the individuals in the two subpopulations focusing on x_i (or in the only subpopulation if x_i is not overlapped). Such an arrangement is illustrated in Figure 2b.

Note that, in order to choose the best individual within two different subpopulations, the function value of each individual used for comparison is the one that has been computed during the optimization of the corresponding subcomponents in the round-robin fashion loop. In this process, individuals in each subcomponent are completed with the variables of the context vector in order to be evaluated. The latter is updated each time a better solution is reached.



(a) Standard CC (b) Overlapped CC

Fig. 2: Management of the context vector within a standard and an overlapped CC framework. The illustrative example relies on the interaction structure presented in Figure 1. Dashed, dotted and solid lines represent individuals from subpopulations 1, 2 and 3 respectively. The context vector is built with the variables values of the best individual in each subpopulation.

4 Experimental settings and results

The performance of the new overlapped framework is assessed on large-scale overlapping benchmark problems derived from the CEC'2013 suite [17] and compared with the standard CC framework based on RDG3 decomposition [15]. The benchmark set contains 6 functions. Two of them, F_5 and F_6, are directly taken from [17]: F_6 is the 1000-d shifted Rosenbrock function and F_5 is the 905-d shifted Schwefel's function with conflicting[3] overlapping subcomponents. The four other

[3] Note that the function f_{13} in [17] also contains overlapping subcomponents but it has not been included in the benchmark set because their overlapping subcomponents are conforming. It means that they have the same optimum value with respect to both subcomponent functions. It can be simply optimized in a standard CC framework.

functions, F_1 to F_4, are obtained by replacing the Schwefel basis function in F_5 by Ackley, Elliptic, Rastrigin and Rosenbrock functions respectively. Therefore, functions F_1 to F_5 contain 20 overlapping subcomponents that share 5 variables with adjacent subcomponents. The F_6 function (Rosenbrock) can be seen as containing 999 subcomponents sharing one variable with adjacent ones.

In order to evaluate the decomposition effects of the newly proposed framework on overlapping problems, the RDG3 and ORDG strategies are used to decompose the benchmark problems presented above. For the RDG3, two different threshold values $\epsilon_n = 50$ and $\epsilon_n = 0$ are tested. The first value is the one used to study optimization results in [15] while the second value aims to identify as many components as possible and systematically cut the overlapping at shared variables. The number of components generated (k), the sum of the number of variables in each group (r) and the number of function evaluations computed (FEs) are reported in Table 1.

Table 1: Decomposition results of RDG3 (with $\epsilon_n = 50$ and $\epsilon_n = 0$) and ORDG strategies. k is the number of components generated, r is the sum of the number of variables in each group and FEs is the number of function evaluations.

Fun	RDG3 ($\epsilon_n = 50$)			RDG3 ($\epsilon_n = 0$)			ORDG		
	k	r	FEs	k	r	FEs	k	r	FEs
F_1	12	905	16273	20	905	16597	12	1011	16702
F_2	12	905	16252	19	905	16666	17	1000	18214
F_3	12	905	16249	20	905	16615	17	1000	18214
F_4	12	905	16252	20	905	16666	17	1000	18214
F_5	13	905	16288	21	905	16669	17	1003	18202
F_6	20	1000	49891	500	1000	25435	999	1998	59848

For the RDG3 decompositions, r is simply equal to the number of variables of the function because there is no overlap. For F_1 to F_5, the ORDG should capture the overlapping subcomponents introduced in [17] and therefore retrieve the 1000 variables involved in the benchmark construction. This is the case for F_2 to F_4. For F_1 and F_5, some additional interactions between independent variables have been identified due to computational round-off errors and lead to a slightly larger value of r. Still according to [17], the number of components k for functions F_1 to F_5 is equal to 20 in the benchmark construction. The RDG3 with $\epsilon_n = 50$ produces only 12 (or 13) components since components that contain less than 50 variables are merged with other ones. The RDG3 with $\epsilon_n = 0$ retrieves the 20 subcomponents (except for F_2 and F_5 for which the small difference is again due to computational round-off errors). The ORDG detects 17 components for functions F_2 to F_5[4]. They correspond to the ones formed in the benchmark construction except that some of them have been merged. Indeed, if

[4] Theoretically, 17 components should also be detected for F_1 but round-off erros affect the results for that particular function.

the ORDG procedure starts the detection with a variable belonging to a component that share some overlapping variables with two adjacent components, the latter are merged to form only one component. Thereafter, adjacent components to these components are also merged and so on. Although this prevents the detection of the 20 subcomponents, the obtained decomposition still agrees with the desired objective. In this particular case, the fact that some overlapping components contain two subsets of variables that do not directly interact will not affect the optimization efficiency. For the F_6 function, the obtained decomposition corresponds to the expected one, 20 (500) components of 50 (2) variables are formed for the RDG3 with $\epsilon_n = 50$ ($= 0$ resp.) and the ORDG produces 999 components of 2 variables. Finally, since the ORDG analyses additional interactions with respect to the RDG3, the cost in terms of FEs is higher. However, the additional cost remains reasonable and will be negligible with respect to the budget in terms of FEs allowed for the optimization.

The influence of the decomposition on the optimization results is analyzed by embedding each kind of decomposition in the overlapped CC framework presented above. In particular, when the latter is coupled with the RDG3 decompositions, it behaves like the standard CC. The evolutionary algorithm used to optimize the subcomponents is a genetic algorithm. In this study, the one implemented in the Minamo software is considered [10]. Here there is an overview of its main features: real-value representation of the individuals; tournament selection to pick up pairs of parents; arithmetic crossovers for recombination; mutation rate of 1 %; elitism of two individuals. Within the CC framework, the population size is set to 10 times the number of variables of the considered component. The round-robin fashion optimization loop is repeated until the maximum number of FEs is reached. It is set to 3×10^6 in total (for the decomposition and the optimization).

The median of the best solution over 51 independent runs and the standard deviation are reported in Table 2. The CC-ORDG produces better solution quality than the CC-RDG3 for 4 of the 6 functions. The CC-RDG3 with $\epsilon_n = 0$ generates the best results for the 2 other functions. Convergence graphs depicting the convergence behavior along the optimization process are also provided in Figure 3. It can be seen that the three algorithms follow the same trend for functions F_1 to F_5[5]. Between the two variants of the CC-RDG3, the slightly different number of components does not have too much influence on the optimization quality. However, for F_6, the CC-RDG3 with $\epsilon_n = 0$ that produces many more subcomponents (each of them focusing on 2 variables) has a better handle of the optimization. Furthermore, the closed results between the CC-RDG3 with $\epsilon_n = 0$ and the CC-ORDG may be surprising. By analyzing the convergence behavior of the overlapping variables in the CC-ORDG in details, it can be seen that most of the time, the variables shared among two subcomponents converge to the same value at the same rate in the two subcomponents. In this context, the overlapped decomposition does not significantly contribute to a

[5] Note that for F_2, the CC-ORDG is stuck in a pseudo-optima for a few runs. It causes the large green-colored area in Figure 3b.

Table 2: Optimization results of the CC-RDG3 (with $\epsilon_n = 50$ and $\epsilon_n = 0$) and the CC-ORDG. The median of the best solution over 51 independent runs and the standard deviation are presented. Best median values are in bold.

Fun	RDG3 ($\epsilon_n = 50$)		RDG3 ($\epsilon_n = 0$)		ORDG	
	median	std	median	std	median	std
F_1	7.03e+07	1.77e+05	7.04e+07	1.48e+05	**7.01e+07**	2.84e+05
F_2	3.95e+13	3.56e+12	**3.53e+13**	5.24e+12	3.85e+13	1.67e+14
F_3	4.83e+08	2.81e+07	5.22e+08	4.45e+07	**4.17e+08**	8.11e+07
F_4	6.01e+11	1.82e+10	**4.27e+11**	1.40e+10	7.80e+11	3.58e+10
F_5	1.04e+11	2.15e+10	1.15e+11	2.66e+10	**9.64e+10**	1.75e+10
F_6	8.68e+05	9.80e+04	1.50e+03	1.21e+02	**1.34e+03**	1.01e+02

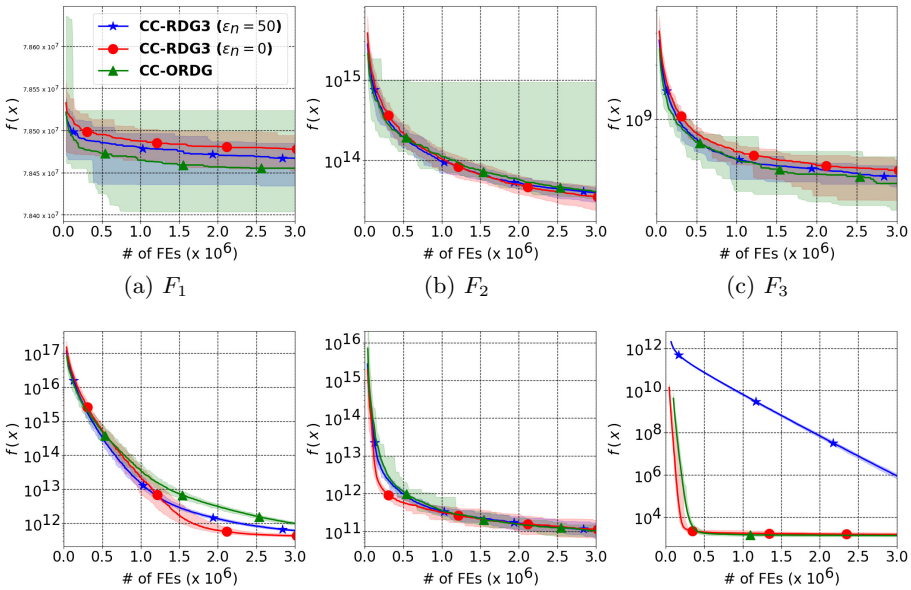
better cooperation between subcomponents in comparison with the cooperation through the sharing of the context vector performed in a standard CC framework. Therefore, although results in Table 2 might indicate that the CC-ORDG provides slightly better results, we can not definitely claim that a strategy is better than the other.

5 Discussion

The new CC framework introduced in this paper is designed to optimize overlapping LSGO problems with an overlapped decomposition strategy. In this context, an overlapped variant of the RDG has been developed to efficiently detect overlapping variables and share them among several subcomponents. The optimization step of the standard CC framework has also been extended in order to efficiently sha-re information between overlapped subcomponents through the context vector.

Numerical experiments were conducted on 6 benchmark functions. The extension of the method to a larger set of test functions is straightforward. However we believe the latter goes beyond the scope of this introductory paper and thus it will be considered in a further work. Similarly, the benchmark set is limited to 905-d and 1000-d functions, which is common practice in LSGO studies. Further research on the scalability may also be carried out to determine how the algorithm performs on more complex problems with larger dimensions.

The experiments presented in this paper show that the new approach produces the desired overlapped decomposition. However, although the optimization results might indicate that the new decomposition helps to get slightly better solutions, we can not definitely claim that the new framework outperforms the standard ones. This may be partly explained by the fact that the exchange of information between subcomponents in a standard CC framework through the context vector is stronger than we could expect. In any way, there is scope for even better progress to further develop the CC concept to deal with overlapping problems. We think that the new strategy that introduces overlapped subcomponents may be a promising way to achieve such an improvement.



(a) F_1 (b) F_2 (c) F_3

(d) F_4 (e) F_5 (f) F_6

Fig. 3: Convergence graphs representing the evolution of $f(x)$ (in log-scale) with respect to the number of FEs. CC-RDG3 with $\epsilon_n = 50$ (blue stars), CC-RDG3 with $\epsilon_n = 0$ (red circles), CC-ORDG (green triangles). The solid line depicts the median value while the light-colored area represents the interval between the best and the worst value over the 51 runs.

References

1. Bellman, R.: Adaptive Control Processes : A Guided Tour. Princeton University Press, Princeton, New Jersey, USA (1961)
2. van den Bergh, F., Engelbrecht, A.P.: A cooperative approach to particle swarm optimization. IEEE Transactions on Evolutionary Computation **8**, 225–239 (2004)
3. Eiben, A., Smith, J.: Introduction to Evolutionary Computing. Natural computing series, Springer, Heidelberg, Germany (2015)
4. Garcia, D.J., You, F.: Supply chain design and optimization: Challenges and opportunities. Computers & Chemical Engineering **81**, 153–170 (2015)
5. Li, L., Fang, W., Wang, Q., Sun, J.: Differential grouping with spectral clustering for large scale global optimization. In: 2019 IEEE Congress on Evolutionary Computation. pp. 334–341. IEEE, Piscataway, New Jersey, USA (2019)
6. Liu, Y., Yao, X., Zhao, Q., Higuchi, T.: Scaling up fast evolutionary programming with cooperative coevolution. In: Proceedings of the 2001 Congress on Evolutionary Computation. vol. 2, pp. 1101–1108. IEEE, Piscataway, New Jersey, USA (2001)
7. Mahdavi, S., Shiri, M.E., Rahnamayan, S.: Metaheuristics in large-scale global continues optimization: A survey. Information Sciences **295**, 407–428 (2015)

8. Omidvar, M.N., Li, X., Mei, Y., Yao, X.: Cooperative co-evolution with differential grouping for large scale optimization. IEEE Transactions on evolutionary computation **10**(10), 1–17 (2013)
9. Potter, M.A., De Jong, K.A.: A cooperative coevolutionary approach to function optimization. In: Davidor Y., Schwefel HP., Mnner R. (eds) Parallel Problem Solving from Nature PPSN III. PPSN 1994. Lecture Notes in Computer Science. vol. 866, pp. 249–257. Springer, Heidelberg, Germany (1994)
10. Sainvitu, C., Iliopoulou, V., Lepot, I.: Global optimization with expensive functions - sample turbomachinery design application. In: Springer (ed.) Recent Advances in Optimization and its Applications in Engineering, pp. 499–509 (2010)
11. Shi, Y., Teng, H., Li, Z.: Cooperative co-evolutionary differential evolution for function optimization. In: Wang L., Chen K., Ong Y.S. (eds) Advances in Natural Computation. ICNC 2005. Lecture Notes in Computer Science, vol. 3611, pp. 1080–1088. Springer, Heidelberg, Germany (2005)
12. Song, A., Chen, W.N., Luo, P.T., Gong, Y.J., Zhang, J.: Overlapped cooperative co-evolution for large scale optimization. In: 2017 IEEE International Conference on Systems, Man, and Cybernetics. pp. 3689–3694. IEEE, Piscataway, New Jersey, USA (2017)
13. Strasser, S., Sheppard, J., Fortier, N., Goodman, R.: Factored evolutionary algorithms. IEEE Transactions on Evolutionary Computation **2**(21), 281–293 (2017)
14. Sun, L., S, Y., Cheng, X., Liang, Y.: A cooperative particle swarm optimizer with statistical variable interdependence learning. Information Sciences **1**(186), 20–39 (2012)
15. Sun, Y., Li, X., Ernst, A., Omidvar, M.N.: Decomposition for large-scale optimization problems with overlapping components. In: 2019 IEEE Congress on Evolutionary Computation. pp. 326–333. IEEE, Piscataway, New Jersey, USA (2019)
16. Sun, Y., Omidvar, M.N., Kirley, M., Li, X.: Adaptive threshold parameter estimation with recursive differential grouping for problem decomposition. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 889–896. GECCO '18, ACM, New York, NY, USA (2018)
17. Xi, L., Tang, K., Omidvar, M.N., Yang, Z., Qin, K.: Benchmark functions for the CEC 2013 special session and competition on large-scale global optimization. Tech. rep., RMIT University, Melbourne (2013)
18. Yang, Z., Tang, K., Yao, X.: Large scale evolutionary optimization using cooperative coevolution. Information Sciences **178**(15), 2985–2999 (2008)

Synthesis of scheduling heuristics by composition and recombination

Dominik Mäckel[1Γ], Jan Winkels[2Γ], and Christin Schumacher[3Γ]

[1] TU Dortmund University,
Department of Computer Science 14 – Software Engineering, Dortmund, Germany,
dominik.maeckel@tu-dortmund.de

[2] TU Dortmund University,
Department of Computer Science 14 – Software Engineering, Dortmund, Germany,
jan.winkels@tu-dortmund.de

[3] TU Dortmund University,
Department of Computer Science 4 – Modeling and Simulation, Dortmund, Germany,
christin.schumacher@tu-dortmund.de

Abstract. In many machine scheduling studies, individual algorithms for each problem have been developed to cope with the specifics of the problem. On the other hand, the same underlying fundamentals (e.g. Shortest Processing Time, Local Search) are often used in the algorithms and only slightly modified for the different problems. This paper deals with the synthesis of machine scheduling algorithms from components of a repository. Especially flow shop and job shop problems with makespan objective are considered to solve with Shortes/Longest Processing Time, NEH, Giffler & Thompson algorithms. For these components, the paper includes an exemplary implementation of an agile scheduling system that uses the Combinatory Logic Synthesizer to recombine components of scheduling algorithms to solve a given scheduling problem. Special attention is given to the composition heuristics and the process of recombination to executable programs. The advantages of this componentization are discussed and illustrated with examples. It will be shown that algorithms can be generalized to deal with scheduling problems of different machine environments and production constraints.

1 Introduction

In production, machine scheduling algorithms help to decide automatically when a certain job should be executed on which machine. Many manufacturers have not yet automated their machine scheduling. One reason is that for each machine scheduling problem with its numerous specific characteristics, suitable algorithms have to be selected, adapted, and implemented individually. Each practical scheduling problem can be categorized into a problem class, for which dedicated heuristics are applicable. If a class is a subset of another class, the heuristics of the superset class can often also be applied to the subset class.

Also, relationships and overlapping between categories can be identified which simplifies the transfer of heuristics between problem classes.

The assignment problem which a combination of heuristics or metaheuristics should be chosen for which practical production environment concerning the applicability, solution quality, and computing time represents a combinatorial challenge. The synthesis framework Combinatory Logic Synthesizer ((CL)S) [1] is suitable for the automated solution of this task. The (CL)S can construct software from a collection of individual components and it is possible to specify components semantically, which enables the (CL)S to select the appropriate components. The framework then automatically generates all possible combinations in the form of executable software.

The objective of the paper is to use the (CL)S-Framework to automatically select and combine different algorithms to solve a given scheduling problem. Therefore, we build a (CL)S repository of algorithms for different machine environments, which takes the relationships of the classes into account and automatically composes selected algorithms for instances of these problems.

This paper is structured as follows: First, we present the general classification scheme of machine scheduling problems. In the related work, we discuss algorithms for scheduling of flow shop and job shop problems and present the framework on which our implementation is based on, the (CL)S. The handling of this framework, as well as the generation and composition of algorithms, is shown in the fourth chapter with example runs. In detail, we show the potential of the tool and the resulting possibilities using the Giffler & Thompson's algorithm.

2 Classification of Machine Scheduling Problems

Machine scheduling problems can be specified by a tuple $\alpha|\beta|\gamma$ [2, pp. 288–290][3, pp. 13–21][4, pp. 1–2]. In the following, parameter values are specified which are considered in this paper.
The parameter α defines the amount and arrangement of machines [3, pp. 14–15]:

- 1: Single Maschine, one machine is available for production.
- Fm: flow shop, m machines with one machine per processing stage. All jobs follow the same route through the machines.
- Jm: job shop, m machines with one machine per stage. Each job has a prescribed route through the stages. The route may differ between the jobs.
- Om: Open Shop, m machines, where each job can visit the machines one after the other in an order that is determined by the planner.

Parameter β can contain as many entries as required and describes characteristics and limitations of the production process:

- $prmu$: Permutation, the processing sequence of jobs from the first processing stage through all machines is to be kept consistent [3, p. 17].

– *skip*: skipping stages of jobs is possible (further example, but not applicated in the paper) [5, pp. 1151–1155, 4, p. 13].

γ specifies the objective function:

– C_{max}: Makespan, interval between production start of the first scheduled job and finish time of the last job.

3 Related Work

In the following, important scheduling algorithms for these machine environments and β-constraints in combination with makespan minimization are described, as well as related work according to the (CL)S.

3.1 Maschine Scheduling Algorithms for Flow Shops and Job Shops

In the context of machine scheduling, an enormous number of papers and algorithms are available. Literature overviews for flow shops and job shops can be found in Komaki, Sheikh, and Malakooti [6], Framinan, Gupta, and Leisten [7] (permutation flow shop with makespan minimization) and Zhang et al. [8]. A comparison between commonly used algorithms for constructive flow shop scheduling can be found in Ruiz and Maroto [9]. Different dispatching rules have been studied in Arisha, Young, and El Baradie [10]. In the following, selectively a few algorithms of the overviews are analyzed that dealt with flow shops or job shops to minimize the makespan and are related to our problem classes (see Section 2).

Some of the most commonly used constructive heuristics for flow shops and job shops are Shortest Processing Time First (SPT), Longest Processing Time First (LPT), and the NEH-heuristic (flow shops) and have therefore been considered in this paper. The benefits of dispatching rules like SPT and LPT are low computational complexities and therefore fast calculations, and transparent behavior for production planners. The NEH-Heuristic, firstly published by Nawaz, Enscore, and Ham [11, pp. 92–94] for permutation flow shops and makespan minimization ($Fm|prmu|C_{max}$) produces good results in most cases. Giffler and Thompson [12] published a constructive algorithm that also applies rules like SPT and LPT to job shops.

3.2 Giffler & Thompson algorithm

Using the algorithm by Giffler & Thompson, job shop as well as flow shop problems can be solved. It schedules exactly one job on a machine in each iteration, so the algorithm returns complete schedules after $m * n$ iterations, where m is the number of machines and n the amount of jobs. The heuristic is only parameterized by the applied dispatching rule. In the algorithm, this dispatching rule decides between several competing jobs on the same machine. The implementation of the complete Giffler & Thompson algorithm is shown in Alg. 1.1.

The algorithm consists of four phases where steps 2 to 4 iterate until all jobs are scheduled [13, S. 75-76]. The calculated schedule and the completion times of the scheduled jobs and for all machines are returned.

1 Let Z_i be the completion time of machine i. Initialize $Z_i = 0$ for $i = 0, ..., m$. Select a dispatching rule.
2 Select machine i^* that first can finish a job out of the set of jobs, which are waiting to be processed next on one of the machines and are not scheduled yet.
3 From the set of all jobs waiting to be processed on this machine i^* select one job by the dispatching rule which is initalized in step 1.
4 Schedule selected job on machine i^* and update Z_{i^*}. If there are jobs left to be scheduled, return to step 2.

Algorithm 1.1: Implementation of the Giffler & Thompson algorithm

Alg. 1.1 works as follows. In each iteration (step 2-4), the machine is determined, which can first complete a job. For this purpose, each not yet fully scheduled job is iterated and the end time after scheduling on the next machine to be visited is compared. Up to this point, it is a greedy algorithm that selects a machine according to the earliest completion time on the next machine the job has to be processed on. Once the machine to be scheduled has been determined, in the second phase the job is varied to meet a prioritization on the machine. This is done by determining all jobs that are also to be scheduled next on the selected machine, including the job determined in the previous phase. If two or more jobs are waiting to be scheduled on the selected machine, the jobs get ranked according to the selected dispatching rule. After selecting a job on the determined machine, it gets scheduled and Z_i, as well as the current end time of the job, gets updated.

3.3 Combinatory Logic Synthesizer

Combinatory Logic Synthesizer, short: (CL)S, is a type-based framework for the synthesis of software from a set of components specified in a repository [1]. The framework was developed in the programming language Scala and is used in this paper. In addition to the synthesis, the framework also allows the immediate execution of the synthesis result. Due to the implementation in the Scala programming language, the synthesis results can also access existing Java and Scala libraries. The framework (CL)S was developed at the chair 14 of the faculty for Computer Science at the TU Dortmund University.

The Combinatory Logic Synthesizer ((CL)S) is particularly suitable for handling unpredictable variability, which makes it well suited for the synthesis of machine allocation algorithms in production planning. (CL)S enables the specification of components, their implementation, as well as the modeling of variability and the automatic composition of components under consideration of the

modeled variability rules [14]. All this is uniformly done within the framework. Thus, the framework provides a solid basis for mapping and specifying individual heuristics and algorithms, and is also suitable as a technological basis for the automatic composition of components [15]. The (CL)S has been used in the past for numerous applications of a similar nature. As an example, we mention the automatic configuration of factory planning projects [16], the automatic generation of BPMN processes [17], and the automated configuration of plans in construction projects [18]. The basis for the use of the framework is that within the target domain, results can be composed of specifiable components. In the (CL)S the specification is done by so-called semantic intersection types. How components can be specified and implemented, and which solutions are then generated automatically, is shown in the following chapters using an example.

4 Implementation

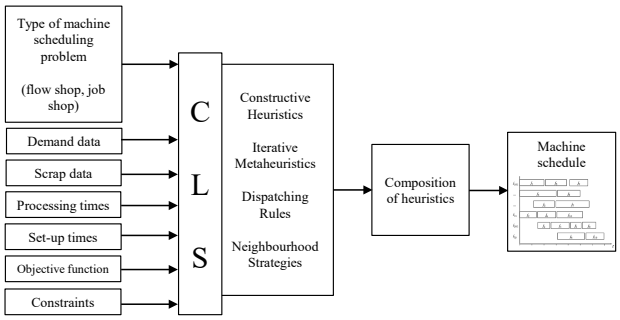


Fig. 1. Concept of schedule generation with (CL)S

The (CL)S-Repository contains all algorithm components as shown in Fig. 1, which can be combined into an executable scheduling system. Through a synthesis request to the (CL)S framework, production characteristics can be used to intersect with the defined types of the algorithm components. The (CL)S only selects those heuristics that are applicable to the given problem class. Available problem classes in this exemplary implementation are flow shop and job shop. After composing the algorithms, they can be utilized to solve the given scheduling problem and produce valid machine schedules. The synthesized algorithms work as transition functions and transfer the given data object into an applicable machine schedule. After scheduling, the makespan is calculated.

Further problem classes can be integrated by adding further possible parameter assignments and therefore extending the intersection types. By specifying additional parameters, further β constraints can be realized, which may exclude further heuristics because they are not applicable for the problem, or

$\Gamma = \{$

Scheduler: (String → String) ∩ (Algorithm ∩ shopClass → Scheduler(shopClass))
NEH: String ∩ (Algorithm ∩ FS)
FSDispatch: (String → String) ∩ (PriorityRule → Algorithm ∩ FS)
GifflerThompson: (String → String) ∩ (PriorityRule → Algorithm ∩ JS ∩ FS)
LPT: String ∩ PriorityRule
SPT: String ∩ PriorityRule

$\}$

Fig. 2. (CL)S repository

include others because they require certain assumptions or additional data such as deadlines.

Our defined (CL)S repository is shown in Fig. 2 and the solution tree calculated by the (CL)S across all combinators of the repository is illustrated in Fig. 3. The repository's first combinator *Scheduler* of Fig. 2 is a wrapping base module, which serves as the common target type for all synthesis requests. Accordingly, it is found on the first level of the solution tree (left square in Fig. 3). As parameter *shopClass* (see Fig. 2) it receives information about the problems' machine environment (α-component). Starting from the base module, the different algorithms for flow shop and job shop problems of the type *Algorithm* are now available according to the parameter *shopClass*. By concretizing the parameter when calling the synthesis, the number of applicable combinators is reduced in such a way that only the algorithms for the corresponding problem class can be used. This is done by using the parameter also as an intersection type of the base module and thus an intersection with combinators of other problem classes is no longer possible.

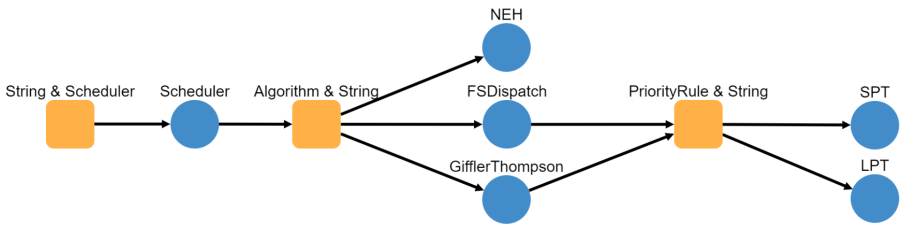


Fig. 3. (CL)S solution tree for flow shops

The first two algorithms *NEH* and *FSDispatch* in our implementation can only be applied to flow shops while the algorithm of Giffler & Thompson can be applied to job shops, which implies that it can also be used for flow shops because flow shops are a real subset of job shops as shown in Fig. 4.

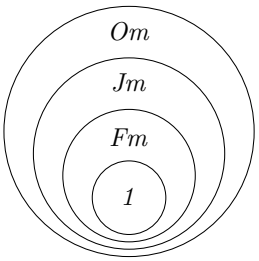


Fig. 4. Relationships between considered scheduling problem classes

The algorithms *FSDispatch* and *GifflerThompson* additionally require a dispatching rule. Fig. 3 shows the reuse of these dispatching rules SPT and LPT for *FSDispatch* and *GifflerThompson*. This shows again one advantage of such a composing method. It is easily possible to integrate and combine new algorithms, heuristics, and dispatching rules into the tool by inserting them into the repository as combinators with corresponding intersection types. New components can reuse already existing ones. Individual components can also be replaced by other possibly better performing components without having to replace them individually at all points. Furthermore, the derivation graph in Fig. 3 shows similarities and differences between algorithms in the sense that the use of similar components is immediately recognizable. The procedure of disassembling an algorithm into reusable components and representing them as (CL)S-combinators is now explained in detail using the example of the Giffler & Thompson algorithm.

5 Results

To show that the same implementation of an algorithm can be effectively used for different machine environments, the Giffler & Thompson algorithm and its implementation is shown in Alg. 1.1 has been applied to a flow shop and a job shop problem. The selection of the dispatching rule takes place inside the dispatching rule combinator that has been selected by CLS and parsed into the program code at this point. The dispatching rule is varied by replacing the code at this point.

To give a concrete example, processing times in Tab. 1 have been randomly generated from a triangular distribution with lower limit 5s, upper limit 15s, and mode 8s. For the job shop problem, also the processing order has been randomized across the stages as shown in Tab. 2. The entry "4" in row "S1" and column "job 1" indicates that job 1 has to be processed on the first stage (S1) in the fourth production step. Before, the job has to visit stage 3, then stage 2 and stage 2 in exactly this sequence. The calculated job shop schedule of the Giffler & Thompson algorithm with LPT-rule is shown in fig. 5.

Since the algorithm was not particularly designed for flow shop problems, it is reasonable to compare its result with the NEH heuristic. The two schedules

Job:	1	2	3	4	5	6	7	8	9	10
S1	6	12	8	9	10	8	9	9	11	7
S2	7	11	7	7	9	7	10	9	10	6
S3	8	12	11	8	9	8	12	13	7	9
S4	9	12	11	7	6	9	10	8	11	10

Table 1. Generated processing times

Job:	1	2	3	4	5	6	7	8	9	10
S1	4	1	4	3	4	4	1	3	2	3
S2	3	3	1	4	1	2	2	2	3	1
S3	1	2	2	1	3	3	4	4	4	4
S4	2	4	3	2	2	1	3	1	1	2

Table 2. Order for job shop production

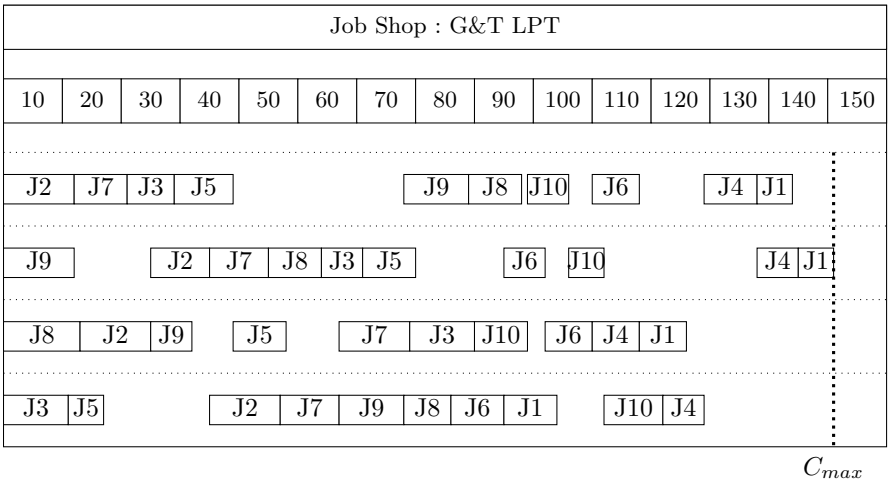


Fig. 5. Jop shop schedule with Giffler & Thompson

are shown in Fig. 6. As expected, the NEH heuristic creates a better schedule than the Giffler & Thompson algorithm. It is worth mentioning that Giffler & Thompson created a valid schedule that can keep up with algorithms specially designed for flow shop algorithms and can therefore be for example used as a starter solution for an iterative algorithm or it can be used if no better solution is available. In addition, Giffler & Thompson algorithm can be executed with different priority rules. To execution of the algorithm with different priority rules as input parameters lead to multiple solutions, the planner team can choose from. The benefit is not having to implement an algorithm for flow shop problems as the job shop algorithm can already handle it.

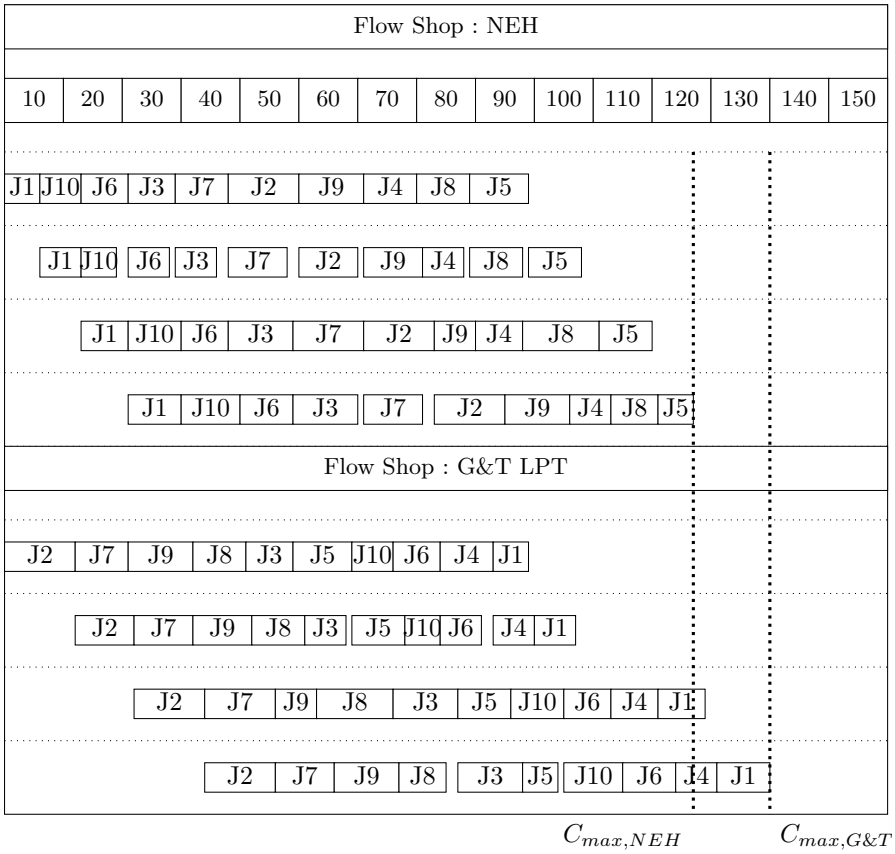


Fig. 6. Comparison of flow shop schedules with NEH and Giffler & Thompson

6 Conclusion

In this paper, we presented a repository for machine scheduling algorithms using the (CL)S, a framework that can generate algorithms automatically and to create solutions that are specially tailored to a previously specified problem. We used this framework for the problem area of machine scheduling in order to solve flow shop and job shop problems with SPT, LPT, NEH and Giffler & Thompson.

We have classified scheduling algorithms and mapped them as components in a (CL)S repository. Through componentization, different algorithms can be integrated into a framework via a uniform interface. This makes it easy to generate different algorithmen to scheduling problems. The recombined algorithms generate valid schedules according to their functionalities. Algorithms can be defined for various problem classes and constraints. According to the synthesis request, only those algorithms are recombined that apply to the current problem.

The shown concept is not limited to constructive algorithm as presented in this study and can also be applied to any iterative metaheuristic in further studies if the given data object already contains a constructive start solution. Concatenations of different constructive and iterative heuristics are conceivable as well. Also, extensions of other objective functions are possible.

References

[1] Jan Bessai et al. "Combinatory Logic Synthesizer". In: *Leveraging applications of formal methods, verification and validation*. Ed. by Tiziana Margaria-Steffen and Bernhard Steffen. Vol. 8802. LNCS sublibrary. SL 1, Theoretical computer science and general issues. Heidelberg: Springer, 2014, pp. 26–40. ISBN: 978-3-662-45233-2. DOI: 10.1007/978-3-662-45234-9.

[2] R. L. Graham et al. "Optimization and approximation in deterministic sequencing and scheduling: a survey". In: *Annals of discrete mathematics: Proceedings of the Advanced Research Institute on Discrete Optimization and Systems Applications of the Systems Science Panel of NATO and of the Discrete Optimization Symposium co-sponsored by IBM Canada and SIAM Banff, Aha. and Vancouver*. Ed. by P. L. Hammer, E. L. Johnson, and Korte B H. Vol. 5. Elsevier, 1979, pp. 287–326. DOI: 10.1016/S0167-5060(08)70356-X.

[3] Michael Pinedo. *Scheduling: Theory, Algorithms, and Systems*. 5th ed. Cham et al.: Springer, 2016. ISBN: 9783319265780. DOI: 10.1007/978-3-319-26580-3. URL: http://dx.doi.org/10.1007/978-3-319-26580-3.

[4] Rubén Ruiz and José Antonio Vázquez-Rodríguez. "The hybrid flow shop scheduling problem". In: *European Journal of Operational Research* 205.1 (2010), pp. 1–18. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2009.09.024.

[5] Rubén Ruiz, Funda Sivrikaya Şerifoğlu, and Thijs Urlings. "Modeling realistic hybrid flexible flowshop scheduling problems". In: *Computers & Operations Research* 35.4 (2008), pp. 1151–1175. ISSN: 03050548. DOI: 10.1016/j.cor.2006.07.014.

[6] G. M. Komaki, Shaya Sheikh, and Behnam Malakooti. "Flow shop scheduling problems with assembly operations: a review and new trends". In: *International Journal of Production Research* 57.10 (2019), pp. 2926–2955. ISSN: 0020-7543. DOI: 10.1080/00207543.2018.1550269.

[7] J M Framinan, J N D Gupta, and R Leisten. "A review and classification of heuristics for permutation flow-shop scheduling with makespan objective". In: *Journal of the Operational Research Society* 55.12 (2004), pp. 1243–1255. DOI: 10.1057/palgrave.jors.2601784. eprint: https://doi.org/10.1057/palgrave.jors.2601784. URL: https://doi.org/10.1057/palgrave.jors.2601784.

[8] Jian Zhang et al. "Review of job shop scheduling research and its new perspectives under Industry 4.0: Journal of Intelligent Manufacturing, 30(4), 1809-1830". In: *Journal of Intelligent Manufacturing* 30.4 (2019), pp. 1809–1830. ISSN: 0956-5515. DOI: 10.1007/S10845-017-1350-2.

[9] Rubén Ruiz and Concepción Maroto. "A comprehensive review and evaluation of permutation flowshop heuristics to minimize flowtime". In: *European Journal of Operational Research* 40 (Sept. 2005), pp. 479–494. DOI: 10.1016/j.ejor.2004.04.017.

[10] Amr Arisha, Paul Young, and Mohie El Baradie. "Flow Shop Scheduling Problem: a Computational Study". In: Sixth International Conference on Production Engineering and Design for Development (PEDD6). Dublin Institute of Technology. Cairo, Egypt, Jan. 1, 2002, pp. 543–557.

[11] Muhammad Nawaz, E. Emory Enscore, and Inyong Ham. "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem". In: *Omega* 11.1 (1983), pp. 91–95. ISSN: 03050483. DOI: 10.1016/0305-0483(83)90088-9. URL: http://www.sciencedirect.com/science/article/pii/0305048383900889.

[12] B. Giffler and G. L. Thompson. "Algorithms for Solving Production Scheduling Problems". In: *Operations Research* 8.4 (1960), pp. 487–503. DOI: 10.1287/opre.8.4.487. URL: http://search.ebscohost.com/login.aspx?direct=true&db=bsu&AN=7687426&site=ehost-live.

[13] Florian Jaehn and Erwin Pesch. *Ablaufplanung: Einführung in Scheduling.* 1st ed. Berlin u.a.: Springer, 2014. ISBN: 978-3-642-54439-2.

[14] Jan Bessai et al. "Combinatory Process Synthesis". In: 9952 (2016), pp. 266–281. DOI: 10.1007/978-3-319-47166-2.

[15] Jan Winkels. *Automatisierte Komposition und Konfiguration von Workflows zur Planung mittels kombinatorischer Logik.* Technische Universität Dortmund. DOI: 10.17877/DE290R-20469.

[16] Jan Winkels et al. "Automatic Composition of Rough Solution Possibilities in the Target Planning of Factory Planning Projects by Means of Combinatory Logic". In: *Leveraging Applications of Formal Methods, Verification and Validation. Industrial Practice.* Ed. by Tiziana Margaria and Bernhard Steffen. Vol. 11247. Theoretical Computer Science and General Issues. Cham: Springer International Publishing, 2018, pp. 487–503. ISBN: 9783030034283. DOI: 10.1007/978-3-030-03427-6.

[17] Jan Bessai et al. “Combinatory Process Synthesis”. In: 9952 (2016), pp. 266–281. DOI: 10.1007/978-3-319-47166-2.

[18] Lisa Theresa Lenz et al. “Smart Factory Adaptation Planning by means of BIM in Combination of Constraint Solving Techniques”. In: *Proceedings of the International Council for Research and Innovation in Building and Construction (CIB), World Building Congress 2019 – Constructing Smart Cities* (2019).

A Learning-based Iterated Local Search Algorithm for Solving the Traveling Salesman Problem

Maryam Karimi-Mamaghan[1][0000−0003−2889−5769], Bastien Pasdeloup[1][0000−0002−8417−7537], Mehrdad Mohammadi[1][0000−0002−3726−8356], and Patrick Meyer[1][0000−0001−6479−213X]

IMT Atlantique, Lab-STICC, UMR CNRS 6285, F-29238 Brest, France
{maryam.karimi, bastien.pasdeloup, mehrdad.mohammadi, patrick.meyer}@imt-atlantique.fr

Abstract. In this paper, we study the use of reinforcement learning in adaptive operator selection within the Iterated Local Search metaheuristic for solving the well-known NP-Hard Traveling Salesman Problem. This metaheuristic basically employs single local search and perturbation operators for finding the (near-) optimal solution. In this paper, by incorporating multiple local search and perturbation operators, we explore the use of reinforcement learning, and more specifically Q-learning as a machine learning technique, to intelligently select the most appropriate search operator(s) at each stage of the search process. The Q-learning is separately used for both local search operator selection and perturbation operator selection. The performance of the proposed algorithms is tested through a comparative analysis against a set of benchmark algorithms. Finally, we show that intelligently selecting the search operators not only provides better solutions with lower optimality gaps but also accelerates the convergence of the algorithms toward promising solutions.

Keywords: Adaptive operator selection · Iterated local search · Reinforcement learning · Q-learning · Traveling salesman problem.

1 Introduction

Combinatorial Optimization Problems (COPs) are a complex class of optimization problems with discrete decision variables and a finite search space. Many COPs are NP-hard for which no polynomial-time algorithm exists. Meta-heuristics (MHs) can solve these problems in reasonable time and provide them with acceptable solutions; however, they do not guarantee the optimality [22]. MHs employ either single or multiple search operators to evolve a single or a population of solutions toward (near-) optimal solutions. When using multiple search operators, the problem of *operator selection* arises.

Individual search operators may be effective in particular stages of the search process and not throughout the search process. The reason is that the search space of COPs is a non-stationary environment that includes different search regions with dissimilar characteristics. Therefore, different search operators act differently in different regions of the search space [7]. Accordingly, solving COPs with single search operators does not necessarily lead to the highest performance of the search process. Intuitively, employing multiple search operators selected in an appropriate way during the search process not only leads to a more robust behavior of a MH with respect to the process of finding the optimal solution [19], but also significantly affects the exploration (i.e., explore undiscovered regions) and exploitation (i.e., intensify the search in promising regions) abilities of a MH, and provides an Exploration-Exploitation balance during the search process. The main question in this regard is in which order the search operators should be employed such that the MH can go toward the global optimum. One efficient way is to dynamically select and apply the most appropriate operator based on their history of performance during the search process. This is referred to as *Adaptive Operator Selection* (AOS) [7]. Adaptive selection strategies may differ from very simple strategies to more advanced ones. In simple strategies, such as score-based selection strategy [16], an initial score is assigned to each search operator and the scores are updated based on the performance of each operator at each step of the search process. In this strategy, the selection chance of each search operators is then proportional to

its accumulated score. Regardless of the neglectable overhead that they impose to the search process, the added-value of simple strategies may not be necessarily significant [24]. Hence, more advanced adaptive strategies should be embedded into the AOS.

In this regard, Machine Learning (ML) techniques can be used in AOS to provide a more intelligent adaptive strategy when selecting the search operators during the search process. The integration of ML techniques into MHs is an emerging research field that has attracted numerous researchers in recent years [3,23,5,20,8,10,15]. In particular, ML techniques help the AOS to use feedback information on the performance of the search operators during the search process. In this situation, operators are selected based on a credit assigned to each operator (i.e., feedback from their historical performance). Considering the nature of the feedback, the learning can be *offline* or *online*. In *offline* learning, knowledge is extracted from a set of training instance with the aim to solve new problem instances. In *online* learning, the knowledge is extracted and incorporated into the resolution process dynamically while solving a problem instance [23,4].

In this paper, we study the use of reinforcement learning (RL), particularly Q-learning as a ML technique, in AOS within the Iterated Local Search (ILS) meta-heuristic [12] for solving the well-known NP-hard Traveling Salesman Problem (TSP). The ILS basically employs single local search and perturbation operators for finding the (near-) optimal solutions. However, there are several specific and efficient local search and perturbation operators for TSP in the literature (e.g., 2-opt, 3-opt, insertion, etc. as local search operators and double-bridge, shuffle-sequence, etc. as perturbation operators) [22] that can be employed simultaneously. In this paper, we incorporate multiple local search and perturbation operators into the ILS and use Q-learning to adaptively select among them during the search process. Indeed, Q-learning is integrated into ILS to adaptively select its operators during the search process. This integrated algorithm is called Q-ILS hereafter. In this paper, two variants of Q-ILS are proposed: in the first algorithm called Q-ILS-1, Q-learning is used to select appropriate local search operators at each stage of the search process, and in the second algorithm called Q-ILS-2, Q-learning is used for selecting appropriate perturbation operators. We will show that both Q-ILS-1 and Q-ILS-2 are able to find good solutions and outperform the ILS with single operator and also ILS with multiple randomly selected operators.

The rest of the paper is organized as follows. Section 2 reviews the recent relevant papers studying Q-learning for AOS in solving different COPs. Section 3 explains the preliminaries and main concepts of this paper. The two Q-ILS algorithms (Q-ILS-1 and Q-ILS-2) are proposed in Section 4. The performances of the proposed algorithms are investigated in Section 5. Finally, the conclusion is given in Section 6.

2 Literature review

AOS has been widely studied within different MHs for adaptively selecting the search operators [7]. Most of the studies use simple score-based methods that select operators based on their accumulated score [7]. Besides simple score-based mechanisms for AOS, RL techniques, in particular Q-learning algorithm, have been used for AOS in recent years [2,19,18,14,9]. In the following, the studies on the use of Q-learning algorithm for AOS for solving different COPs are elaborated.

In [19], Q-learning has been integrated into a Variable Neighborhood Search algorithm to solve the symmetric TSP. The role of Q-learning is to select appropriate local search operators during the search process, where both the states and actions are a set of local search operators (i.e., interchange, insertion, 2-opt, and double-bridge). The authors show that using Q-learning to intelligently select the local search operators achieves satisfactory results for small-sized instances of the TSP. In [18], the Q-learning algorithm is used to select the search operators of a Genetic algorithm, namely mutation and crossover operators, during the search process for solving TSP. The authors discuss that adaptive operator selection based on the immediate performance of the operators might lead to a short-sighted optimization. Therefore, to overcome this shortcoming, they recommended using RL that can learn a policy to maximize the expected reward in a long term prospect. In [14], the authors have used

Q-learning algorithm to select appropriate local search operators of a Simulated Annealing algorithm. The proposed algorithm is applied to mixed-model sequencing problem to select among exchange, shift, and knowledge sharing operators. The states are defined as the number of successful neighbor moves (i.e., moves that improve the objective function) occurred during an episode, and actions are a set of triplet local search operators. They show that the integration of Q-learning into Simulated Annealing significantly improves its performance comparing to other Simulated Annealing-based algorithms. In [2], the authors have employed Q-learning to select the order of applying mutation and crossover operators in each generation of the Genetic algorithm. In their algorithm, five states are defined depending on the number of chromosomes within the population that are replaced by executing an action, and there are two possible actions; apply crossover first and mutation next or apply mutation first and crossover next. To show the performance of the proposed method, it is applied to job sequencing and tool switching problem. The authors show that the proposed algorithm is competitive and even superior to the state-of-the-art algorithms for solving some instances of the problem.

As shown by the reviewed papers, the use of Q-learning in AOS has provided promising results in solving different COPs, and even in some cases it has been superior to some state-of-the-art algorithms. Motivated by the good performance of Q-learning, this paper aims at investigating the integration of Q-learning into AOS to select local search (Q-ILS-1 algorithm) and perturbation (Q-ILS-2 algorithm) operators of the ILS for solving the TSP.

The main contributions of this paper compared to the literature are threefold: 1) for the first time, this paper investigates the use of Q-learning in ILS for intelligently selecting the search operators throughout the search process, 2) the Q-learning is integrated into ILS in two levels for selecting local search and perturbation operators, with the aim of investigating the effect of intelligent AOS in each level, and 3) a new design of Q-learning is proposed where a set of appropriate states and actions are defined according to the level of integration. In Q-ILS-1 the states are defined as the sequence of last k local search operators and the actions are the local search operators. In Q-ILS-2, we define two states; 0 if there is no improvement in the best found solution during an episode and 1; otherwise, and the actions are a set of perturbation operators.

3 Preliminaries

In this section, first a short introduction to the TSP is provided. Next, the basics of the ILS algorithm and the Q-learning algorithm, are explained.

3.1 Traveling Salesman Problem

TSP is a classical NP-hard COP, which requires exponential time to be solved to optimality [11]. TSP can be formally defined by means of a weighted graph $G = (V, A)$ where V is the set of vertices representing cities and A is the set of edges that connect the vertices of V. The edge that connects cities i and j has a weight of d_{ij}, which represents the distance between cities i and j; $i, j \in V$. In TSP, the aim is to find the Hamiltonian cycle of minimum total travel distance such that all vertices are visited exactly once.

3.2 Iterated Local Search

Iterated Local Search (ILS) is a well-known MH for its effectiveness in both exploration and exploitation and its simplicity in practice. When the search gets trapped in a local optimum, ILS attempts to escape from the trap without losing many of the good properties of the current solution [12]. Considering s_{best} as the best solution found in the history of ILS, the general pseudo code of ILS is given in Algorithm 1. For a given initial solution s_0, a `LocalSearch(.)` function is performed on solution s_0 to search its neighborhood with the hope to find better solutions, particularly the local optimal solution s^*. Subsequently, s^* is archived as the current best solution s_{best}. Then, the main loop of the ILS starts

by performing a `Perturbation(.)` function over the current local optimum solution s^* to help the search process to escape from the local optimum; whereby an intermediate solution s' is generated. The `LocalSearch(.)` function is performed on the intermediate solution s' to obtain a new local optimal solution $s^{*'}$. Next, the `Acceptance`$(s^*, s^{*'}, s_{best})$ function is employed to check whether the new local optimal solution $s^{*'}$ is accepted. The `Acceptance`(.) function can only accept better solution (i.e., *Only Improvement* strategy) or it can even accept worse solution with a small gap (i.e., *Metropolis acceptance* strategy [13]). Finally, the best solution s_{best} is updated. The algorithm terminates when the termination criterion is satisfied.

Algorithm 1. Pseudo code of the ILS

```
get an initial solution s_0
s* := LocalSearch(s_0)
s_best := s*
while termination criterion not reached do
    s' := Perturbation(s*)
    s*' := LocalSearch(s')
    s* := Acceptance(s*, s*', s_best)
end
return the best found solution s_best
```

3.3 The Q-learning algorithm

In RL, an agent interacts with the environment and aims to iteratively learn which action to take at a given state of the environment to achieve a goal. At each interaction depending on the state s ($s \in S$) the agent takes an action a ($a \in A(s)$) and receives a numerical feedback from the environment. Through this process, the agent attempts to iteratively maximize the cumulative received reward. Classical RL methods need the complete model of the environment (i.e., all possible states of the system, the set of possible actions per state and the matrix of transition probabilities as well as the expected values of the feedback). However, in most problems including COPs, it is not possible to have a complete model of the environment [26]. In such cases, Monte Carlo and Temporal Differences algorithms can be used [21]. The Q-learning algorithm [25] is a model-free RL algorithm based on temporal differences. In Q-learning, a Q-value is associated with each state-action pair (s, a) that represents the expected gain of the choice of action a at state s. The Q-value of each state-action pair (s, a) is updated using Expression (1).

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)] \tag{1}$$

where r is the reward (punishment) received after performing action a in state s and γ ($0 \leq \gamma < 1$) and α ($0 \leq \alpha < 1$) are the discount factor and the learning rate, respectively.

One strategy to select the actions in Q-learning is to always select the action with the maximum Q-value. In this strategy, the best state-action pairs with the maximum Q-values are exploited sufficiently, while other state-action pairs remain unexplored. To cope with this issue and to make a balance between exploration and exploitation, the ϵ-greedy strategy (Expression (2)) [21] is an efficient strategy that assigns an ϵ selection probability to other actions to give them a chance to be explored.

$$a = \begin{cases} \underset{a}{\operatorname{argmax}}\, Q(s,a) & \text{with probability } 1-\epsilon \\ \text{any other action} & \text{with probability } \epsilon \end{cases} \tag{2}$$

To move from exploration of new actions toward exploitation of the best actions, the value of ϵ gradually degrades throughout the search process using a parameter β called ϵ-decay.

4 Proposed Q-ILS algorithms

This section proposes two Q-ILS-1 and Q-ILS-2 algorithms and explains their corresponding operators and properties.

4.1 Q-ILS-1 algorithm

The novelty of the proposed Q-ILS-1 algorithm is development of a new local search procedure for ILS based on the ideas from AOS and Q-learning. The proposed local search procedure adaptively selects appropriate operators during the search process based on the current employed operator and operators' history of performance. In the first step, a pool of local search operators are incorporated into the algorithm. Then, the proposed Q-learning algorithm is integrated into AOS to select local search operators.

General framework In Q-ILS-1, the local search operators perform a descent-based search and continue until no more improvements are found. As the perturbation operator, we employ *double-bridge* operator wherein four edges are removed from the route of the cities and sub-routes are reconnected in another way to explore a new route [22]. The `Acceptance(.)` function in Q-ILS-1 applies a *Metropolis acceptance* strategy [13] that accepts all improved solutions and even non-improved solutions with a probability of $\exp \frac{\Delta f}{T}$, where Δf is the difference between the objective function before and after applying the local search operator, and parameter T denotes the temperature. The higher the value of T, the higher the chance to accept worse moves and vice versa.

Local search operators In Q-ILS-1, three efficient local search operators are used; the *basic 2-opt* [22], *a new 2-opt*, and *a new insertion* operators. The basic 2-opt removes two edges from the route of the cities and reconnects the sub-routes with new edges.

In this paper, we propose a new 2-opt operator based on the idea of best-move 2-opt presented in [6]. In the best-move 2-opt, in each iteration of the local search, all the improving moves are identified and sorted based on their improvement value, and only the best improving move is performed. In this way, the information gathered about other improving moves is neglected and remain unused. However, in the proposed 2-opt, the main idea is to use the gathered information about the improving moves and to perform all possible moves simultaneously as long as they can be done independently (i.e., they do not share any segment of the route). In this way, in an iteration of the local search, a greater value of improvement achieves. We call this new 2opt, the best-independent-moves 2-opt. To explain the procedure of the proposed 2-opt, consider a simple example of Figure 1. In the first step, all the improving moves are identified (moves 1, 2, 3, and 4 with improving values in parenthesis). Then, the improving moves are sorted based on their improvement values in a descending order (moves 2, 4, 1, 3). Finally, starting from the first move, all the independent moves are performed simultaneously (moves 2, 4, and 3). Indeed, move 1 cannot be applied immediately after move 4 since they share the same segment "Q-R-A".

In addition, we propose a new insertion operators in this paper. In the new proposed insertion operator, four types of moves are employed: *forward-left*, *forward-right*, *backward-left*, and *backward-right*. Let's consider two $k \rightarrow i \rightarrow l$ and $m \rightarrow j \rightarrow n$ segments of the route, where the first segment is visited before the second segment. In addition, consider that two cities i and j undergo the insertion operator. The four above-mentioned insertion moves produce $m \rightarrow i \rightarrow j \rightarrow n$, $m \rightarrow j \rightarrow i \rightarrow n$, $k \rightarrow j \rightarrow i \rightarrow l$, and $k \rightarrow i \rightarrow j \rightarrow l$, respectively. Finally, the best insertion move among all four moves are applied to the solution.

Action, state, and reward In Q-ILS-1, the actions are the set of local search operators to be selected and applied at each iteration and the states are the sequence of last k local search operators (k is equal to 1). At the end of each iteration, the performance of the employed perturbation operator is evaluated.

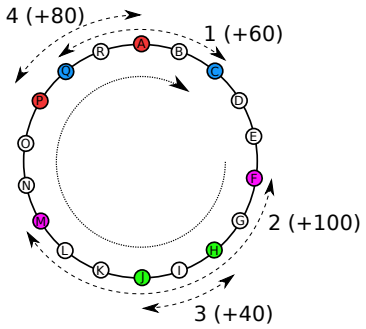


Fig. 1: Independent improving moves in the best-independent-moves 2-opt operator

Then, a reward or punishment is assigned to the employed operator. If the operator has been able to improve the best found solution, it receives a reward equal to the proportional improvement of the objective function; otherwise, it receives a punishment and is deleted from the set of available actions for the next iteration. In some cases, where no operator is able to improve the solution and set of available operators is empty, one operator is selected and applied randomly.

4.2 Q-ILS-2 algorithm

The novelty of the proposed Q-ILS-2 algorithm is development of a new perturbation procedure for ILS based on the ideas from AOS and Q-learning. In this algorithm, the type of perturbation operators and the number of times to apply them are adaptively selected based on the status of the search using the Q-learning algorithm. The aim of the proposed perturbation procedure is to adapt the exploration level to the status of the search. The general framework of Q-ILS-2 is the same as Q-ILS-1 except that Q-ILS-2, employs *the best-independent-moves 2-opt* operator as its single local search operator.

Perturbation operators In Q-ILS-2, a pool of three different perturbation operators are employed; the *Double-bridge* operator, the *Shuffle-sequence* operator that perturbs the solution by re-ordering a randomly selected sequence at random, and the *Reversion-sequence* operator that perturbs the solution by reversing a randomly selected sequence from the solution.

Action, state, and reward In Q-ILS-2, the actions are tuples (P, R), where P is the type of the perturbation operator and R is the repetition number of the perturbation operator P. Each action is given a chance of one episode equal to a fixed number of iterations to help the solution to escape from the local optimum. Accordingly, the states are the set of $S = \{0, 1\}$. $s = 1$ if the current perturbation operator P with R number of repetition followed by the local search has been able to improve the best found solution in an episode and $s = 0$, otherwise. After evaluation of the current action at the end of each episode, a reward (punishment) is assigned to the corresponding action. If the operator has been able to improve the best found solution, it receives a reward equal to the proportional improvement of the objective function; otherwise, it receives a punishment.

5 Results and discussion

In this section, the performance of the two proposed algorithms, Q-ILS-1 and Q-ILS-2 are validated through a set of experimental results. For this aim, the experiments are designed in Section 5.1. Next, the numerical results are presented in Section 5.2.

5.1 Experimental design

The performance of the proposed algorithms are validated through a set of 24 randomly selected symmetric TSP instances from the TSPLIB library [1] with different number of cities ranging from 50 to 2150. Two different experiments are done in this paper. First, the performance of the proposed algorithms in finding the optimal solution is investigated. Second, a comparative study is done to assess the efficiency of employing Q-learning in AOS. For this aim, first, in order to show the effectiveness of intelligent operator selection, Q-ILS-1 and Q-ILS-2 are compared to their corresponding Random ILS (R-ILS) with the same set of operators selected randomly. Second, in order to show the effectiveness of incorporating multiple operators into ILS, Q-ILS-1 and Q-ILS-2 are compared to their corresponding S-ILS algorithms, each employing single local search and perturbation operators.

For the Q-ILS-2, the maximum number of repetitions R of *double-bride*, *shuffle-sequence* and *reversion-sequence* are considered equal to 3, 1, and 1, respectively. The input parameters of the proposed algorithms are tuned using Design of Experiments [17] where $\epsilon = 0.8, \alpha = 0.6, \gamma = 0.5, \beta = 0.999$, and $episode = 3$. Each algorithm has been executed 30 times on each instance and is stopped after $0.2N$ number of iterations without improvement, where N is the number of cities in each instance. All algorithms have been coded in Python 3 and executed on an Intel Core i5 with 2.7GHz CPU and 16G of RAM.

The performance of the algorithms is measured using two main criteria [22]:

- The *solution quality* represented as the Relative Percentage Deviation (RPD). The RPD is calculated as $RPD = \frac{OF - OF^*}{OF^*} \times 100$, where OF is the objective function (i.e., tour length) of the best found solution by each algorithm and OF^* is the objective function of the optimal solution.
- The *convergence behavior* of the algorithms that measures how fast (i.e., when/ at which iteration) an algorithm converges to the best found (optimal) solution.

5.2 Numerical results

The performance of Q-ILS-1 and Q-ILS-2 in achieving the (near-) optimal solution are investigated through Tables 1 and 2. In these tables, the columns "Best RPD" and "Best time" are the gap of the best found solution and its corresponding CPU time, respectively and the columns "Average RPD" and "Average time" are the average gap and average CPU time over 30 executions.

Table 1 indicates that Q-ILS-1 is able to find optimal solution in both small- and medium-sized instances and it is able to find near-optimal solutions with an optimality gap of 3.83% for the largest instance with 2152 cities. By looking at the "Best RPD" and the "Average RPD" results, it can be seen that Q-ILS-1 has produced small gaps over all 30 executions. In terms of the CPU time, the higher the size of the instance, the higher the CPU time of the algorithm. By looking at the "Best Time" and the "Average Time" results, it can be seen how expensive certain instances are in terms of CPU time. For example, instance "ts225" with 225 cities is much less expensive comparing to instance "ch130" with 130 cities. Accordingly, the number of cities is not the only factor that affects the computational complexity of the instance, but the geographical distribution of the cities is also an important factor.

Some of the observations from Table 1 can be also generalized to the results of Table 2. Besides the zero optimality gap for small- and medium-sized instances, Q-ILS-2 is even able to find optimal solution for some large-sized instances up to 300 cities. For larger instances, small gaps have been also reported with an optimality gap of 3.94% for the largest instance with 2152 cities. Similar to Q-ILS-1, the results of "Best RPD" and "Average RPD" show that Q-ILS-2 produces small gaps over all 30 executions for almost all instances.

Table 1: Result of the proposed Q-ILS-1 in comparison to the optimal solutions

Instance	Optimal	Best RPD(%)	Best time (s)	Average RPD(%)	Average time (s)
berlin52	7542	0	0.1	0	0.7
st70	675	0	2.4	0.037	9.9
kroA100	21282	0	0.3	0.005	10.2
rd100	7910	0	9.3	0.173	25.9
lin105	14379	0	0.5	0	15.2
pr124	59030	0	4.7	0.004	34.1
ch130	6110	0.262	42.9	0.546	73.4
ch150	6528	0.077	50.5	0.465	50.8
u159	42080	0	5.8	0	55.4
d198	15780	0.165	286.8	0.263	220.9
kroA200	29368	0.051	97.3	0.352	237.5
ts225	126643	0	1.1	0	48.0
pr264	49135	0	303.5	0.402	207.1
a280	2579	0	451.8	0.587	396.8
pr299	48191	0.151	734.7	0.805	704.3
lin318	42029	0.895	737.2	1.559	688.7
fl417	11861	0.430	605.5	0.696	796.1
pr439	107217	0.755	840.1	2.308	778.2
pcb442	50778	1.061	747.7	1.568	758.4
d493	35002	1.451	16.5	2.135	643.4
vm1084	239297	3.025	1461.4	4.217	2012.2
d1291	50801	3.173	782.4	4.295	1835.3
u1817	57201	4.142	254.0	4.858	664.4
u2152	64253	3.836	235.3	4.841	919.8

Table 2: Result of the proposed Q-ILS-2 in comparison to the optimal solutions

Instance	Optimal	Best RPD(%)	Best time (s)	Average RPD(%)	Average time (s)
berlin52	7542	0	0	0	0.2
st70	675	0	0.7	0.109	4.5
kroA100	21282	0	1.3	0	7.3
rd100	7910	0	3.6	0.201	14.2
lin105	14379	0	0.4	0	4.5
pr124	59030	0	1.8	0	8.7
ch130	6110	0	17.4	0.359	40.4
ch150	6528	0	19.8	0.374	33.2
u159	42080	0	9.6	0.112	42.9
d198	15780	0.057	149.4	0.180	147.5
kroA200	29368	0	117.3	0.150	194.8
ts225	126643	0	56.5	0.002	108.9
pr264	49135	0	61.4	0.172	135.8
a280	2579	0	246.4	0.498	276.5
pr299	48191	0	453.1	0.274	440.7
lin318	42029	0.302	502.4	0.795	479.6
fl417	11861	0.211	574.5	0.339	553.0
pr439	107217	0.438	573.9	1.392	528.2

To be continued ...

Table 2 (continued)

Instance	Optimal	Best RPD(%)	Best time (s)	Average RPD(%)	Average time (s)
pcb442	50778	0.640	518.4	1.294	521.6
d493	35002	0.923	571.5	1.499	587.8
vm1084	239297	3.061	2729.6	3.717	2608.3
d1291	50801	2.281	1985.9	3.418	2060.1
u1817	57201	3.449	412.1	4.366	1416
u2152	64253	3.947	1967.5	4.531	1022.3

Table 3 shows the comparative results of Q-ILS-1 and Q-ILS-2 against R-ILS and S-ILS algorithms. Considering Q-ILS-1 with three local search operators, there are three S-ILS; S-ILS-1 to S-ILS-3 that stand for the use of *basic 2-opt*, *best-independent-moves 2-opt* and *insertion* local search operators, respectively. Considering Q-ILS-2 with three perturbation operators, S-ILS-1 to S-ILS-3 stand for the use of *double-bridge*, *shuffle-sequence*, and *reversion-sequence* operators perturbation operators repeated only once, respectively. The values in Table 3 represent the RPD of other algorithms (i.e., R-ILS and S-ILS) comparing to Q-ILS-1 and Q-ILS-2 which is calculated as Equation 3:

$$RPD^{R(S)} = \frac{OF^{R(S)} - OF^{Q}}{OF^{Q}} \times 100 \tag{3}$$

where $OF^{R(S)}$ is the average tour length obtained by R-ILS (S-ILS) for each instance and OF^{Q} is the average tour length obtained by Q-ILS-1/Q-ILS-2. A positive RPD value for an algorithm represents that the corresponding algorithm has a positive gap compared to the Q-ILS-1/Q-ILS-2. The RPD values equal to 0 shows both the two algorithms have led to the same solution. It can be seen that both R-ILS and S-ILS for almost all the instances have positive gap comparing to Q-ILS-1 and Q-ILS-2. This highlights the outperformance of the proposed Q-ILS algorithms over R-ILS and S-ILS in terms of the optimality gap. Investigating the results of R-ILS with positive gaps illustrates the efficiency of integrating the knowledge from the Q-learning algorithm into the operator selection mechanism of the ILS algorithm. Furthermore, the performance of Q-ILS algorithms are better than the S-ILS algorithms with single local search or perturbation operators. It shows the efficiency of employing different operators when solving TSP instances. Based on the obtained results, it can be concluded that intelligent selection of the operators at each stage of the search process using Q-learning provides promising results when solving TSP instances.

Table 3: The RPD (%) of R-ILS and S-ILS comparing to Q-ILS-1 and Q-ILS-2

Instance	Q-ILS-1				Q-ILS-2			
	R-ILS	S-ILS			R-ILS	S-ILS		
		1	2	3		1	2	3
berlin52	2.04	1.34	2.09	3.01	3.34	3.48	3.17	2.10
st70	1.05	1.03	1.21	1.00	1.19	1.25	0.74	0.83
kroA100	0.34	0.76	0.43	2.17	0.97	1.23	0.58	0.64
rd100	1.23	1.53	1.63	3.00	1.43	1.79	1.20	1.62
lin105	1.39	1.07	1.19	1.02	0.83	1.84	1.24	0.85
pr124	0.60	0.72	0.93	1.87	1.06	1.21	0.65	0.70
ch130	0.99	1.51	1.12	2.60	1.68	2.18	1.12	1.23
ch150	0.95	1.19	1.29	3.20	1.31	2.36	1.01	1.17
u159	1.78	1.44	1.26	2.28	2.28	2.99	1.28	1.74
d198	0.43	0.39	0.51	2.65	0.73	1.92	0.71	0.96

To be continued ...

Table 3 (continued)

Instance	Q-ILS-1				Q-ILS-2			
	R-ILS	S-ILS			R-ILS	S-ILS		
		1	2	3		1	2	3
kroA200	0.53	0.31	0.37	1.07	1.33	1.91	0.60	1.02
ts225	0.72	0.81	0.97	3.67	0.72	1.54	0.74	0.80
pr264	1.00	1.15	1.22	5.05	1.67	3.39	1.12	1.14
a280	1.96	1.84	2.27	4.23	2.23	4.17	1.53	2.00
pr299	1.63	2.07	1.56	5.06	2.22	4.23	1.65	2.18
lin318	0.58	0.48	0.58	0.97	1.81	3.34	1.44	1.79
fl417	1.67	2.00	2.19	5.11	1.14	3.88	2.19	1.99
pr439	1.51	1.22	1.68	3.16	2.78	3.65	1.75	1.45
pcb442	0.77	1.34	1.40	3.36	1.26	3.16	1.66	1.97
d493	0.85	0.87	1.49	3.28	1.26	3.58	2.18	2.06
vm1084	0.67	1.01	1.44	3.58	1.33	2.74	1.40	1.89
d1291	1.15	1.08	1.74	2.82	2.06	3.33	2.30	2.62
u1817	2.19	1.35	2.82	5.17	1.57	3.18	3.14	3.24
u2152	2.45	1.26	2.46	5.59	1.75	2.72	2.43	2.52

In addition to investigating the performance of Q-ILS-1 and Q-ILS-2 over R-ILS and S-ILS in finding (near-) optimal solutions, the algorithms are also compared based on their convergence behavior. In this regard, the average gap to the optimal solution for the instance "d493" for different algorithms at different stages of the search are depicted in Figure 2 for Q-ILS-1 and Figure 3 for Q-ILS-2.

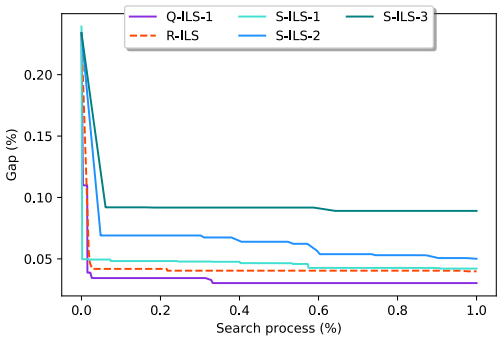


Fig. 2: Convergence behavior of Q-ILS-1 comparing to its benchmarks for instance d493

As Figures 2 and 3 illustrate, the convergence of Q-ILS-1 and Q-ILS-2 happens at earlier stages of the search, about 40% of the search process for Q-ILS-1 and 60% of the search process for Q-ILS-2, which leads to solutions with higher quality in both algorithms. Considering Figure 3, although R-ILS converges at earlier stages, it is a premature convergence which cannot be improved by the end of the search process. Considering both Figures 2 and 3, all algorithms are competitive but the Q-ILS-1 and Q-ILS-2 always converge faster to the good solutions. The faster convergence of Q-ILS-1 and Q-ILS-2 is also observed for all the TSP instances.

Based on the obtained results, the integration of Q-learning into ILS in both levels provides promising results. Comparing the performance of the proposed Q-ILS-1 and Q-ILS-2 algorithms, it can be seen

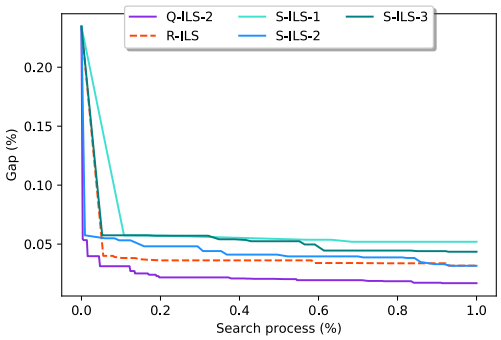


Fig. 3: Convergence behavior of Q-ILS-2 comparing to its benchmarks for instance d493

that Q-ILS-2 outperforms the Q-ILS-1 in all the selected instances. It can be concluded that incorporating multiple efficient perturbation operators with different characteristics into ILS and intelligently selecting among them significantly enhances the exploration ability of the ILS.

6 Conclusion

In this paper, we have integrated the Q-learning algorithm as a machine learning technique to select the most appropriate search operators in the ILS algorithm for solving TSP. For this aim, the Q-learning has been integrated into the ILS algorithm in two levels: 1) selecting the appropriate local search operators and 2) selecting the appropriate perturbation operators at each stage of the search process. In the first integration level, a set of three local search operators including the basic 2-opt, a new 2opt, and a new insertion operator are considered. In the second integration level, the selection is done among three perturbation operators including double-bridge, shuffle-sequence and reversion-sequence.

The performance of the proposed algorithms has been tested on a set of 24 symmetric TSP instances from the TSPLIB library. In addition, a comparative study has been conducted to investigate the efficiency of intelligently selecting search operators using Q-learning algorithm. The results showed that the proposed algorithms are able to find optimal solutions for small- and medium-sized instances and near-optimal solutions for large-sized instances with small gaps. Through the comparative analysis, it was observed that employing several search operators provides better performance for the ILS when solving the TSP instances. Furthermore, the impact of the Q-learning for intelligently selecting the appropriate search operators at each stage of the search process was significant.

Finally, it was concluded that employing different perturbation operators provides better results in comparison to employing different local search operators. Indeed, the ILS algorithm is inherently powerful in exploitation while it gets trapped easily in local optimum. Accordingly, considering different perturbation operators and selecting the most appropriate one at each stage of the search process helps the ILS to escape from the local optimum. ILS with multiple perturbation operators becomes more and more efficient when the knowledge obtained from the Q-learning algorithm is injected into its operator selection mechanism.

Testing the performance of the proposed algorithms on TSP instances with larger sizes could be an interesting future research direction. In addition, considering other types of local search search and perturbation operators and testing their performance is another future research direction that is worth of further investigation. Finally, comparing the performance of the proposed algorithms against the benchmark algorithms in the literature and statistically checking their differences could be another future research work.

References

1. www.elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html
2. Ahmadi, E., Goldengorin, B., Süer, G.A., Mosadegh, H.: A hybrid method of 2-tsp and novel learning-based ga for job sequencing and tool switching problem. Applied Soft Computing **65**, 214–229 (2018)
3. Bengio, Y., Lodi, A., Prouvost, A.: Machine learning for combinatorial optimization: a methodological tour d'horizon. European Journal of Operational Research **290**(2), 405–421 (2021)
4. Burke, E.K., Hyde, M.R., Kendall, G., Ochoa, G., Özcan, E., Woodward, J.R.: A classification of hyper-heuristic approaches: revisited. In: Handbook of Metaheuristics, pp. 453–477. Springer (2019)
5. Calvet, L., de Armas, J., Masip, D., Juan, A.A.: Learnheuristics: hybridizing metaheuristics with machine learning for optimization with dynamic inputs. Open Mathematics **15**(1), 261–280 (2017)
6. El Krari, M., El Benani, B., et al.: Breakout local search for the travelling salesman problem. Computing and Informatics **37**(3), 656–672 (2018)
7. Fialho, Á.: Adaptive operator selection for optimization. Ph.D. thesis, Université Paris Sud - Paris XI (2010)
8. Karimi-Mamaghan, M., Mohammadi, M., Jula, P., Pirayesh, A., Ahmadi, H.: A learning-based metaheuristic for a multi-objective agile inspection planning model under uncertainty. European Journal of Operational Research **285**(2), 513–537 (2020)
9. Karimi-Mamaghan, M., Mohammadi, M., Pasdeloup, B., Billot, R., Meyer, P.: An online learning-based metaheuristic for solving combinatorial optimization problems. In: 21ème congrès annuel de la société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF) (2020)
10. Karimi-Mamaghan, M., Mohammadi, M., Pirayesh, A., Karimi-Mamaghan, A.M., Irani, H.: Hub-and-spoke network design under congestion: A learning based metaheuristic. Transportation Research Part E: Logistics and Transportation Review **142**, 102069 (2020)
11. Karp, R.M.: On the computational complexity of combinatorial problems. Networks **5**(1), 45–68 (1975)
12. Lourenço, H.R., Martin, O.C., Stützle, T.: Iterated local search. In: Handbook of metaheuristics, pp. 320–353. Springer (2003)
13. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equation of state calculations by fast computing machines. The journal of chemical physics **21**(6), 1087–1092 (1953)
14. Mosadegh, H., Ghomi, S.F., Süer, G.A.: Stochastic mixed-model assembly line sequencing problem: Mathematical modeling and q-learning based simulated annealing hyper-heuristics. European Journal of Operational Research **282**(2), 530–544 (2020)
15. Pasdeloup, B., Karimi-Mamaghan, M., Mohammadi, M., Meyer, P.: Autoencoder-based generation of individuals in population-based metaheuristics. In: ROADEF 2020: 21ème Congrès Annuel de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (2020)
16. Peng, B., Zhang, Y., Gajpal, Y., Chen, X.: A memetic algorithm for the green vehicle routing problem. Sustainability **11**(21), 6055 (2019)
17. Ridge, E., Kudenko, D.: Tuning an algorithm using design of experiments. In: Experimental methods for the analysis of optimization algorithms, pp. 265–286. Springer (2010)
18. Sakurai, Y., Takada, K., Kawabe, T., Tsuruta, S.: A method to control parameters of evolutionary algorithms by using reinforcement learning. In: 2010 Sixth International Conference on Signal-Image Technology and Internet Based Systems. pp. 74–79. IEEE (2010)
19. dos Santos, J.P.Q., de Melo, J.D., Neto, A.D.D., Aloise, D.: Reactive search strategies using reinforcement learning, local search algorithms and variable neighborhood search. Expert Systems with Applications **41**(10), 4939–4949 (2014)
20. Song, H., Triguero, I., Özcan, E.: A review on the self and dual interactions between machine learning and optimisation. Progress in Artificial Intelligence **8**(2), 143–165 (2019)
21. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT press (2018)
22. Talbi, E.G.: Metaheuristics: from design to implementation, vol. 74. John Wiley & Sons (2009)
23. Talbi, E.G.: Combining metaheuristics with mathematical programming, constraint programming and machine learning. Annals of Operations Research **240**(1), 171–215 (2016)
24. Turkeš, R., Sörensen, K., Hvattum, L.M.: Meta-analysis of metaheuristics: quantifying the effect of adaptiveness in adaptive large neighborhood search. European Journal of Operational Research **292**(2), 423–442 (2021)
25. Watkins, C.J.C.H.: Learning from delayed rewards (1989)
26. Wauters, T., Verbeeck, K., De Causmaecker, P., Berghe, G.V.: Boosting metaheuristic search using reinforcement learning. In: Hybrid Metaheuristics, pp. 433–452. Springer (2013)

A semi-infinite optimization approach to shape-constrained regression

Jochen Schmid[1][0000−0001−6527−537X], Martin von Kurnatowski[1][0000−0001−6123−8723], Patrick Link[2][0000−0002−1451−6396], Lukas Morand[3][0000−0002−8566−7642], and Jan Schwientek[1][0000−0001−9740−3233]

[1] Fraunhofer Institute for Industrial Mathematics ITWM, Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany
[2] Fraunhofer Institute for Machine Tools and Forming Technology IWU, Reichenhainer Straße 88, 09126 Chemnitz, Germany
[3] Fraunhofer Institute for Mechanics of Materials IWM, Wöhlerstraße 11, 79108 Freiburg, Germany
jochen.schmid@itwm.fraunhofer.de

Abstract. We present a regression method for enhancing the predictive power of a model by exploiting expert knowledge in the form of shape constraints such as monotonicity or convexity constraints. Incorporating such information is particularly beneficial when the available data sets are sparse. We set up the regression subject to the considered shape constraints as a semi-infinite optimization problem and propose an efficient adaptive solution algorithm. It turns out that, in manufacturing applications with their typically sparse data, the predictive power of the models obtained with our method is generally superior to those obtained with standard monotonization methods.

Keywords: shape-constrained regression · semi-infinite optimization · informed machine learning.

1 Introduction

Conventional machine learning models are purely data-based. Accordingly, the predictive power of such models is generally bad if the underlying training data $\mathcal{D} = \{(\boldsymbol{x}_l, t_l) : l \in \{1, \ldots, N\}\}$ is insufficient. Such data insufficiencies occur quite often in applications, and they can come in one of the following forms: on the one hand, the available data sets can be too small and have too little variance in the input data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ and on the other hand, the output data $t_1, \ldots, t_N$ can be too noisy.

Aside from potentially insufficient data, however, one often also has additional knowledge about the relation between the input variables and the responses to be learned. Such extra knowledge about the considered process is referred to as expert knowledge in the following. Informed machine learning [1] techniques combine data and expert knowledge to build hybrid or gray-box models predicting the responses more accurately than purely data-based models.

An important and common type of expert knowledge is prior information about the monotonicity behavior of the unknown functional relationship $\boldsymbol{x} \mapsto y(\boldsymbol{x})$ to be learned. Along with convexity constraints, monotonicity constraints are probably the most intensively studied shape constraints in the literature and correspondingly, there exist plenty of different approaches to incorporate monotonicity knowledge in a machine learning model. See [2] for an extensive overview. Very roughly, these approaches can be categorized according to when the monotonicity knowledge is taken into account: in or only after the training phase. In this talk, we propose a novel in-training approach to monotonic regression and compare it to the standard after-training approaches [3], [4].

2 Semi-infinite optimization for monotonic regression

In our approach to monotonic regression, multivariate polynomial models

$$\boldsymbol{x} \mapsto \widehat{y}_{\boldsymbol{w}}(\boldsymbol{x}) = \sum_{|\alpha| \leq m} w_\alpha \boldsymbol{x}^\alpha \in \mathbb{R} \tag{1}$$

are used for all input-output relationships $\boldsymbol{x} \mapsto y(\boldsymbol{x})$ to be learned. In the above relation (1), the sum extends over all d-dimensional multi-indices $\alpha = (\alpha_1, \ldots, \alpha_d) \in \mathbb{N}_0^d$ with degree $|\alpha| := \alpha_1 + \cdots + \alpha_d$ less than or equal to some total degree $m \in \mathbb{N}$. Also, the terms $\boldsymbol{x}^\alpha := x_1^{\alpha_1} \cdots x_d^{\alpha_d}$ are the monomials in d variables of degree less than or equal to m and the N_m numbers w_α are the model parameters to be tuned by regression. As is well-known, standard polynomial regression is about solving the unconstrained optimization problem

$$\min_{\boldsymbol{w} \in \mathbb{R}^{N_m}} \frac{1}{2} \sum_{l=1}^{N} \big(\widehat{y}_{\boldsymbol{w}}(\boldsymbol{x}_l) - t_l \big)^2 \tag{2}$$

or, in other words, about optimally adapting the model parameters $w_\alpha \in \mathbb{R}$ of the polynomial model (1) to the available data $\mathcal{D} = \{(\boldsymbol{x}_l, t_l) : l \in \{1, \ldots, N\}\}$. In general, the resulting model $\boldsymbol{x} \mapsto \widehat{y}_{\boldsymbol{w}}(\boldsymbol{x})$ will not exhibit the monotonicity behavior an expert expects for the underlying true physical relationship $\boldsymbol{x} \mapsto y(\boldsymbol{x})$. In order to enforce the expected monotonicity behavior, the constraints

$$\sigma_j \cdot \partial_{x_j} \widehat{y}_{\boldsymbol{w}}(\boldsymbol{x}) \geq 0 \quad \text{for all } j \in J \text{ and } \boldsymbol{x} \in X \tag{3}$$

are added to the unconstrained standard regression problem (2). The numbers $\sigma_j \in \{-1, 0, 1\}$ indicate the expected monotonicity behavior for each coordinate direction $j \in \{1, \ldots, d\}$. Also, $J := \{j \in \{1, \ldots, d\} : \sigma_j \neq 0\}$ is the set of all directions for which a monotonicity constraint is imposed, and the input space $X := [a_1, b_1] \times \cdots \times [a_d, b_d]$ is a hyperbox in $\mathbb{R}^d$. Since X contains infinitely many points, the monotonic regression problem (2)-(3) takes the form

$$\min_{\boldsymbol{w} \in \mathbb{R}^{N_m}} f(\boldsymbol{w}) \quad \text{s.t.} \quad g_j(\boldsymbol{w}, \boldsymbol{x}) \geq 0 \quad \text{for all } j \in J \text{ and } \boldsymbol{x} \in X \tag{4}$$

of a semi-infinite optimization problem [5]. In order to solve (4), we adapt the adaptive discretization algorithm from [6] to our specific situation.

Algorithm 1 *1. Choose a finite subset $X^0 \subset X$ and set $k = 0$.*
2. Solve the kth discretized upper-level problem

$$\min_{\boldsymbol{w} \in \mathbb{R}^{N_m}} f(\boldsymbol{w}) \quad s.t. \quad g_j(\boldsymbol{w}, \boldsymbol{x}) \geq 0 \quad \text{for all } j \in J \text{ and } \boldsymbol{x} \in X^k \tag{5}$$

to obtain optimal model parameters $\boldsymbol{w}^k \in \mathbb{R}^{N_m}$.
3. Solve the (k, j)th lower-level problem $\min_{\boldsymbol{x} \in X} g_j(\boldsymbol{w}^k, \boldsymbol{x})$ δ_k-approximately for every $j \in J$ to obtain δ_k-approximate global minimizers $\boldsymbol{x}^{k+1,j} \in X$. Add those of the points $\boldsymbol{x}^{k+1,j}$, for which substantial monotonicity violations occur, that is, for which $g_j(\boldsymbol{w}, \boldsymbol{x}^{k+1,j}) < -\varepsilon_j$, to the current discretization X^k and go to Step 2 with $k = k + 1$. If for none of the points $\boldsymbol{x}^{k+1,j}$ substantial monotonicity violations occur, go to Step 4.
4. Check for monotonicity violations on a fixed, fine reference discretization $X_{\text{ref}} \subset X$. If there are no such violations, that is, if $g_j(\boldsymbol{w}^k, \boldsymbol{x}) \geq -\varepsilon_j$ for all $j \in J$ and $\boldsymbol{x} \in X_{\text{ref}}$, then terminate. If there are such violations, then for every direction j with violations, add the reference grid point $\boldsymbol{x}_{\text{ref}}^{k+1,j}$ with the largest violation to X^k and go to Step 2 with $k = k + 1$.

Clearly, the upper-level problems (5) are convex quadratic problems and the lower-level problems are multivariate polynomial optimization problems. Since in contrast to [6] we require only approximate solutions of the lower-level problems, we can use a wide range of global solvers, for example, global polynomial solvers based on semidefinite relaxation hierarchies. We present convergence results for this and related adaptive discretization algorithms. We also apply the algorithms to two real-world manufacturing applications, namely laser glass bending and press hardening of sheet metal. In these processes, experimental data are fairly costly and therefore sparse (with little variance), but monotonicity expert knowledge is available. Compared to the well-known after-training approaches to monotonization (via rearrangement [3] or projection [4]), our in-training semi-infinite optimization approach yields monotonic models with generally better predictive power. Another advantage of our approach is that it directly extends not only to convexity constraints but, in fact, to any shape constraint that can be expressed in terms of the partial derivatives of the underlying model $\boldsymbol{x} \mapsto \widehat{y}_{\boldsymbol{w}}(\boldsymbol{x})$.

References

1. L. v. Rueden et al.: Informed machine learning – a taxonomy and survey of integrating knowledge into learning systems. arxiv.org/pdf/1903.12394v2
2. M. Gupta et al.: Calibrated interpolated look-up tables. J. Mach. Learn. Res. **17** (109), 1-47 (2016)
3. H. Dette, R. Scheder: Strictly monotone and smooth nonparametric regression for two or more variables. Can. J. Stat. **34** (4), 535-561 (2006)
4. L. Lin, D.B. Dunson: Bayesian monotone regression using Gaussian process projection. Biometrika **101** (2), 303-317 (2014)
5. O. Stein: How to solve a semi-infinite optimization problem. Eur. J. Op. Res. **223** (2), 312-320 (2012)
6. J.W. Blankenship, J.E. Falk: Infinitely constrained optimization problems. J. Opt. Th. Appl. **19** (2), 261-281 (1976)

Improved SAT models for NFA learning

Frédéric Lardeux[0001−8636−3870] and Eric Monfroy[0001−7970−1368]

LERIA, University of Angers, France firstname.lastname@univ-angers.fr

Abstract. Grammatical inference is concerned with the study of algorithms for learning automata and grammars from words. We focus on learning Nondeterministic Finite Automaton of size k from samples of words. To this end, we formulate the problem as a SAT model. The generated SAT instances being enormous, we propose some model improvements, both in terms of the number of variables, the number of clauses, and clauses size. These improvements significantly reduce the instances, but at the cost of longer generation time. We thus try to balance instance size vs. generation and solving time. We also achieved some experimental comparisons and we analyzed our various model improvements.

Keywords: Constraint problem modeling · SAT · model reformulation.

1 Introduction

Grammatical inference [7] is concerned with the study of algorithms for learning automata and grammars from words. It plays a significant role in numerous applications, such as compiler design, bioinformatics, speech recognition, pattern recognition, machine learning, and others. The problem we address in this paper is learning a finite automaton from samples of words $S = S^+ \cup S^-$, which consist of positive words (S^+) that are in the language and must be accepted by the automaton, and negative words (S^-) that must be rejected by the automaton. A non deterministic automaton (NFA) being generally a smaller description for a language than an equivalent deterministic automaton (DFA), we focus here on NFA inference. An NFA is represented by a 5-tuple $(Q, \Sigma, \Delta, q_1, F)$ where Q is a finite set of states, the vocabulary Σ is a finite set of symbols, the transition function $\Delta : Q \times \Sigma \to \mathcal{P}(Q)$ associates a set of states to a given state and a given symbol, $q_1 \in Q$ is the initial state, and $F \subseteq Q$ is the set of final states.

Not to mention DFA (e.g., [6]), the problem for NFA has been tackled from a variety of angles. In [15] a wide panel of techniques for NFA inference is given. Some works focus on the design of ad-hoc algorithms, such as *DeLeTe2* [3] that is based on state merging methods. More recently, a new family of algorithms for regular languages inference was given in [14]. Some approaches are based on metaheuristic, such as in [12] where hill-climbing is applied in the context of regular language, or [4] which is based on genetic algorithm. In contrast to metaheuristics, complete solvers are always able to find a solution if there exists one, to prove the unsatisfiablility of the problem, and to find the optimal solution in case of optimization problems. In this case, generally, the problem is modeled

as a Constraint Satisfaction Problem (CSP [11]). For example, in [15], an Integer Non-Linear Programming (INLP) formulation of the problem is given. Parallel solvers for minimizing the inferred NFA size are presented in [8, 9]. The author of [10] proposes two strategies, based on variable ordering, for solving the CSP formulation of the problem.

In this paper, we are not interesting in designing or improving a solver, but we focus in improving models of the problem in order to obtain faster solving times using a standard SAT solver. Modeling is the process of translating a problem into a CSP consisting in decision variables and constraints linking these variables. The INLP model for NFA inference of [15] cannot be easily modified to reduce the instances: to our knowledge, only Property 1 of our paper could be useful for the INLP model, and we do not see any other possible improvement. We thus start with a rather straightforward conversion of the INLP model into the propositional satisfiablity problem (SAT [5]). This is our base SAT model to evaluate our improvements. The model, together with a training sample, lead to a SAT instance that we solve with a standard SAT solver. The generated SAT instances are very huge: the order of magnitude is $|S|.(|\omega|+1).k^{|\omega|}$ clauses, where k is the number of states of the NFA, ω is the longest word of S, and $|S|$ is the number of words of the training sample. We propose three main improvements to reduce the generated SAT instances. The first one prevents generating subsumed constraints. Based on a multiset representation of words, the second one avoid generating some useless constraints. The last one is a weaker version of the first one, based on prefixes of words. The first improvement returns smaller instances than the second one, which in turn returns smaller instances than the third one. However, the first improvement is very long and costly, whereas the third one is rather fast. We are thus interested in balancing generation and solving times against instance sizes. We achieved some experiments with the Glucose solver [1] to compare the generated SAT instances. The results show that our improvements are worth: larger instances could be solved, and faster. Generating the smallest instances can be too costly, and the best results are obtained with a good balance between instance sizes and generation/solving time.

This paper is organized as follows. In Section 2, we describe the problem and we give the basic SAT model. We also evaluate the size of the generated instances. Section 3 presents 3 model improvements, together with sketches of algorithms to generate them. Section 4 exposes our experimental results and some analysis. We finally conclude in Section 5.

2 Modeling the problem in SAT

The non-linear integer programming (INLP) model of [15, 9] cannot be easily improved or simplified. Indeed, the only improvement proposed in [15] and [9] corresponds to Property 1 (given in the next section). In this section, we thus present a SAT formulation of the NFA inference problem. This SAT model permits many improvements to reduce the size of the generated SAT instances.

The NFA inference problem Consider an alphabet $\Sigma = \{s_1, \ldots, s_n\}$ of n symbols; a training sample $S = S^+ \cup S^-$, where S^+ (respectively S^-) is a set of *positive words* (respectively *negative words*) from Σ^*; and an integer k. The problem consists in building a NFA of size k which validates words of S^+, and rejects words of S^-. The problem can be extended to an optimization problem: it consists in inferring a minimal NFA for S, i.e., an NFA minimizing k. However, we do not consider optimization in this paper.

Notations Let $A = (Q, \Sigma, q, F)$ be a NFA with: $Q = \{q_1, \ldots, q_k\}$ a set of states, Σ a finite alphabet (a set of symbols), q the initial state, and F the set of final states. The symbol λ represents the empty word. We denote by K the set $\{1, \ldots, k\}$. A transition from q_j to q_k with the symbol s_i is denoted by $\tau_{s_i,q_j \to q_k}$. Consider the word $w = w_1 \ldots w_n$ with $w_1, \ldots, w_n$ in Σ. Then, the notion of transition is extended to w by $T_{w,q_{i_1} \to q_{i_{n+1}}}$ which is a sequence of transitions $\tau_{w_1,q_{i_1} \to q_{i_2}}, \ldots, \tau_{w_n,q_{i_n} \to q_{i_{n+1}}}$. The set of candidate transitions for w between the states q_{i_1} and q_{i_l} in a NFA of size k is $\mathcal{T}_{w,q_{i_1} \to q_{i_l}} = \{T_{w,q_{i_1} \to q_{i_l}} \mid \exists i_2, \ldots i_{i_l - 1} \in K,\ T_{w,q_{i_1} \to q_{i_l}} = \tau_{w_1,q_{i_1} \to q_{i_2}}, \ldots, \tau_{w_l,q_{i_l - 1} \to q_{i_l}}\}$.

A SAT model Our base model is a conversion into SAT of the nonlinear integer programming problem given in [15] or [9]. Consider the following variables:

- k the size of the NFA we want to build,
- $F = \{f_1, \ldots, f_k\}$ a set of k Boolean variables determining whether states q_1 to q_k are final or not,
- and $\Delta = \{\delta_{s,q_i \to q_j} | s \in \Sigma \text{ and } i, j \in K\}$ a set of $n.k^2$ variables determining whether there is or not a transition $\delta_{s,q_i \to q_j}$, i.e., a transition from state q_i to state q_j with the symbol s, for each q_i, q_j, and s.

A transition $T_{w_1 \ldots w_n,q_{i_1} \to q_{i_{n+1}}} = \tau_{w_1,q_{i_1} \to q_{i_2}}, \ldots \tau_{w_n,q_{i_n} \to q_{i_{n+1}}}$ exists if and only if the conjunction $d = \delta_{w_1,q_{i_1} \to q_{i_2}} \wedge \ldots \wedge \delta_{w_n,q_{i_n} \to q_{i_{n+1}}}$ is true. We call d a c_transition, and we say that d models $T_{w_1 \ldots w_n,q_{i_1} \to q_{i_{n+1}}}$. We denote by D_{w,q_i,q_j} the set of all c_transitions for the word w between states q_i and q_j.

The problem can be modeled with 3 sets of equations:

1. If the empty word λ is in S^+ or in S^-, we can determine whether the first state is final or not:

$$\text{if } \lambda \in S^+, \quad f_1 \tag{1}$$
$$\text{if } \lambda \in S^-, \quad \neg f_1 \tag{2}$$

2. For each word $w \in S^+$, there is at least a transition starting in q_1 and ending in a final state q_j:

$$\bigvee_{j \in K} \bigvee_{d \in D_{w,q_1,q_j}} (d \wedge f_j) \tag{3}$$

With the Tseitin transformations [13], we create one auxiliary variable for each combination of a word w, a state $j \in K$, and a transition $d \in D_{w,q_1,q_j}$:

$$aux_{w,j,d} \leftrightarrow d \wedge f_j$$

For each w, we obtain a formula in CNF:

$$\bigwedge_{j \in K} \bigwedge_{d \in D_{w,q_1,q_j}} [(\neg aux_{w,j,d} \vee (d \wedge f_j))] \tag{4}$$

$$\bigwedge_{j \in K} \bigwedge_{d \in D_{w,q_1,q_j}} (aux_{w,j,d} \vee \neg d \vee \neg f_j) \tag{5}$$

$$\bigvee_{j \in K} \bigvee_{d \in D_{w,q_1,q_j}} aux_{w,j,d} \tag{6}$$

d is a conjunction, and thus $\neg aux_{w,j,d} \vee d$ is a conjunction of $|w|$ binary clauses: $(\neg aux_{w,j,d} \vee \delta_{w_1,q_1 \to q_{i_2}}) \wedge \ldots \wedge (\neg aux_{w,j,d} \vee \delta_{w_{|w|},q_{i_{|w|}} \to q_{i_{|w|+1}}})$.
$|D_{w,q_1,q_j}| = k^{|w|-1}$ since for each symbol of w there is k possible moves in the NFA, except for the last symbol which leads to q_j. Thus, we have $(|w|+1).k^{|w|}$ binary clauses for Constraints (4), $k^{|w|}$ $(|w|+2)$-ary clauses for Constraints (5), and one $k^{|w|}$-ary clause for Constraints (6). We have added $k^{|w|}$ auxiliary variables.

3. For each $w \in S^-$ and each state q_j, either there is no complete transition from state q_1 to q_j, or q_j is not final:

$$\neg \left[\bigvee_{j \in K} \bigvee_{d \in D_{w,q_1,q_j}} (d \wedge f_j) \right] \tag{7}$$

Constraints (7) are already in CNF, and we have $k^{|w|}$ $(|w+1|)$-ary clauses.

Thus, the constraint model M_k for building a NFA of size k is:

$$M_k = \bigwedge_{w \in S^+} \Big((4) \wedge (5) \wedge (6) \Big) \wedge \bigwedge_{w \in S^-} (7)$$

and is possibly completed by (1) or (2) if $\lambda \in S^+$ or $\lambda \in S^-$.

Size of the models Considering ω_+, the longest word of S^+, and ω_-, the longest word of S^-, the number of constraints in model M_k is bounded by:

- $|S^+|.(|\omega_+|+1).k^{|\omega_+|}$ binary clauses;
- $|S^+|.k^{|\omega_+|}$ $(|\omega_+|+2)$-ary clauses;
- $|S^+|$ $k^{|\omega_+|}$-ary clauses;
- $|S^-|.k^{|\omega_-|}$ $(|\omega_-|+1)$-ary clauses.

The number of Boolean variables is bounded by:

- k variables in F determining final states;
- $n.k^2$ variables determining existence of transitions;
- $|S^+|.k.^{|\omega_+|}$ auxiliary variables $aux_{w,j,d}$.

It is thus obvious that it is important to improve the model M_k.

3 Improving the SAT model

We now give some properties that can be used for improving the SAT model. By abuse of language, we will say that a model M_1 is smaller than a model M_2 whereas we should say that the SAT instance generated with M_1 and data D is smaller than the instance generated with M_2 and D. A first and simple improvement is based on the following property.

Property 1 (Empty word λ). If $\lambda \in S^-$, then each c_transition ending in q_1 does not have to be considered when generating the constraints related to the word $w \in S$.

Indeed, if w is positive, it cannot be accepted by a transition ending in q_1; similarly, if w is negative, $\neg d \vee \neg f_1$ is always true. When $\lambda \in S^+$, the gain is not very interesting: f_1 can be omitted in Constraints (7), (4), and (5). This does not really reduce the instance, and a standard solver would simplify it immediately.

Whereas a transition is an ordered sequence, the order of conjuncts in a c_transition is not relevant, and equal conjuncts can be deleted. Thus, a c_transition may model several transitions, and may correspond to several words. By abuse of language, we say that a c_transition ends in a state q_j if it corresponds to at least a transition ending in q_j. Thus, a c_transition may end in several states. We consider an order on c_transitions. Let d and d'' be two c_transitions. Then, $d \preceq d''$ if and only if there exists a c_transition d' such that $d \wedge d' = d''$. In other words, each transition variable $\delta_{s,q_i \to q_j}$ appearing in d also appears in d''. This order is used in the two first model improvements which are based on c_transitions. The third model improvement is based on transitions. We now consider some redundant constraints.

Property 2 (Redundant constraints). When a state q_i cannot be reached, each outgoing transition becomes free (it can be assigned true or false), and q_i can be final or not. In order to help the solver, all the corresponding variables can be assigned an arbitrary value. For each state q_j, $j \neq 1$:

$$\Big(\bigwedge_{i \in K, i \neq j} \bigwedge_{s \in \Sigma} \neg \delta_{s,q_i \to q_j}\Big) \to \neg f_j \wedge \Big(\bigwedge_{i \in K} \bigwedge_{s \in \Sigma} \neg \delta_{s,q_j \to q_i}\Big)$$

In CNF, these constraints generate (for all q_j), $(k-1).(k.n+1)$ redundant clauses of size $n.(k-1)+1$.

These constraints are useful when looking for a NFA of size k when k is not the minimal size of the NFA. Compared to SAT instance size, these redundant constraints can be very helpful without being too heavy.

Note that in our implementation, for all the models, we always simplify instances using Property 1 and removing duplicate transition variables in c_transitions (i.e., $\delta_{s,q_i \to q_j} \wedge \ldots \wedge \delta_{s,q_i \to q_j}$ is simplified into $\delta_{s,q_i \to q_j} \wedge \ldots$). Moreover, we also generate the redundant constraints as defined in Property 2.

Improvement based on c_transitions subsumption. This first improvement consists in removing tautologies for negative words, and some constraints and unsatisfiable disjuncts for positive words.

Property 3 (c_transition subsumption). Let v be a negative word from S^-, and $\neg d_v \vee \neg q_j$ be a Constraint (7) generated for the c_transition d_v for v ending in state q_j. We denote this constraint c_{v,d_v,q_j}. Consider a positive word w from S^+, and d_w a c_transition for w ending in q_j such that $d_v \preceq d_w$. Then, each $d_w \wedge f_j$ will be false due to c_{v,d_v,q_j}. Thus, Constraints (4) and (5) corresponding to w, d_w, and q_j will force to satisfy $\neg aux_{w,j,d_w}$; hence, they can be omitted and aux_{w,j,d_w} can be removed from Constraints (7). Similarly, consider ω from S^-, and d_ω a c_transition for w ending in q_j such that $d_v \preceq d_w$. Then, Constraint (7), $\neg d_v \vee \neg q_j$, will always be true (due to the constraint c_{v,d_v,q_j}), and can be omitted.

We can compute the size of the reduced SAT instance when the smaller word is a prefix. Let $v \in S^-$ and $w \in S$ be words such that $w = v.v'$, i.e., $v \subseteq w$ and v is a prefix of w. Then, using Property 3: if $w \in S^-$, the number of clauses generated for w is reduced to $(k-1).k^{|w|-1}$ clauses of size $|w+1|$; if $w \in S^+$, the number of clauses generated for w is reduced to $(|w|+1).(k-1).k^{|w|-1}$ binary clauses for Constraints (4), $(k-1).k^{|w|-1}$ $(|w|+2)$-ary clauses for Constraints (5), and one clause of size $(k-1).k^{|w|-1}$ for Constraint (6). The number of auxiliary variables is reduced to $(k-1).k^{|w|-1}$.

Operationally, we have a two step mechanism. First, for each negative word, each c_transition together with its ending state is generated and stored in a database of couples (c_transition, ending state) that we call c_couple. Then, for generating constraints for a word w, each of its c_couple is compared to the database. If a c_transition for w ending in q_j is smaller than a c_transition from the database also ending in q_j, then the corresponding constraints are not generated, as shown above. We call $M_{k,all}$ this reduced model.

Improvement based on Multisets. Although efficient in terms of generated instance sizes, the previous improvement is very costly in memory and time. It becomes rapidly intractable. This second improvement also uses Property 3. It is a weakening of the above operational mechanism that does not omit every subsumed c_transition. This mechanism is less costly. Hence, generated instances will be a bit larger, but the balance generation time against instance size is very good. The idea is to order words in order to search in a very smaller database of c_couples (c_transition, ending state) when generating constraints for a word w. Moreover, this order will also imply the order for generating constraints.

We associate each word to a multiset which support is the vocabulary Σ. The word w, is thus associated with the multiset $ms(w) = \{s_1^{|w|_{s_1}}, \ldots, s_n^{|w|_{s_n}}\}$ where $|w|_{s_i}$ is the number of occurrences of the symbol s_i in w. Note that several words can have the same multiset representation. Based on multiset inclusion $(\{s_1^{a'_1}, \ldots, s_n^{a'_n}\} \subseteq_{\mathcal{M}} \{s_1^{a_1}, \ldots, s_n^{a_n}\} \Leftrightarrow \forall i, a'_i \leq a_i)$, we can now define the notion of word inclusion, noted $\subseteq_\omega$. Consider w and w', two words of Σ^*, then:

$$w' \subseteq_\omega w \Leftrightarrow ms(w') \subseteq_{\mathcal{M}} ms(w)$$

Consider a sample $S = S^+ \cup S^-$. Let $\top(S)$ be the multiset defined as

$$\top(S) = \{s_1^{1+max_{w\in(S)}\{|w|_{s_1}\}}, \ldots, s_n^{1+max_{w\in(S)}\{|w|_{s_n}\}}\}$$

and $\bot = \{s_1^0, \ldots, s_n^0\}$. Then, $\top(S)$ represents words which are not in the sample S, and $\bot$ represents the empty word λ which may be in S.

Consider the sample $S = S^+ \cup S^-$. Let $MS(S) = \{ms(w)|w \in S^+ \cup S^-\}$ be the set of the representations of words of S. Then, $(MS(S) \cup \{\bot, \top(S)\}, \subseteq_{\mathcal{M}})$ is a lattice. Let m be a multiset of $MS(S)$. Then, $inf(m)$ is the set of multisets $\{m' \in MS(S) \mid m' \subseteq_{\mathcal{M}} m\}$. This lattice of multisets defines the data structure used for constraint generation. For generating constraint of a word w of a multiset m, we now only compare its c_couples with the database of c_couples of words $w' \in S^-$ with $w' \subseteq_\omega w$, i.e., words represented by multisets smaller than m.

The negative words that allow to reduce the most, are the ones represented by the smallest multiset. We thus also propose a mechanism to reduce the database (c_transition, ending state) with the most useful c_couples, i.e., the ones from smallest words. Let $level(m)$ be the "level" of the multiset defined by: $level(m) = 0$ if $m = \bot$, $1 + max_{m'\in inf(m)}(level(m'))$ otherwise. Given a multiset m, and a threshold l, the *base* function returns all the multisets m' of level smaller than l, and such that $m' \subseteq_{\mathcal{M}} m$: $base(p, l) = \{n \in inf(p) \mid level(n) \leq l\} \bigcup \left(\bigcup_{p'\in inf(p)} base(p', l)\right)$ if $p \neq \bot$, $\emptyset$ otherwise.

Based on Property 3, c_couples of the negative words of these multisets will be used to reduce constraint generation of the words of m. We call this model $M_{k,mset,l}$, with l a given threshold. If *base* is called with the threshold 0, the database will be empty and the complete instance will be generated: $M_{k,mset,0} = M_k$. If *base* is called with the maximum level of the lattice, then, the database will be the largest one built with all the smaller words, and we will thus obtain the smallest instances with this notion of lattice. However, the larger the threshold, the longer the generation time, and the smaller the SAT instance. With the maximal threshold, the generated instances will be a bit larger than with the previous improvement ($M_{k,all} \subseteq M_{k,mset,max}$), but the generation is significantly faster. For lack of space, we cannot give here the complete algorithms for generating this improved model.

Improvements based on Prefixes. Although faster to generate, the second model is still costly. We now propose a kind of weakening of Property 3, restricting its use to prefix.

Property 4 (Prefix). Let $w \in S$ be a word from the sample. Consider D^*_{w,q_i,q_j} the set of c_transitions defined by:

$$D^*_{w,q_i,q_j} = \bigvee_{l\in K, l\neq j} \left(\left(\bigvee_{d_u \in D^*_{u,q_i,q_l}} d_u \wedge \left(\bigvee_{d_v \in D^*_{v,q_l,q_i}} d_v \right) \right) \right)$$

if $w = u.v$, and $u \in S^-$; otherwise, $D^*_{w,q_i,q_j} = D_{w,q_i,q_j}$. Then,

$$\forall d \in D_{w,q_i,q_j} \setminus D^*_{w,q_i,q_j}, \neg d \vee \neg f_j$$

Hence, this property allows us to directly generate the reduced constraints, for negative or positive words, without comparing c_couples with a database.

Let $w = u_1 \dots u_n$ be a word from S such that $u_1 \in S^-$, $u_1.u_2 \in S^-$, and $u_1 \dots u_{n-1} \in S^-$ and for each $i < n$, there does not exist a decomposition $u_i = u'_i.u''_i$ such that $u_1 \dots u_{i-1}.u'_i \in S^-$. Then, if $w \in S^+$, using several times Property 4, Constraints (4), (5), and (6) can be replaced by Constraints (8), (9), and (10) where $l_0 = q_1$ and $N = [1, \dots, n]$:

$$\bigwedge_{i \in N, l_i \in K \setminus \{l_j | 1 \leq j < i\}} \quad \bigwedge_{i \in N, d_i \in D_{u_i, q_{l_i - 1}, q_l}} [(\neg aux_{w,l_1,\dots,l_n} \vee (d_1 \wedge \dots \wedge d_n \wedge f_j))] \quad (8)$$

$$\bigwedge_{i \in N, l_i \in K \setminus \{l_j | 1 \leq j < i\}} \quad \bigwedge_{i \in N, d_i \in D_{u_i, q_{l_i - 1}, q_l}} (aux_{w,l_1,\dots,l_n} \vee \neg d_1 \vee \dots \vee \neg d_n \vee \neg f_j) \quad (9)$$

$$\bigvee_{i \in N, l_i \in K \setminus \{l_j | 1 \leq j < i\}} \quad \bigvee_{i \in N, d_i \in D_{u_i, q_{l_i - 1}, q_l}} aux_{w,l_1,\dots,l_n} \quad (10)$$

Similarly, if $w \in S^-$, using several times Property 4, Constraints (7) can be replaced by Constraints (11):

$$\bigwedge_{i \in N, l_i \in K \setminus \{l_j | 1 \leq j < i\}} \quad \bigwedge_{i \in N, d_i \in D_{u_i, q_{l_i - 1}, q_l}} (\neg d_1 \vee \dots \vee \neg d_n \vee \neg f_j) \quad (11)$$

The number of clauses and variables generated for $w \in S^+$ is reduced to:

- $(|w| + 1).\big(\prod_{i=1}^{n}(k - i + 1)\big).k^{|w|-n}$ binary clauses for Constraints (8),
- $\big(\prod_{i=1}^{n}(k - i + 1)\big).k^{|w|-n}$ $(|w| + 2)$-ary clauses for Constraints (9),
- one clause of size $\big(\prod_{i=1}^{n}(k - i + 1)\big)$ for Constraint (10),
- and the number of auxiliary variables is reduced to $\big(\prod_{i=1}^{n}(k - i + 1)\big)$.

For $w \in S^-$, Constraints (11) are already in CNF and they correspond to $\big(\prod_{i=1}^{n}(k - i + 1)\big).k^{|w|-n}$ $(|w + 1|)$-ary clauses. Interestingly, these new counts of clauses (and more especially the factor $k - i + 1$ with $i = n$) also give us a lower bound for k: k must be greater than or equal to n, the number of nested prefixes in a word. This new improved model, that we call $M_{k,pref}$, is not much larger than $M_{k,mset}$, but it is significantly faster to generate.

Improvement order. We have defined various models for inference of NFA of size k that can be ordered by their sizes: $M_{k,all} \subseteq M_{k,mset,l\_max} \subseteq m_{k,pref} \subseteq M_k$. Note that $M_{k,mset,l}$ with $l \neq l\_max$, and $M_{k,pref}$ cannot be compared in the general case; their sizes depend on the instance, the number and size of prefixes, and on the given level l. In the next section, we compare these models not only in terms of instance size, but also in terms of generation and resolution time.

4 Experimental results

We suspect that, with respect to their generation time, the models are in reverse order of the order given above. Thus, we are interested in findng the best balance between three parameters: model size v.s. generation time + SAT solving time.

The experiments were carried out on a computing cluster with Intel-E5-2695 CPUs and 128 GB of memory. Running times were limited to 2 hours for the generation of SAT instances, and 3 hours to solve them. We used the Glucose [1] SAT solver with the default options.The benchmarks are based on the training set of the StaMinA Competition (http://stamina.chefbe.net). We selected 12 instances[1] with a sparsity $s \in \{12.5\%, 25\%, 50\%, 100\%\}$ and an alphabet size $|\Sigma| \in \{2, 5, 10\}$. For each of them, we limited the number of words to $|S^+| = |S^-| = 10$ and 20 for a maximal size of words equal to 7 and to $|S^+| = |S^-| = 20$ for a maximal size of words equal to 10. We generate CNF instances for different NFA sizes ($k \in \{3, 4, 5\}$). Consequently, we obtained 96 instances.

Table 1 presents a synthetic view of our experiments. The 4 first columns detail the instances: size of the NFA (k), size of the longest word ($|\omega|$), number of positive (and negative) words ($|S^+|$), and the model. The next columns provide average values over the 12 instances for the modeling time (T_{Model}), the number of variables ($\#Var$), the number of clauses ($\#Cl$), the solving time (T_{solve}), and the total modeling+solving time (T_{total}). We do not indicate the standard deviations but they are very close to zero. "-" indicates that no result was obtained before the time-out. From Table 1, we can draw some general conclusions about model improvements. As expected, $M_{k,all}$ always returns the smallest instances, and also the instances that Glucose solve the fastest. However, the generation time of these instances is very long. Thus, the total CPU time, i.e., generation + solving, is not the best. We can also see that when we increase the maximum length of words, this model does not permit to generate the instances in less than 2 hours (e.g., Table 1, for $k = 4$, $\omega = 10$, and $|S^+| = 20$). This model is thus tractable, but only for small instances, with short words and small samples.

$M_{k,mset,l_{max}}$ generates instances a bit larger than $M_{k,all}$. Consider the negative word $v = aaab$, and the positive word $w = ba$. $M_{k,all}$ uses some c_transitions of v to ignore some clauses of w that $M_{k,mset,l_{max}}$ will not detect. For example, a loop on aaa from v with the same transition in v is used in $M_{k,all}$ but not in $M_{k,mset,l_{max}}$. However, with the multiset data structure, we obtain a much faster generation of instances. The total time is thus more interesting with $M_{k,mset,l_{max}}$ than with $M_{k,all}$. The generation time of $M_{k,mset,l_{max}}$ is still very high, and its interest is not always significant. For large instances, not presented in the table, $M_{k,mset,l_{max}}$ could not be generated in less than 2 hours.

For $M_{k,pref}$, we can see that the generation time becomes reasonable, and much smaller than with the two previous improvements. Although smaller than with M_k, the instances are larger than with $M_{k,mset,l_{max}}$. In various experiments, this improvement was the best for the total time. Note also that our training samples are not so big, and that the number of prefixes is not so important. With larger $|S^+|$, for a fixed k, we should obtain better performances of $M_{k,pref}$.

We also tried two more improvements of $M_{k,mset,l}$ with $l \in \{1, 3\}$. The generation time of these models is logically faster than the ones of $M_{k,mset,l_{max}}$; as planned, the SAT instances are also larger. However, we were pleasantly surprised by the total time which is much better than for $M_{k,mset,l_{max}}$. The three

[1] We conserved the "official" name used during the Stamina Competition.

Table 1. Comparison on 96 generated instances between the models $m_{k,all}$, $m_{k,mset,l_{max}}$, $m_{k,mset,1}$, $m_{k,mset,3}$, and $m_{k,pref}$. Instances are grouped by size of the NFA (k), size of the longest word ($|\omega|$), and number of positive (and negative) words ($|S^+|$). For each line, obtained values are average on 12 instances.

k	$\|\omega\|$	$\|S^+\|$	Model	T_{model}	#Var.	#Cl.	T_{solve}	T_{total}
3	7	10	m_k	0.19	6742	61366	0.22	0.41
			$m_{k,all}$	0.68	4310	37789	0.14	0.82
			$m_{k,mset,l_{max}}$	0.17	4742	42020	0.14	0.31
			$m_{k,mset,1}$	0.18	5517	49484	0.16	0.34
			$m_{k,mset,3}$	0.17	4822	42850	0.14	0.31
			$m_{k,pref}$	0.18	6466	58645	0.2	0.38
		20	m_k	0.48	14830	134302	1.58	2.06
			$m_{k,all}$	2.62	8274	72569	1.64	4.26
			$m_{k,mset,l_{max}}$	0.42	8929	79030	1.22	1.64
			$m_{k,mset,1}$	0.45	11179	99811	1.39	1.84
			$m_{k,mset,3}$	0.46	9148	81188	1.27	1.73
			$m_{k,pref}$	0.43	13689	123390	1.71	2.14
	10	20	m_k	11	303519	3276974	397.68	408.68
			$m_{k,all}$	746.08	108417	1172093	79.98	826.06
			$m_{k,mset,l_{max}}$	9.87	122423	1313463	143.32	153.19
			$m_{k,mset,1}$	9.04	208610	2255307	233.97	243.01
			$m_{k,mset,3}$	9.06	134720	1443357	156.24	165.3
			$m_{k,pref}$	8.88	281408	3040802	270.04	278.92
4	7	10	m_k	1.46	45014	428775	10.3	11.76
			$m_{k,all}$	19.42	32956	302835	5.59	25.01
			$m_{k,mset,l_{max}}$	1.64	35362	328938	5.58	7.22
			$m_{k,mset,1}$	1.42	39242	369600	7.12	8.54
			$m_{k,mset,3}$	1.56	36048	336637	5.58	7.14
			$m_{k,pref}$	1.3	43655	414141	10.69	11.99
		20	m_k	3.93	100984	950473	83.55	87.48
			$m_{k,all}$	93.48	64428	588293	74.55	168.03
			$m_{k,mset,l_{max}}$	4.33	68041	628400	43.08	47.41
			$m_{k,mset,1}$	3.65	83463	777005	32.32	35.97
			$m_{k,mset,3}$	4.27	70720	653396	41.36	45.63
			$m_{k,pref}$	3.37	94829	887943	55.88	59.25
	10	20	m_k	187.59	4670833	53350566	2084.78	2272.37
			$m_{k,all}$	-	-	-	-	-
			$m_{k,mset,l_{max}}$	919.56	2304788	26010946	651	1570.56
			$m_{k,mset,1}$	173.82	3336332	38121787	658.7	832.52
			$m_{k,mset,3}$	375.34	2345238	26693196	107.13	482.47
			$m_{k,pref}$	162.45	4405201	50260648	1331.92	1494.37
5	7	10	m_k	6.61	201651	1962754	215.06	221.67
			$m_{k,all}$	232.47	161828	1526044	51.82	284.29
			$m_{k,mset,l_{max}}$	14.38	169816	1619550	171.92	186.3
			$m_{k,mset,1}$	7.24	182445	1759734	180.98	188.22
			$m_{k,mset,3}$	10.76	172660	1653301	210.1	220.86
			$m_{k,pref}$	6.26	196894	1908623	176.12	182.38
		20	m_k	19.37	456976	4382919	1268.18	1287.55
			$m_{k,all}$	1158.5	320689	2995308	631.14	1789.64
			$m_{k,mset,l_{max}}$	44.01	333799	3148787	1115.9	1159.91
			$m_{k,mset,1}$	20.24	398074	3784691	1192.49	1212.73
			$m_{k,mset,3}$	32.82	348339	3288509	1309.17	1341.99
			$m_{k,pref}$	16.54	434008	4141453	1203.36	1219.9

models $M_{k,pref}$, $M_{k,mset,1}$, and $M_{k,mset,3}$ are very difficult to compare. Depending on the instance, on the number and size of prefixes, on multiset inclusion, one can be better than the other. But for all the instances we tried, one of this 3 models was always the best of the 6 models, and they were better than M_k. Table 2 presents a focus on 2 specific instances (25_training and 35_training, both with $|\Sigma| = 5$) with a fixed value for k, $|\omega|$, and $|S^+|$. The columns correspond exactly to those of Table 1. For the first instance, we clearly see the order presented in Section 3 for instance sizes of improved models. We can also see the reverse order

Table 2. Focus on 2 specific instances.

k	$\|\omega\|$	$\|S^+\|$	Model	T_{model}	#Var.	#Cl.	T_{solve}	T_{total}
25_training								
5	7	20	m_k	16.72	378030	3748314	934.92	951.64
			$m_{k,all}$	854.47	271338	2626880	841.22	1695.69
			$m_{k,mset,l_{max}}$	48.71	275331	2678349	1538.06	1586.77
			$m_{k,mset,1}$	14.25	280899	2733709	895.92	910.17
			$m_{k,mset,3}$	23.67	277359	2696089	1147.41	1171.08
			$m_{k,pref}$	11.76	338880	3377124	687.79	699.55
35_training								
4	10	20	m_k	163.10	5253332	59504339	-	-
			$m_{k,all}$	-	-	-	-	-
			$m_{k,mset,l_{max}}$	676.22	4234500	47661301	2322.42	2998.64
			$m_{k,mset,1}$	209.86	4969772	56092438	-	-
			$m_{k,pref}$	184.56	5253332	59504339	7145.62	7330.18

in terms of generation time. When $|\Sigma|$ is small, the probability of having prefixes is higher than with larger vocabularies, and for this instance, $M_{k,pref}$ returns the best instance in terms of generation+solving time. For the second instance, $M_{k,all}$ could not be generated in less than 2 hours. M_k and $M_{k,mset,3}$ could be generated rather quickly, but could not be solved. $M_{k,pref}$ was even faster for generating the SAT instance. However, we see that there was not prefix in the training set (the size of instances of M_k and $M_{k,pref}$ are the same). The overhead for taking prefixes into account is rather insignificant (12% of generation time). Since the solving time was close to the timeout, the M_k instance did not succeed to be solved while the $M_{k,pref}$ instance succeeded (the small difference of 55 s., i.e., less than 0,8 %, is certainly due to clause order in the SAT instance). This instance shows that $M_{k,mset,l_{max}}$ can be the best model in terms of total time. This is due to the fact that there is no negative word being prefix of another word from S, and that the lattice is rather "wide", with a long branch. Hence, $M_{k,mset,l}$ is interesting when l is large for this training sample.

5 Conclusion

In the context of grammatical inference, we proposeed various model improvements for learning Nondeterministic Finite Automaton of size k from samples of words. Our base model, M_k, is a conversion from an INLP model [15]. The first improvement, $M_{k,all}$, leads to the smallest SAT instances, which are also solved quickly. However, generating this model is too costly. Thus, when problems grow (in terms of k, $|S|$, or length of words), $M_{k,all}$ cannot be generated anymore. We proposed a set of improvements based on multiset representation of words, $M_{k,mset,l}$. The generated SAT instances are a bit larger with the maximal level than with $M_{k,all}$, but generation is still costly. We thus defined a third improvement based on prefix. On average, the best balance between generation and solving time is obtained with $M_{k,pref}$, $M_{k,mset,1}$, or $M_{k,mset,3}$: the generation is rather light and the reductions are significant. The interest of our work is that, to our knowledge, we are the only ones working on CSP model improvements. It is very complicated to compare our results with previous works. Many works on this topics are only formal and experimental results are also difficult

to compare. For examples, the authors of [8, 9] focus on a parallel solver for optimizing k. In [10], experiments are based on samples issued from the Waltz-DB database [2] of amino acid sequences, i.e., all the words are of size 6, and there cannot be any prefix word: in the tests we performed, only anagrams could be used in multisets. Moreover, for all the 50 instances we tried issued from this database, the M_k model could be generated and solved in a reasonable time, without need of any model improvement.

In the future, we plan to hybridize $M_{k,mset,l}$ for small values of l with $M_{k,pref}$. The second idea is to simplify the work of the SAT solver and of the instance generation with simplified and incomplete training samples. We would then evaluate our SAT models with respect to the accurateness of the generated NFA on test set of words.

References

1. Audemard, G., Simon, L.: Predicting learnt clauses quality in modern SAT solvers. In: Proc. of IJCAI 2009. pp. 399–404 (2009)
2. Beerten, J., van Durme, J.J.J., Gallardo, R., Capriotti, E., Serpell, L.C., Rousseau, F., Schymkowitz, J.: WALTZ-DB: a benchmark database of amyloidogenic hexapeptides. Bioinform. **31**(10), 1698–1700 (2015)
3. Denis, F., Lemay, A., Terlutte, A.: Learning regular languages using rfsas. Theor. Comput. Sci. **313**(2), 267–294 (2004)
4. Dupont, P.: Regular grammatical inference from positive and negative samples by genetic search: the GIG method. In: Proc. of ICGI 94. LNCS, vol. 862, pp. 236–245. Springer (1994)
5. Garey, M.R., Johnson, D.S.: Computers and Intractability, A Guide to the Theory of NP-Completeness. W.H. Freeman & Company, San Francisco (1979)
6. Heule, M., Verwer, S.: Software model synthesis using satisfiability solvers. Empirical Software Engineering **18**(4), 825–856 (2013)
7. de la Higuera, C.: Grammatical Inference: Learning Automata and Grammars. Cambridge University Press (2010)
8. Jastrzab, T.: On parallel induction of nondeterministic finite automata. In: Proc. of ICCS 2016. Procedia Computer Science, vol. 80, pp. 257–268. Elsevier (2016)
9. Jastrzab, T.: Two parallelization schemes for the induction of nondeterministic finite automata on pcs. In: Proc. of PPAM 2017. LNCS, vol. 10777, pp. 279–289. Springer (2017)
10. Jastrzab, T.: A comparison of selected variable ordering methods for NFA induction. In: Proc. of ICCS 2019. LNCS, vol. 11540, pp. 741–748. Springer (2019)
11. Rossi, F., van Beek, P., Walsh, T. (eds.): Handbook of Constraint Programming. Elsevier Science, 1st edn. (2006)
12. Tomita, M.: Dynamic construction of finite-state automata from examples using hill-climbing. Proc. of the Fourth Annual Conference of the Cognitive Science Society pp. 105–108 (1982)
13. Tseitin, G.S.: On the Complexity of Derivation in Propositional Calculus, pp. 466–483. Springer Berlin Heidelberg, Berlin, Heidelberg (1983)
14. Vázquez de Parga, M., García, P., Ruiz, J.: A family of algorithms for non deterministic regular languages inference. In: Proc. of CIAA 2006. LNCS, vol. 4094, pp. 265–274. Springer (2006)
15. Wieczorek, W.: Grammatical Inference - Algorithms, Routines and Applications, Studies in Computational Intelligence, vol. 673. Springer (2017)

Guiding Representation Learning in Deep Generative Models with Policy Gradients

Luca Lach[1,3], Timo Korthals[2], Francesco Ferro[3], Malte Schilling[1] and Helge Ritter[1]

[1]Neuroinformatics Group, Bielefeld University, Bielefeld, Germany
llach@techfak.uni-bielefeld.de
[2]Cognitronics and Sensor Systems Group, Bielefeld Univeristy, Bielefeld, Germany
tkorthals@techfak.uni-bielefeld.de
[3]PAL Robotics, Barcelona, Spain
luca.lach@pal-robotics.com

Abstract. Variational Auto Encoder (VAE) provide an efficient latent space representation of complex data distributions which is learned in an unsupervised fashion. Using such a representation as input to Reinforcement Learning (RL) approaches may reduce learning time, enable domain transfer or improve interpretability of the model. However, current state-of-the-art approaches that combine VAE with RL fail at learning good performing policies on certain RL domains. Typically, the VAE is pre-trained in isolation and may omit the embedding of task-relevant features due to insufficiencies of its loss. As a result, the RL approach can not successfully maximize the reward on these domains. Therefore, this paper investigates the issues of joint training approaches and explores incorporation of policy gradients from RL into the VAE's latent space to find a task-specific latent space representation. We show that using pre-trained representations can lead to policies being unable to learn any rewarding behaviour in these environments. Subsequently, we introduce two types of models which overcome this deficiency by using policy gradients to learn the representation. Thereby the models are able to embed features into its representation that are crucial for performance on the RL task but would not have been learned with previous methods.

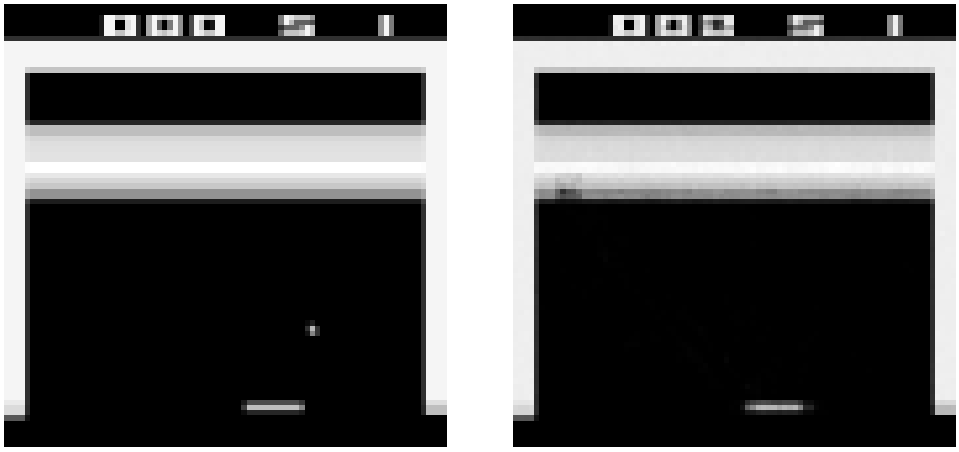
1 Introduction

Reinforcement Learning (RL) gained much popularity in recent years by outperforming humans in games such as *Atari* ([1], [2]), *Go* ([3], [2]) and *Starcraft 2* [4]. These results were facilitated by combining novel machine learning techniques such as deep neural networks [5] with classical RL methods. The RL framework has shown to be quite flexible and has been applied successfully in many further domains, for example, robotics [6], resource management [7] or physiologically accurate locomotion [8].

The goal of representation learning is to learn a suitable representation for a given application domain. Such a representation should contain useful information for a particular downstream task and capture the distribution of explanatory factors [9]. Typically, the choice of a downstream task influences the choice of method for representation learning. While Generative Adversarial Network (GAN) are frequently used for tasks that require high-fidelity reconstructions or generation of realistic new data, auto-encoder based methods have been more common in RL. Recently, many such approaches employed the Variational Auto Encoder (VAE) [10] framework which aims to learn a smooth representation of its domain. For a large number of RL environments, the usage of VAEs as a preprocesser improved sample efficiency and performance ([11], [12]).

Many of the current methods combining RL with representation learning follow the same pattern, called unsupervised pre-training [13]. First, they build a dataset of states from the RL environment. Second, they train the VAE on this static dataset and lastly train the RL mode using the VAE's representation. While this procedure generates sufficiently good results for certain scenarios, there are some fundamental issues with this method. Such an approach assumes that it is possible to collect enough data and observe all task-relevant states in the environment without knowing how to act in it. As a consequence, when learning to act the agent will only have access to

This work was supported by the European Union Horizon 2020 Marie Curie Actions under Grant 813713 NeuTouch.



(a) Input frame after pre-processing (b) Reconstruction of 1a

Fig. 1: A frame from Atari Breakout. The original image 1a was passed through a pre-trained VAE yielding the reconstruction 1b. Note the missing ball in the lower right hand corner.

a representation that is optimized for the known and visited states. As soon as the agent becomes more competent, it might experience novel states that have not been visited before and for which there is no good representation (in the sense that the experienced states are out of the original learned distribution and the mapping is not appropriate).

Another issue arises from the manner the representation is learned. Usually, the VAE is trained in isolation, so it decides what features are learned based on its own objective function and not on what is helpful for the downstream task. Mostly, such a model is tuned for good reconstruction. Without the information from the RL model, such a representation does not reflect what is important for the downstream task. As a consequence, the VAE might omit learning features that are crucial for good performance on the task because they appear negligible with respect to reconstruction ([14], Chapter 15, Figure 15.5). For example, small objects in pixel-space are ignored as they affect a reconstruction based loss only marginally. Thus, any downstream task using such a representation will have no access to information about such objects. A good example for such a task is Atari Breakout, a common RL benchmark. Figures 1a and 1b show an original Breakout frame and its reconstruction. While the original frame contains the ball in the lower right hand corner, this crucial feature is missing completely in the reconstruction.

We approach this issue through simultaneously learning representation and RL task, that is by combining the training of both models. As an advantage, this abolishes the need of collecting data before knowing the environment as it combines VAE and RL objectives. In consequence the VAE has an incentive to represent features that are relevant to the RL model. The main contributions of this paper are as follows: First we show, that using unsupervised pre-training on environments that have underrepresented task-relevant features fails to produce good RL policies. Second, we show that by jointly training representation and policy leads to a model that encodes task-relevant information and thus enabling significantly higher performing policies. This will be shown by comparing achieved rewards and by an analysis of the trained model and its representation.

2 Related Work

[15] explored Auto Encoder (AE) ([16]; [17]; [18]) as a possible pre-processor for RL algorithms. The main focus in their work was finding good representations for high dimensional state spaces that enables policy learning. As input, rendered images from the commonly used grid world environment were used. The agent had to manoeuvre through a discretized map using one of four discrete movement actions per timestep. It received a positive reward once reaching the goal tile and negative rewards elsewhere. The AE bottleneck consisted only of two neurons, which corresponds to the dimensionality of the environemnt's state. Fitted Q-Iteration (FQI) [19] was used to estimate

the Q-function, which the agent then acted ϵ-greedy upon. Besides RL, they also used the learned representation to classify the agents position given an encoding using a Multi-Layer Perceptron (MLP) [20]. For these experiments, they found that adapting the encoder using MLP gradients lead to an accuracy of 99.46 %. However, they did not apply this approach to their RL task.

A compelling example for separate training of meaningful representation is provided by [21] who proposed a framework called *DARLA*. They trained RL agents on the encoding of a β-VAE ([22]; [23]) with the goal of zero-shot domain transfer. In their approach, β-VAE and agent were trained separately on a source domain and then evaluated in a target domain. Importantly, source and target domain are similar to a certain extent and only differ in some features, e.g. a blue object in the source domain might be red in the target domain. During training of the β-VAE, the pixel-based reconstruction loss was replaced with a loss calculated in the latent space of a Denoising Auto Encoder (DAE) [24]. Thereby their approach avoids missing task relevant feature encodings at the cost of training another model. For one of their evaluation models, they allowed the RL gradients to adapt the encoder. Their results show that subsequent encoder learning improves performance of Deep Q-Learning (DQN) but decreases performance of Asynchronous Advantage Actor-Critic (A3C) [25].

[26] proposed a combination of VAE, Recurrent Neural Networks (RNN) [27] and a simple policy as a controller. They hypothesized that by learning a good representation of the environment and having the ability to predict future states, learning the policy itself becomes a trivial task. Like in most other models, the VAE was pre-trained on data collected by a random policy. Only the RNN and the controller were trained online. The compressed representation from the VAE was passed into a RNN in order to estimate a probability density for the subsequent state. The controller was deliberately chosen as a single linear layer and could thus be optimized with Covariance Matrix Adaptation - Evolution Strategy (CMA-ES) [28].

This work demonstrated how a VAE can provide a versatile representation that can be utilized in reinforcement learning. In addition, such an approach allows to predict the subsequent encoded state. While these findings encourage the usage of VAE in conjunction with RL, this is only possible in environments where the state space can be explored sufficiently by a random policy. However, if the policy can only discover important features after acquiring a minimal level of skill, sampling the state space using a random policy will not yield high-performing agents. Learning such features would only be possible if the VAE is continuously improved during policy training.

In the work of PlaNet [29], the authors also use a VAE to learn a latent state representation of a pixel input. Based on the learned representation, they use the Cross Entropy Method to learn various robotics control tasks. They refine this method in their subsequent publications Dreamer [30] and DreamerV2 [31] where the agent is trained purely on imagined trajectories from the VAE. Their works are similar to ours to the extent that they also continuously adapt the learned latent state representation. However their environments do not contain task relevant features that are underrepresented, hence their focus does not lie on training them.

Another interesting combination of VAEs and RL was recently proposed by [32], with their so called Action-Conditional Variational Auto-Encoder (AC-VAE). Their motivation for creating this model was to train a transparent, interpretable policy network. Usually, the β-VAEs decoder is trained to reconstruct the input based on the representation the encoder produced. In this work though, the decoders objective was to predict the subsequent state s_{t+1}. As input it got the latent space vector $\boldsymbol{z}$ combined with an action-mapping-vector, which is the action vector a_t with a zero-padding to match the latent spaces dimensionality. Inspecting the decoder estimates for s_{t+1} when varying one dimension of the latent space showed, that each dimension encoded a possible subsequent state that is likely to be encountered if the corresponding action from this dimension was taken. Unfortunately, the authors did not report any rewards they achieved on Breakout, hence it was not possible for us to compare model performances.

3 Combination of Reinforcement and Representation Learning Objectives

In this section, we will first revisit the fundamentals of RL and VAEs and discuss their different objective functions. Then, we propose a joint objective function that allows for joint training of both models using gradient descent based learning methods.

3.1 Reinforcement Learning with Policy Optimization

RL tries to optimize a Markov Decision Process (MDP) [33] that is given by the tuple $\langle \mathcal{S}, \mathcal{A}, r, p, \gamma \rangle$. $\mathcal{S}$ denotes the state space, $\mathcal{A}$ the action space and $p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ the environment's dynamics function that, provided a state-action pair, gives the state distribution for the successor state. $r : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$ is the reward and $\gamma \in [0, 1)$ the scalar discount factor. The policy $\pi_\theta(a|s)$ is a stochastic function that gives a probability distribution over actions for state s. θ denotes the policy's parameter vector which is typically subject to optimization. A trajectory $\tau = (s_0, a_0, ..., s_T, a_T)$ consisting of an alternating sequence of states and actions can be sampled in the environment, where T stands for the final timestep of the trajectory and $a_i \sim \pi_\theta(a_i|s_i)$.

The overarching goal of RL is to find a policy that maximizes the average collected reward over all trajectories. This can be expressed as the optimization problem $\max \mathbb{E}_{\tau \sim p(\tau)} \left[\sum_t r(s, a) \right]$, which can also be written in terms of an optimal policy parameter vector $\theta^* = \arg\max_\theta \mathbb{E}_{\tau \sim p(\tau)} \left[\sum_t r(s, a) \right]$. When trying to optimize the policy directly be searching for θ^*, policy optimization algorithms like A3C, Actor-Critic with Experience Replay (ACER) [34], Trust Region Policy Optimization (TRPO) [35] or Proximal Policy Optimization (PPO) [36] are commonly used. The fundamental idea behind policy optimization techniques is to calculate gradients of the RL objective with respect to the policy parameters:

$$\nabla_\theta J(\theta) = \mathop{\mathbb{E}}_{\tau \sim p(\tau)} \Big[\nabla_\theta \log \pi_\theta(\tau)\, r(\tau) \Big] \tag{1}$$

where we defined $\sum_{t=0}^{T} r(s, a) = r(\tau)$ for brevity. However, most policy optimization methods introduce heavy modifications to this vanilla gradient in order to achieve more stable policy updates. Throughout our work, we have used PPO as RL algorithm because it is quite sample efficient and usually produces stable policy updates. For an in-depth description of PPO, we refer to our A.1 or the original work [36].

3.2 Learning Representations using Variational Auto-Encoders

[10] introduced the VAE as a method to perform Variational Inference (VI) [37] using function approximators, e.g. deep neural networks. VI tries to approximate a distribution over the generative factors of a dataset which would otherwise involve calculating an intractable integral. The authors present an algorithm that utilizes the auto encoder framework, an unsupervised learning method which learns data encodings by reconstructing its input. Therefore, the input is first compressed until it reaches a given size and is afterwards decompressed to its original size. When using deep neural networks, these transformations can be achieved by using for example fully connected or convolutional layers. In order for the VAE to approximate a distribution over generative factors, the authors used the so called "reparametrization trick". It allows for gradient based optimization methods to be used in searching for the distribution parameters. For training the VAE, a gradient based optimizer tries to minimize the following loss:

$$\begin{gathered} \mathcal{L}^{VAE}(\boldsymbol{x}, \phi, \psi) = -D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,||\, p(\boldsymbol{z})) + \mathop{\mathbb{E}}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})} \big[\log p_\psi(\boldsymbol{x}|\boldsymbol{z}) \big] \\ \text{with } \boldsymbol{z} = l(\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\epsilon}) \text{ and } \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}) \end{gathered} \tag{2}$$

where D_{KL} denotes the Kullback-Leibler Divergence (KL) [38] of the approximated distribution over generative factors produced by the encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and some prior distribution $p(\boldsymbol{z})$. The expectation is often referred to as reconstruction loss that is typically calculated on a per-pixel basis. Lastly, $l(\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\epsilon})$ is a sampling function that is differentiable w.r.t. the distribution parameters, for example $\boldsymbol{z} = \boldsymbol{u} + \boldsymbol{\sigma}\boldsymbol{\epsilon}$.

3.3 Joint Objective Function

Combining both loss functions such that both models can be trained at the same time is rather straight-forward. Adding both individual losses and using an optimizer such as ADAM [39] to minimize them is sufficient to achieve joint training. During backpropagation, gradients from the policy and the VAE are combined in the latent space. Due to different topologies of the networks,

gradient magnitudes differ significantly. Therefore, we introduced the hyperparameter κ which can be used to either amplify or dampen the gradients and we arrive at the following loss:

$$\mathcal{L}^{\text{joint}} = \kappa \mathcal{L}^{\text{PG}}(\theta_k, \theta_{k-1}, \phi_k) + \mathcal{L}^{VAE}(\boldsymbol{x}, \phi, \psi, \beta) \tag{3}$$

where $\mathcal{L}^{\text{PG}}$ is some policy gradient algorithm's objective function. As mentioned before, we used PPO's loss $\mathcal{L}^{\text{PPO}}$ (equation 4 in the appendix).

4 Experiments

In order to test our model with the combined objective function given by Equation 3, we have used the well-known benchmark of Atari Breakout. This environment has several properties that make it appealing to use: it is easily understandable by humans, used often as a RL task and the conventional pre-trained methods fail at mastering it. The ball is the most important feature that is required to be encoded in order to perform well, is heavily underrepresented (approximately 0.1% of the observation space). Therefore, the VAE's incentive to encode it is very low whereas our model succeeds in encoding it. In the following, we compare the pre-trained approach to two different continuously trained models that use the loss from Equation 3.

4.1 Data Collection and Pre-Processing

The raw RGB image data produced by the environment has a dimensionality of $210 \times 160 \times 3$ pixels. We employ a similar pre-precessing as [1], but instead of cropping the grey-scaled frames, we simply resize them to 84×84 pixels. As we will first train models similar to those introduced in previous works with a pre-trained VAE, we needed to construct a dataset containing Breakout states. We used an already trained policy to collect a total of $25,000$ frames, the approximate equivalent of 50 episodes.

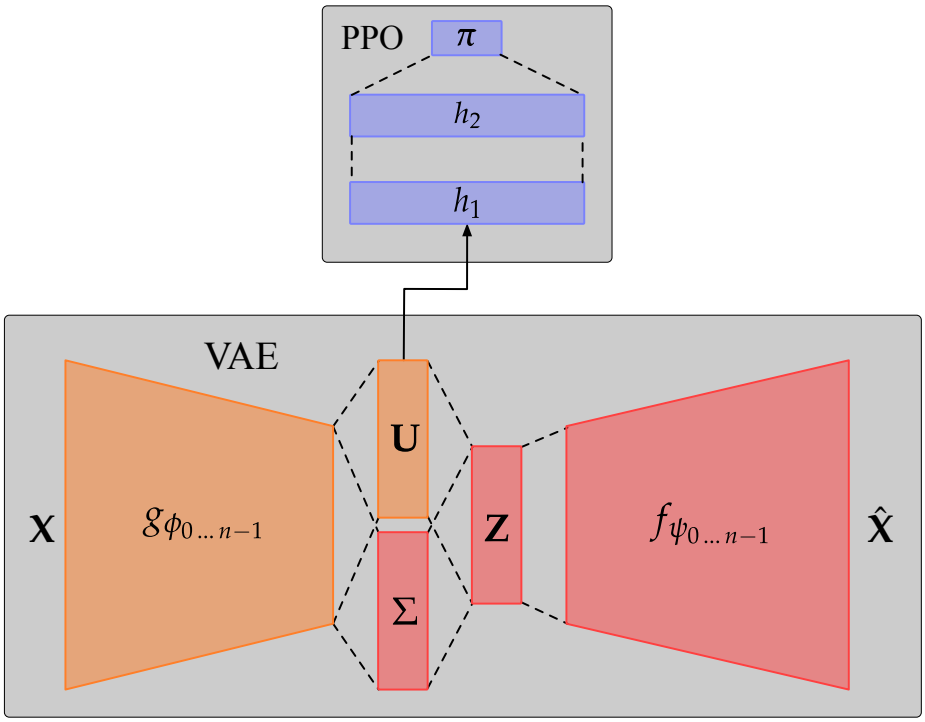


Fig. 2: Model combining PPO and a VAE. Depending on the model configuration, the colored parts are trained differently. $\mathbf{X}$ is the VAE's input and $\hat{\mathbf{X}}$ the reconstructions. PPO receives the mean vectors $\mathbf{U}$ as input and calculates a distribution over actions π. Note that we use capital letters in the VAE to emphasize that we pass n frames at the same time when a policy is trained.

4.2 Pre-training the Variational Auto-Encoder

Our first model is based on those of the previously introduced works which involve isolated pre-training the VAE on a static dataset. Figure 2 shows the individual parts of the complete training process. For the first model, PPO$^{\text{fixed}}$, the encoder and decoder (shown in orange and red) are pre-trained before policy training. During this phase, there is no influence from the RL loss. Once the VAE training is finished, the decoder shown in red in Figure 2 is discarded completely. Later during policy training, we use n instances of the same encoder with shared weights that receive a sequence of the last n frames as input. Stacking allows us to incorporate temporal information and for the policy to predict the ball's trajectory. By sharing the weights, we ensure that the resulting encodings originate from the same function. $\mathbf{U}$ then represents the concatenated encodings of the sequence.

Prior to policy training, we trained the VAE on the dataset we have collected before, with hyperparameters from Table 1. Once pre-training was finished, we discarded the decoder weights and used the stacked encoder as input for the policy MLP. The MLP was then trained 10M steps with hyperparameters from Table 2. During this training, the encoder weights were not changed by gradient updates anymore but remained fixed.

The second model we introduce is called PPO$^{\text{adapt}}$, which has the same structure and hyperparameters as the first model. For this model, we also train the VAE in isolation first, however the encoder weights are not fixed anymore during policy training. Gradients from the RL objective are back propagated through the encoder, allowing it to learn throughout policy training. We hypothesize that features that are important for policy performance can be incorporated in an already learned representation.

Figure 3 compares the median rewards of three rollouts with different random seeds for all models. PPO$^{\text{fixed}}$ was not once able to achieve a reward of 10 or higher, while PPO$^{\text{adapt}}$ steadily improved its performance with final rewards well over 50. The learning curve of PPO$^{\text{adapt}}$ shows that the model is able to learn how to act in the environment, whereas PPO$^{\text{fixed}}$ does not. The non-zero rewards from PPO$^{\text{fixed}}$ are similar to those of random agents in Breakout. From these results, we can assume that training the VAE in isolation on a static dataset for Breakout results in a deficient representation for RL. Therefore, using policy gradients to adapt an already learned representation can be beneficial in environments where the VAE fails to encode task-relevant features.

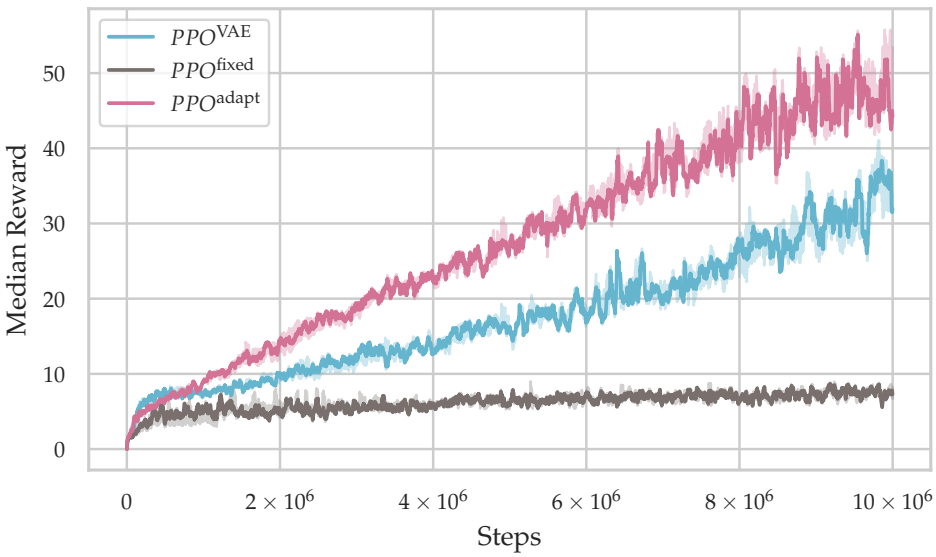


Fig. 3: Reward of the three proposed models across three random seeds each. PPO$^{\text{fixed}}$ is not able to achieve high rewards while the other two models consistently improve their performance.

4.3 Jointly Learning Representation and Policy

The last model we introduce, PPO$^{\text{VAE}}$, combines a complete VAE with a policy MLP that receives $\mathbf{U}$, the concatenated state encodings, as input. As opposed to the first two models, all weights are initialized randomly before policy training and the VAE is not pre-trained. For this procedure an already trained agent that gathers a dataset for the VAE beforehand is not necessary. The decoder is trained exactly as in the isolated setting, meaning its gradients are also only computed using the VAE's loss function. During backpropagation, the gradients coming from $\mathbf{Z}$ and h_1 are added together and passed through the encoder. This model has the same network configuration and hyperparameters as the first two, with the only difference that we also evaluated different values for κ from the joint loss 3 (see A.3). For the results reported here, we chose $\kappa = 20$. All hyperparameters can be found in Table 3.

By simultaneously training representation and policy, we expect the VAE to learn task-relevant features from the beginning of training. This assumption is supported by the learning curve shown in Figure 3, which compares PPO$^{\text{VAE}}$ to the previous two models. The curve shows a steady increase in reward over the course of training with PPO$^{\text{VAE}}$ achieving slightly higher rewards than PPO$^{\text{adapt}}$ in the beginning. This characteristic changes after less than 1M steps and from that point on PPO$^{\text{adapt}}$ consistently outperforms PPO$^{\text{VAE}}$. This difference in performance is likely attributed to the fact, that in PPO$^{\text{VAE}}$ the decoder is trained throughout the complete training. The gradients of PPO$^{\text{adapt}}$ can change the latent space without restrictions and they only optimize the RL objective. In PPO$^{\text{VAE}}$ however, gradients are also produced by the decoder that presumably do not contain information about the ball. Therefore PPO$^{\text{VAE}}$'s latent space is constantly changed by two different objectives, thus leading to lower rewards for the RL part.

4.4 Analyzing the Value Function Gradients

So far, the results imply that PPO$^{\text{VAE}}$ and PPO$^{\text{adapt}}$ do indeed learn encodings of the ball. One difficulty when analyzing the representation is, that the decoder still has no incentive to reconstruct the ball, even if it is present in the latent space. In a work that enhances DQN algorithm [40], the authors visualized the Jacobian of the value function w.r.t. the input images. These visualizations showed which features or regions from the input space are considered as important in terms of future reward. As we also learn a value function, we did the same and visualized what our model considered important and what not.

In Figure 4 we illustrate a pre-processed frame and added the values of the Jacobian to the blue channel if the were greater than the mean value of the Jacobian. Only visualizing above-mean Jacobian values removes some noise in the blue channel makes the images much easier to interpret and only highlights regions of high relevance. We can clearly see, that the Jacobian has high values at missing blocks as well as around the ball, meaning that these regions are considered to have high impact on future rewards. By visualizing the Jacobian we have confirmed that the policy gradients encourage the VAE to embed task-relevant features.

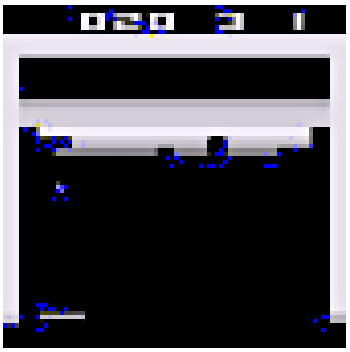


Fig. 4: The Jacobian of PPO's value function. Highlighted areas mean high importance in terms of future rewards. Note the high Jacobian values around the ball and the blocks.

5 Conclusion

This paper focused on the issue of pre-training VAEs with the purpose of learning a policy for a downstream task based on the VAE's representation. In many environments, the VAE has little to no incentive to learn task-relevant features if they are small in observation space. Another issue arises if the observation of these features depends on policy performance and as a result, they are underrepresented in a dataset sampled by a random agent. In both cases, fixing encoder weights during policy training prevents the VAE to learn these important features and policy performance will be underwhelming.

We carried out experiments on the popular RL benchmark Atari Breakout. The goal was to analyze whether policy gradients guide representation learning towards incorporating performance-critic features that a VAE would not learn on a pre-recorded dataset. First experiments confirmed, that the common pre-trained approach did not yield well-performing policies in this environment. Allowing the policy gradients to adapt encoder weights in two different models showed significant improvements in terms of rewards. With policy gradients guiding the learned representation, agents consistently outperformed those that were trained on a fixed representation.

Out work verifies the fundamental issue with pre-trained representations and provides methods that overcome this issue. Nonetheless, future work can still explore a variety of improvements to our models. For once, training not only the encoder but also the decoder with RL gradients can improve interpretability of the VAE and enable it to be used as a generator again that also generates task-relevant features. Another direction is to impose further restrictions on the latent space during joint training of VAE and policy. The goal there would be to maintain the desired latent space characteristics of VAEs while still encoding task-relevant features.

References

1. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.
2. Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *arXiv preprint arXiv:1911.08265*, 2019.
3. David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.
4. Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.
5. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.
6. OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.
7. Hongzi Mao, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula. Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, pages 50–56. ACM, 2016.
8. Łukasz Kidziński, Sharada Prasanna Mohanty, Carmichael F Ong, Zhewei Huang, Shuchang Zhou, Anton Pechenko, Adam Stelmaszczyk, Piotr Jarosik, Mikhail Pavlov, Sergey Kolesnikov, et al. Learning to run challenge solutions: Adapting reinforcement learning methods for neuromusculoskeletal environments. In *The NIPS'17 Competition: Building Intelligent Systems*, pages 121–153. Springer, 2018.
9. Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.
10. Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
11. Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018.
12. Karol Gregor, Danilo Jimenez Rezende, Frederic Besse, Yan Wu, Hamza Merzic, and Aaron van den Oord. Shaping belief states with generative environment models for rl. *Advances in Neural Information Processing Systems*, 32:13475–13487, 2019.
13. Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. Why does unsupervised pre-training help deep learning? In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 201–208. JMLR Workshop and Conference Proceedings, 2010.
14. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
15. Sascha Lange and Martin A. Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *IJCNN*, pages 1–8. IEEE, 2010.
16. Yann Lecun. *PhD thesis: Modeles connexionnistes de l'apprentissage (connectionist learning models)*. Universite P. et M. Curie (Paris 6), June 1987.
17. Herve Bourlard and Y Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59:291–4, 02 1988.

18. Geoffrey E Hinton and Richard S Zemel. Autoencoders, minimum description length and helmholtz free energy. pages 3–10, 1994.
19. Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556, 2005.
20. David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.
21. Irina Higgins, Arka Pal, Andrei Rusu, Loic Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1480–1490. JMLR. org, 2017.
22. Irina Higgins, Loic Matthey, Xavier Glorot, Arka Pal, Benigno Uria, Charles Blundell, Shakir Mohamed, and Alexander Lerchner. Early visual concept learning with unsupervised deep learning. *arXiv preprint arXiv:1606.05579*, 2016.
23. Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 3, 2017.
24. Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.
25. Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. pages 1928–1937, 2016.
26. David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 2450–2462. Curran Associates, Inc., 2018.
27. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
28. Nikolaus Hansen. The cma evolution strategy: a comparing review. In *Towards a new evolutionary computation*, pages 75–102. Springer, 2006.
29. Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pages 2555–2565. PMLR, 2019.
30. Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.
31. Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.
32. John Yang, Gyuejeong Lee, Simyung Chang, and Nojun Kwak. Towards governing agents efficacy: Action-conditional β-vae for deep transparent reinforcement learning. volume 101 of *Proceedings of Machine Learning Research*, pages 32–47, Nagoya, Japan, 17–19 Nov 2019. PMLR.
33. Richard Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, pages 679–684, 1957.
34. Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*, 2016.
35. John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. pages 1889–1897, 2015.
36. John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
37. Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.
38. Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.
39. Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
40. Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003, 2016.
41. Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.
42. John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.
43. OpenAI Spinning Up - PPO. Spinning up explanation of ppo, 2018.
44. Sham M Kakade. A natural policy gradient. pages 1531–1538, 2002.

A Appendix

A.1 Stable Policy Learning with Proximal Policy Optimization

Most actor-critic algorithms successfully reduce the variance of the policy gradient, however they show high variance in policy performance during learning and are at the same time very sample inefficient. Natural gradient ([41]) methods such as TRPO from [35] greatly increase sample efficiency and learning robustness. Unfortunately, they are relatively complicated to implement and are computationally expensive as the require some second order approximations. PPO ([36]) is a family of policy gradient methods that form pessimistic estimates of the policy performance. By clipping and therefore restricting the policy updates, PPO prohibits too large of a policy change as they have been found to be harmful to policy performance in practice. PPO is often combined with another type of advantage estimation ([42]) that produces high accuracy advantage function estimates.

We define the PPO-Clip objective is defined as

$$J^{\text{PPO}}(\theta_k, \theta_{k-1}) = \mathbb{E}\left[\min\Big(o(\theta)A^{\pi_{\theta_k}}(s,a),\ \text{clip}\Big(o(\theta), 1-\epsilon, 1+\epsilon\Big)A^{\pi_{\theta_k}}(s,a)\Big)\right] \quad \text{s.t. } \delta_{\text{MB}} < \delta_{\text{target}} \tag{4}$$

where $o(\theta) = \frac{\pi_{\theta_k}(a|s)}{\pi_{\theta_{k-1}}(a|s)}$ denotes the probability ratio of two policies.

This objective is motivated by the hard KL constraint that TRPO enforces on policy updates. Should a policy update result in a policy that deviates too much from its predecessor, TRPO performs a line search along the policy gradient direction that decreases the gradient magnitude. If the constraint is satisfied during the line search, the policy is updated using that smaller gradient step. Otherwise the update is rejected after a certain number of steps. This method requires to calculate the second order derivative of the KL divergence, which is computationally costly. PPO uses its clipping objective to implicitly constrain the deviation of consecutive policies. In some settings, PPO still suffers from diverging policy updates ([43]), so we included a hard KL constrained on policy updates. The constraint can be checked after each mini-batch update analytically and is therefore not very computationally demanding.

PPO extends the policy gradient objective function from [44]. With the probability ratio $o(\theta)$, we utilize importance sampling in order to use samples collected with any policy to update our current one. Thereby we can use samples more often than in other algorithms, making PPO more sample efficient. Using importance sampling, we still have a correct gradient estimate. Combining the new objective with actor-critic methods yields algorithm 1. K denotes the number of optimization epoch per set of trajectories and B denotes the mini-batch size. In the original paper, a combined objective function is also given with:

$$\mathcal{L}^{\text{PPO}}(\theta_k, \theta_{k-1}, \phi_k) = \mathbb{E}\left[c_1 J^{\text{PPO}}(\theta_k,\theta_{k-1}) - c_2\mathcal{L}^{V^{\pi_\theta}}(\phi_k) + \mathcal{H}(\pi_{\theta_k})\right] \quad \text{s.t. } \delta_{\text{MB}} < \delta_{\text{target}} \tag{5}$$

where $\mathcal{H}(\pi_{\theta_k})$ denotes the policy entropy. Encouraging the policy entropy not to decrease too much prohibits the policy from specializing on one action. As discussed in [43], there are two cases for $J^{\text{PPO}}(\theta_k, \theta)$: either the advantage function was positive or negative. In case the advantage is positive, it can be written as:

$$J^{\text{PPO}}(\theta_k, \theta) = \mathbb{E}\left[\min\left(o(\theta), (1+\epsilon)\right)A^{\pi_{\theta_k}}(s,a)\right] \tag{6}$$

$A^{\pi_{\theta_k}}(s,a) > 0$ indicates that the action yields higher reward than other actions in this state, hence we want its probability $\pi_{\theta_k}(a|s)$ to increase. This increase is clipped to $(1+\epsilon)$ once $\pi_{\theta_k}(a|s) > \pi_{\theta_{k-1}}(a|s)(1+\epsilon)$. Note however, that updates that would worsen policy performance are neither clipped nor bound. If the the advantage is negative, it can be expressed as:

$$J^{\text{PPO}}(\theta_k, \theta) = \mathbb{E}\left[\max\left(o(\theta), (1-\epsilon)\right)A^{\pi_{\theta_k}}(s,a)\right] \tag{7}$$

This equation behaves conversely to 6: $A^{\pi_{\theta_k}}(s,a) < 0$ indicates that we chose a suboptimal action, thus we want to decrease its probability. Once $\pi_{\theta_k}(a|s) < \pi_{\theta_{k-1}}(a|s)(1-\epsilon)$, the max bounds the magnitude by which the action's probability can be decreased.

Algorithm 1 Proximal Policy Optimisation with KL constraint

Initialize policy parameters θ_0 and value function parameters ϕ_0
for $k = 0, 1, 2, ...$ **do**
    Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ with π_{θ_k} and compute $\hat{R}_t$
    $\delta_{\text{MB}} \leftarrow 0$
    for $0, 1, 2, ...K$ **do**
        for each mini-batch of size B in $\{\tau_i\}$ **do**
            Update the policy by maximizing the PPO-Clip objective 4
            Minimize $\mathcal{L}^{V^{\pi_\theta}}$ on the mini-batch
        end for
    end for
    if $\delta_{\text{MB}} > \delta_{\text{target}}$ **then**
        $\theta_{k+1} = \theta_k$
    end if
end for

A.2 Hyperparameter Tables

Parameter	**Value**
epochs	100
batch size	128
input size	$(84, 84, 1)$
optimizer	ADAM
learning rate	1×10^{-4}
encoder	Conv2D $32 \times 4 \times 4$ (stride 2) - $64 \times 4 \times 4$ (stride 2) - FC 512 (ReLU)
latents	20 (linear)
decoder	FC 512 (ReLU) - $64 \times 4 \times 4$ (stride 2) - $32 \times 4 \times 4$ (stride 2) Conv2D Transpose

Table 1: Hyperparameter table for VAE training on Breakout

Parameter	**Value**
timesteps	1×10^7
environments	16
batch size	32
$t_{\max}$	2048
K	10
c_1	1.0
c_2	0.5
c_3	0.0
γ	0.99
λ	0.95
network	FC 64 (tanh) - FC 64 (tanh)
optimizer	ADAM
learning rate	3×10^{-4}

Table 2: Policy hyperparameters of $\text{PPO}^{\text{fixed}}$ and $\text{PPO}^{\text{adapt}}$

Parameter	**Value**
timesteps	1×10^7
environments	16
batch size	32
$t_{\max}$	2048
K	10
c_1	1.0
c_2	0.5
c_3	0.0
γ	0.99
λ	0.95
network	FC 64 (tanh) - FC 64 (tanh)
optimizer	ADAM
learning rate	3×10^{-4}
κ	(1, 10, 20)

Table 3: Policy hyperparameter table of PPO^{VAE}

A.3 Choosing appropriate values for κ

In Equation 3, we introduced the hyperparameter κ to balance VAE and PPO gradients. We found empirically, that tuning κ is straight forward and requires only few trials. In order to simplify the search for κ, one can evaluate gradient magnitudes of the different losses at the point where they are merged at $\mathbf{U}$. Our experiments showed PPO's gradients to be significantly smaller, thus scaling up the loss function was appropriate. This will likely differ if the networks are configured differently. Increasing κ from 1 to 10 led to considerably higher rewards, however the difference in performance was small when increasing κ further to 20. Therefore, we chose $\kappa = 20$ in our reported model performances.

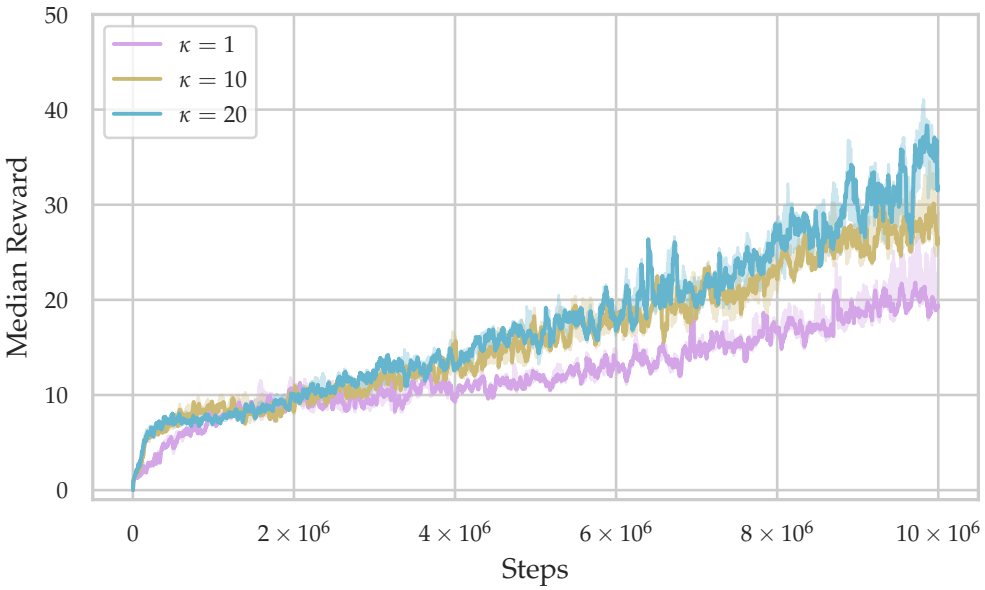


Fig. 5: Performance comparison of PPO^{VAE} with different values for κ

OPTIMAL DEMAND RESPONSE SCHEDULING OF A RESIDENTIAL MICROGRID

Mohamed Saâd EL HARRAB [1] **Michel NAKHLA** [1]

[1] Centre de Gestion Scientifique (CGS),
UMR CNRS i^3 9217, MINES ParisTech,
PSL Research University

{Mohamed-Saad.EL_HARRAB, Michel.NAKHLA}@mines-paristech.fr

ABSTRACT

The electrical system must handle increasing production from renewable sources that are difficult to predict, highly variable and not controllable. This shift in the electrical paradigm makes power grid operation, and therefore the exercise of supply/demand balance, increasingly complex. Microgrids (MG) enable a more flexible management of the grid. These intelligent bidirectional systems allow to reach new sources of flexibility from consumers using Demand Response (DR).

Based on Stochastic Optimization and Deep Learning approaches, we propose an optimal demand response scheduling under load uncertainty in a residential Microgrid. Our approach is based on load forecasting techniques, clustering and pattern recognition procedures and a stochastic optimal power flow scheduling algorithm.

For each household in the Microgrid, we forecast the day ahead load profile and perform pattern recognition and clustering in order to identify transferable loads and then we evaluate household's micro-flexibilities potential. Thus the flexibility potential of the whole Microgrid will be the aggregation of household's micro-flexibilities.

The Microgrid optimal schedule strategy is obtained by maximizing the Microgrid operator's DR payoff while satisfying the load demand and user's comfort constraints. Simulation results show that the proposed DR scheduling is beneficial to both service provider's and prosumer.

Keywords: Microgrid ; Smartgrid ; Demand Response ; Optimal scheduling ; Load management ; Optimization.

SELECTED REFERENCES

[1] S. Chapaloglou *et al.*, « Smart energy management algorithm for load smoothing and peak shaving based on load forecasting of an island's power system », *Applied Energy*, vol. 238, p. 627-642, mars 2019, doi: 10.1016/j.apenergy.2019.01.102.

[2] S. Rahim, S. A. Khan, N. Javaid, N. Shaheen, Z. Iqbal, et G. Rehman, « Towards Multiple Knapsack Problem Approach for Home Energy Management in Smart Grid », in *2015 18th International Conference on Network-Based Information Systems*, sept. 2015, p. 48-52, doi: 10.1109/NBiS.2015.11.

[3] M. R. Staats, P. D. M. de Boer-Meulman, et W. G. J. H. M. van Sark, « Experimental determination of demand side management potential of wet appliances in the Netherlands », *Sustainable Energy, Grids and Networks*, vol. 9, p. 80-94, mars 2017, doi: 10.1016/j.segan.2016.12.004.

[4] Jiang Chen, Bo Yang, et Xinping Guan, « Optimal demand response scheduling with Stackelberg game approach under load uncertainty for smart grid », in *2012 IEEE Third International Conference on Smart Grid Communications (SmartGridComm)*, nov. 2012, p. 546-551, doi: 10.1109/SmartGridComm.2012.6486042.

[5] V. V. S. N. M. Vallem et A. Kumar, « Optimal energy dispatch in microgrids with renewable energy sources and demand response », *International Transactions on Electrical Energy Systems*, vol. 30, n° 5, p. e12328, 2020, doi: https://doi.org/10.1002/2050-7038.12328.

[6] N. I. Nwulu et X. Xia, « Optimal dispatch for a microgrid incorporating renewables and demand response », *Renewable Energy*, vol. 101, p. 16-28, févr. 2017, doi: 10.1016/j.renene.2016.08.026.

[7] S. Choi, S. Park, D. Kang, S. Han, et H. Kim, « A microgrid energy management system for inducing optimal demand response », in *2011 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, oct. 2011, p. 19-24, doi: 10.1109/SmartGridComm.2011.6102317.

[8] Jiang Chen, Bo Yang, et Xinping Guan, « Optimal demand response scheduling with Stackelberg game approach under load uncertainty for smart grid », in *2012 IEEE Third International Conference on Smart Grid Communications (SmartGridComm)*, nov. 2012, p. 546-551, doi: 10.1109/SmartGridComm.2012.6486042.

A Bayesian Optimisation Approach for Multidimensional Knapsack Problem

Hanyu Gu[0000−0003−2035−2583], Alireza Etminaniesfahani, and Amir Salehipour

School of Mathematical and Physical Sciences, University of Technology Sydney
15 Broadway Ultimo NSW 2007, Australia
Hanyu.Gu@uts.edu.au, Alireza.Etminaniesfahani@student.uts.edu.au,
amir.salehipour@uts.edu.au

Abstract. This paper considers the application of Bayesian optimisation to the well-known multidimensional knapsack problem which is strongly NP-hard. For the multidimensional knapsack problem with a large number of items and knapsack constraints, a two-level formulation is presented to take advantage of the global optimisation capability of the Bayesian optimisation approach, and the efficiency of integer programming solvers on small problems. The first level makes the decisions about the optimal allocation of knapsack capacities to different item groups, while the second level solves a multidimensional knapsack problem of reduced size for each item group. To accelerate the Bayesian optimisation guided search process, various techniques are proposed including variable domain tightening, initialisation by the Genetic Algorithm, and optimisation landscape smoothing by local search. Computational experiments are carried out on the widely used benchmark instances with up to 100 items and 30 knapsack constraints. The preliminary results demonstrate the effectiveness of the proposed solution approach.

Keywords: Bayesian optimisation · Multidimensional knapsack problem · Meta-heuristics.

1 Introduction

The Bayesian optimisation (BO) is a powerful machine learning based method for the optimisation of expensive black-box functions, which typically only allow point-wise function evaluation [23, 22]. Although BO has been widely used in the experimental design community since the 1990s [15, 13], it is not until the last decade that BO has become extremely popular in the machine learning community as an efficient tool for tuning hyper-parameters in various algorithms, e.g., deep learning [5, 7], natural language processing [29], and preference learning [10]. The BO is also embraced by new areas such as robotics [16], automatic control [1], and pharmaceutical product development [21].

The Multidimensional Knapsack Problem (MKP) is an extension of the classic Knapsack Problem (KP). It comprises of n items and m knapsacks with limited capacities. Each item contributes a certain amount of profit if selected

and consumes "resources" simultaneously in each knapsack. The MKP aims for a subset of items that achieves the highest total profit while abiding by the capacities of all knapsacks. The MKP is a well-known, and strongly NP-hard combinatorial optimisation problem, and has found applications in many practical areas involving resource allocation [11, 17]. In spite of the tremendous progress made in exact solution techniques, many instances from the widely used Chu and Beasley MKP test set [4] cannot be solved to optimality [8, 12, 28], especially when the number of knapsacks is large. The best known solutions on the hard instances are all obtained by specialised meta-heuristics which require exorbitant computation time [24, 25, 27, 3, 6]. The simplicity of problem statement and computational hardness makes the MKP an ideal test bed for new solution ideas and techniques [14, 18].

The BO encounters insurmountable issues to solve the MKP. Firstly, the BO is designed to solve problems with simple feasible set of continuous variables [9], while the MKP has only binary variables with many knapsack constraints. Whereas a lot of efforts have been committed to consider feasible set with combinatorial structures, all the reported computational studies investigated problems with just a few dozen categorical/integer/binary variables [2, 19]. Secondly, the BO is only efficient for low dimensional problems with less than 20 variables, while the MKP can have hundreds of binary variables. Although the BO with random embedding can solve problems with billions of variables, it relies on the "low effective dimensionality" which can be an issue for MKP [26]. Finally, the MKP has a linear function which is "cheap" to calculate, which makes it hard for the BO to compete with other meta-heuristic and artificial intelligence algorithms.

Based on the idea of divide and conquer, a novel two-level model for MKP (TL-MKP) is proposed in this paper to take advantage of the special structure of MKP, i.e., the number of items (variables) is much larger than the number of knapsacks (constraints). In particular, the items are divided into groups, and the knapsack capacities allocated to each group are determined by the first level, or master problem, of the TL-MKP. With assigned knapsack capacities, each group can be solved as a MKP of reduced size in the second-level of TL-MKP, or subproblem. It is shown in Section 2 that the master problem has a non-continuous, multi-modal, and expensive to evaluate objective function with simple feasible set, which is suitable for the application of BO. Since the subproblem has a much smaller number of binary variables, it can be efficiently solved to optimality with commercial integer programming solvers.

It is essential to incorporate prior knowledge in the BO, which was designed to be a black-box global optimisation method. Two novel techniques are presented in this paper to make use of the information provided by mathematical programming solver and meta-heuristics. Indeed, when a good solution is known, e.g., by using other meta-heuristics, an efficient heuristic is proposed in this paper to tighten the domain bounds of the master problem in the TL-MKP. Inspired by the simulation approach used in robotics control algorithms to initialise the BO [20], the Genetic Algorithm (GA) is used in this paper to generate initial

trial points for the BO. To take advantage of the linear structure of the objective function of MKP, the GA is run on the MKP instead of the master problem of TL-MKP. These techniques can significantly accelerate the search process of BO as demonstrated by the computational experiments in this paper.

The paper is organised as follows. The novel two-level model for MKP is presented with discussion of the properties of the master problem in Section 2. The BO based optimisation approach and some acceleration techniques are described in Section 3. The implementation details are discussed in Section 4. Computational results are presented in Section 5. The conclusion is given in Section 6.

2 Two-level Model for MKP

Given m knapsacks with capacities b_i, $i = 1, \ldots, m$, and a set of n items $I = \{1, 2, \ldots, n\}$, each item j requires a resource consumption of $a_{i,j}$ units in the i-th knapsack, $i = 1, \ldots, m$, and yields c_j units of profit upon inclusion, $j = 1, \ldots, n$. The goal is to find a subset of items that yields maximum profit, denoted by z^*, without exceeding the knapsack capacities. The MKP can be defined by the following integer linear programming model:

$$\text{(MKP)} \qquad z^* = \max\{c^T x : Ax \leq b, x \in \{0,1\}^n\}, \tag{1}$$

where $c = [c_1, c_2, \ldots, c_n]^T$ is an n-dimensional vector of profits, $x = [x_1, x_2, \ldots, x_n]^T$ is an n-dimensional vector of 0-1 decision variables indicating whether an item is included or not, $A = [a_{i,j}]$, $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$ is an $m \times n$ coefficient matrix of resource requirements, and $b = [b_1, b_2, \ldots, b_m]^T$ is an m-dimensional vector of resource capacities. It is further assumed that all parameters are non-negative integers.

Assume the items are divided into two groups, i.e., $I = I_1 \cup I_2$, and $I_1 \cap I_2 = \emptyset$. Each group is formulated as a MKP with profit vector $c^i = c_{I_i}$, resource requirement matrix $A^i = A_{I_i}$, and capacity vector $b^i \in \mathbb{R}^m$. The two groups share the capacities of the m knapsacks, i.e.,

$$b^1 + b^2 = b \tag{2}$$

The first level of the TL-MKP (the two-level model for MKP), or the master problem is defined as

$$\text{(L1-MKP)} \qquad f^* = \max\{f(t) : t \in \mathbb{R}^m, 0 \leq t \leq b\}, \tag{3}$$

where

$$f(t) = z_1^*(t) + z_2^*(b - t) \tag{4}$$

is calculated by solving the second level of the TL-MKP, or subproblems:

$$\text{(L2-MKP)} \qquad z_i^*(u) = \max\{c^i x^i : A^i x^i \leq u, x^i \in \{0,1\}^{|I_i|}\}, \quad i = 1, 2 \tag{5}$$

Since each solution of the TL-MKP can be easily converted to a solution to the MKP with the same objective value, and each solution of the MKP can be used to define a value of t for the master problem of TL-MKP (3), the following proposition holds.

Proposition 1. *$(t^* = A^1 x^{1*}, x^{1*}, x^{2*})$ is an optimal solution of TL-MKP if and only if x^*, defined as $x^*_{N_1} = x^{1*}$ and $x^*_{N_2} = x^{2*}$, is an optimal solution of MKP. Furthermore, $f^* = z^*$.*

Example 1. Consider an instance of MKP with three items and one knapsack, where $c = [1, 2, 3]$, $A = [1, 2, 3]$, and $b = 4$. The two groups are $I_1 = \{1, 2\}$ and $I_2 = \{3\}$. It is straightforward to show that the first level objective function is

$$f(t) = \begin{cases} 3 & t \in [0, 1) \\ 4 & t = 1 \\ 1 & t \in (1, 2) \\ 2 & t \in [2, 3) \\ 3 & t \in [3, 4] \end{cases}.$$

Example 2. Consider an instance of MKP with 20 items and two knapsacks. The two groups have the same number of items. Fig. 1 shows the contour graph of the first level objective function $f(t)$. The optimal value is equal to 75.

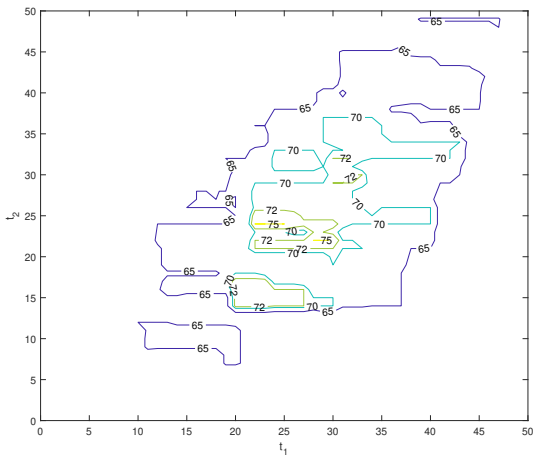


Fig. 1. Contour of the first level objective function $f(t)$; $t_1(t_2)$ is the capacity allocated to group 1 from knapsack 1 (2).

Examples (1) and (2) clearly demonstrate that the objective function of the master problem in TL-MKP is non-continuous, and can have many local optima. Although the subproblems have much smaller sizes, they are still more expensive to evaluate than the linear function of MKP.

It can be observed that $f(t)$ is not differentiable when at least one knapsack has no slack capacity in one of the subproblems. That leads to the following proposition,

Proposition 2. *$f(t)$ is differentiable almost everywhere in the sense of Lebesque measure with $f'(t) = 0$.*

Although $f(t)$ is differentiable almost everywhere, the derivative is constantly zero and consequently, useless for the design of optmisation algorithms.

3 Bayesian Optimisation and Acceleration

The BO is a promising option to deal with the challenges presented by the master problem of TL-MKP such as no closed form, non-continuity, multiple local optima, absense of useful derivatives, and high cost of function evaluation. In this section, the basic principles of BO are described first [9], then followed by techniques to incorporate prior knowledge to accelerate the search process.

The BO builds a probabilistic model for the unknown $f(t)$ of the master problem of TL-MKP. In particular, $f(t)$ is assumed to be drawn from a Gaussian process (GP), which is determined by a mean function $\mu_0 : \mathbb{R}^m \to \mathbb{R}$, and a positive definite covariance function $k_0 : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$, also known as the kernel of the GP. The BO sequentially generates points to evaluate within the feasible region of TL-MKP. Assume that n points have been evaluated with observations $\mathcal{D}_n = \{(t^1, f(t^1)), (t^2, f(t^2)), \ldots, (t^n, f(t^n))\}$. Using Bayes' rule, the conditional distribution of $f(t)$ is derived as a Normal distribution:

$$P(f(t)|D_n, t) = \mathcal{N}(\mu_n(t), \sigma_n^2(t)) \tag{6}$$

$$\mu_n(t) = \Sigma_0(t, t_{1:n})\Sigma_0(t_{1:n}, t_{1:n})^{-1}(f(t_{1:n}) - \mu_0(t_{1:n})) + \mu_0(t) \tag{7}$$

$$\sigma_n^2(t) = k_0(t, t) - \Sigma_0(t, t_{1:n})\Sigma_0(t_{1:n}, t_{1:n})^{-1})\Sigma_0(t_{1:n}, t) \tag{8}$$

where $f(t_{1:n}) = [f(t^1), \ldots, f(t^n)]^T$, $\mu_0(t_{1:n}) = [\mu_0(t^1), \ldots, \mu_0(t^n)]^T$, and

$$\Sigma_0(t_{1:n}, t_{1:n}) = \begin{pmatrix} k_0(t^1, t^1) & \cdots & k_0(t^1, t^n) \\ \vdots & \ddots & \vdots \\ k_0(t^n, t^1) & \cdots & k_0(t^n, t^n) \end{pmatrix}.$$

The BO selects the next most promising point to evaluate, i.e., t^{n+1}, by optimising an acquisition function, which balances exploration (uncertainty $\sigma_n(t^{n+1})$ is large) against exploitation (objective expected value $\mu_n(t^{n+1})$ is large). Different types of acquisition function have been proposed in the literature, while the most commonly used is *Expected Improvement* (EI). The EI acquisition function is defined as

$$\mathbf{EI}_n(t) = E_n(max(f(t) - max_{i=1}^n f(t^i), 0)), \tag{9}$$

where $E_n(\cdot)$ is the expectation taken under the posterior distribution (6).

The next point to evaluate is selected as

$$t^{n+1} = \text{argmax}_t \mathbf{EI}_n(t). \tag{10}$$

With new point $(t^{n+1}, f(t^{n+1}))$, the conditional probability of $f(t)$ can be updated according to (6), and the iterative process stops when a sampling budget is reached.

3.1 Variable Domain Tightening

The efficiency of BO depends on the size and dimensionality of the search space of TL-MKP, which is defined in (3) as $[0, b] \subset \mathbb{R}^m$. If a good lower bound of MKP f_L is known, e.g., through a quick meta-heuristic, the search space can be reduced to $F = \{t | f(t) \geq f_L, t \in [0, b] \subset \mathbb{R}^m\}$. However, this will make the EI acquisition function harder to optimise in (10) since F has no simple representation. In this paper, an optimisation based approach is employed to find the smallest hypercube $H = [t^L, t^U]$ that contains F, i.e., $F \subset H$. The upper bound of H along the i-th coordinate, t_i^U, $i = 1, \ldots, m$, can be obtained by solving

$$t_i^U = \max\{A^1 x^1 : c^T x \geq f_L, Ax \leq b, x^1 = x_{I_1}, x \in \{0, 1\}^n\}. \tag{11}$$

The lower bound of H along the i-th coordinate, t_i^L, $i1, \ldots, m$, can be obtained by solving

$$t_i^L = \min\{A^1 x^1 : c^T x \geq f_L, Ax \leq b, x^1 = x_{I_1}, x \in \{0, 1\}^n\}. \tag{12}$$

The exact solution of (11) and (12) is time-consuming. Therefore, t^U (t^L) can be replaced by a upper (lower) bound of (11) ((12)), e.g., using the linear programming relaxation by replacing $x \in \{0, 1\}^n$ with $x \in [0, 1]^n$.

3.2 Initialisation with Genetic Algorithm

The BO randomly generates the initial trial points in the search space which can lead to slow convergence. In this paper, The GA is used to generate initial points that have good solution quality as well as diversity in the search space. The GA is a population based meta-heuristic which evolves by generations through genetic operators such as cross-over and mutation. In the early stage of GA the population has good diversity but low percentage of good solutions; while in the later stage, the population has high percentage of good solutions but with less diversity.

It is computationally infeasible to run GA on the TL-MKP since the objective evaluation involves solving two MIP problems and consequently expensive. Instead, the GA is directly run on the MKP, and the population is mapped to initialise the BO for TL-MKP. In particular, let $\tilde{x}$ be a solution from a population of GA. The mapped solution for TL-MKP becomes

$$\tilde{t} = A^1 \tilde{x}_{N_1}. \tag{13}$$

It is easy to see that

$$f(\tilde{t}) \geq c^T \tilde{x}. \tag{14}$$

3.3 Optimisation Landscape Smoothing

At each sampling point of BO, a feasible solution to the MKP is also generated according to Proposition 1. This solution can be improved by a local search which is efficient to cope with large number of items and constraints. We define the neighbourhood of a solution x as the set of solutions with at most k different items:

$$N_k(x) = \{y \in \{0,1\}^n : Ay \leq b, \|\|x - y\|\|_1 \leq k\}. \tag{15}$$

For Example 1, with $k = 1$, the first level objective function becomes

$$f(t) = \begin{cases} 4 & t \in [0,2) \\ 3 & t \in [2,4] \end{cases},$$

which is "smoother" in terms of the optimisation landscape.

4 Implementation

The BO approach for the MKP (BO-MKP) can be described as in Alg. 1, and a prototype of BO-MKP was implemented in Matlab R2020b. In Step 1 of BO-MKP, the linear relaxation of (11) and (12) are solved to tighten the bounds of the feasible set of TL-MKP using the function *linprog* in Matlab *Optimization Toolbox*. Using the function *ga* in the *Global Optimization Toolbox* of Matlab, an initial set of trial points are generated in Step 2 as input for BO according to (13). In Step 3, the BO is implemented with the function *bayesopt* in the *Global Optimization Toolbox* of Matlab. The acquisition function is set to "expected-improvement", and the maximum number of evaluation, "MaxObjectiveEvaluations", is set to N which is a user specified parameter. The subproblems of TL-MKP (5) are solved by the mixed integer programming solver *intlinprog* in Matlab *Optimization Toolbox*. In Step 4, The best solution of TL-MKP found by BO is converted to a solution of MKP with the same objective function value according to Proposition 1.

The selection of kernel function for GP can have a strong influence on the performance of BO. *bayesopt* uses the ARD Matérn 5/2 kernel

$$k(x_i, x_j|\sigma_f, \sigma_l) = \sigma_f^2(1 + \frac{\sqrt{5}r}{\sigma_l} + \frac{5r^2}{3\sigma_l^2}) \exp(-\frac{\sqrt{5}r}{\sigma_l})$$

where $r = \sqrt{(x_i - x_j)^T(x_i - x_j)}$, and the parameters are estimated by Gaussian process regression *fitrgp*.

Algorithm 1: The BO approach for the MKP (BO-MKP).

Input: MKP, item groups I_1 and I_2, lower bound of MKP f_L, maximum number of evaluation N for BO.
Output: a feasible solution of MKP.
Step 1: tighten the bounds of feasible set of TL-MKP based on f_L;
Step 2: generate initial trial points using GA;
Step 3: search for the global optimum of TL-MKP using BO within a sampling budget of N evaluations;
Step 4: convert the best solution found by BO to the solution of MKP;
return

5 Computational Experiments

All experiments are carried out on the widely used Chu and Beasley MKP test set in [4]. The Chu and Beasley test set contains classes of randomly generated instances for each combination of $n \in \{100, 250, 500\}$ items, $m \in \{5, 10, 30\}$ constraints, and tightness ratios $\alpha \in \{0.25, 0.5, 0.75\}$ with smaller α representing tighter resource capacities. In the Chu and Beasley MKP test set, the resource consumption values a_{ij} are integers uniformly chosen from (0, 1000), which leads to large values of the knapsack capacities b. Since the search space of BO for TL-MKP is defined by b in (3), the Chu and Beasley MKP test set is an challenging test bed for the proposed BO approach.

To show the effect of tightening bounds in section 3.1, the BO is tested on three selected instances with $n = 100$, and the results are reported in Table 1. The rows correspond to the instances with the number of knapsack constraints $m = 5$, 10 and 30. The optimal values of these instances are obtained by CPLEX and reported in the column titled "Opt." The columns are divided into two groups for the BO results, one for the cases without bound tightening ("Without tightening") and the other one for the cases with bound tightening ("With tightening"). To have a better understanding of the convergence behavior of BO, two values are applied for the maximum number of evaluations, i.e., $N = 25, 50$. Since the BO is a stochastic algorithm, the average objective function value of 5 runs is reported for each pair of (m, N) in the columns titled "Ave.". The relative gap for the solution found by the BO is calculated as $100 \times (z^* - f)/z^*$ and reported in the columns titled "gap(%)". It can be seen that the performance of BO deteriorates dramatically when m increases. When $m = 30$, the BO reaches a massive relative gap of 63.8% after 50 function evaluations. This observation is consistent with BO's behavior for other optimisation problems. When bound tightening technique is applied, the performance of BO is improved on all (m, N) pairs. The improvement is more dramatic when m becomes large. For $m = 10$ and $N = 50$ the relative gap is reduced from 9.8% to 4%. However, the solution quality for $m = 30$ is still not satisfactory with a large gap of 31.8%.

Table 2 presents the results of BO-MKP which initialises the BO with GA. The initial trial points provided by the GA should be diverse enough while also having good solution quality. Therefore, the maximum number of iterations of

Table 1. Effects of bound tightening for BO on the TL-MKP.

		Without tightening				With tightening			
		N = 25		N = 50		N = 25		N = 50	
m	Opt.	Ave.	gap(%)	Ave.	gap(%)	Ave.	gap(%)	Ave.	gap(%)
5	24381	22897	6.1	23849	2.2	23913	1.9	24017	1.5
10	23064	17145	25.7	20806	9.8	20581	10.8	22149	4.0
30	21946	5710.2	74.0	7955	63.8	14659	33.2	14978	31.8

GA is limited to 55 in BO-MKP. It can be seen that the GA initialisation is not helpful when $m = 5$, which suggests that the BO has strong global search capability when the dimension is low. In sharp contrast, the BO-MKP dramatically reduces the relative gap for larger dimension. Indeed, the relative gap is just 4% for $m = 30$ with 50 function evaluations.

Table 2. Effects of GA initialisation for BO on the TL-MKP.

		$N = 25$			$N = 50$		
	Opt.	w/o GA	Ave.	gap(%)	w/o GA	Ave.	gap(%)
$m = 5$	24381	23913	23928	1.9	24017	24063	1.3
$m = 10$	23064	20581	22471	2.6	22149	22396	2.9
$m = 30$	21946	14659	20727	5.6	14978	21060	4.0

Table 3 shows the impact of employing the local search in solving the BO-MKP. With $k = 5$ for the neighbourhood defined in (15), the three instances with $m = 5$, 10 and 30 are all solved to optimality.

Table 3. Effects of local search for BO on the TL-MKP.

		$N = 25$		
	Opt.	w/o LS	with LS	gap(%)
$m = 5$	24381	23928	24381	0.0
$m = 10$	23064	22471	23064	0.0
$m = 30$	21946	20727	21946	0.0

The overall performance of BO-MKP on all the 90 instances with 100 items, i.e., $n = 100$ is presented in Table 4. For the groups with $m = 5$ and $m = 10$, we set $N = 25$ and $k = 5$. For all instances with $m = 5$ and 26 instances with $m = 10$, the optimal solutions are obtained. The remaining 4 instances in the group with $m = 10$ can also be solved to optimality by increasing N to 50. We set $k = 10$ and $N = 50$ for all instances with $m = 30$. This group of instances is particularly challenging to BO due to the high dimensions of

the search space. However, with a strong local search procedure to smooth the optimisation landscape, high quality solutions are obtained on all instances.

Table 4. Computational results for all instances with 100 items.

		$\alpha = 0.25$	$\alpha = 0.5$	$\alpha = 0.75$
$m = 5$	Average	24197.2	43252.9	60471.0
	Best	24197.2	43252.9	60471.0
	Opt.	24197.2	43252.9	60471.0
	gap %	0.0	0.0	0.0
	time	146.7s	128.8s	83.3s
$m = 10$	Average	22601.0	42660.2	59555.6
	Best	22601.9	42660.6	59555.6
	Opt.	22601.9	42660.6	59555.6
	gap %	0.0	0.0	0.0
	time	191.5s	195.3s	152.3s
$m = 30$	Average	21638.2	41420.3	59201.8
	Best	21652.9	41427.2	59201.8
	Opt.	21660.4	41440.4	59201.8
	gap %	0.1	0.0	0.0
	time	359.0s	359.3s	311.3s

6 Conclusion and Future Work

In this paper, a two-level model is presented for the multidimensional knapsack problem. The master problem has much smaller dimensions, which makes it amenable to Bayesian optimisation. Three techniques are introduced to accelerate the search process of BO. Preliminary test results show the effectiveness of the proposed approach. It strongly demonstrates that incorporating prior knowledge and smoothing the optimisation landscape by the local search are crucial for the success of BO for large MKP.

Future work includes the investigation of the proper kernels in BO for combinatorial optimisation problems, the automatic tuning of hyper-parameters, and comparison with other meta-heuristics. It is also interesting to extend the models to combinatorial optimisation problems with more complex structures.

References

1. Baheri, A., Bin-Karim, S., Bafandeh, A., Vermillion, C.: Real-time control using bayesian optimization: A case study in airborne wind energy systems. Control Engineering Practice **69**, 131 – 140 (2017).
2. Baptista, R., Poloczek, M.: Bayesian optimization of combinatorial structures. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 462–471. PMLR, Stockholmsmässan, Stockholm Sweden (10–15 Jul 2018)

3. Boussier, S., Vasquez, M., Vimont, Y., Hanafi, S., Michelon, P.: A multi-level search strategy for the 0–1 multidimensional knapsack problem. Discrete Applied Mathematics **158**(2), 97–109 (2010)
4. Chu, P.C., Beasley, J.E.: A genetic algorithm for the multidimensional knapsack problem. J. Heuristics **4**(1), 63–86 (1998)
5. Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., aurelio Ranzato, M., Senior, A., Tucker, P., Yang, K., Le, Q.V., Ng, A.Y.: Large scale distributed deep networks. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 25, pp. 1223–1231. Curran Associates, Inc. (2012)
6. Della Croce, F., Grosso, A.: Improved core problem based heuristics for the 0/1 multi-dimensional knapsack problem. Computers & Operations Research **39**(1), 27–31 (2012)
7. Domhan, T., Springenberg, J.T., Hutter, F.: Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In: Proceedings of the 24th International Conference on Artificial Intelligence. p. 3460–3468. IJCAI'15, AAAI Press (2015)
8. Drake, J.: Or library mkp - best known solutions, http://www.cs.nott.ac.uk/ jqd/mkp/bestresults.html
9. Frazier, P.I.: Bayesian optimization. INFORMS TutORials in Operations Research p. 255–278 (2018).
10. Freno, A., Saveski, M., Jenatton, R., Archambeau, C.: One-pass ranking models for low-latency product recommendations. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 1789–1798. KDD '15, Association for Computing Machinery, New York, NY, USA (2015).
11. Fréville, A.: The multidimensional 0 - 1 knapsack problem: An overview. European Journal of Operational Research **155**(1), 1–21 (2004)
12. Fréville, A., Hanafi, S.: The multidimensional 0-1 knapsack problem - bounds and computational aspects. Annals of Operations Research **139**(1), 195–227 (2005)
13. Greenhill, S., Rana, S., Gupta, S., Vellanki, P., Venkatesh, S.: Bayesian optimization for adaptive experimental design: A review. IEEE Access **8**, 13937–13948 (2020)
14. Gu, H.: Improving problem reduction for 0-1 multidimensional knapsack problems with valid inequalities. Comput. Oper. Res. **71**(C), 82–89 (Jul 2016).
15. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. Journal of Global optimization **13**(4), 455–492 (1998)
16. Junge, K., Hughes, J., Thuruthel, T.G., Iida, F.: Improving robotic cooking using batch bayesian optimization. IEEE Robotics and Automation Letters **5**(2), 760–765 (2020)
17. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack problems. Springer (2004)
18. Lai, X., Hao, J., Glover, F.W., Lü, Z.: A two-phase tabu-evolutionary algorithm for the 0-1 multidimensional knapsack problem. Information Science **436-437**, 282–301 (2018).
19. Oh, C., Tomczak, J., Gavves, E., Welling, M.: Combinatorial bayesian optimization using the graph cartesian product. In: Wallach, H., Larochelle, H., Beygelzimer, A., d' Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 2914–2924. Curran Associates, Inc. (2019)
20. Rai, A., Antonova, R., Meier, F., Atkeson, C.G.: Using simulation to improve sample-efficiency of bayesian optimization for bipedal robots. Journal of Machine Learning Research **20**(49), 1–24 (2019)

21. Sano, S., Kadowaki, T., Tsuda, K., Kimura, S.: Application of bayesian optimization for pharmaceutical product development. Journal of Pharmaceutical Innovation (2019).
22. Shahriari, B., Swersky, K., Wang, Z., Adams, R.P., de Freitas, N.: Taking the human out of the loop: A review of bayesian optimization. Proceedings of the IEEE **104**(1), 148–175 (2016)
23. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 25, pp. 2951–2959. Curran Associates, Inc. (2012)
24. Vasquez, M., Vimont, Y.: Improved results on the 0-1 multidimensional knapsack problem. European Journal of Operational Research **165**(1), 70–81 (2005)
25. Vimont, Y., Boussier, S., Vasquez, M.: Reduced costs propagation in an efficient implicit enumeration for the 01 multidimensional knapsack problem. Journal of Combinatorial Optimization **15**(2), 165–178 (2008)
26. Wang, Z., Hutter, F., Zoghi, M., Matheson, D., De Freitas, N.: Bayesian optimization in a billion dimensions via random embeddings. J. Artif. Int. Res. **55**(1), 361–387 (Jan 2016)
27. Wilbaut, C., Hanafi, S.: New convergent heuristics for 0–1 mixed integer programming. European Journal of Operational Research **195**(1), 62–74 (2009)
28. Wilbaut, C., Hanafi, S., Salhi, S.: A survey of effective heuristics and their application to a variety of knapsack problems. IMA Journal of Management Mathematics **19**(3), 227–244 (2008)
29. Yogatama, D., Kong, L., Smith, N.A.: Bayesian optimization of text representations. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 2100–2105. Association for Computational Linguistics, Lisbon, Portugal (Sep 2015).

Multiple-tour constrained optimization for waste sites inspections

F.Nebula, F.Gargiulo, G.Gigante, D.Pascarella, L.Cicala

CIRA - Italian Aerospace Research Center, via Maiorise snc, 81043 Capua, Italy
f.nebula@cira.it, f.gargiulo@cira.it, g.gigante@cira.it, d.pascarella@cira.it, l.cicala@cira.it

Keywords: combinatorial optimization, constrained optimization, multiple-tour optimization.

1 Introduction

Within the *Crowd for the Environment* (C4E) project, several emerging information technologies are integrated in an innovative framework to discover and monitor illegal dumping sites. Such dumps typically contain urban or agricultural waste and often flammable materials, then they could be open-burned because spontaneous combustion or malicious intents. In any case, the open-burnings are source of cancerogenic and toxic substances, causing many diseases in the neighbor population. Such hazardous dumping sites, once detected by citizens or by some technologies, must be confirmed and characterized on the field by environmental authorities. In particular, in the C4E project, the detection of the dumping sites comes from spontaneous reports of citizens (that use a specific mobile app), social network crawlers (that process natural language in order to extract synthetic reports) and periodic satellite image acquisitions (artificial intelligence extracts areas potentially affected by illegal dumping). The characterization performed by the authorities, in terms of volume and composition, is used to assess the health risk on the neighborhood in case of fire. The risk analysis is needed for remediation, that can be planned when the characterization of a confirmed site is sent to the local government. The C4E project proposes to perform a risk analysis also before a site is confirmed, in order to establish a *priority* for the of-field inspections. In fact, the resources that can be deployed on the territory (men, cars, drones) are expensive and limited then they should be used in an optimized way in order to mitigate the risk of open-fires near populated zones. The number of sites can be very large and also the vehicles used for the inspections can be numerous, as well as heterogeneous and even starting from different depots. Moreover, usually the patrols have to respect some time constraints (represented by the work shift, for example), considering both the travel time and the time needed to inspect the site, hereafter referred as "time to collect its priority". For the above reasons, the problem of finding the vehicles routes, in such a way that a "priority index" referred as "the overall collected priority" is maximum, is very hard to solve, unless the sites are very few. Currently, the tours are assigned by human operators according to their own experience but this way forward takes much effort and often underperforms. Consequently, it is strongly desired a fast and effective algorithm capable to find at least a quasi-optimal solution. Indeed, being the problem belonging to the class of the NP-hard problems, it cannot be solved exactly in reasonable time, already when the dimensionality (sites number) is moderate.

In this work an approximated method from Literature [1] is properly modified and evaluated. It has shown good accuracies ensuring, at same time, acceptable computational load, i.e. compatible with interactions with human operators. Common requirements for the solution error and for the computation time are approximately not greater than 20% and half an hour, respectively.

2 Problem formulation

Our problem is to maximize the priority index over n geographical points within a time deadline T, using not more than m available vehicles, starting from the same depot. It is worth to note that both the order of visit and the subset of nodes to be visited must be found, being often impossible to visit all of them due to the limited available resources. Since the cardinality of the problem is usually very high (hundreds of thousands of places to be visited), a decomposition in *strategic* and *tactical* sub-problems comes to help. In this work, only the strategic phase is considered: the waste places are initially grouped in geographical zones, characterized by their own centroids identifying the *nodes* mentioned in this work. All the priorities,

all the trip times and all the collection times of the original places are assumed cumulated in these nodes. Moreover, each node can be visited by one vehicle only and only once. When the strategic step is concluded, all the places within the selected zones must be visited (the Floyd-Warshall algorithm [2] could be used to solve the tactical phase).

Hereafter we formulate the problem for homogeneous vehicles starting from a unique depot (0) because the general case could be decomposed in problems like this. Then, let us define a complete graph G=[N,A] where N={0,1,…,n} is the set of nodes (centroids) and A={a_{ij}} is the set of the arcs between the i-th and j-th nodes (i,j∈N-{0}). Furthermore, we assume that D={d_{ij}}={d_{ji}} is the "time matrix" composed by the trip times between i-th and j-th nodes, p_i is the priority of the node i-th and b_i is time required to collect p_i (p_0=b_0=0). A tour is defined as feasible through a subset of G if starts and ends in the depot, if visits each node not more than once and if ends within the given maximum time T. Hence we must find the feasible tours (not more than m) such that the priority index – collected over all tours – is maximized.

3 Algorithm description

The described problem of combinatorial optimization is classified as NP-hard, therefore it is unlikely that there is an optimal polynomial algorithm for problem instances with more than a dozen of nodes. The selected approximated algorithm is known as MAXIMP ([1]), working deterministically and independently for each tour. Basically it relies on the idea that, in order to maximize the collected priority, each patrol has to spend more possible time, in compliance with the time constraint. This means that the aim of the routine is to maximize the priority in the time unit. Actually, not all the time units will be used to collect priorities but nevertheless this approach quickly provides "good" solutions for our application.

Two different types of weights are defined, both depending on the collected priority (p_i+p_j) in the elementary loop tour composed by the depot, the i-th node, the j-th node, and the depot again. They are defined as W_{1ij}=((p_i+p_j)/T)t_{ij} and W_{2ij}=((p_i+p_j)/t_{ij})T, where t_{ij}=d_{0i}+d_{ij}+d_{j0}+b_i+b_j is the overall travel and collection time. These weights make us know the best pairs of nodes in terms of the largest priority collected in the time required to be visited (t_{ij}/T) and in terms of the time needed to be collected, respectively. Consequently, W_{1ij} and W_{2ij} will tend to select the pairs farther from and closer to the depot. Moreover, each pair of nodes has associated a convex combination of these weights defined as $W_{ij}(\alpha)=\alpha W_{1ij}+(1-\alpha)W_{2ij}$ and providing the definitive ranking of the graph pairs. In its definition, α∈[0,1] is a tradeoff parameter, unfortunately providing optimal results for one geometry only. As a rule of thumb, α<0.5 should be used for n>20. A tour is built scrolling the array composed by the W_{ij} values and looking for additional nodes satisfying the time constraint.

In this work some modifications to [1] are carried out, obtaining a custom MAXIMP algorithm. If the weights' array does not remain fully empty (very probable for large n), a brute force algorithm is applied on the remaining nodes. Furthermore, different α parameters are used to provide different solutions to choose from, after selecting a statistics (Maximum, Standard Deviation, RMS, etc.) of the times of the used patrols, to be minimized or maximized in order to find a unique solution (0, 0.25 and 0.5 are used instead of 0.1). As expected, these upgrades strongly improve the optimization accuracy but require additional computation time, therefore they can be applied only for less than about 100 nodes.

4 Performance assessment

In this section, the standard and custom MAXIMP algorithms are compared in terms of solution accuracy and computational load. The reported assessment involves, as reference, a brute force algorithm (i.e. based on an exhaustive evaluation of all possible tours in order to find the optimal solution).

The Monte Carlo (MC) methodology is applied in order to try filtering the effects of different graph geometries. The parameters scattered within a session are the time matrix (D), the priorities and the collection times (p_i, b_i with i=1, … , n) of the nodes. Instead, the number of nodes (n), the numbers of patrols (m=3) and the deadline (T=10000) are constant over the 100 simulations performed for each session. For less than 10 nodes, the brute force can be used as reference because capable to find the optimal solution in reasonable time. For more nodes, only some random runs can be performed and, by this way, the custom MAXIMP performances are generalized.

In Fig. 1 the performances of the standard and custom MAXIMP are reported for few nodes. Note that the brute force takes about 15 min already for n=9, after that it cannot be used easily (Intel® Core™ i7-8665U CPU @ 1.90-2.11GHz, 8GB RAM). Zero error means that the set of the visited nodes is optimal but the order of visit and/or the patrol assignment could change with respect to the exact algorithm. Indeed, the brute force is implemented to use less time as possible, while MAXIMP tends to use the whole time span (actually, different sets of nodes may generate the same total priority but this case is very rare). Sometimes, for particular graph geometries, the standard MAXIMP finds a zero overall priority because, in the sorted array of weights, there is no tour compliant with the time constraint (the custom MAXIMP overcomes this situation). In other words, the standard MAXIMP fails more than the modified one because of the unique α value that cannot manage all the nodes configurations. As the number of nodes increases, the standard MAXIMP is the only tool capable to provide a solution in acceptable time (Fig. 2). A simple way to take advantage of the very good accuracy of the custom MAXIMP is to perform an appropriate initial clustering, generating a limited number of nodes.

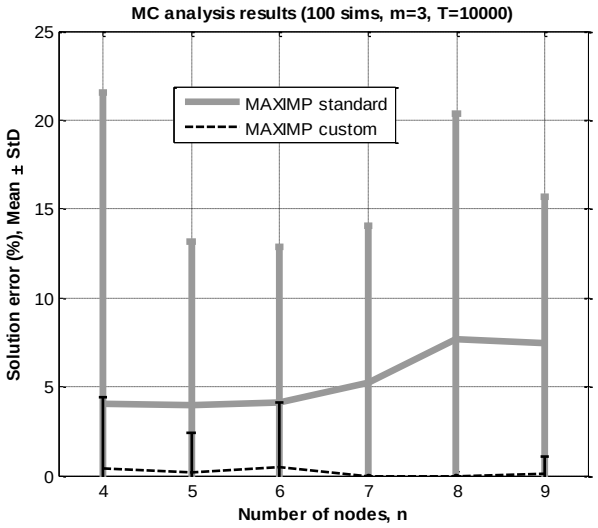


Fig. 1 Accuracies of standard and custom MAXIMP algorithms

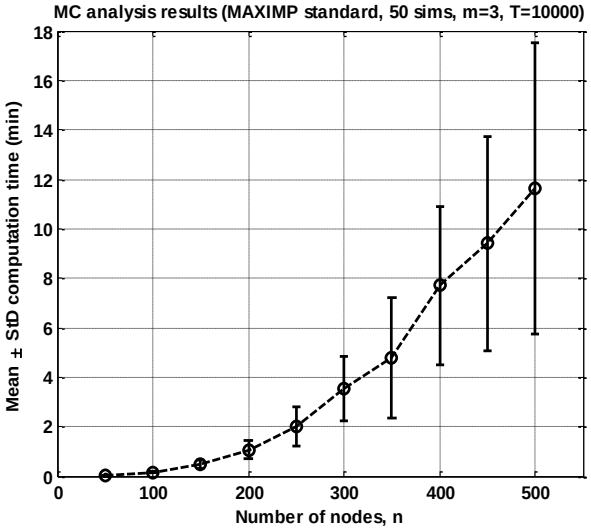


Fig. 2 Computation times of standard MAXIMP algorithm

5 Conclusions

In the field of environmental monitoring, some vehicles have to visit many critical waste sites within a specified deadline, trying to maximize the overall priority index. Currently human operators provide themselves these paths but this is a challenging issue, being the tour optimizations in the NP-hard class problem. Thus, it is strongly desired to dispose of an algorithm capable to provide good solutions in reasonable times. In this work, a multi-tour time-constrained algorithm from Literature has been applied, also with some modifications. The reported performance evaluation confirms that the MAXIMP algorithms are highly promising to achieve the abovementioned objective, in terms of simplicity, limited computation time (~12 min for 500 nodes, in its standard version), and fair accuracy (~10%) regardless of geometry and matrix time. The standard version of MAXIMP is best suited for handling hundreds of nodes and if a high accuracy is not required. On the other hand, custom MAXIMP can be applied when, for less nodes, a quasi-optimal solution is desired in reasonable time, also achieving high accuracy (~2%). The use of the custom MAXIMP is facilitated applying a low dense clustering of the initial waste places, in order to obtain a graph with a moderate number of nodes.

References

[1] Butt, S.E. and T.M. Cavalier. (1994). "A Heuristic for the Multiple Tour Maximum Collection Problem", Computers and Operations Research 21, 101–111.

[2] Eugène L. Lawler, Floyd-Warshall Method, in "Combinatorial optimization: networks and matroids". Courier Dover Publications, 2001, pp. 86–92.

Rescheduling in textile industry : an unrelated parallel machines problem with setup times

A. Berthier[1], H. Chehade[1], A. Yalaoui[1], F. Yalaoui[1], L. Amodeo[1] and C. Bouillot

Laboratory of Inudstrial Systems Optimization (LOSI), University of Technology of Troyes,
12 rue Marie Curie, CS42060, 10004 Troyes Cedex, France
alice.berthier@utt.fr

1 Introduction

Nowadays, in textile industry as in other industries facing the industry 4.0 revolution, the work is changing. This study takes place in the original factory of a famous company created in 1893, a textile industry located in Troyes (France). This factory is facing a market trend evolution and have to adapt his production process. The company is very known for its high quality baby's clothing. They also develop adults garments collection. The product diversity increases while the total volume decreases. This allows a permanent renewal of the products displayed. Furthermore, the development of online-business requires more and more flexibility and reactivity. This distribution channel is more important during COVID 19 crisis context as e-business have greatly increased. The adaptation needed is reflected in the entire process of the clothing manufacturing industry from the knitting of the fabric to the assembling stage. Figure 1 shows the three principals stages in textile factory.

Fig. 1. The three fabrication stages in textile industry

This study focuses on the first production stage: the knitting workshop. This workshop needs a new organization in this context. In the past, all machines worked at the same time. Nowadays, only some machines work simultaneously. A rotation between the machines takes place regularly. This work is related to a previous one [2] which focused on a dynamic layout of such as workshop. The main subject was about the creation of groups of machines. The objective was to balance the workload between all the operators and to propose a new implementation method. A good workshop layout was a prerequisite before optimizing the scheduling
An other study has been conducted previously on the scheduling problem of this workshop. A method to solve the unrelated parallel machines scheduling problem with setup time and limited resources has been implemented on the industrial partner workshop.
In order to go further on this subject, the study propose in this study focuses on the rescheduling of the knitting workshop. The problem tackled is to propose a method to adapt the production scheduling when disruption occurs with a limited impact on the workshop organization and the performance of productivity.

The rest of the study is organized as follows. Section 2 gives a description of the industrial problem. The third section gives a literature review on this kind of problem. Section 4 provides contribution of this study. A conclusion and future research directions end up the study in section 5.

2 Problem description

The main objective of this study is to improve the scheduling software implemented on the knitting workshop by proposing a rescheduling module to adapt the production scheduling to any disruption that can occur. This, in the main objective of helping the knitting workshop manager to make better and quicker decisions in his daily work at any time.
The workshop is composed of M unrelated parallel machines. An example of a circular knitting machine is given in the figure 2. The machines are grouped in W areas. The notation Z_{iw} is used

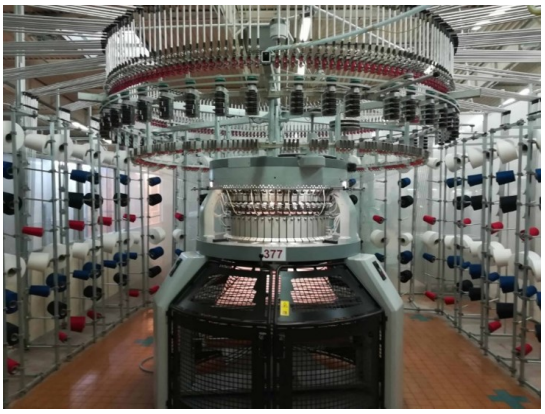


Fig. 2. Circular knitting machine

to define the assignment of machine i to the area w. Each operator is monitoring a machine area. The number of machines able to run at the same time in the area w is denoted by U_w. This is the guarantee of good quality fabrics. The operators are not able to manage more machines than a specific number due to his qualification level. The objective is to schedule a known number of N jobs on the machines in a time horizon of H discredited in different t slots. The machines are considered as unrelated as the processing time p_{ij} of product j on machine i of each job are not related neither to the machine i or the product j. Their is no relation between processing time of an identical product on two different machines. Furthermore, this study takes into account machines and sequence dependent setup times s_{ijk}. The data s_{ijk} is the necessary time to make the transition when the job j preceding the job k on the machine i. The setup adjustments are made by members of a crew. So, this study has to consider them as limited resource. The notation B_t is the number of this resource type available at the moment t. This study is also tackled with a machine restriction constraint. Only the product j with a data e_{ij} equal to 1 can be processed by the machine i.
An initial production planning is provided by the scheduling algorithm already implemented in the software. However, perfect production conditions are very unrealistic, disruption can occurs and the initial planning is no longer up to date. The different disruptions that can occurs in this problem are:

- Arrival of a new job
- Deleting a job
- Machine breakdown
- Lack of human resources

The rescheduling objective is to find the best possible planning to finish all the jobs as soon as possible by keeping stability in the planning initially provided. This is why this study is focused on the objective of maximizing performance ($min\ c_{max}$) while maintaining stability.
To summarize this study investigate an unrelated parallel machine rescheduling problem with sequence and machine dependent setup times. Machine eligibility restrictions and two different types of common server (operators and adjusters) are included. Four different disruptions can occurs. The objective is to minimize the maximum completion time while maintaining stability between the initial production scheduling and the rescheduling one.

3 State of the art

In order to position the problem regarding to the literature, a review of the parallel machines rescheduling problem has been done. Different rescheduling approaches are proposed in the literature.
The first one is to use a standard scheduling method with the new data after disruption. This can rich high quality solution on the performance objective. However, stability on solution are not guarantee ([4]). On real life production, getting a totally different schedule is very unfavourable to a good workshop organization and management.
The second one is to use a proactive scheduling. This is generated by inserting idle time between the pre-scheduling activities, enabling the disruptions to be smoothed out through the system in order to maintain the schedule quality ([1]). Stochastic approaches is an other way to do it.
The last one is reactive scheduling, commonly referred to as rescheduling. It is a procedure to modify the existing schedule during processing to adapt to changes in a production or operational environment. Kim ([3]) recently studies a rescheduling problem of unrelated parallel machines with job-dependent setup times under forecasted machine breakdown.
On a majority of studies, two conflicting objectives are taken into consideration: performance and stability. Multiple indicators in the literature are proposed. However, each industry has its own characteristics that involves specific indicators.

4 Contribution

The contribution of this study is to explore different performance evaluation of rescheduling solution in the specific case of textile industry production. The resolution method is based on a genetic algorithm developed for this specific problem. The performance measure is the minimization of completion time (c_{max}). The stability in rescheduling is more complex to evaluate. Different indicators are explored and combination of them are analysed in order to propose the most pertinent and efficient rescheduling planning to the company. These different indicators can be compare to a limit parameter. If the limit is crossed so the objective function will be penalized. This allows a tolerance and plays up on the performance objective.

5 Conclusion

The originality of the problem studied in this paper is the specific application to textile industry. New evaluation method of stability in rescheduling problem will be tested in order to propose to the industrial partner an efficient solution. This work will be based on the previous study with this company on the unrelated parallel machines scheduling problem with setup time and limited resources. The continuity of this study is to allow to the company to rescheduling the workshop production every time an unpredictable disruption occurs. It is a very import prerequisite to have an agile and reactive production. It is also a first step on the road to the 4.0 factory transformation.

References

1. Jean-Paul Arnaout. Rescheduling of parallel machines with stochastic processing and setup times. *Journal of Manufacturing Systems*, 33(3):376–384, 2014.
2. A Berthier, Alice Yalaoui, Hicham Chehade, Farouk Yalaoui, Lionel Amodeo, and Guillem Coquelet. Machines group and load balancing: an industrial case. *IFAC-PapersOnLine*, 52(13):415–420, 2019.
3. Young-In Kim and Hyun-Jung Kim. Rescheduling of unrelated parallel machines with job-dependent setup times under forecasted machine breakdown. *International Journal of Production Research*, pages 1–23, 2020.
4. Jatoth Mohan, Krishnanand Lanka, and A Neelakanteswara Rao. A review of dynamic job shop scheduling techniques. *Procedia Manufacturing*, 30:34–39, 2019.

A stochastic and dynamic approach for the one-commodity pickup & delivery travelling salesman problem

Seddik Hadjadj and Hamamache Kheddouci

Laboratoire d'informatique en image et systèmes d'information, Bâtiment Nautibus
43, bd du 11 novembre 1918, 69622 VILLEURBANNE, FRANCE

Abstract. In this paper, we consider the one-commodity travelling salesman problem. One vehicle has to visit n customers to deliver small containers of ready-mixed concrete or pick them up after they have been used. Unexpected detours to recycling centres are considered, which makes tours uncertain beforehand and involves dealing with stochastic and dynamic tours.
We propose two approaches to tackle the problem. An estimation-based local search that generates a priori optimized tours considering potential detours that may occur, and an online approach which starts from an initial a priori tour and adapts it to fit the unexpected detours dynamically. We provide some experimental results that show the effectiveness of our approaches, especially when detours are more likely to occur.

Keywords: Dynamic vehicle routing · stochastic vehicle routing · local search · pickup & delivery problems...

1 Introduction

This work is carried out in collaboration with a company which specializes in the sale of ready-mixed concrete.
Ready-mixed concrete is normally delivered in *mixer trucks*. This type of truck is heavy, cumbersome, expensive, and can be disproportionate in some cases, especially when delivering small quantities of concrete.
Therefore, the company wants to propose a new delivery method using small containers (500 litre bins) to reduce delivery costs and deal more effectively with orders of small quantities. This new method is a two-step process. A vehicle delivers a number of bins of concrete to the customer, then, returns the next day to pick them up after they have been emptied by the customer.
To ensure the profitability of this method, the company needs a decision support system that can generate efficient **pickup & delivery tours** taking into account the vehicle capacity constraint and **recycling constraint**. Thus, if a bin is empty and clean when it is picked up from a customer, it could be directly supplied to another customer. Otherwise, it must be immediately routed to a recycling centre before it can be delivered again (unconsumed concrete is then

recycled and the bin cleaned). Knowing that the state of a bin is uncertain before the vehicle arrives at the customer's location, the a priori planned vehicle route may change during the time to include recycling centres whenever necessary. This uncertainty involves dealing with **stochastic vehicle routing**.
This paper aims to provide efficient approaches to build pickup & delivery tours minimizing the total travel distance by minimizing the loss of quality caused by potential detours to recycling centres.
The paper is structured as follows. Section 2 provides a description of pickup & delivery and stochastic/dynamic vehicle routing problems. Section 3 gives a formulation for the problem tackled in this work. In section 4, we present our first heuristic which is an estimation-based local search that provides a priori optimized tours. Section 5 describes our dynamic online algorithm that provides a priori optimized tours that can be dynamically adjusted to fit recycling centres detours. Section 6 proposes some experimental results, and section 7 concludes the paper.

2 Literature Review

We consider the One-Commodity pickup & delivery travelling salesman problem in a dynamic context.

2.1 Pickup & Delivery Problems

There are three main classes of pickup & delivery problem in the literature :

One-to-One Problems One or more vehicles have to carry n commodities, where each commodity has an origin and a destination. One of the best known examples of this class is the *Dial-a-ride problem* which consists of transporting people from an origin to a destination.

One-to-Many-to-One Problems Commodities are divided into "delivery commodities" and "pickup commodities". One or more vehicles have to carry the delivery commodities from the depot to the customers and the pickup commodities from the customers to the depot. Assuming that n_p is a set of pickup customers, and n_d a set of delivery customers, two cases have been distinguished for these problems : single demands, where $n_p \cap n_d = \emptyset$, and combined demands, where $n_p \cap n_d \neq \emptyset$. Several heuristics have been proposed for both single and multi-vehicle cases [10, 5]...

Many-to-Many Problems One or more vehicles have to transport goods between customers knowing that each customer can be a source or a destination of any type of good. Among the problems of this class, the *One-Commodity pickup and delivery travelling salesman problem* was introduced in [8]. A single vehicle with a known and finite capacity has to carry a single commodity between pickup

customers and delivery customers. Picked up commodities can be supplied to delivery customers. This problem is known to be *NP-Hard*. Moreover, checking the existence of a feasible solution is an *NP-Complete* problem [7]. Studies on such problems are relatively scarce. A branch and cut algorithm has been proposed in [8] for small instances, and two heuristics have been developed in [9] to tackle larger instances.
For a detailed survey on pickup and delivery problems, we refer the reader to [1].

2.2 Stochastic/Dynamic Vehicle Routing Problems

Vehicle routing problems can be classified according to the information quality and evolution. Thus, an input information can be deterministic or stochastic, and it can be known in advance or revealed during the tour.
A taxonomy of vehicle routing problems based on these two dimensions is proposed by [12]. Four types of vehicle routing problems are then distinguished :

Static and Deterministic Problems Input is known in advance and doesn't change over time. This is the most studied type of problem, but it generally doesn't fit with real-world applications, where some information cannot be known beforehand.

Static and Stochastic Problems Here, some information is a stochastic variable which is revealed gradually during the execution of the tour. However, the a priori planned routes cannot change during the execution of the tour except in some special cases. For example, if the stochastic variable considered is the customer's request, or in other words, if customers may request a visit with a certain probability, the a priori planned route may change only to skip customers that do not require a visit. Several types of stochastic variables have been studied in the literature : stochastic travel times [11], where travel times between customers is a random variable, stochastic customers, where customers may request a visit with a certain probability [3]...

Dynamic and Deterministic Problems Some information is totally unknown beforehand and is revealed only during the execution of the tour. Vehicle tours are then changed in real time, during the execution of the tour according to new information.

Dynamic and Stochastic Problems This type of problem is a combination of the latter two types described above. Some information is a stochastic variable that can be used to build a priori tours taking into account possible future events, and routes are adapted in real time according to information changes.
For more details on stochastic and dynamic vehicle routing problems, we refer the reader to the surveys of [12] and [13].

2.3 Stochastic/Dynamic Pickup & Delivery Problems

Most studies tackling pickup & delivery problems consider the static case in which all information is known beforehand and does not change during the time. However, some papers deal with the dynamic case where some information is only revealed during the tour and the a priori tour is adapted progressively in real time. A few of these works exploit stochastic information to anticipate future events, [2] present some of these papers. However, to the best of our knowledge, there is no work dealing with the one-commodity travelling salesman problem in a stochastic case.
The problem considered in this paper can be classified as a stochastic one-commodity travelling salesman problem. It is stochastic because we have probabilistic information through historical data about potential future detours. It is a one-commodity travelling salesman problem because a single vehicle has to carry one commodity from a set of pickup customers to a set of delivery customers.

3 Problem Formulation

The pickup & delivery travelling salesman problem(1-PDTSP) can be defined on a complete graph $G = (V, E)$ as follows :

- $V = \{0, 1, ..., n\}$ is a set of $n + 1$ nodes representing the n customers ($n = n_d + n_p$, where n_d is the number of delivery customers and n_p the number of pickup customers). Node 0 represents the depot ;
- $E = \{(i, j), i, j \in V, i \neq j\}$ is a set of edges representing connections between customers ;
- $C = \{c_{i,j}, (i, j) \in E\}$ represents the travel distance between customer i and customer j ($c_{i,j} = c_{j,i}, \forall (i, j) \in E$) ;
- $D = \{d_i, i \in V\}$ is a set of customers' demands ($|d_i|$ is the number of bins to deliver to / pick up from a customer i, $d_i < 0$ for delivery customers and > 0 for pickup customers) ;

Given a vehicle with a known and finite maximum capacity Q, and assuming that :

- $x_{i,j}$ is a boolean variable such that $x_{i,j} = 1$ if customer j is visited immediately after customer j, 0 otherwise ;
- q_i the number of bins in the vehicle after his visit to the customer i.

Our objective is to find a Hamiltonian cycle that minimizes the total travel distance, i.e. :

$$min \quad \sum_{i=0}^{n} \sum_{j=0}^{n} x_{i,j} c_{i,j} \tag{1}$$

Subject to :

$$\sum_{j \in N} x_{i,j} = 1 \qquad \forall i \in \{D_1\} \cup N \tag{2}$$

$$\sum_{i \in N} x_{i,j} = 1 \qquad \forall j \in \{D_1\} \cup N \tag{3}$$

$$\sum_{j \in N_d} y_{i,j} = 1 \qquad \forall i \in \{D_2\} \cup N_d \tag{4}$$

$$\sum_{i \in N_d} y_{i,j} = 1 \qquad \forall j \in \{D_2\} \cup N_d \tag{5}$$

$$x_{i,D_2} = 0 \qquad \forall i \in \{D_1\} \cup N \tag{6}$$

$$x_{D_2,i} = 0 \qquad \forall i \in \{D_1\} \cup N \tag{7}$$

$$y_{i,j} = 0 \qquad \forall i,j \in \{D_1\} \cup N_p \tag{8}$$

$$x_{i,j}(q_i + d_j - q_j) = 0 \qquad \forall i,j \in \{D_1\} \cup N \tag{9}$$

$$q_i \leq Q \qquad \forall i \in \{D_1\} \cup N \tag{10}$$

$$q_i \geq 0 \qquad \forall i \in \{D_1\} \cup N \tag{11}$$

$$q_{D_1} = Q_{init} \tag{12}$$

Constraints (2) and (3) ensure that each customer is visited exactly once, while constraints (4) and (5) relate to vehicle capacity.
Picked up bins can be supplied to a delivery customer if necessary. However, if a bin is not totally clean and empty when it is picked up from a customer, it must be firstly routed to one of the R available recycling centres around the customer's location before it can be supplied again. We can consider the R available recycling centres as a "priority customer" that may require a visit after each of the n_p pickup customers.
Thus, we define for each customer i :

- ω_i such that $\omega_i = 1$ if a detour is required immediately after i, 0 otherwise (note that the value of ω_i is unknown beforehand) ;
- p_i, the probability to require a detour immediately after customer i, or, in other words, the probability that $\omega_i = 1$ (note that $p_i = 0$ for all delivery customers).

4 A priori Optimization Approach

To tackle the 1-PDTSP described above, we first propose an a priori optimization approach which considers potential detours to recycling centres to build static vehicle tours. The proposed algorithm is an estimation-based heuristic adapted from the approach presented in [4] for the probabilistic travelling salesman problem.
This approach is based on a local search method which starts from an initial feasible solution S, and tries to improve it by moving to S', a feasible neighbouring solution of S, such that $f(S') < f(S)$. The process is repeated until no improvement can be found.

4.1 Neighbourhood Structure

We use the 1-shift algorithm introduced in [3] to generate the neighbourhood of a given solution S. This method consists in changing the position of a customer in a tour from i to j. Customers who are in positions $i+1, i+2, ..., j$ of the tour are then shifted backwards (see Fig. 1).

4.2 Feasibility Checking

For each generated solution, we ensure that capacity constraints described in section 3 are respected. A feasible solution is a tour in which the number of bins loaded on the vehicle never exceeds the maximum capacity Q of the vehicle, and is never negative. Assuming that q_i is the number of bins in the vehicle after visiting customer i, Fig.1 presents an example of a feasible and an infeasible solution.

Given a feasible solution S and a 1-shift neighbouring solution S' of S obtained

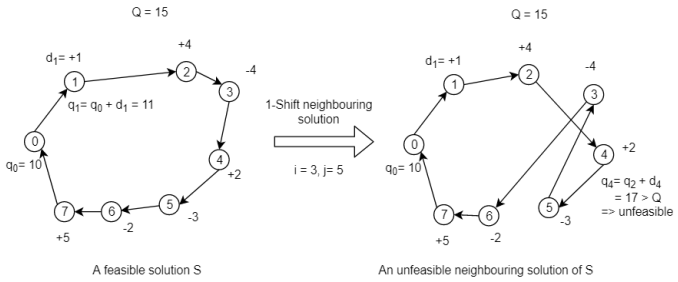


Fig. 1. 1-Shift algorithm

by shifting a customer from position i to j. It can easily be shown that S' is feasible if and only if the partial tour from customer i to customer j is feasible. Indeed, to check the feasibility of a neighbouring solution, we only check the feasibility of the tour between position i and position j.

4.3 Objective Function

In our case, the objective function f to minimize is the total travel distance of the vehicle. However, since we cannot know in advance the travel distance of an a priori solution S due to potential detours to recycling centres (see Fig. 2), we use the following unbiased estimator of $f(S)$ as a criterion to move from a solution to another one :

$$\hat{f}_M(S) = \frac{1}{M} \sum_{r=1}^{M} f(S, \omega_r)$$

This estimator was proposed by [4] for the probabilistic travelling salesman problem. The idea is to estimate the quality of an a priori solution S from a set of M simulations of possible a posteriori solutions. An a posteriori solution is obtained by associating a binary vector ω with the a priori solution such that $\omega[i] = 1$ if a detour to a recycling centre is required immediately after visiting a customer i, 0 otherwise (see the vector ω in Fig. 2).
Thus, given an a priori solution S (that does not include recycling detours) :

1. M possible a posteriori solutions (including potential detours) are generated by associating M vectors ω with the a priori solution S;
2. For each generated a posteriori solution, $f(S, \omega_i)$, the travel distance of the a posteriori solution given by ω_i is calculated. $f(S, \omega_i) = f(S) + TDL - \sum_{i=1}^{n} \sum_{j=1}^{n} \omega_i x_{i,j} c_{i,j}$, where :
   - $f(S)$ is the travel distance of the a priori solution S (without detours) ;
   - TDL is the Total Detour Length of the a posteriori solution (see the example in Fig. 2).
3. $\hat{f}_M(S) = \frac{1}{M} \sum_{r=1}^{M} f(S, \omega_r)$ is calculated and considered as an estimator of $f(S)$.

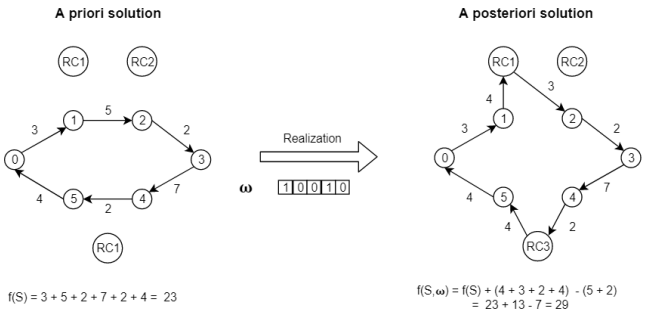


Fig. 2. A priori solution VS a posteriori solution

Note that ω is generated according to the set $P = \{p_i, i \in V\}$ of probabilities of requiring detour after visiting customer i. Therefore, $\omega[i] = 0$ for all delivery customers because detours may occur only when picking up bins.

4.4 Recycling Centre Choice

Since we consider R available recycling centres in our problem, each time a detour to recycling centre is required, we must choose among the R possibilities we have. Therefore, we calculate the travel distance caused by the detour to each of the R available recycling centres to choose the one that minimizes the detour length.

5 Online Optimization Approach

The second proposed approach is a dynamic algorithm that starts from an initial a priori optimized tour and adjusts it during the time whenever it is necessary (before each detour).

Algorithm 1 Dynamic routing algorithm

```
1: InitialSolution ← GenerateIntialSolution()
2: // generate an initial optimized a priori solution including the n customers
3: S ← LocalSearch(InitialSolution)
4: // during the tour, at each customer's location...
5: for (i ∈ S) do
6:    // if a detour is required...
7:    if (ω_i = 1) then
8:       BestSubSolution ← nil
9:       // for each available recycling centre...
10:      for R_j ∈ R do
11:         // extract the sub-tour S_j that has not been travelled yet (from i + 1 to n)
12:         S_j ← SubSolution(i + 1, n)
13:         // add R_j at the beginning of S_j
14:         Add(S_j, R_j)
15:         // start a local search with S_j as an initial solution
16:         S_j ← LocalSearch(S_j)
17:         if BestSubSolution = nil OR f(S_j) < f(BestSubSolution) then
18:            BestSubSolution ← S_j
19:         end if
20:      end for
21:      // replace the sub-tour i + 1...n by BestSubSolution
22:      Replace(SubSolution(i + 1, n), BestSubSolution)
23:   end if
24: end for
```

As it is described in Algorithm 1, we start by generating an initial a priori optimized tour using the local search described in section 4. Then, at each customer's location, we check if a detour is needed. If so, the algorithm has to choose one recycling centre among the R available. To this end, a local search is performed to find an efficient sub-tour including all customers that do not have been visited yet with each of the available recycling centres. The best sub-tour obtained by the R local searches is chosen to replace the current sub-tour. An example is illustrated in Fig. 3.

6 Computational Results

The two approaches described above were implemented in Java, and executed on AMD A10-7700K Radeon R7, 3.40 GHz With 8 GB RAM.

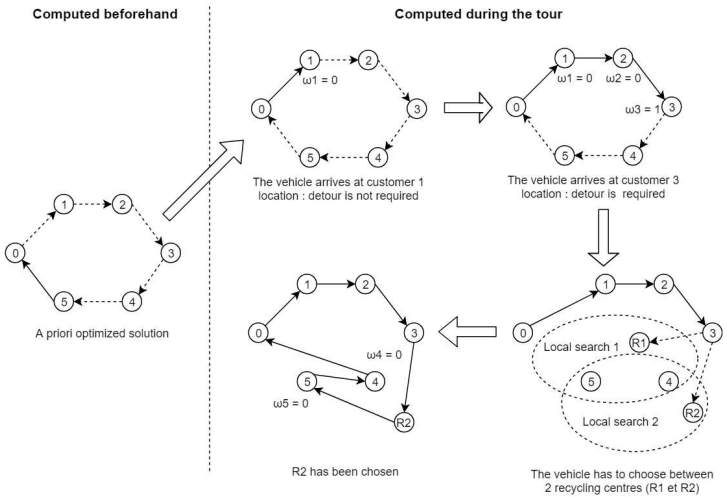


Fig. 3. Dynamic routing example

We tested the performance of our algorithms on the Euclidian PDTSP instances generated by [6]. The number of customers in these instances varies between 25 and 200. The first four customers of each instance have been chosen to be the recycling centres, the remaining nodes are assumed to be the customers. For each customer i, we determined whether a recycling detour is required after visiting i or not (we determined an "effective scenario" for each instance). The boolean variables were generated according to a fixed probability P. We generated scenarios for $P \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. Then, our algorithms were evaluated according to the fixed scenarios.
Furthermore, the first approach (a priori optimization) was tested with different values for the parameter M, the number of simulated a posteriori solutions (see section 4.3). Note that for $M = 0$, the algorithm doesn't simulate a posteriori solutions. It is then equivalent to a classic local search which ignores stochastic information.
Table 1 shows the average solution cost obtained by the a priori optimization approach for the instances described above (we fixed the neighbourhood size to 200).

First, we observe that, for each class of instances, solution costs increase as the parameter p increases. This is due to the fact that a higher probability p involves a greater risk of requiring detours and thus, a greater risk of increasing the solution cost. However, we can see that this increase is smaller as the number of a posteriori solutions generated (M) is greater.
Fig. 4 shows the percentage of travel distance due to detours when $p = \{0.1, 0.3, 0.5, 0.7, 0.9\}$, and for $M = \{0, 25, 50, 100\}$. The results show the effectiveness of our approach in minimizing the detours' impact on the solution cost, especially when $p \geq 0,5$. Indeed, the objective of our estimation-based local search is to anticipate pos-

Table 1. A priori optimization solutions for the Euclidian PDTSP instances

		Number of sample solutions M			
Probability	Number of customers	0	25	50	100
$p = 0.1$	25	**544,26**	555.9	551,5	565,4
	50	**843,33**	895	893.1	860.6
	75	1121,1	1119	1138,66	**1111,5**
	100	**1414,56**	1432,5	1494	1457,5
	150	2007,07	2017	2013	**1993,3**
	200	2603,73	2601.1	2496	**2489,16**
$p = 0.3$	25	618,5	596	**574,26**	577,3
	50	1020,03	1018,2	1051,3	**1001,1**
	75	1446,9	1474,4	1398,12	**1392,8**
	100	1874,26	1862,2	**1813.7**	1841,1
	150	2673,83	2641,14	2594	**2569,4**
	200	3592,26	3617,14	3504,2	**3495,6**
$p = 0.5$	25	706,66	688,3	681.13	**680,6**
	50	1281,33	1210,15	1187	**1161,7**
	75	1848,86	1817,21	1832,2	**1804,4**
	100	2233,96	2214,1	2157	**2149,4**
	150	3416,86	3411,14	**3378,9**	3386.6
	200	4445,53	4431,37	4376,1	**4348,4**
$p = 0.7$	25	782,06	771	**734,3**	746,2
	50	1480,83	1535,2	1457,5	**1447,3**
	75	2172	2267,3	2169	**2125,2**
	100	2666,5	2517,3	2500,1	**2491,7**
	150	4047,16	4006,7	4011,6	**3992,8**
	200	5490,2	5397,3	**5325,8**	5332,3
$p = 0.9$	25	867,8	804,4	**786,4**	799,1
	50	1682,06	1633,9	**1526,1**	1589,1
	75	2472	2480,4	2366,5	**2306,2**
	100	3108,8	3116,5	2915,2	**2903**
	150	4741,8	4886,3	4605,36	**4552,52**
	200	6457,8	6384,8	6376,6	**6301,1**

sible detours and take them into account when generating a priori solutions. Therefore, the more detours may occur during a tour, the more interesting our approach is. In other words, and as we can observe in Table 1, our estimation-based heuristic always obtains the best solutions in comparison with the classic local search (the one with $M = 0$) when $p > 0,1$. Moreover, the results are generally better when $M = 100$. Thus, more a posteriori simulations gives generally a more accurate evaluation of an a priori solution.

Table 2 shows a comparison between the a priori approach and the online approach (for the first approach, we fixed $M = 100$ since Table 1 shows better results for $M = 100$).

The online optimization heuristic (OOH) clearly outperforms the a priori optimization heuristic (APOH), regardless of the detour probability considered.

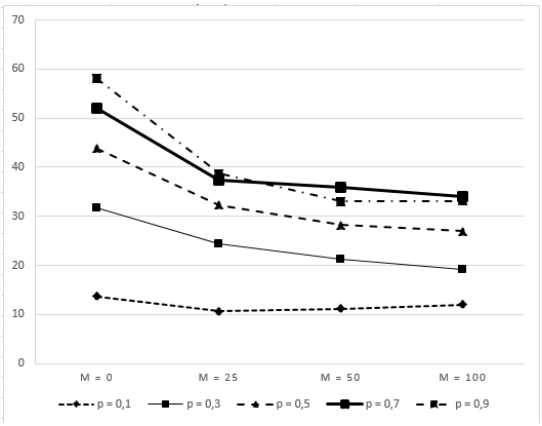


Fig. 4. Detour travel distance / Total travel distance (%)

Thus, among all the instances tested, the OOH obtained better results than APOH in 80% of the cases, and in 100% of the cases for large instances ($n \geq 75$). This is due to the flexibility of this approach which actually changes the tour to include the recycling centres, contrary to APOH which only tries to anticipate future detours and minimize their impact.

7 Conclusions

We presented in this paper two approaches to tackle a stochastic one-commodity pickup & delivery travelling salesman problem. The first approach is an a priori optimization heuristic. It is an estimation-based local search which exploits probabilistic information about possible detours that may occur during a tour to build static vehicle tours that minimize the total travel distance by minimizing the impact of these unexpected detours. The second approach is an online optimization heuristic which starts with an initial a priori tour and updates it dynamically during the tour, whenever it is necessary, to minimize the total travel distance.
We tested our algorithms on the Euclidian PDTSP instances proposed in [6]. We adapted the instances to fit our constraints and collected the results considering different parameter values. The experiments show the effectiveness of our approaches in minimizing the loss of quality due to unexpected detours, especially when detours are more likely to occur.
Future works will be devoted to the development of other approaches exploiting other types of metaheuristics, and including other constraints such as time windows, multiple vehicles...

Table 2. A priori optimization VS Online optimization

Probability	Number of customers	A priori optimization	Online optimization
$p = 0.1$	25	565,4	**533,16**
	50	860,6	**803**
	75	1111,5	**1043,6**
	100	1457,5	**1218.36**
	150	1993,3	**1613,86**
	200	2489,16	**2091,16**
$p = 0.3$	25	**577,3**	619,6
	50	**1001,1**	1051,5
	75	1392,8	**1310,16**
	100	1841,1	**1686,46**
	150	2569,4	**2381,63**
	200	3495,6	**3004,4**
$p = 0.5$	25	**680,6**	699,43
	50	**1161,7**	1245,4
	75	1804,4	**1678,7**
	100	2149,4	**2146**
	150	3386.6	**3017,73**
	200	4348,4	**4050,2**
$p = 0.7$	25	**746,2**	766,83
	50	1447,3	**1414,93**
	75	2125,2	**2038,33**
	100	2491,7	**2486,8**
	150	3992,8	**3709,5**
	200	5332,3	**5013,5**
$p = 0.9$	25	**799,1**	864,16
	50	1589,1	**1578,4**
	75	2306,2	**2287,23**
	100	2903	**2887,73**
	150	4552,52	**4472,4**
	200	6301,1	**6015,7**

References

1. Berbeglia, G., Cordeau, J.F., Gribkovskaia, I., Laporte, G.: Static pickup and delivery problems: a classification scheme and survey. Top **15**(1), 1–31 (2007)
2. Berbeglia, G., Cordeau, J.F., Laporte, G.: Dynamic pickup and delivery problems. European journal of operational research **202**(1), 8–15 (2010)
3. Bertsimas, D.: Probabilistic combinatorial optimization problems. Ph.D. thesis, Massachusetts Institute of Technology (1988)
4. Birattari, M., Balaprakash, P., Stützle, T., Dorigo, M.: Estimation-based local search for stochastic combinatorial optimization using delta evaluations: a case study on the probabilistic traveling salesman problem. INFORMS Journal on Computing **20**(4), 644–658 (2008)
5. Chen, J.F., Wu, T.H.: Vehicle routing problem with simultaneous deliveries and pickups. Journal of the Operational Research Society **57**(5), 579–587 (2006)

6. Gendreau, M., Laporte, G., Vigo, D.: Heuristics for the traveling salesman problem with pickup and delivery. Computers & Operations Research **26**(7), 699–714 (1999)
7. Hernández-Pérez, H.: Traveling salesman problems with pickups and deliveries. Disertation, University of La Laguna, Spain (2004)
8. Hernández-Pérez, H., Salazar-González, J.J.: A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. Discrete Applied Mathematics **145**(1), 126–139 (2004)
9. Hernández-Pérez, H., Salazar-González, J.J.: Heuristics for the one-commodity pickup-and-delivery traveling salesman problem. Transportation Science **38**(2), 245–255 (2004)
10. Hoff, A., Løkketangen, A.: Creating lasso-solutions for the traveling salesman problem with pickup and delivery by tabu search. Central European Journal of Operations Research **14**(2), 125–140 (2006)
11. Kenyon, A.S., Morton, D.P.: Stochastic vehicle routing with random travel times. Transportation Science **37**(1), 69–82 (2003)
12. Pillac, V., Gendreau, M., Guéret, C., Medaglia, A.L.: A review of dynamic vehicle routing problems. European Journal of Operational Research **225**(1), 1–11 (2013)
13. Ritzinger, U., Puchinger, J., Hartl, R.F.: A survey on dynamic and stochastic vehicle routing problems. International Journal of Production Research **54**(1), 215–231 (2016)

A Hybrid Approach for Data-based Models Using a Least-squares Regression*

Malin Lachmann[1 [0000−0002−6313−4415]] and Christof Büskens[1 [0000−0001−7385−4670]]

Center for Industrial Mathematics, University of Bremen, Bibliothekstr. 5, 28359 Bremen, Germany mlachmann@uni-bremen.de, bueskens@math.uni-bremen.de

Abstract. An increased use of renewable energy could significantly reduce greenhouse gas emissions but is difficult to realize since most renewable energy sources underlie volatile availability. Making use of storage devices and scheduling consumers to times when energy is available allows to increase the amount of renewable energy that can be used. For this purpose, adequate models for forecasting the energy generated and consumed as well as for the behavior of storage devices are essential. Many data-based modeling approaches are computationally costly and therefore difficult to apply in real-world systems. Hence we present a computationally efficient modeling approach using a least-squares regression. Besides, we propose to use a hybrid model approach and evaluate it on real-world data at the examples of modeling the state of charge of a battery storage and the temperature inside a milk cooling tank. The experiments indicate that the hybrid approach leads to better forecasting results, especially for modeling more complicated behavior. Also, it is investigated if the behavior of the models is qualitatively realistic and we find that the battery model fulfills this requirement and is thus suitable for the application in a smart energy management system. Even though forecasts for the hybrid milk cooling model have even lower error values than the ones for the battery storage, further steps need to be taken to avoid undesired effects when using this model in such a sophisticated system.

Keywords: Data-based modeling · Least-squares regression · Hybrid models · Multiple models

1 Introduction

Even though its greenhouse gas emissions are decreasing, the energy supply sector is still the sector causing most of these environmentally hazardous emissions [1]. Increased use of renewable energy sources could reduce those emissions and thus allow climate change to decelerate. However, most of these sources have volatile availability. On the one hand, there are times where the demand

* This research is based on a project funded by the Federal Ministry for Economic Affairs and Energy of Germany (project title SmartFarm, project number 0325927).

is higher than the availability, on the other hand, if the demand is low at times of high availability, the grid stability might be in danger. To still allow increasing the use of renewable energy sources, the installation of storage devices can help to absorb this undesired behavior. In this context, storage devices are not limited to electrical storages but could also include devices that can be used as thermal storage such as cooling systems or heat pumps. Locally installed smart energy management systems can now allow using such storage devices in an optimal way. For this purpose, models that forecast the local energy generation and consumption as well as the behavior of storage devices are essential.

Modeling approaches are usually classified into two groups: Physics-based and data-based modeling [2, 3]. The first aims at finding a model by analyzing the underlying physical laws and requires a deep understanding of the dependencies in the system while the latter determines a model by data for input and output values that is recorded during a training horizon and highly depends on the quality of this data. Their biggest advantage is that they are transferable to many different devices. For data-based approaches, it is often distinguished between models based on statistical methods and techniques using artificial intelligence (AI) [4]. AI techniques include fuzzy regression models, artificial neural networks and support vector machines while examples for statistical methods are (linear) regression models and autoregressive and moving average models. All these techniques are often used to model energy generation, consumption or storage behavior. In [5] and [6], for instance, forecasts for the energy generation are made by applying regression methods and a neural network, respectively. In [7–9], models for batteries' states of charge are determined with a neural network together with a Kalman filter, using a neuro-fuzzy inference system and a resistor-capacitor model.

In this work, we extend the data-based technique used in [10–12] such that the model can forecast complex behavior better. To achieve that, a data-based method based on a least-squares regression is combined with a hybrid model approach as introduced in [13] allowing multiple models for one device, each identified on and valid for a subset of the data. This paper investigates the application of these approaches on real-world data and evaluates if hybrid models are likely to be more plausible, i.e. to show more realistic qualitative model behavior.

In Section 2 the modeling approach is presented and the extension to hybrid models is explained. The subsequent section deals with numerical results of applying this modeling approach to real-world data using the examples of a battery storage and the temperature inside a milk cooling tank. In Section 4 this work is closed with a conclusion and possible future work is outlined.

2 Data-based Modeling Approach

One approach to data-based modeling is to fit the data by applying a least-squares regression. This method has two advantages over other data-based techniques. First, it allows fast computation of models even for large data sets which

is very convenient in case calculations need to be done on locally installed hardware with little computational power. Second, the method can be extended to include an even more efficient adaptation of the models to new data without recalculating the model on the entire data set, but only on the newly acquired data. This method will only be sketched here since it is frequently applied e.g. in [14] where it is also explained in more detail. In [11] and [10], this method is applied to a similar problem where it is extended to determining probabilistic forecasts and analyzing the capability of adapting to new data as well as improving forecasts by taking very recent data into account. In contrast to that, the focus in this paper is on a hybrid model approach that is explained in this section.

2.1 Data-based Modeling with Least-squares Regression

When identifying data-based models using a least-squares regression, we want to find a model $f : \mathbb{R}^m \to \mathbb{R}$ that best fits to a given data set with output data $y_i \in \mathbb{R}$, $i \in \{1, \dots, n\}$, measured at n different points in time, and input data $x_i \in \mathbb{R}^m$, $i \in \{1, \dots, n\}$ measured at the same time points from m different inputs $x^1, \dots, x^m$.

According to Taylor's Theorem, such a function f can be approximated around a point $x_0 \in \mathbb{R}^m$ by its derivatives at that point if it is sufficiently smooth. Since the function is not given, the derivatives which determine the coefficients of the polynomial are not available. Nevertheless, assuming a normally distributed error, the coefficients can be determined as minimizers of the mean-square deviation between the model $f(x_i)$ and the measured output y_i at all times. This problem can be reformulated as a linear least-squares problem which can be solved by QR-decomposition very efficiently even for large data sets [15].

2.2 Introducing Hybrid Models

In [10], it is found that high polynomial degrees often result in an overfitted model, i.e. one that fits very well to the training data but does not generalize well. It is also mentioned that if the model might be extrapolated to a bigger data range, a polynomial degree of one is best to use. However, a low polynomial degree does often not allow to model complex behavior. Using *hybrid models* is a possible approach to reduce the chance of overfitting but still allow modeling complex dependencies. For more details on this, we refer to [13].

Since the behavior of the device to be modeled might be very different in different phases, i.e. in $\kappa \in \mathbb{N}$ different subsets of the data set, we want to identify κ different models on the respective subsets, each valid only for these data subsets. If we denote the κ disjoint data subsets as $X_1, \dots, X_\kappa$, then we

can write our model as

$$f(x_i) = \begin{cases} f_1(x_i) & \text{if } x_i \in X_{k_1} \\ f_2(x_i) & \text{if } x_i \in X_{k_2} \\ \vdots & \vdots \\ f_\kappa(x_i) & \text{if } x_i \in X_{k_\kappa}. \end{cases}$$

There exist many different approaches for choosing these subsets (see [13]) and it can be expected that the modeling results are sensitive to the data partitioning strategy, although the investigation of the effects is not part of this paper. Our choice in the following is to divide the data into subsets depending on the value of one integer input x having κ different values. This input can be measured data, obtained by a classification algorithm or be generated by hand. When calculating a forecast for time i, it is determined to which subset X_k the point x_i belongs and the corresponding function f_k is chosen for calculating the forecasted model value at time i, that means $f(x_i) = f_k(x_i)$ if $x_i \in X_k$.

3 Experimental Results Comparing Non-hybrid and Hybrid Models on Real-world Data

3.1 Setup and Data

The modeling approaches are now evaluated on real-world data at the example of the state of charge of a battery storage and the temperature of a milk cooling tank. The data was recorded by a measurement system comprised of one- and three-phase smart meters, current terminals and 1-wire temperature sensors on two demonstration sites in Lower Saxony, Germany within the scope of a research project aiming at developing an energy management system that controls storage devices and shiftable consumers such that the use of self-produced energy is maximized.

At both demonstration sites, a photovoltaic plant produces energy that can either be locally used or exported to the grid. One of the sites is a four-person household in which a lithium-ion battery storage with a capacity of 106 A h, a usable energy of 5.0 kW h and a one-phase inverter with a maximum apparent power flow of 6 kV A is installed. Internal values from the inverter can be accessed via a modbus interface and are also used. On the other site, a milk farm, a milk cooling tank with an energy consumption of up to 13 kW is installed which can be used as thermal storage by cooling the milk to a lower temperature within constraints that guarantee no quality loss.

The data used for the battery storage was recorded on 37 days in April and May 2017 and interpolated to 30 minutes with a moving average filter to reduce noise in the measurements. The first 20 days of data are used for training a model while all other data is used for validating that model. For modeling the milk cooling tank temperature, active power and temperature data measured between February 18, 2018 and April 10, 2018 is used and interpolated to five

minutes again using a moving average filter. Here, further preprocessing was required since up to eight values per day (i.e. from 1440 values) exceeded all other values by several orders of magnitude. They were replaced by the value measured before that value since the data does often not change within a minute. The 52 days of data are again divided into a training period and a validation period where the first comprises the first 20 days of that set.

Within that setup, we now want to determine non-hybrid and hybrid models for the state of charge of the battery and the temperature inside the milk cooling tank. Both values show a dynamic behavior and depend on the state of charge or the temperature that was measured before as well as the active power which is available as a forecast in the energy management system. In the experiments, the actual measurements of the active power will be used since the quality of the power forecasts could influence the results. Also, the state of charge or temperature measured one time step before will be used as an input to the model. In addition to that, setpoints for the milk cooling tank would be determined in an energy management system indicating whether the cooling is *on* or *off*. These were reconstructed from the power data and are also used in the experiments.

Within the energy management system in which the models presented here will be applied, forecasts for 24 hours are required. Since the models use the state of charge or temperature measured one step earlier which is not available at time points in the future, we evaluate the models iteratively to simulate the models' predictions within that energy management system. This means for a forecast starting at time t_0 that is one step into the future, we use the state of charge or temperature measured one step ago which is available. For the next time step t_1 we do not have a measurement at the time t_0 to use, but instead calculate the output value at time t_1 using the forecasted value at time t_0. This value is then used as an input to forecasting the output at time t_2. This procedure is continued until the end of the forecast horizon, e.g. until values for 24 hours into the future are calculated.

Naturally, those forecasts will be more accurate during the beginning of the forecast horizon since small deviations from the data within the first hours can propagate and lead to huge differences in the last hours. Thus, the energy management system requests forecasts minutely and recalculates the optimal operation schedule to reduce this effect. However, in the following evaluations, we will consider forecasts for 24 hours since these are of interest to the energy management system. To allow clear depictions, we will simulate that forecasts are requested only at midnight.

To evaluate the quality of a model, the deviation between the forecast by a model and the actual measurements is considered during the training and the validation period separately. To measure this deviation, we calculate the *normalized root-mean-square deviation* (nRSMD) which is the root-mean-square deviation (RSMD) (often also referred to as *root-mean-square error* (RSME)) normalized to the biggest value $y_{\max}$ measured for the output y, i.e.

$$\text{nRSMD} = \frac{1}{y_{\max}} \sqrt{\frac{\sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2}{n}}.$$

3.2 Comparing Non-Hybrid and Hybrid Models on Real-world Data

Forecasting the State of Charge of a Battery Storage To identify the parameters of not only a non-hybrid, but also a hybrid model, the data for the battery storage device as mentioned above needs to be divided into subsets that show similar behavior. For this division, we add an input x^{m+1} to the data that indicates whether the battery storage is charged or not, i.e. $\kappa = 2$ and

$$x_i^{m+1} = \begin{cases} 1 & \text{if the battery storage is charged at time } i \\ 0 & \text{otherwise.} \end{cases}$$

Based on this data, one non-hybrid and one hybrid model for the state of charge of a battery storage device are determined and forecasts are calculated. An excerpt of the results is depicted in Figure 1 and shows five days at the beginning of the validation period. It can be observed that on some days the forecast calculated using the hybrid model (green) is closer to the measured data (purple) than the one with the non-hybrid model (blue) and on other days they are very similar. For both models, the forecasts do usually not reach the maximum state of charge of 90 %. This behavior can similarly be observed on the other days of the validation period indicating that the model predicts the charging process to be slower than in the measurements. The discharging, however, can be predicted better. In total, forecasts from both models are close to the measurements while the hybrid model seems to yield better predictions than the non-hybrid one.

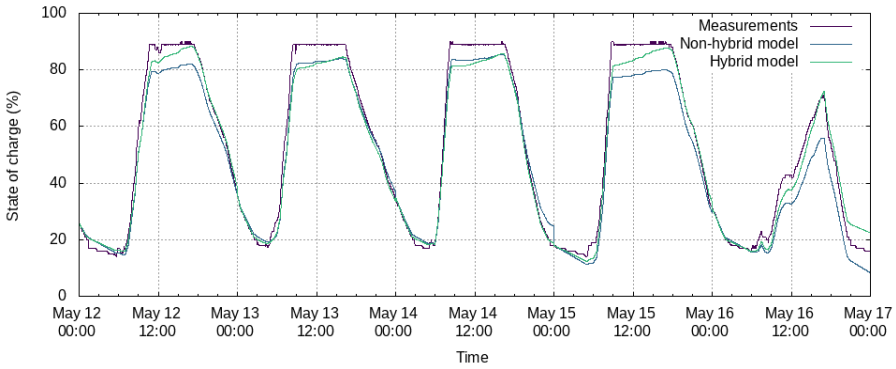


Fig. 1. Model for the state of charge of a battery storage during five days from the validation data. Measurements are depicted in purple, the results of the iteratively computed forecast by the non-hybrid model in blue and the ones obtained by the hybrid model in green.

This can also be observed in the error values of both models. The nRSMD for the non-hybrid model is 8.6 % during the training period and 6.9 % during the

validation period. This indicates that the model is not overfitted to the training data but generalizes well. The error values for the hybrid model are 8.0 % on the training data and 6.0 % on the validation data indicating that the hybrid model has the potential to improve forecasts even for devices in which a non-hybrid approach already leads to good models. Regarding the fact that the iterative forecast calculation is determined for a horizon of 24 hours, the error values here are comparable to the results of other works, e.g. [9] in which the state of charge of batteries is forecasted with an error of less than 5 %.

Forecasting the Temperature Inside a Milk Cooling Tank To forecast the temperature in the milk cooling tank, it is interesting to know the general behavior of the temperature. In the measurements, it can be observed that there is a pattern that repeats with a periodicity of two days. In Figure 2, the measurements, depicted in purple, are often constantly at about 5 °C which is the temperature at which the milk is stored. In the power data, it can be seen that the isolation of the tank allows it to keep that temperature constant without cooling after it is reached. Twice a day, pre-cooled milk is added to the tank raising the temperature to about 8 °C depending on the amount of milk in the tank. Every other day the tank is emptied and cleaned with warm water resulting in temperatures of up to 53 °C. After that, no cooling is activated and the tank is left open until the next milking. The first milking after the cleaning procedure occurs while the tank still has a temperature of about 15 °C.

From this knowledge an additional input x^{m+1} is added to the data set indicating whether it is cleaned, milk is added or none of these, i.e. $\kappa = 3$ and

$$x_i^{m+1} = \begin{cases} 1 & \text{if milk is added to the tank at time } i \\ -1 & \text{if the milk tank is cleaned at time } i \\ 0 & \text{otherwise} \end{cases}$$

while the cleaning process is considered to start once the tank is cleaned with hot water and ends when the subsequent milking starts. An addition of milk occurs when a rise in temperature can be observed even though the power indicates that the cooling is active.

Based on this indicator vector for the hybrid model, again a non-hybrid and a hybrid model for the temperature inside the milk cooling tank are determined based on temperature data from one time step earlier, active power data and reconstructed setpoints. These results are depicted in Figure 2 which shows four days from the validation period. First, it can be observed that the non-hybrid model (blue) is neither close to the measurements (purple) nor able to predict the behavior. It shows a periodicity of two days but the predicted temperature decreases when the one in the measurement increases and vice versa.

In contrast, the hybrid model is much closer to the measurements. During the cleaning process, i.e. when temperatures are above 10 °C, it fails to forecast the decrease in the data but instead predicts a constant temperature of about 25 °C. This could be since the model calculated during cleaning times is – as

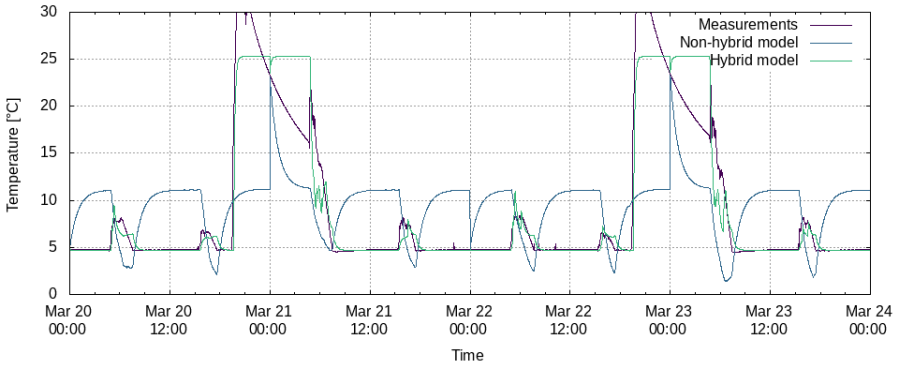


Fig. 2. Model for the temperature inside a milk cooling tank during five days of the validation data. Measurements are depicted in purple, the results of the iteratively computed forecast by the non-hybrid model in blue and the ones obtained by the hybrid model in green.

all models – linear in its inputs and can thus not model the decrease in the data but chooses an average temperature. At all other times it can be observed that the forecast qualitatively behaves as the measurement: At times where the measurement is constant, the hybrid model forecasts constant behavior at the correct temperature and at times where the temperature is changing, this change is also visible in the forecasts even though the temperature is often lower than the measurements and decreases to a constant value later than the actual measurement does.

The hybrid model outperforms the non-hybrid one also when regarding the error values. The error (nRSMD) of the non-hybrid model during the training period is 10.6 % which does not seem to be very high. However, during the validation period, the error is 16.0 % which is much higher and indicates that this model does not generalize well. The error values of the hybrid model are much lower, being 4.9 % on the training data and 6.4 % on the validation data. In [12], the error for the temperature of a milk cooling model is calculated with an error of 11 % indicating that the hybrid model is not only an improvement over the non-hybrid model but also better than what other approaches have achieved.

3.3 Checking the Models' Plausibility

After finding that the non-hybrid and hybrid models are mostly close to the actual measurements, it is of interest if they could be used in a smart energy management system. That requires that the models behave logically correct even if the storage device is operated differently than in the measurements, e.g. the battery storage could be charged or discharged at different times than in the measurement. In case of the milk cooling, a smart system could decide to cool it

to a slightly lower temperature at times of an energy surplus, such that during the next milking less energy needs to be used for cooling. To check whether the models identified above could be used in such a context, we now simulate their qualitative behavior on artificial data.

For the battery storage, it is checked how the non-hybrid and hybrid model forecast the state of charge during one day for given power values. For that, the power is set to 0 W during the first quarter of the forecast horizon, then charging at constant 200 W is simulated during the second quarter, followed by a quarter where the active power is −200 W which means that the battery is discharged. In the last quarter of the forecast horizon, the power is again set to 0 W. This power curve (black) is depicted in Figure 3 together with the forecasts generated by the non-hybrid (blue) and hybrid (green) model. Discharging and charging can be observed in the forecasts as expected. Furthermore, when the simulated active power is 0 W the state of charge also decreases, but slower than at a power of −200 W. This passive discharging, i.e. the process of slowly decreasing the state of charge even though no energy is actively used, matches the actual behavior of all battery storages. The hybrid model forecasts a slightly slower passive discharging than the non-hybrid one. The biggest difference between the non-hybrid and the hybrid model can be observed during the charging in the second quarter where both models show an increase in the state of charge which is realistic. However, the hybrid model reaches much higher states of charge, but since the constant charging with 200 W for a longer time cannot be found in the measurements, it cannot be assessed which of the models is more realistic. Nevertheless, the fact that the hybrid model forecasts higher states of charge after a charging period could explain why in Figure 1 forecasts by the hybrid model were closer to the measurements than the ones by the non-hybrid model. In summary, both models show realistic behavior here and could thus be applied in an energy management system where they would be suitable to forecast behavior that has not occurred in the data.

For the temperature inside a milk cooling tank, we simulate an additional cooling period at noon after the temperature has been constant for several hours to check if the models would predict a decrease in the temperature and stay at that lower temperature once the cooling is turned off. For simulating the additional cooling process, we set the active power to 11 850 W, the average power measured during all cooling processes, and adjust the setpoints to indicate cooling. The non-hybrid model predicts a decrease in temperature, but is, as in Figure 2, not able to predict the constant temperature. For the hybrid model, the simulated additional cooling leads to a rise in temperature once the cooling starts and the temperature falls to the constant temperature of 5 °C once the process stops. Thus both models are not able to predict an additional cooling process qualitatively correct and also other choices of the additional input x^{m+1} do not lead to better qualitative behavior. This might be since the training data does not include temperatures below 5 °C. Another explanation is that the models learn that the temperature rises once the cooling starts since at the beginning of each cooling warm milk is added. Adding data from a flow sensor as an input

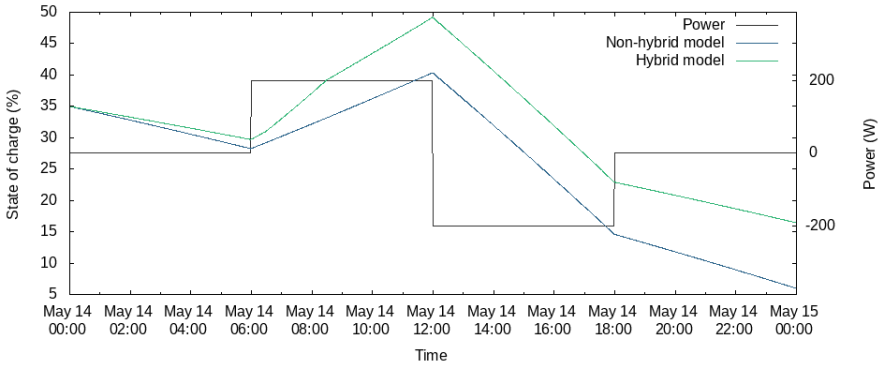


Fig. 3. Model for the state of charge of a battery storage evaluated on data simulating an active power profile as depicted by the black line. The forecast by the non-hybrid model is depicted in blue and the one obtained by the hybrid model in green.

could thus be interesting. Also, other data, such as the amount of milk inside the tank might improve the models. Furthermore, it would be interesting to choose the subsets for the hybrid model in a more sophisticated way, e.g. by a clustering approach, or to generate data containing additional cooling periods.

4 Conclusion

In this work it is evaluated to which extent computationally efficient data-based models can be applied to forecast the behavior of storage devices. The modeling approach uses a least-squares regression and is extended to hybrid models, where each submodel is trained on a subset of the data. The division into subsets is based on integer-valued indicator vectors that are added to the data manually. These two approaches are evaluated at the examples of the state of charge of a battery storage and the temperature inside a milk cooling tank. For both devices, two different models, a non-hybrid and a hybrid one, are calculated and compared to the actual measurements. It is found that the hybrid model is closer to the measurements in both cases. For the battery storage device, the error values of the models are not too far apart, but for the milk cooling tank, the non-hybrid model fails to forecast the temperature inside the tank while the hybrid model's prediction is close to the measurements. Additionally, it is investigated whether the models show plausible behavior which would be essential for their application in an energy management system. It is found that both battery models show realistic behavior while the hybrid model predicts a higher state of charge after a period of charging. In contrast, the models for the milk cooling tank both do not show plausible behavior. This can be explained by the fact that the data might lack information such as the flow of milk into the tank that influences the temperature heavily or the fact that the model is extended to data not contained in the training data.

To tackle this, adding further data would be interesting as well as a repetition of the experiments once data is available where the milk cooling tank is controlled and thus includes behavior that could not be predicted in the experiments in this paper. Additionally, it would be very interesting to evaluate if other divisions of the data into subsets lead to better results, e.g., if the data is divided into subsets by a clustering approach.

In summary, we show on real-world data that a hybrid data-based modeling approach can indeed improve forecasts calculated by models identified using least-squares regression. However, even though the hybrid models are much closer to the actual measurements, when applying them in a smart energy management system it must be carefully checked if their qualitative behavior is plausible.

References

1. World Resources Institute: World Greenhouse Gas Emissions: 2016 https://www.wri.org/resources/data-visualizations/world-greenhouse-gas-emissions-2016. Last accessed 30 Nov 2020
2. Foley, A. M., Leahy, P. G., Marvuglia, A., McKeogh, E. J.: Current methods and advances in forecasting of wind power generation. Renewable Energy **37**(1), 1–8 (2012)
3. Antonanzas, J., Osorio, N., Escobar, R., Urraca, R., Martinez-de-Pison, F.J., Antonanzas-Torres, F.: Review of photovoltaic power forecasting. Solar Energy **136**, 78–111 (2016)
4. Pedro, H. T. C., Inman, R. H., Coimbra, C. F. M.: 4 - Mathematical Methods for Optimized Solar Forecasting. In: Kariniotakis, G. (eds.), Renewable Energy Forecasting. From Models to Applications, pp. 111–152 Woodhead Publishing (2017). https://doi.org/10.1016/B978-0-08-100504-0.00004-4
5. Tsekouras, G., Dialynas, E., Hatziargyriou, N., Kavatza, S.: A Non-Linear Multivariable Regression Model for Midterm Energy Forecasting of Power Systems. Electrical Power Systems Research **77**(12), 1560–1568 (2007)
6. Li, S., Wunsch, D.C., OHair, E.A., Giesselmann, M.G.: Using Neural Networks to Estimate Wind Turbine Power Generation. IEEE Transactions on energy conversion **16**(3), 276–282 (2001)
7. Chen, Z., Qiu, S., Masrur, M.A., Murphey, Y.L.: Battery State of Charge Estimation Based on a Combined Model of Extended Kalman Filter and Neural Networks. In: Proceedings of the 2011 International Joint Conference on Neural Networks, pp. 2156–2163. IEEE, San Jose, USA (2011)
8. Cai, C.H., Du, D., Liu, Z.Y.: Battery State-of-Charge (SOC) Estimation Using Adaptive Neuro-Fuzzy Inference System (ANFIS). In: Proceedings of the 12th IEEE International Conference on Fuzzy Systems (FUZZ 03), pp. 1068–1073. IEEE, St. Louis, USA (2003)
9. Eichi, H.R., Chow, M.: Adaptive Parameter Identification and State-of-Charge Estimation of Lithium-Ion Batteries. In: Proceedings of the 38th Annual Conference of the IEEE Industrial Electronics Society, pp. 4012–4017. IEEE, Montreal, Canada (2012)
10. Lachmann, M., Maldonado, J., Bergmann, W., Jung, F., Weber, M., Bskens, C.: Self-Learning Data-Based Models as Basis of a Universally Applicable Energy Management System. Energies **13**(8), 2084 et seqq. (2020)

11. Jung, F., Büskens, C.: Probabilistic Data-Based Models for a Reliable Energy Management. In: Proceedings of the 2018 IEEE International Conference on Environment and Electrical Engineering and 2018 IEEE Industrial and Commercial Power Systems Europe, pp. 1–6. IEEE, Palermo, Italy (2018)
12. Lachmann, M., Jung, F., Büskens, C.: Computationally efficient identification of databased models applied to a milk cooling system. In: Conference of Computational Interdisciplinary Science, 2019, pp. 1–10. Galoa, Atlanta, USA (2020)
13. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Data Mining, Inference, and Prediction. 2nd edn. Springer Science & Business Media, New York, USA (2009)
14. Chen, S.; Wassel, D.; Büskens, C. High-Precision Modeling and Optimization of Cogeneration Plants. Energy Technology **4**, 177–186 (2016)
15. Hanke-Bourgeois, M.: Grundlagen der numerischen Mathematik und des wissenschaftlichen Rechnens. Vieweg+Teubner Verlag/GWV Fachverlage GmbH, Wiesbaden, Germany (2009), pp. 119 et seqq.

An Application of BnB-NSGAII: Initializing NSGAII to Solve 3 Stage Reducer Problem *

Ahmed Jaber[1,2][0000−0001−5508−1299], Pascal Lafon[1], and Rafic Younes[2]

[1] University of Technology of Troyes UTT, France
https://www.utt.fr/
[2] Lebanese University, Beirut, Lebanon
https://ul.edu.lb/

Abstract. The 3 stage reducer problem is a point of interest for many researchers. In this paper, this problem is reformulated to a bi-objective problem with additional constraints to meet the ISO mechanical standards. Those additional constraints increase the complexity of the problem, such that, NSGAII performance is not sufficient. To overcome this, we propose to use BnB-NSGAII [10] method - a hybrid multi-criteria branch and bound with NSGAII - to initialize NSGAII before solving the problem, seeking for a better initial population. A new feature is also proposed to enhance BnB-NSGAII method, called the legacy feature. The legacy feature permits the inheritance of the elite individuals between - branch and bound - parent and children nodes. NSGAII and BnB-NSGAII with and without the legacy feature are tested on the 3 stage reducer problem. Results demonstrate the competitive performance of BnB-NSGAII with the legacy feature.

Keywords: NSGAII · multi-objective · MINLP · branch-and-bound · 3-stage reducer.

1 Introduction

In [3], the design of the 3 Stage Reducer (3SR) optimization problem has been introduced to illustrate the optimal design framework of the power transmission mechanism. This problem has been a point of interest for many researchers in different domains. Engineering researchers enhance the problem for mechanical engineering applications. In [4], the problem is extended to a mixed variables optimization problem. And recently a similar problem is stated in [5] to illustrate the optimization of the volume and layout design of 3SR. Due to the problem complexity, optimization researchers are interested to test optimization methods on it. In [14], the authors use the 3SR problem to examine the performance of the constraint propagation method.

In this paper, the 3SR problem is reformulated to a bi-objective problem with additional constraints to meet the ISO mechanical standards. Those additional constraints increase the complexity of the problem, such that, the well-known Non-Dominated Sorting Genetic Algorithm 2 (NSGAII) [1] performance is not sufficient.

In [10], the authors enhance the performance of NSGAII by hybridizing it with the multi-criteria branch and bound method [12], the proposed method is called BnB-NSGAII. In this paper, we propose to use the BnB-NSGAII method to initialize NSGAII before solving the 3SR problem, seeking a better initial population. The initial population seeding phase is the first phase of any

* Supported by organization ERDF, Grand Est and Lebanese University

genetic algorithm application. It generates a set of solutions randomly or by heuristic initialization as input for the algorithm. Although the initial population seeding phase is executed only once, it has an important role to improve the genetic algorithm performance [2].

Furthermore, we propose a new feature to enhance the BnB-NSGAII method, called the legacy feature. The legacy feature permits the inheritance of elite genes between branch-and-bound nodes.

The rest is organized as follows. Section 2 presents the 3SR problem and its complexity. The proposed BnB-NSGAII legacy feature is explained in section 3. The computational results are reported in section 4. Finally, an overall conclusion is drawn in section 5.

2 3 Stage Reducer Problem

The design problem consists in finding dimensions of main components (pinions, wheels and shafts) of the 3 stage reducer (figure 1) to minimize the following bi-objective problem :

1. The volume of all the components of the reducer :

$$f_1(\boldsymbol{x}) = \pi \left(\sum_{s=0}^{s=3} l_{a_s} {r_{a,s}}^2 + \sum_{s=1}^{s=3} \left[b_s \frac{m_{\text{ns}}^2}{2} (Z_{s,1}^2 + Z_{s,2}^2) \right] \right) \tag{1}$$

2. The gap between the required reduction ratio $\bar{u}$ and the ratio of the reducer (tolerance):

$$f_2(\boldsymbol{x}) = \frac{1}{\bar{u}} \left| \bar{u} - \prod_{s=1}^{s=3} \frac{Z_{s,2}}{Z_{s,1}} \right|, \ \bar{u} > 1 \tag{2}$$

The problem is designed assuming the following are known:

- The power to be transmitted, P_t and the speed rotation of input shaft N_e.
- The total speed rotation reduction ratio $\bar{u}$, the position of the output shaft from the input shaft position (figure 2).
- The dimension of the casing box.

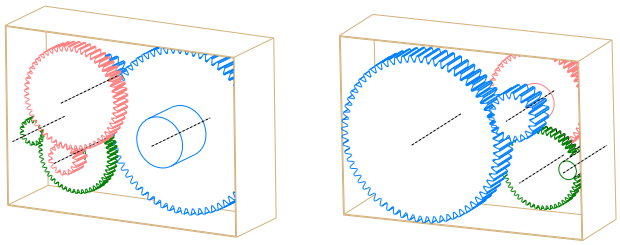


Fig. 1. Front and back view of a 3 stage reducer with closure.

The 3SR problem is formulated with 2 objective functions, 41 constraints (presented in Appendix A), 3 categorical variables (gears modules), 6 integer variables (number of teeth), and 11 continuous

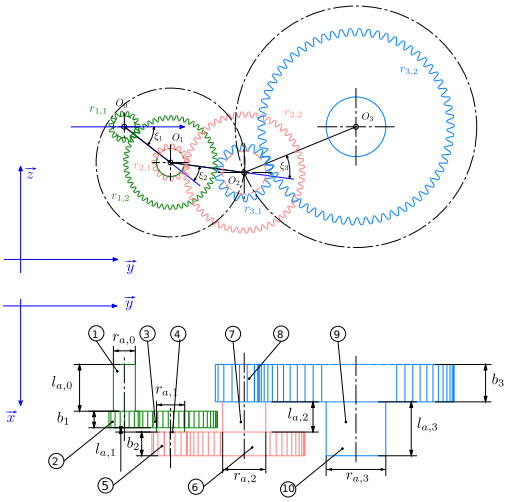


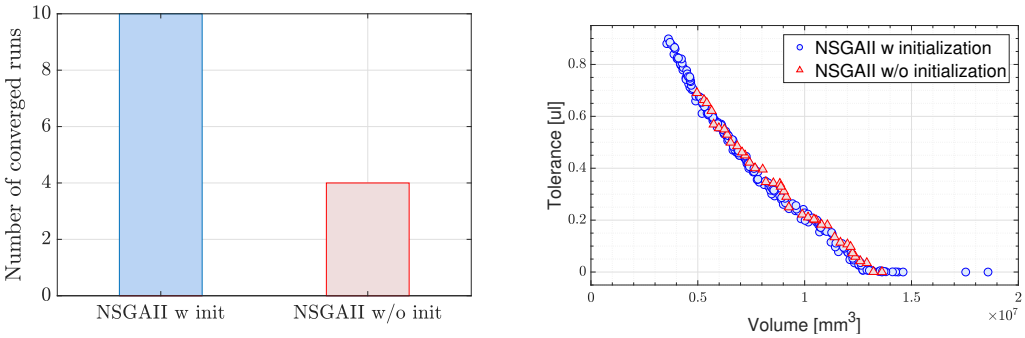
Fig. 2. Detailed view of the 3 stage reducer.

variables. Gears modules have 41 possibilities, pinion number of teeth ranges from 14 to 30 and wheel number of teeth ranges from 14 to 150. Hence, the combinatorial space of the 3SR problem consists in $41^3 + (30-14)^3 + (150-14)^3 \simeq 8.7 \times 10^{14}$. Thus, the problem is considered a mid-sized problem concerning the number of variables and constraints, but, huge combinatorial space.

The additional constraints increase the complexity of the problem. This is noticed by solving the problem using NSGAII with different initial conditions as follows. In first hand, NSGAII is initialized with 1 feasible individual. On the other hand, NSGAII is randomly initialized. Each was run 10 times with the same parameters shown in Table 1. Figure 3(a) shows how many run each method converged to a feasible solution out of 10. Figures 3(a) and 3(b) show that if the initial population contains at least 1 feasible individual, NSGAII converges to a good approximated Pareto front every time. Whilst, if NSGAII is initialized with a random population, NSGAII either fails to converge to a feasible solution, or it converges to a low-quality Pareto front.

Table 1. Parameters used for NSGAII algorithm

Parameters	**Value**
Cross over probability	0.8
Mutation Probability	0.9
Population size	200
Allowable generations	500
Constraint handling	Legacy method [1]
Crossover operator	Simulated Binary crossover (SBX) [11]
ETAC	100
Mutation operator	Partially-mapped crossover (PMX) [11]
ETAM	10



(a) Number of converged runs. (b) Best Pareto front.

Fig. 3. Results of 3SR problem solved by NSGAII with (blue) and without (red) initial feasible seed.

Figure 4 shows part of the domain of the 3SR problem explored by NSGAII with feasible initial population. The explored domain shows the complexity of the problem, where both feasible and infeasible solutions share the same domain on the projected objective domain. Moreover, all the feasible solutions are too near to the infeasible ones.

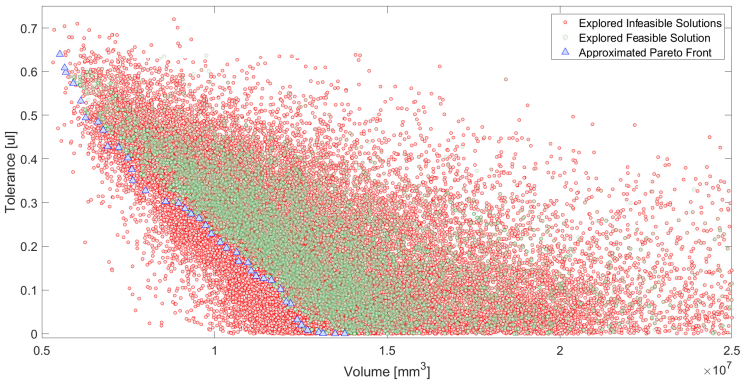


Fig. 4. Explored portion of the domain, showing the 3SR problem complexity.

To enhance the quality of the solution of this problem - and accordingly any similar problem - where feasible solutions are not known, our proposal is first to use BnB-NSGAII proposed in [10] to search for feasible individuals. These individuals are then injected in the random initial population of NSGAII.

3 BnB-NSGAII

In [10] the authors proposed the BnB-NSGAII approach. In this approach, Multi-Criteria Branch and Bound (MCBB) [13] is used to enhance the exploration force of NSGAII by investigating the mixed-integer domain space through branching it to subdomains, then NSGAII bounds each one. In this way, MCBB guides the search using the lower bounds obtained by NSGAII. Our proposal is to furthermore enhance the exploration potential of BnB-NSGAII by adding the legacy feature.

3.1 General Concept of BnB-NSGAII

The general multi-Objective MINLP problem ($\mathcal{P}_{\text{MO-MINLP}}$) is written as

$$\begin{aligned} \underset{\boldsymbol{x},\boldsymbol{y}}{\text{minimize}} \quad & \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y}) = f_1(\boldsymbol{x},\boldsymbol{y}),\ldots,f_p(\boldsymbol{x},\boldsymbol{y}) \\ \text{subject to} \quad & \\ & c_j(\boldsymbol{x},\boldsymbol{y}) \leq 0,\ j = 1,...,m \\ & \boldsymbol{x} \in \boldsymbol{X}, \qquad \boldsymbol{X} \in \mathbb{R}^{n_c} \\ & \boldsymbol{y} \in \boldsymbol{Y}, \qquad \boldsymbol{Y} \in \mathbb{N}^{n_i}, \end{aligned} \tag{3}$$

where p and m are the number of objectives and constraints respectively. $\boldsymbol{X}$ and $\boldsymbol{Y}$ denote the set of feasible solutions of the problem for n_c continuous and n_i integer variables respectively.

$\mathcal{P}_{\text{MO-MINLP}}$ is complex and expensive to solve. The general idea is thus to solve several simpler problems instead. BnB-NSGAII divides $\mathcal{P}_{\text{MO-MINLP}}$ by constructing a combinatorial tree that aim to partition the root node problem - $\mathcal{P}_{\text{MO-MINLP}}$ - into a finite number of subproblems $Pr_1,\ldots,Pr_i,\ldots,Pr_n$. Where i and n are the current node and the total number of nodes respectively. Each Pr_i is considered a node. Each node is then solved by NSGAII. Solving a node is to determine its lower and upper bounds. The upper bound of a node P_i^N is the Pareto front captured by NSGAII, which is then stored in an incumbent list P^N. Whilst the lower bound is the ideal point P_i^I of the current node.

$$P_i^I = \min f_k(\boldsymbol{x}_i,\boldsymbol{y}_i); \quad k = 1,\ldots,p. \tag{4}$$

By solving Pr_i, one of the following is revealed:

- Pr_i is infeasible, means that NSGAII didn't find any solution that satisfies all constraints. Hence, Pr_i is pruned (*fathomed*) by *infeasibility*.
- Pr_i is feasible, but, the current lower bound P_i^I is dominated by a previously found upper bound P^N. Therefore, Pr_i is fathomed by *optimality*.
- Pr_i is feasible, and, P_i^I is not dominated by P^N, $P_i^I \leq P^N$. P^N is then updated by adding P_i^N to it.

In the 3$^{\text{rd}}$ case, the combinatorial tree is furtherly branched by dividing Pr_i into farther subproblems, called children nodes. If a node cannot be divided anymore, it is called a leaf node. Leaves are the simplest nodes, since all integer variables are fixed such that $\boldsymbol{y} = \bar{\boldsymbol{y}}$. NSGAII then solve leaves as Multi-Objective continuous Non-Linear problem ($\mathcal{P}_{\text{MO-NLP}}$):

$$\begin{aligned} \underset{\boldsymbol{x},\bar{\boldsymbol{y}}}{\text{minimize}} \quad & \boldsymbol{f}(\boldsymbol{x},\bar{\boldsymbol{y}}) = f_1(\boldsymbol{x},\bar{\boldsymbol{y}}),\ldots,f_p(\boldsymbol{x},\bar{\boldsymbol{y}}) \\ \text{subject to} \quad & \\ & c_j(\boldsymbol{x},\bar{\boldsymbol{y}}) \leq 0,\ j = 1,...,m \\ & \boldsymbol{x} \in \boldsymbol{X}_i, \end{aligned} \tag{5}$$

where $\boldsymbol{X}_i$ denotes the set of feasible solutions of the current node. P_i^N of each leaf is then added to P^N. The overall Pareto front is obtained by removing the dominated elements from P^N.

3.2 BnB Legacy Feature

In NSGAII, the best population is that found in the last generation, since it contains the elite individuals among all the previous generations. In BnB-NSGAII, each node is solved independently. The output of each node is the captured Pareto front only. The last population in the node is thus discarded, although it might be valuable to other nodes.

We propose to permit the legacy between nodes. Where each child node inherits the last population from its parent node. The child node then initializes NSGAII by this population.

The children nodes are subproblems of their parent node. Thus, the boundary of parent node is different than that for the children nodes, $\boldsymbol{Y}_{parent} \neq \boldsymbol{Y}_{child}$. Hence, the population is rebounded before initializing NSGAII. Rebounding the population may lead to the loss of the elite individuals, though some of the elite genes are still conserved.

3.3 An Application of BnB-NSGAII

BnB-NSGAII is characterized by high exploration potential. Thus, in this paper, BnB-NSGAII is used to search for at least one feasible solution for the 3SR problem. For this aim, BnB-NSGAII is properly modified to 1) continue enumeration of the combinatorial tree even if the root node is infeasible. 2) stop whenever a feasible solution(s) is found. Then, NSGAII is called to solve the 3SR problem by initializing it with the feasible solution(s) found as shown in Figure 5.

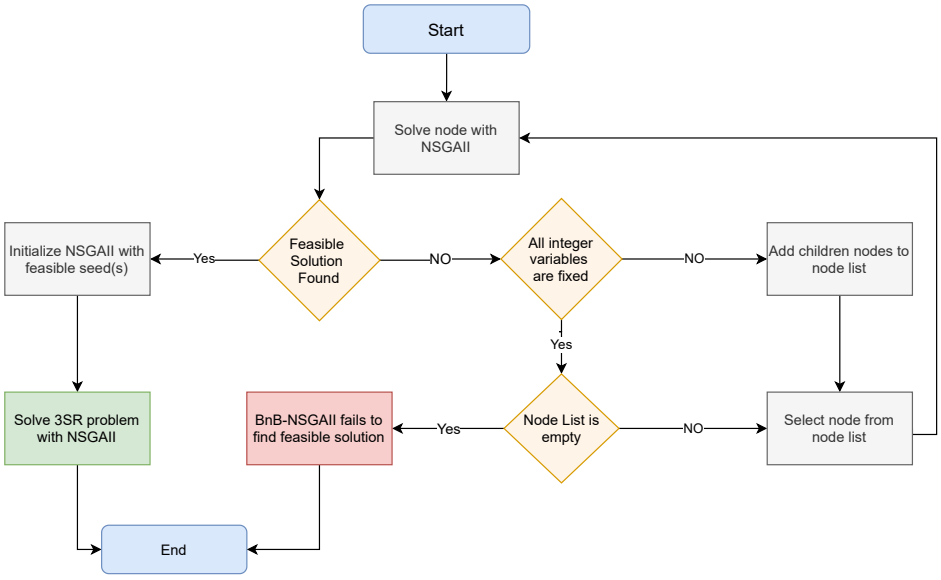


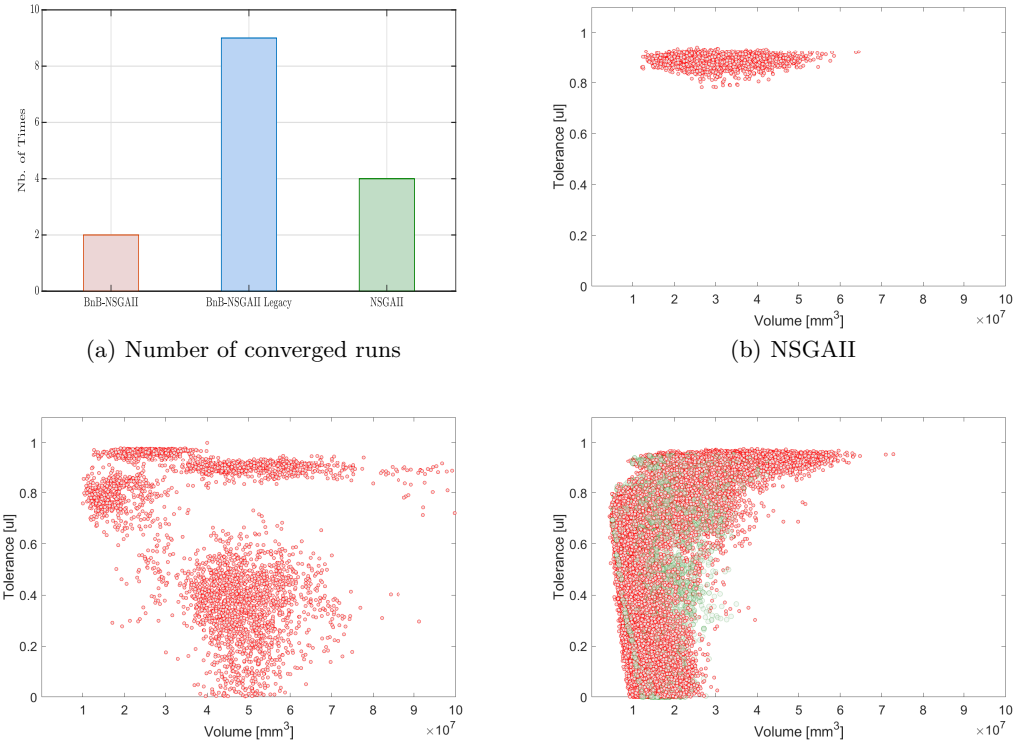
Fig. 5. Flowchart of BnB-NSGAII application.

4 Numerical Experiment

NSGAII and BnB-NSGAII with and without the legacy feature were tested on the 3SR problem. Each method was run 10 times. The test was done using the same parameters for the 3 solvers. Table 1 shows the parameters used in this experiment.

4.1 Results and Discussion

In this experiment, the evaluation of the performance of each method is limited to how many times the method finds at least 1 feasible solution over the 10 runs. Figure 6(a) shows the number of times each method succeeded the test. It can be obviously concluded that BnB-NSGAII legacy method overcomes the performances of NSGAII and BnB-NSGAII. It should be noted that the computational effort is not regarded since all the runs converge within 30 minutes. Which is considered an acceptable time for such a problem.



(a) Number of converged runs (b) NSGAII

(c) BnB-NSGAII (d) BnB-NSGAII Legacy

Fig. 6. Explored domain by (b) NSGAII, (b) BnB-NSGAII and (d) BnB-NSGAII legacy methods. Feasible and infeasible individuals are plotted in green and red respectively.

Figure 6(b) shows that NSGAII explored local space of the domain depending on the initial population. While Figure 6(c) shows that BnB-NSGAII explored random spaces of the domain. Figure 6(d) shows that the legacy feature guides the exploration force of BnB-NSGAII towards the feasible solutions.

5 Conclusion

The 3 stage reducer problem is a point of interest of many researchers, either to use/ enhance it for engineering applications, or to examine the performance of optimization methods. The 3SR problem is desirable for such experiments for its complexity.

The 3SR problem was reformulated to a bi-objective problem in this paper to demonstrate a proposed application of BnB-NSGAII. The proposed application is to use BnB-NSGAII as an initiator of NSGAII, where BnB-NSGAII initially seeks feasible individuals before injecting them into the initial population of NSGAII.

BnB-NSGAII was enhanced by adding the legacy feature. The legacy feature is a generic feature that can be implemented in any branch and bound algorithm. Any parameter that is tuned during the node solving process could be the legacy. In this paper, the legacy was the last population in the father node in BnB-NSGAII. The latter was then used to initialize the child node.

The performances of NSGAII and BnB-NSGAII with and without the legacy feature were tested on the bi-objective version of the 3SR problem. Results show that the legacy feature guides the exploration force of BnB-NSGAII leading it to a better solution than that obtained by NSGAII and BnB-NSGAII.

References

1. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (Apr 2002). https://doi.org/10.1109/4235.996017, http://ieeexplore.ieee.org/document/996017/
2. Deng, Y., Liu, Y., Zhou, D.: An Improved Genetic Algorithm with Initial Population Strategy for Symmetric TSP. Mathematical Problems in Engineering (Oct 2015). https://doi.org/https://doi.org/10.1155/2015/212794, https://www.hindawi.com/journals/mpe/2015/212794/, iSSN: 1024-123X Pages: e212794 Publisher: Hindawi Volume: 2015
3. Fauroux, J.C.: Conception optimale de structures cinématiques tridimensionnelles. Application aux mécanismes de transmission en rotation. phdthesis, INSA de Toulouse (Jan 1999), https://tel.archives-ouvertes.fr/tel-00006691
4. Fauroux, J.C., Lafon, P.: Optimization of a Multi-Stage Transmission Mechanism. In: IFToMM. vol. 2, pp. 813–817 (2004)
5. Han, L., Liu, G., Yang, X., Han, B.: Dimensional and Layout Optimization Design of Multistage Gear Drives Using Genetic Algorithms. Mathematical Problems in Engineering **2020**, 3197395 (Jan 2020). https://doi.org/10.1155/2020/3197395, https://doi.org/10.1155/2020/3197395, publisher: Hindawi
6. International Organization for Standardization (Ginebra): ISO 6336-1: Calculation of Load Capacity of Spur and Helical Gears. Part 1. ISO (2019)
7. International Organization for Standardization (Ginebra): ISO 6336-2: Calculation of Load Capacity of Spur and Helical Gears. Part 2. ISO (2019)
8. International Organization for Standardization (Ginebra): ISO 6336-3: Calculation of Load Capacity of Spur and Helical Gears. Part 3. ISO (2019)

9. International Organization for Standardization (Ginebra): ISO 6336-5: Calculation of Load Capacity of Spur and Helical Gears. Part 5. ISO (2019)
10. Jaber, A., Lafon, P., Younes, R.: A branch-and-bound algorithm based on NSGAII for multi-objective mixed integer nonlinear optimization problems. Engineering Optimization **0**(0), 1–19 (Apr 2021). https://doi.org/10.1080/0305215X.2021.1904918, https://doi.org/10.1080/0305215X.2021.1904918, publisher: Taylor & Francis _eprint: https://doi.org/10.1080/0305215X.2021.1904918
11. Maruyama, S., Tatsukawa, T.: A parametric study of crossover operators in pareto-based multiobjective evolutionary algorithm. In: Advances in Swarm Intelligence - 8th International Conference, ICSI 2017, Proceedings. pp. 3–14. Springer Verlag (Jan 2017). https://doi.org/10.1007/978-3-319-61833-3_1
12. Mavrotas, G., Diakoulaki, D.: A branch and bound algorithm for mixed zero-one multiple objective linear programming. European Journal of Operational Research **107**(3), 530–541 (Jun 1998). https://doi.org/10.1016/S0377-2217(97)00077-5
13. Mavrotas, G., Diakoulaki, D.: Multi-criteria branch and bound: A vector maximization algorithm for Mixed 0-1 Multiple Objective Linear Programming. Applied Mathematics and Computation **171**(1), 53–71 (Dec 2005). https://doi.org/10.1016/j.amc.2005.01.038
14. Yvars, P.A., Zimmer, L.: System Sizing with a Model-Based Approach: Application to the Optimization of a Power Transmission System. Mathematical Problems in Engineering **2018**, 1–14 (2018). https://doi.org/10.1155/2018/6861429

A 3SR Problem Constraints

A.1 Closure condition

Interference and fitting constraints are adopted from [5]. In [4], the closure condition was expressed with the distance between the terminal point O_3 shown in Figure 2 and required position of the center of the output shaft. The coordinate of O_3 can be easily compute with the center distance of each stage and the angle ξ_1, ξ_2 and ξ_3. But, if we consider that center distance of each stage allow this closure condition, we can compute the value of ξ_2 and ξ_3. By this way can reduce he number of variables in the optimization problem.

For a given value of ξ_1 and $r_{1,1}$, $r_{1,2}$, center distance of each stage allow a closure if we have :

$$\|O_1\vec{O}_3\| \leq \|O_1\vec{O}_2\| + \|O_2\vec{O}_3\|$$

Assuming the previous condition is true, we can compute the two intersection of circle of center O_1 of $\|O_1\vec{O}_2\|$ radius and circle of center O_3 of $\|O_2\vec{O}_3\|$ radius.

With $a_2 = \|O_1\vec{O}_2\|$ and $a_3 = \|O_2\vec{O}_3\|$ we have :

$$\begin{cases} a_2 \sin\alpha_1 - a_3 \sin\alpha_3 = 0 \\ a_2 \cos\alpha_1 + a_3 \cos\alpha_3 = \|O_1\vec{O}_3\| \end{cases}$$

which give :

$$\cos\alpha_1 = \frac{\overline{O_1H}}{\overline{O_1O_2}} = \frac{a_2^2 - a_3^2 + (O_1O_3)^2}{2a_2\overline{O_1O_3}}$$

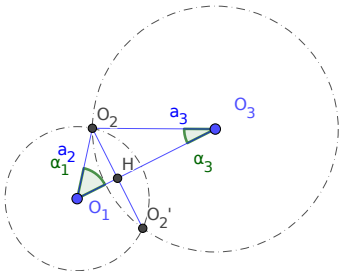


Fig. 7.

Knowing α_1, computation of coordinate of O_2 and $O_2^{'}$ is straightforward. if the two position O_2 and $O_2^{'}$ allow the wheel of the 2nd stage to fit in the casing box, then $O_2^{'}$ is preferred for lubrication reason.

A.2 Mechanical constraint for one stage of the mechanism

Constraints related to the gear pair Following the recommendation from International Standard ISO 6336, [6],[7],[8] we can calculate, knowing the geometry of gear pair, the material and the working conditions the contact and σ_{H} the bending stress σ_{F} in the gear pair. These stresses must be less of equal to the respective permissible value σ_{HP} and σ_{FP}, depending on the material and the working conditions.

From [8] the bending stress σ_{F} is given by (1 for the pinion and 2 for the wheel)::

$$\sigma_{\mathrm{F}(1,2)} = \sigma_{\mathrm{F0}} \left(K_{\mathrm{A}} K_{\mathrm{V}} K_{\mathrm{F}\alpha} K_{\mathrm{F}\beta}\right)$$

with $\sigma_{\mathrm{F0}(1,2)}$, the nominal tooth stress :

$$\sigma_{\mathrm{F0}(1,2)} = \frac{F_t}{b m_{\mathrm{n}}} \left(Y_{\mathrm{F}} Y_{\mathrm{S}} Y_{\beta} Y_{\mathrm{B}} Y_{\mathrm{DT}}\right)$$

where :

- F_t : is the tangential load from [6].
- b : is the facewidth.
- m_{n} : is the normal module.

Factors K_{A}, K_{V}, $K_{\mathrm{F}\alpha}$, $K_{\mathrm{F}\beta}$ are related to dynamic ad loading conditions in the gear. Factors Y_{F}, Y_{S}, Y_{β}, Y_{B}, Y_{DT} are related to the geometry effect on stress.

From [8], the permissible bending stress σ_{FP} is given by :

$$\sigma_{\mathrm{FP}} = \frac{\sigma_{\mathrm{FLim}}}{S_{\mathrm{Fmin}}} \left(Y_{\mathrm{ST}} Y_{\mathrm{NT}} Y_{\delta \mathrm{relT}} Y_{R\mathrm{relT}} Y_{\mathrm{X}}\right)$$

with σ_{FLim} is the nominal stress number (bending) from reference test gears [9] and S_{Fmin} the minimal required safety factor. Factors Y_{ST}, Y_{NT}, $Y_{\delta \mathrm{relT}}$, $Y_{R\mathrm{relT}}$, Y_{X} are related to the reference test gears and the geometry and material conditions of the gear pair.

From [7] the contact stress is given by (1 for the pinion and 2 for the wheel):

$$\sigma_{\mathrm{H}(1,2)} = Z_{(\mathrm{B,D})}\sigma_{\mathrm{H0}}\sqrt{K_{\mathrm{A}}K_{\mathrm{V}}K_{\mathrm{H}\alpha}K_{\mathrm{H}\beta}}$$

with σ_{H0} is the nominal contact stress :

$$\sigma_{\mathrm{H0}} = (Z_{\mathrm{H}}Z_{\mathrm{E}}Z_{\varepsilon}Z_{\beta})\sqrt{\frac{F_t}{bd_1}\frac{u+1}{u}}$$

Factors Z_{H}, Z_{E} ,Z_{ε}, Z_{β} are related to the Hertzian theory of contact, and take into account geometry and material in the gear pair.

From [7] the permissible contact stress σ_{HP} is :

$$\sigma_{\mathrm{HP}} = \frac{\sigma_{\mathrm{HLim}}}{S_{\mathrm{Hmin}}}(Z_{\mathrm{NT}}Z_{\mathrm{L}}Z_{\mathrm{V}}Z_{\mathrm{R}}Z_{\mathrm{W}}Z_{\mathrm{X}})$$

with σ_{HLim} is the allowable contact stress number and S_{Hmin} is the minimum required safety factor for surface durability. Factors Z_{NT}, Z_{L}, Z_{V}, Z_{R} ,Z_{W}, Z_{X} are related to lubrication conditions, surface roughness and hardened conditions and size of the tooth.

So to respect the requirement specification of a given power to be transmitted, the gear pair must respect :

$$\sigma_{\mathrm{F}(1,2)} \leq \sigma_{\mathrm{FP}}$$
$$\sigma_{\mathrm{H}(1,2)} \leq \sigma_{\mathrm{HP}}$$

Considering that σ_{F} is proportional to F_t and σ_{H} is proportional to $\sqrt{F_t}$ for a given gear pair, we can rewrite these 2 conditions with P_t the power to be transmitted :

$$\frac{\sigma_{\mathrm{FP}}}{\sigma_{\mathrm{F}(1,2)}}P_t \geq P_t$$
$$\left(\frac{\sigma_{\mathrm{HP}}}{\sigma_{\mathrm{H}(1,2)}}\right)^2 P_t \geq P_t$$

Usually, some factors are slightly for the pinion and the wheel so transmitted power is different for the pinion (1) and the wheel (2). We will keep the minimal value.

So finally, for the stage number s on the reducer, the following conditions must be fulfilled :

$$\min\left(\frac{\sigma_{\mathrm{FP}s}}{\sigma_{\mathrm{F}(1,2)s}}\right)P_t \geq P_t \tag{6}$$
$$\min\left(\frac{\sigma_{\mathrm{HP}s}}{\sigma_{\mathrm{H}(1,2)s}}\right)^2 P_t \geq P_t \tag{7}$$

Following condition must be respected :

- For the transverse contact ratio : $\varepsilon_{\alpha} \geq 1.3$.
- For the minimal face width : $b \geq 0.1d_2$
- For the maximal face width : $b \leq d_1$

In order to use pinion with at least $Z_{\min} = 14$ teeth, the value of the profile shift coefficient must be adjusted to avoid gear meshing with the relation :

$$Z_{\min} \geq \frac{2(1-x_1)}{\sin\alpha_{\mathrm{n}}^2} \Rightarrow x_1 \geq 1 - Z_{\min}\frac{\sin\alpha_{\mathrm{n}}^2}{2} \Rightarrow x_1 \geq 0.1812$$

Constraint related to shaft's reducer In each of the 4 shafts of the mechanism, the transmitted torque produce shear stress. This stress must not exceed the allowable shear of the material of shafts $\tau_{\max}$. We assume here that all the shaft are using the same steel and that all shaft can be consider as beam. So, with $r_{a,0}$, the radius of input shaft, and $r_{a,s}$, $s = 1 \dots 3$ the radius of output shaft of the three stages, we have :

$$\tau_s = \frac{2C_s}{\pi {r_{a,s}}^3} \leq \tau_{\max} \text{ for } s = 1 \dots 3 \tag{8}$$

C_s is the output torque of each stage and C_e the torque on the input shaft, where $Z_{i,1}$ and $Z_{i,2}$ are the number of teeth for pinion $(_1)$ and wheel $(_2)$ of stage number i:

$$C_s = C_e \prod_{i=1}^{i=s} \frac{Z_{i,2}}{Z_{i,1}}$$

For the input shaft we have :

$$\tau_0 = \frac{2C_e}{\pi {r_{a,0}}^3} \leq \tau_{\max} \tag{9}$$

The total rotation angle between the initial section of the input shaft and the final section of the output shaft is :

$$\theta = \frac{2C_e l_{a,0}}{G\pi {r_{a,0}}^3} + \sum_{s=1}^{s=3} \frac{2C_s l_{a,s}}{G\pi {r_{a,s}}^3}$$

For some reasons (dynamic behaviour of the reducer, ...) this total rotation angle should be limited by a maximal value $\theta_{\max}$.

$$\theta \leq \theta_{\max} \tag{10}$$

Reinforcement Learning for the Pick-up & Delivery Problem

Jacopo Pierotti[1], Maximilian Kronmuller[1], J. Theresia van Essen[1], Javier Alonso Mora[1], and Wendelin Böhmer[1]

TU Delft, Delft, Netherlands
J.Pierotti@tudelft.nl, M.Kronmuller@tudelft.nl, J.T.vanEssen@tudelft.nl, J.AlonsoMora@tudelft.nl, J.W.Bohmer@tudelft.nl

1 Background

In recent years, machine learning (ML) has proved super-human capabilities in speech recognition, language translations and image classifications, to name a few [1]. Lately, more and more combinatorial optimization (CO) problems have been studied under the lens of machine learning [16]. Among CO problems, NP-hard problems are particularly interesting because solving them to optimality (via the so-called exact methods) takes exponential time. Smart decisions during the branch-and-bound process (or other exact methods) could reduce the computation time drastically ([4],[5],[6]). Nonetheless, even a state-of-the-art fine-tuned exact method could take too much time to output the optimal solution; for this reason, sometimes we need to resort to heuristics. Heuristics have the disadvantage to not return a guaranteed optimal solution (or to return it in infinite time), but they have the critical advantage to be extremely fast with respect to exact methods. Even in the field of heuristics, machine learning has brought new insights and novel ideas ([2],[3]).

When ML is used to quickly generate a single solution from an input instance it is often called an 'end-to-end' method. Among all the machine learning methods to solve for combinatorial problems end-to-end, deep reinforcement learning (RL) seems to be the most promising one so far [7]. The general idea behind reinforcement learning (for more details see [8]) is that an agent interacts with an environment in order to learn a policy according to which it will behave during the evaluation phase. At each time step (at least in the discrete case), an agent observes the state of the environment and performs an action. Based on the internal dynamics of the environment and the action of the agent, the environment will change state and the agent will observe a reward. The goal of the agent is to maximize its reward. Usually, the agent is modelled by a neural network (NN) which extrapolates features from a state in order to associate the state with a set of values (one value for each possible action); then, the evaluation policy is to choose the action which will lead to states with the highest values.

For large or even medium sized instances of many classes of CO problems, we can obtain good solutions in reasonable time only with heuristics. This is the case for most vehicle routing problems (VRPs). In general, VRPs are a class of problems where a set of vehicles has to visit a set of clients while minimizing the total travelled distance and satisfying capacity, timing and flow constraints. Among VRPs, we want to talk in more details about the travelling salesman problem (TPS), the capacitated vehicle routing problem (CVRP) and the pick-up and delivery problem (PDP).

The TSP is probably the most studied problem in CO [9] and it was among the first ones to be tackled via RL [10]. In the TSP, a single vehicle has to visit a set of clients on a fully connected Euclidean graph, where capacity and time constraints are never binding. End-to-end methods can quickly (after training) return high quality solutions for the TSP on medium sized graphs, but they cannot yet generalise to large graphs [11].

The CVRP is an interesting and practical problem where a set of homogeneous vehicles, all parked in a depot, has to visit a set of clients, each of which has a demand to be delivered. Clients may have time windows and the visits to the clients gradually consume the capacity of the vehicles. To the best of our knowledge, the state-of-the-art for solving the CVRP via RL is detailed in [12]. Their implementation is based on an actor-critic (AC) network with convolutional and recurring neural networks (CNN and RNN). An AC is a method where a neural network is used to determine the value of a state (critic) and another neural network is used to determine which action to take (actor). CNNs are a type of NN able to extrapolate local features based on the values and position of the elements of a state and RNNs are a type of NN developed to process sequences of inputs.

Finally, the PDP is a VRP problem where on a graph ($G = (N, A)$) a set of homogeneous vehicles (V) has to visit a set of clients (C) subject to time windows and capacity constraints. The main novelty being that, in the PDP, clients are determined by pairs of node; one pick-up node and one delivery. The same vehicle has to visit (in order) both the pick-up and the delivery node which both have time windows. To the best of our knowledge, there are yet no RL algorithms to solve the PDP.

These recent advancements in the field of ML for CO has shown that end-to-end methods can be (after training) as performing and fast as state-of-the-art heuristics; which raises the fundamental question: will ML based methods outperform conventional heuristics in the future?

Our opinion is that ML based methods will indeed outperform conventional heuristic and this research focuses its effort in proposing a quite general method for solving the PDP. As said by [12] while talking about possible extension of their algorithm from the CVRP to more complex VRPs: '..designing such a scheme might be a challenging task, possibly harder than solving the optimization problem itself'. We decided to investigate the design of such a novel scheme and we chose the PDP because it is one of the most rich and constrained VRP. Our hope is that, if our proposed approach works on the PDP, it will most likely work on many other VRPs.

2 Approach

Most end-to-end approaches for CO problems involve CNNs, RNNs or a combination of the two. We conjecture that, for the PDP, these are not needed or, equivalently, the effectiveness of an algorithm does not lie within those structures.

Although CNNs are an excellent tool to extract local features [13], they are useful when there is a clear ordered object structure (for instances, pixels in an image). In the PDP, and in most CO problems, there is no clear ordered object structure. Even if we introduce an arbitrary order in the sets of vehicles and customers, the problem would be permutation invariant. In fact, a permutation of the elements of either set would not change the optimal solution of the problem or its structure. In theory, local features could be extracted from the graph itself (where a structure is indeed present), but this is not a standard practice and it is not clear how graph features would propagate into solutions features.

The idea behind RNNs is to sequentially embed a sequence of inputs, where the output of an input depends also on the sequence of inputs before it. This is very useful when there is an underlying sequentiality of the problem as, for instance, in a written text [14]. It is possible to argue that in end-to-end problems there is often sequentiality. Indeed, the most commonly used method relies on choosing one action at a time, update the state, and repeat until the end of the task. Nevertheless, the majority of problems satisfy the Markov property; i.e. the distribution of future states depends only on the current state and not on past ones. For example, given a vehicle at any time instant, to choose the next node to visit, it does not matter the order in which other clients were visited previously. What matters is the vehicles current position, its remaining capacity, the unvisited clients and the set of clients required to be visited (consequences of pick-ups), all of which can be modelled into the current state. This memoryless property makes the problem Markovian; hence, we conjecture that RNNs would not increase the performance of our algorithm.

Thus, given the Markovian property of our problem and the absence of an underlying ordered structure, we decide to base our implementation on a variation of the transformer [15] without relying on CNNs or RNNs as the previous methods. The transformer, which in turn is based on the self-attention mechanism, has the property of being permutation invariant and is not sensitive to the input dimension (except for the softmax operation). The transformer is a fundamental object in the state-of-the-art language translation and has proven its effectiveness in many other fields. The transformer is composed by a series of multi-head attention mechanisms in a layer structure. In ML, attention is a powerful mechanism that allows to look at the input and generate a context vector based on how much each part of the input is relevant for the output. Doing so, the algorithm learns to isolate from a set of input features the one(s) relevant for that particular state.

We plan to implement and test a double Q-learning (DQN) algorithm [17] able to solve the PDP in a very short time (after training), where the deep layers are mostly based on the transformer architecture. We hypothesise that this algorithm will be competitive with state-of-the-art ones both for time and solution quality.

3 Conclusion

More and more CO problems are being tackled via ML techniques, and RL seems to be the most effective end-to-end method. We want to give our contribution to this field by solving the PDP, which is a complex NP-hard problem. Due to the PDP richness, we hope that a method for the PDP could quite easily generalized to other VRPs (where the Markovian property holds). Given the Markov property of the problem and the absence of an underline ordered object structure, our implementation will be based on a DQN framework with a transformer-based NN as the main deep component.

References

1. Najafabadi, Maryam M., et al. "Deep learning applications and challenges in big data analytics." Journal of Big Data 2.1 (2015): 1.
2. Smith, Kate A. "Neural networks for combinatorial optimization: a review of more than a decade of research." INFORMS Journal on Computing 11.1 (1999): 15-34.
3. Khalil, Elias, et al. "Learning combinatorial optimization algorithms over graphs." Advances in neural information processing systems 30 (2017): 6348-6358.
4. Balcan, Maria-Florina, et al. "Learning to branch." arXiv preprint arXiv:1803.10150 (2018).
5. Prouvost, Antoine, et al. "Ecole: A Gym-like Library for Machine Learning in Combinatorial Optimization Solvers." arXiv preprint arXiv:2011.06069 (2020).
6. Lodi, Andrea, and Giulia Zarpellon. "On learning and branching: a survey." Top 25.2 (2017): 207-236.
7. Bello, Irwan, et al. "Neural combinatorial optimization with reinforcement learning." arXiv preprint arXiv:1611.09940 (2016).
8. Sutton, Richard S., and Andrew G. Barto. "Reinforcement learning: An introduction". MIT press, 2018.
9. Applegate, David L., et al. "The traveling salesman problem: a computational study." Princeton university press, 2006.
10. La Maire, Bert FJ, and Valeri M. Mladenov. "Comparison of neural networks for solving the travelling salesman problem." 11th Symposium on Neural Network Applications in Electrical Engineering. IEEE, 2012.
11. Joshi, Chaitanya K., et al. "Learning TSP requires rethinking generalization." arXiv preprint arXiv:2006.07054 (2020).
12. Nazari, Mohammadreza, et al. "Reinforcement learning for solving the vehicle routing problem." Advances in Neural Information Processing Systems 31 (2018): 9839-9849.
13. O'Shea, Keiron, and Ryan Nash. "An introduction to convolutional neural networks." arXiv preprint arXiv:1511.08458 (2015).
14. Bontemps, Loïc, James McDermott, and Nhien-An Le-Khac. "Collective anomaly detection based on long short-term memory recurrent neural networks." International Conference on Future Data and Security Engineering. Springer, Cham, 2016.
15. Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017): 5998-6008.
16. Bengio, Yoshua, Andrea Lodi, and Antoine Prouvost. "Machine learning for combinatorial optimization: a methodological tour d'horizon." European Journal of Operational Research (2020).
17. Van Hasselt, Hado, Arthur Guez, and David Silver. "Deep reinforcement learning with double q-learning." arXiv preprint arXiv:1509.06461 (2015).

Automatic Synthesis of Boolean Networks from Biological Knowledge and Data

Athénaïs Vaginay[1,2][0000−0001−5062−7993], Taha Boukhobza[2][0000−0003−1046−3554], and Malika Smaïl-Tabbone[1][0000−0002−8119−2117]

[1] Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
[2] Université de Lorraine, CNRS, CRAN, F-54000 Nancy, France
athenais.vaginay@loria.fr

Abstract. Boolean Networks (BNs) are a simple formalism used to study complex biological systems when the prediction of exact reaction times is not of interest. They play a key role to understand the dynamics of the studied systems and to predict their disruption in case of complex human diseases. BNs are generally built from experimental data and knowledge from the literature, either manually or with the aid of programs. The automatic synthesis of BNs is still a challenge for which several approaches have been proposed. In this paper, we propose `ASKeD-BN`, a new approach based on Answer-Set Programming to synthesise BNs constrained in their structure and dynamics. By applying our method on several well-known biological systems, we provide empirical evidence that our approach can construct BNs in line with the provided constraints. We compare our approach with three existing methods (`REVEAL`, `Best-Fit` and `caspo-TS`) and show that our approach synthesises a small number of BNs which are covering a good proportion of the dynamical constraints, and that the variance of this coverage is low.

Keywords: Boolean Network Synthesis · Answer-Set Programming.

1 Introduction

Models of biological systems are important to understand the underlying processes in living organisms [10]. Once built, the model is an artefact that can be used to study a system through simulation. Several formalisms have been proposed to model biological systems [11], and they all have their own strengths and weaknesses. The choice of a formalism is guided by the question at hand: the best formalism is the most abstract formalism which can answer the question [3]. For example, differential equations are a formalism suited to run detailed dynamic simulations because they contain information on kinetic parameters. However, they do not scale to large systems.

Boolean Networks (BNs) are a formalism used to study complex biological systems where prediction of exact reaction times is not of interest [1]. They play a key role to understand the dynamics of biological systems and predict their disruption in case of complex human diseases [2]. The key notions of BNs

are presented in Section 2.2. BNs are built from available knowledge about the structure of the system and data about the behaviour of its components (Section 2.3). The knowledge and data are used as constraints for the BN synthesis. The automatic synthesis of BNs from biological data and knowledge is still a challenge for which several methods have been developed. In Section 3, we review three state-of-the-art approaches: `REVEAL`, `Best-Fit` and `caspo-TS`.

In Section 4, we present `ASKeD-BN`, a new automatic approach for the synthesis of BNs constrained in their structure and dynamics. We rely on the Answer-Set Programming framework to generate non-redundant BNs fulfilling the given constraints. We compare the performances of our approach with `REVEAL`, `Best-Fit` and `caspo-TS` on several biological systems with experimental and synthetic data (Section 5). Finally, we discuss the results and conclude.

2 Boolean Networks and their Synthesis

2.1 Prior Knowledge Network (PKN)

Part of the knowledge one has about a biological system is the list of components (genes, proteins...) constituting the system and how these components influence each other. Influences have a **polarity**: activation (polarity "+") or inhibition (polarity "−"). The **parents** of a component X are the components which influence X. A **Prior Knowledge Network** (PKN) encodes this knowledge. The nodes of the network are the components of the system. The edges are directed from parent components to child components and labelled "+" or "−" according to the polarity of the influences. Fig. 1 shows an example PKN for a system of three components. In this PKN, C and A are the parents of C.

- "A activates C"
- "B interacts with itself"
- "C activates A"
- "C interacts with B"
- "C inhibits itself"

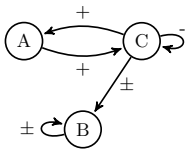


Fig. 1. PKN example of a three-components system.

2.2 Boolean Networks (BNs)

BNs were introduced by Kauffman [7] to model genetic regulatory networks. Concepts used in BNs are described in a recent review [17]. Two examples of BNs are given in Fig. 2.

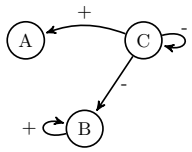
The components of a BN are the components of the considered biological system. For example, a BN modelling a system of three proteins called A, B and C has three components named A, B and C. A **configuration** of a BN

$$\mathcal{B}_1 = \begin{cases} f_{\mathrm{A}} := \mathrm{C} \\ f_{\mathrm{B}} := \mathrm{B} \wedge \neg \mathrm{C} \\ f_{\mathrm{C}} := \neg \mathrm{C} \end{cases}$$

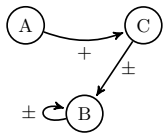
(a) Transition functions of $\mathcal{B}_1$

$$\mathcal{B}_2 = \begin{cases} f_{\mathrm{A}} := 0 \\ f_{\mathrm{B}} := (\mathrm{B} \wedge \neg \mathrm{C}) \vee (\neg \mathrm{B} \wedge \mathrm{C}) \\ f_{\mathrm{C}} := \mathrm{A} \end{cases}$$

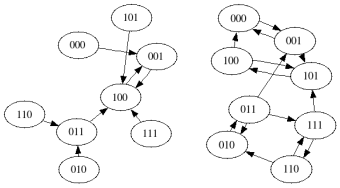
(b) Transition functions of $\mathcal{B}_2$



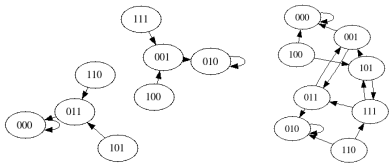
(c) Interaction graph of $\mathcal{B}_1$



(d) Interaction graph of $\mathcal{B}_2$



(e) Synchronous (left) and asynchronous (right) state transition graphs of $\mathcal{B}_1$



(f) Synchronous (left) and asynchronous (right) state transition graphs of $\mathcal{B}_2$

Fig. 2. The transition functions, derived interaction graph, and state transition graphs according to synchronous and asynchronous update schemes of two BNs.

is a vector which associates a Boolean value (1/active or 0/inactive) to each component of the BN. A BN with n components has 2^n possible configurations. For example, the $2^3 = 8$ possible configurations of a BN with 3 components are: 000, 001, 010, 011, 100, 101, 110 and 111.

Each component has an associated **transition function** ($\mathbb{B}^n \to \mathbb{B}$) which maps the configurations of the BN to the next value of the component. The transition functions are usually written as Boolean expressions. In this paper, these expressions are in Disjunctive Normal Form (DNF), *i.e.*, disjunctions of conjunctions. The conjunctions are *satisfiable*, which means they do not contain a literal and its contrary. The operators $\neg$, $\wedge$, $\vee$ represent respectively negation, conjunction and disjunction. Figs. 2a and 2b show examples of transition functions. The transition function associated with B in $\mathcal{B}_2$ states that the value of B will be 1 if either the value of B or of C was 1 in the previous configuration.

Like for the PKN, the structure of a BN is defined in terms of parent-child relationships between the components. A component P which appears in the transition function of a component X is called a **parent** of X. If the parent is negated in the DNF, we say that the *polarity* of the influence of P on X is negative. Conversely, if the parent is not negated, the polarity is positive. The **Interaction Graph (IG)** summarises these relationships as a directed graph. The directed edge P $\to$ X is labelled with "+" or "−" depending on the polarity

of the influence P has on X. For example, the IG of $\mathcal{B}_1$ contains B $\xrightarrow{+}$ B and C $\xrightarrow{-}$ B because B appears positively and C appears negatively in the transition function associated with B. As we will see in Section 2.3, the PKN will act as a hard constraint on the IG of the BNs we want to synthesise.

The BN **dynamics** is obtained by applying iteratively the transition functions starting from each possible configuration. The order of application of the transition functions is defined by the **update scheme**. The **synchronous**, **asynchronous** and **mixed** update schemes are the most commonly used. In the synchronous update scheme, the transition functions are applied all at once, while in the asynchronous scheme, they are applied one by one. In the mixed update scheme, any number of components can be updated at each step. Thus, the update possibilities from both the synchronous and asynchronous update schemes are included in this third update scheme.

The **State Transition Graph (STG)** is a directed graph whose nodes are the 2^n possible configurations of the BN. In this graph, there is a directed edge from c to c' if c' is the result of applying to c the transition function(s) according to the chosen update scheme. Fig. 2 shows examples of synchronous and asynchronous STGs. Later, we discuss how dynamical constraints are enforced in the STGs, and how we use the mixed STG to quantify how well the synthesised BNs match the dynamical constraints.

2.3 Synthesis of BNs from PKN and Multivariate TS

In general, BNs that model biological systems have to satisfy two categories of *constraints*. On one hand, the BNs have to comply with a PKN. The PKN constrains the structure of the synthesised BNs by defining which components can appear as variables in each transition function and the polarity of those variables. Hence, a component P is allowed to appear in the transition function of a component X with a polarity s if the PKN contains an edge P $\xrightarrow{s}$ X. Formally, a BN is compatible with a PKN if its IG is a *spanning subgraph* of the PKN. In other words, the IG of a BN compatible with a given PKN is formed from the vertices and a subset of the edges of the PKN. For example, the two BNs presented in Figs. 2a and 2b are compatible with the PKN given in Fig. 1. On the contrary, a BN containing the transition function $f_{\mathrm{A}} :=$ B is not, since the IG of this BN contains the edge B $\xrightarrow{+}$ A, which is not in the PKN. A BN having $f_{\mathrm{A}} := \neg$C is also incompatible: despite C being a possible parent of A, the negative polarity is not allowed, since the PKN does not contain the edge C $\xrightarrow{-}$ A.

On the other hand, the synthesised BNs are expected to reproduce as well as possible the sequence of configurations extracted from an observed continuous multivariate Time Series (TS) of the concentration of the components over time. An example of a multivariate TS is given in Table 1. Various strategies for extracting the sequence of configurations and fitting the transition functions to the observations have been proposed in the literature, but they all roughly

result in enforcing the STG of the synthesised BNs to contain specific edges, corresponding to specific transitions of configuration.

We focus here on the *automatic synthesis* of BNs that respect the structure of a given PKN and are designed to reproduce as well as possible the observations from one given multivariate TS. For each synthesised BN, this ability of reproducing the observations is measured in terms of *coverage* proportion, *i.e.*, the proportion of transitions observed in the multivariate TS that are retrieved by the BN when computing its STG according to the mixed update scheme. Ideally, an identification method would only return BNs with a perfect coverage proportion (*i.e.*, 1).

Table 1. Multivariate TS of the three-components system given as example. The continuous concentrations of the components have been sampled for 20 time steps. Here, all the observations range from 0 to 100. The value resulting from the binarisation with a threshold of 50 is indicated by the colour of the cells: green if the result of the binarisation is 1 and red if 0. The resulting binary vectors are the configurations. Here there are four configurations (010, 011, 100 and 001) lasting respectively 4, 3, 3 and 10 time steps. Vertical bars indicate a change of configuration.

	configurations sequence:																			
	010				→ 011			→ 100			→ 001									
time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A	0	3	7	13	20	30	49	61	100	63	36	25	2	3	1	1	3	0	0	0
B	100	86	64	57	54	53	51	49	45	37	33	28	22	19	14	12	9	5	2	0
C	0	27	36	42	60	75	54	44	38	48	60	72	88	90	100	100	100	100	100	100

3 State-of-the-Art Methods of BN Synthesis from PKN and TS

Several studies have been dedicated to the automatic synthesis of BNs from PKNs and observed multivariate TS. Here, we review three main state-of-the-art approaches: `REVEAL` [12], `Best-Fit` [9] and `caspo-TS` [16].

For each component of the system, `REVEAL` tests all the possible combinations of its parent nodes, and attempts to find the functions that explain *all* the observations of the binarised TS. For example with the multivariate TS from Table 1: `REVEAL` tries to explain 010 → 010 → 010 → 010 → 011 → 011 → 011 → 100 → ... Hence, it cannot handle *inconsistencies*—such as a configuration being associated to distinct successor configurations. Such inconsistencies are frequent when sampling concentrations along time, because the processes involved can have different speeds. In the example (Table 1), observing both 010 → 010 and 010 → 011 is an inconsistency which causes the failure of `REVEAL`. Furthermore, `REVEAL` cannot use the influence signs from the PKN, and since it uses an already binarised TS, it is possibly biased by the chosen binarisation.

Like `REVEAL`, `Best-Fit` tests every possible combination of the parent nodes of each component. It cannot use the influence signs and works on the binarised TS as well. Unlike `REVEAL` it can manage inconsistencies from the TS since it returns the functions that explain the maximal number of time steps. In Table 1, since $010 \rightarrow 010$ is observed three times and $010 \rightarrow 011$ only once, `Best-Fit` will focus on explaining the former instead of the latter.

`caspo-TS` was designed to manage several multivariate TS, corresponding to several experiments where the system is perturbed (forced activation or inhibition of some components), and where some measurements are potentially missing. Unlike `REVEAL` and `Best-Fit`, `caspo-TS` takes the influence signs into account, but it can only generate locally monotonous BNs, *i.e.*, BNs for which a parent of a component cannot be both its activator and its inhibitor. $\mathcal{B}_2$ is an example of a BN `caspo-TS` cannot generate because it is not locally monotonic. Indeed, in f_{B}, the components B and C act both as activator and inhibitor of B. `caspo-TS` works as the following: first, it derives the set of BNs that are compatible with the given PKN and an over-approximation of the dynamics of the TS, using the so-called most-permissive semantics [4]. Because of this over-approximation, the result can contain many false positive BNs, *i.e.*, BNs optimising the cost function used under the hood of `caspo-TS`, while their asynchronous dynamics is not able to reproduce the configurations sequence of the multivariate TS. These false positive BNs are subsequently ruled out using exact model checking. This filtering is PSPACE-hard, but thanks to the first step, a large set of BNs has already been excluded.

4 Our Approach: `ASKeD-BN`

4.1 Details of the Approach

We propose an approach for the Automatic Synthesis of Boolean Networks from Knowledge and Data (`ASKeD-BN`). It computes a non-redundant set of BNs complying with a given PKN and one observed multivariate TS. Unlike `REVEAL` and `Best-Fit`, `ASKeD-BN` is capable of using the influence signs provided in the given Prior Knowledge Network (PKN) and the raw values of the input multivariate Time-Series (TS). Unlike `caspo-TS`, `ASKeD-BN` directly fits the behaviour of each component with the TS. Also, it is not limited to the synthesis of locally-monotonous BNs.

For each component of the studied system, our approach searches among all possible transition functions. All the transition functions that do not respect the given PKN are ruled out. Then, every remaining candidate is evaluated on the basis of both their simplicity and their ability to reproduce the given observations. The candidate transition functions for the component X might not be able to explain all the binary state transitions happening at time $t \rightarrow t'$. The set of unexplained t' is denoted $\mathcal{U}$. Every time step t' in $\mathcal{U}$ is associated with a measure stating "how far" the continuous value X'_t is from the binarisation threshold θ: $|\theta - \mathrm{X}'_t|$. These spotted errors are then averaged on the T time steps of the TS

through the Mean Absolute Error (MAE):

$$\mathrm{MAE}_{\mathrm{X}} = \frac{\sum_{t' \in \mathcal{U}} |\theta - \mathrm{X}_{t'}|}{T}$$

Among the candidates having the smallest MAE, we select the ones that has the smallest number of influences. Finally, we create all the possible BNs by generating all the combinations of the selected functions.

We implemented our approach using Python and the Answer-Set Programming framework (ASP) with the system *clingo* [6]. ASP is a declarative programming language oriented towards difficult (NP-hard) search problems. The possible solutions of a problem are described with the constraints they must fulfill. These constraints are written as a logic program. The ASP solver is tasked with finding the solutions of the program. To do so, it uses a Conflict-Driven Clause Learning (CDCL) algorithm inspired by SAT solvers. In our case, the CDCL algorithm avoids the evaluation of all the possible transition functions by learning from conflicts: whenever it finds that a candidate is in conflict with the constraints, it creates a new constraint that explains the conflict. These learned constraints subsequently eliminate other conflicting candidates, pruning the search space. Thanks to these pruning heuristics, our approach is efficient. ASP and in particular *clingo*, have already been used in similar contexts including `caspo-TS`.

4.2 Illustration on the Toy Example

Let us illustrate our approach on the toy example consisting of the PKN in Fig. 1 and the multivariate TS in Table 1.

When no PKN is available, the default PKN is a complete graph assuming that each component can inhibit / activate all the others (including itself). In this setup, a component with n parents have 2^{2^n} possible transition functions. In the toy example, each component can be explained by $2^{2^3} = 256$ distinct functions, which correspond to 16 777 216 potential BNs (formed by all the possible combinations of all the candidates of each component). Thanks to the available PKN, the number of candidate functions for each components A, B and C falls respectively to 3, 16 and 6. Besides the CDCL pruning, `ASKeD-BN` virtually evaluates all the candidates, but for illustration purpose we will focus on the two that are present in $\mathcal{B}_1$ and $\mathcal{B}_2$ (Figs. 2a and 2b).

For the component A, the candidate $f_{\mathrm{A}} := 0$ does not contain any literal and it cannot explain the transition of configuration for A at $t_7 \to t_8$. Hence, the set $\mathcal{U}$ of unexplained time steps is $\{t_8\}$. The concentration of A at time t_8 is 61, and the candidate's MAE is thus $|50-61|/20 = 0.55$. The candidate $f_{\mathrm{A}} := \mathrm{C}$ involves one literal (which is C). This candidate can explain all transitions. Hence, $\mathcal{U} = \emptyset$ and the MAE associated with this candidate is 0. Despite requiring more literals, $f_{\mathrm{A}} := \mathrm{C}$ is a better candidate than $f_{\mathrm{A}} := 0$ because its MAE is smaller. The comparisons of the candidates proposed for the components B and C in $\mathcal{B}_1$ and $\mathcal{B}_2$ are summarised in Table 2.

For the toy example, our approach returns $\mathcal{B}_1$ as the only solution. It retrieves the 3 configuration transitions extracted from the binarised TS, thus its coverage proportion is 1. `REVEAL` does not find any BN, and the BN returned by `Best-Fit` does not comply with the PKN. `caspo-TS` finds 5 BNs with coverage proportions ranging from 0.33 to 1 (standard deviation of 0.25).

Table 2. Number of influences and MAE for the candidate functions in $\mathcal{B}_1$ (Fig. 2a) and $\mathcal{B}_2$ (Fig. 2b). A checkmark indicates the candidate selected by our approach, and the best for each criterion: (1) minimal MAE and (2) minimal number of influences.

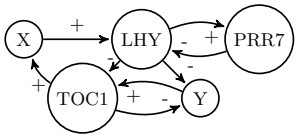
candidate	$f_B := B \wedge \neg C$	✓	$f_B := (B \wedge \neg C) \vee (\neg B \wedge C)$		$f_C := \neg C$	✓	$f_C := A$	
MAE ($\mathcal{U}$)	0 ($\emptyset$)	✓	0 ($\emptyset$)	✓	0 ($\emptyset$)	✓	0.5 ($\{t_5\}$)	
# influences	2	✓	4		1	✓	1	✓

5 Datasets and Procedure for the Comparative Evaluation

5.1 Datasets

In order to compare our approach with `REVEAL`, `Best-Fit` and `caspo-TS`, we used eight biological systems. For two of these systems (*yeast*'s cell cycle and *A. thaliana*'s circadian clock), their PKN and experimental multivariate TS are taken from [13] and [18] respectively. These two systems are summarised in Table 3. They respectively involve 4 and 5 components.

Table 3. Summary of two biological systems and their corresponding datasets

System	Genes	PKN	TS	Source
yeast (cell cycle)	Fkh2, Swi5, Sic1 & Clb1	Sic1 does not influence itself nor Fkh2	14 time steps 6 transitions	[18]
A. thaliana (circadian clock)	LHY, PRR7, TOC1, X & Y		50 time steps 11 transitions	[13]

For the six other systems[3], we conducted our experiments on multivariate TS that we simulated from existing BNs taken from the repository of example BNs of the package `PyBoolNet` [8]. For these systems, the number of components

[3] `raf`, `randomnet_n7k3`, `xiao_wnt5a`, `arellano_rootstem`, `davidich_yeast` and `faure_cellcycle`

ranges from 3 to 10. For each system, the used PKN is the IG of the associated BN. As for the generation of the multivariate TS, three parameters are taken into consideration: the update scheme (in $\{\text{synchronous}, \text{asynchronous}\}$), the maximum number of introduced repetitions of each configuration (in $\{1, 4\}$) and the standard deviation of the added noise (in $\{0, 0.1\}$). For each setting of these parameters, we follow a procedure similar to what is implemented in the `generateTimeSeries` function of the R package `BoolNet` [15]:

1. choose randomly a configuration of the considered BN,
2. on this configuration, apply the update function(s) 20 times w.r.t the chosen update scheme,
3. duplicate randomly each configuration in the obtained sequence (added in contrast to `generateTimeSeries`),
4. add a Gaussian noise with a standard deviation of N.

For a given setting of the 3 parameters and a given system, we run the procedure 7 times (with different random seeds). In the following, we denote *ARN* the setting with the Asynchronous update scheme, Repetitions (of 4) and Noise (of 0.1). We believe that this setting allows us to obtain multivariate TS which are quite close to real TS.

We illustrate here how to generate a synthetic multivariate TS in the ARN setting for $\mathcal{B}_1$ (Fig. 2a). We would start from a random configuration. Let it be 010. Then we apply 20 times the transition functions of $\mathcal{B}_1$ with the asynchronous update scheme. This process is not deterministic as any path from Fig. 2e (right) starting from 010 and of length 20 is valid. Let's say we obtain a path starting with $010 \to 011 \to 010 \to 011 \to 111 \to 101 \to \dots$ Then we add a random number of duplications (in bold). The beginning of the sequence could for example look like $010 \to 011 \to \mathbf{011} \to 010 \to 011 \to \mathbf{011} \to \mathbf{011} \to 111 \to 101 \to \mathbf{101} \to \mathbf{101} \to \mathbf{101} \to \dots$ Finally, we add a random Gaussian noise with a standard deviation of 0.1. The synthetic multivariate TS could now start with $(0.02; 0.92; -0.16) \to (0.04; 0.8; 0.7) \to (-0.05; 1.06; 0.7) \to \dots$

5.2 Details on the Evaluation Procedure

For `REVEAL` and `Best-Fit` we use the implementation from the R package `BoolNet` [15]. `caspo-TS` is ran with the option `mincard`, that asks for BNs with functions minimising the number of influences. Note that this is also what our method optimises.

In the following, we define an *experiment* as a BN identification method applied on a system with *one* multivariate TS. The unicity of the multivariate TS makes the problem under-specified and allows us to evaluate the performances of the different approaches in this context.

`REVEAL`, `Best-Fit` and our approach need the binarised multivariate TS in their inputs. We use a simple form of binarisation: the binarisation threshold is defined as $\min + (\max - \min)/2$. All values from the multivariate TS greater or equal to the threshold are binarised to 1, and to 0 otherwise. For the two

systems with real TS, the theoretic range of the values is not know in advance, so the binarisation threshold is determined component-wise: the components are binarised taking into account their observed minimum and maximum. For the six systems with the synthetic multivariate TS, we know *a priori* that the values of all the components are between 0 and 1 ($\pm$ the noise). In case of noisy data, the fluctuations of a constant component are interpreted as state changes when using a threshold computed component-wise. However, the identification methods are not capable to detect these spurious transitions in the binarised TS. Hence, we compute the binarisation threshold globally, on all the observations of all the components.

In order to have a fair comparison of the methods, and since `caspo-TS` is making the binarisation itself and is not aware that the theoretical minimum and maximum of the components are 0 and 1 ($\pm$ the noise), we correct *a posteriori* the transition functions it returned. The value of the constant is set to the binarised value that is the most present in the binarised TS of the component concerned. Also, since `caspo-TS` does not return a function for the components without parents in the PKN nor for the components that it founds constant for all the TS (in the case where no noise is involved), we use the same technique to set the transition functions to their correct values. We also added a step to filter out BNs returned by `REVEAL` and `Best-Fit` which do not respect the polarities given in the PKN.

For all the BNs returned by the four methods (and after the PKN-based filtering for `REVEAL` and `Best-Fit`), we use `PyBoolNet` [8] to compute the STG of each retrieved BN according to the mixed update scheme. Finally, we evaluate the results of each experience according to three criteria:

- the number of BNs returned;
- the median of the coverage ratios: the proportion of configuration transitions extracted from the input TS that are present in the mixed STG;
- the standard deviation of the coverage ratios.

All data and programs needed to reproduce the presented results are accessible at `https://gitlab.inria.fr/avaginay/OLA2021`.

6 Results

6.1 Results on Systems with Real PKN and Experimental Multivariate TS

Yeast (Fig. 3 left). For this system `caspo-TS` find 61 BNs while `Best-Fit` and `ASKeD-BN` both find 16 BNs. As for `REVEAL`, due to inconsistencies in the TS, it does not return any BN. Concerning the coverage, on the 7 transitions observed in the TS, the BNs synthesised by `Best-Fit` recover 4 and the BNs synthesised by `ASKeD-BN` recover five. The best coverage ratio (6 retrieved transitions over 7) is obtained for 8 BNs synthesised by `caspo-TS` (among the total of 61). Nevertheless, as the box plot shows, the BNs synthesised by `caspo-TS` present a large variance in their coverage.

A. thaliana (Fig. 3 right). For this system, `REVEAL` returns no BN. The only BN returned by `Best-Fit` has all the components set to 1 and recovers 5 transitions over the 10 observed. `ASKeD-BN` also returns a single BN with a perfect coverage since the BN recovers all the 10 transitions. As for the 5 BNs synthesised by `caspo-TS`, we can make the same observation as before: they present a variability in their coverage. The best coverage obtained by `caspo-TS` are from 2 different BNs including the one synthesised by `ASKeD-BN`.

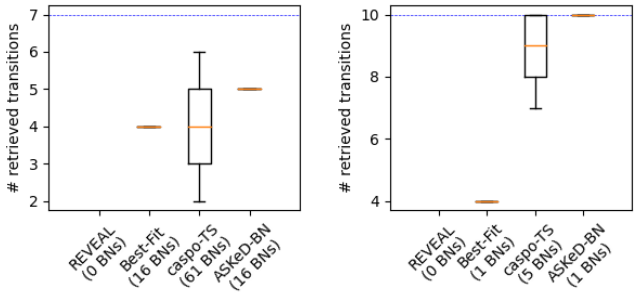


Fig. 3. Number of transitions retrieved by the BNs synthesised using the different methods on the systems *yeast* (left) and *A. thaliana* (right). The blue dashed line indicates the number of transitions that were observed in the multivariate TS.

To sum up, the results on these two real examples show that:

- `REVEAL` constantly fails to return any BN. At the opposite, `caspo-TS` returns more BNs than the other methods;
- the coverage of the BNs returned by both our approach and `caspo-TS` are better than for `Best-Fit`;
- `caspo-TS` presents worse variability in the coverage ratio of its BNs compared to our approach.

6.2 Results on Systems with Generated Multivariate TS

Number of Synthesised BNs: The total number of BNs returned on the synthetic datasets and the number of times the identification methods failed returning any BNs are reported in Table 4. The table shows that a large proportion of BNs generated by `REVEAL` and `Best-Fit` were not complying with the influence signs from the input PKN. The following reported results do not take into account these non-compliant BNs. `REVEAL` is the method which returns the smallest number of BNs, in particular in the ARN setting. This is due to the inconsistencies in the TS, which are frequent in the ARN setting (as in real TS). On the opposite, `caspo-TS` is the method that returned the largest number of BNs. Moreover, when considering all experiments, there are 18 experiments for which `caspo-TS`

generated more than 100 BNs. In these cases, we stopped the enumeration and analysed the 100 first BNs `caspo-TS` returned. Despite this limit, `caspo-TS` returned between 5 and 7 times more BNs than our method.

Table 4. Number of experiments for which each method failed to return any BN, number of BNs returned over all 336 experiments with synthetic TS and number of BNs returned over the 42 experiments with the ARN setting. The labels "before" and "after" refers to the filtering step which rules out the BNs not respecting the signs of the given PKN (see Section 5.2).

measure (setting)		`REVEAL` before	`REVEAL` after	`Best-Fit` before	`Best-Fit` after	`caspo-TS`	`ASKeD-BN`
# failing experiments	(all)	230	240	0	64	20	0
# BNs returned	(all)	100 677 500	406	100 678 198	724	8481	1210
# BNs returned	(ARN)	3	3	51	35	720	85

From here on, we focus on the results of the experiments corresponding to the ARN setting (Asynchronous update scheme, random Repetition of configurations, and Noise addition) after having remove the BNs from `REVEAL` and `Best-Fit` which does not respect the given PKN.

Coverage ratio: To assess the coverage ratio criterion, instead of plotting the boxplots for the 42 experiments of this setting (6 systems times 7 replicates), we summarised them in Fig. 4. In the scatter plot, each experiment is represented by a point whose coordinates are the coverage ratio median of the synthesised BNs and the associated standard deviation (std). The more top-right a point is, the better the corresponding identification method is (*i.e.*, it produces BNs with high coverage ratio and low std). We can see that for the few experiments for which `REVEAL` was able to return BNs, the median coverage is actually excellent. The median coverage of the BNs returned by `Best-Fit` is almost uniform: `Best-Fit` lacks regularity in finding BNs with good coverage. But the high pick around 0 on the plot of std distribution shows that for a given experiment, the BNs returned by `Best-Fit` have similar coverage rates. `caspo-TS` and our approach have a very similar distribution of median coverage. They are both good at finding BNs with very good coverage. But here again, for a given experiment, the BNs synthesised by `caspo-TS` present a bigger variation of their coverage proportions than the ones synthesised by our approach.

7 Conclusion and Perspectives

We presented `ASKeD-BN`, a novel method to create BNs from a PKN and a multivariate TS. The results on 8 biological systems showed that our approach has the best trade-off on the evaluation criteria: it returns a small set of BNs with a high coverage median and low variance. Our results actually confirm

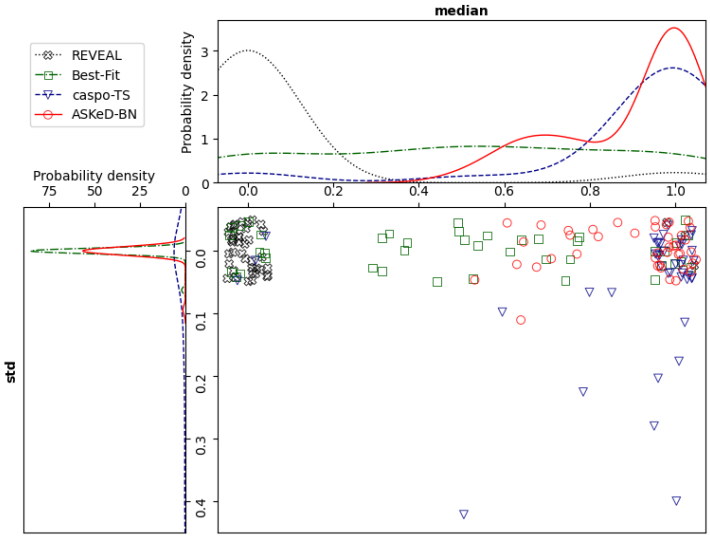


Fig. 4. On the scatter plot, each point represents an experiment in the ARN condition for which a given method potentially returned several BNs with different coverage ratios. The horizontal coordinate of the point is the median of these ratios. The vertical coordinate is their standard deviation (std). For a better visualisation, the coordinates have been jittered with a variance of 0.1 on both axes. The curves on the top (resp. on the left) of the scatter plot are the probability densities of the median (resp. the std) of the points in the scatter plots. The densities have been estimated from the non-jittered coordinates of the points with the Gaussian kernel density estimation method. The smoothing parameter of the estimator was determined automatically (with the Scott method). The areas under all these curves are 1, and the picks show where the points are the most concentrated.

that although `caspo-TS` finds good BNs, too many sub-optimal BNs are also retrieved. Indeed a new version of `caspo-TS` was recently proposed to tackle this problem [5].

We now present two perspectives to improve our approach and the study. First of all, real datasets may contain outlier measurements which could mislead the computation procedure of the binarisation thresholds we used in this paper. It would be interesting to see how such cases impact the performances of the identification methods and to propose a better binarisation procedure with prior outliers detection for instance. Second, contrarily to `REVEAL`, `Best-Fit` and `caspo-TS`, our approach does not handle multiple multivariate TS. However, biologists often have several multivariate TS generated with perturbations forcing some components to stay either active or inactive. However, exploiting such supplementary data gives more information about the behaviour of the studied system in specific conditions (*e.g.*, pathological states). This knowledge allows to constrain even more the space of solutions.

Finally, we are currently working on an automatic pipeline for BN synthesis from a curated mathematical model repository, namely BioModels [14]. This requires (i) automatic extraction of the PKN from the model structure encoded in the SBML[4] file format and (ii) generation of a multivariate TS by simulation of these models.

Acknowledgements We thank Julie Lao and Hans-Jörg Schurr for their valuable comments and suggestions.

References

[1] R. Albert and J. Thakar. "Boolean Modeling: A Logic-Based Dynamic Approach for Understanding Signaling and Regulatory Networks and for Making Useful Predictions". In: *Wiley Interdisciplinary Reviews. Systems Biology and Medicine* 6.5 (2014), pp. 353–369. DOI: 10.1002/wsbm.1273.

[2] C. Biane, F. Delaplace, and T. Melliti. "Abductive Network Action Inference for Targeted Therapy Discovery". In: *Electronic Notes in Theoretical Computer Science* 335 (2018), pp. 3–25. DOI: 10.1016/j.entcs.2018.03.006.

[3] S. Bornholdt. "Less Is More in Modeling Large Genetic Networks". In: *Science* 310.5747 (2005), pp. 449–451. DOI: 10.1126/science.1119959.

[4] T. Chatain, S. Haar, J. Kolčák, and L. Paulevé. *Most Permissive Semantics of Boolean Networks*. Research Report. 2020. URL: https://hal.archives-ouvertes.fr/hal-01864693.

[5] S. Chevalier, V. Noël, L. Calzone, A. Zinovyev, and L. Paulevé. "Synthesis and Simulation of Ensembles of Boolean Networks for Cell Fate Decision". In: *Computational Methods in Systems Biology*. Ed. by A. Abate, T. Petrov, and V. Wolf. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 193–209. DOI: 10.1007/978-3-030-60327-4_11.

[6] M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub. *Answer Set Solving in Practice*. Morgan & Claypool Publishers, 2012. ISBN: 978-1-60845-971-1.

[7] S. A. Kauffman. "Metabolic Stability and Epigenesis in Randomly Constructed Genetic Nets". In: *Journal of Theoretical Biology* 22.3 (1969), pp. 437–467. DOI: 10.1016/0022-5193(69)90015-0.

[8] H. Klarner, A. Streck, and H. Siebert. "PyBoolNet: A Python Package for the Generation, Analysis and Visualization of Boolean Networks". In: *Bioinformatics* (2016). DOI: 10.1093/bioinformatics/btw682.

[9] H. Lähdesmäki, I. Shmulevich, and O. Yli-Harja. "On Learning Gene Regulatory Networks under the Boolean Network Model". In: *Machine Learning* 52.1 (2003), pp. 147–167. DOI: 10.1023/A:1023905711304.

[4] Systems Biology Markup Language

[10] Y. Lazebnik. "Can a Biologist Fix a Radio?—Or, What I Learned While Studying Apoptosis". In: *Cancer Cell* 2.3 (2002), pp. 179–182. DOI: 10.1016/S1535-6108(02)00133-2.

[11] N. Le Novère. "Quantitative and Logic Modelling of Molecular and Gene Networks". In: *Nature Reviews Genetics* 16.3 (2015), pp. 146–158. DOI: 10.1038/nrg3885.

[12] S. Liang, S. Fuhrman, and R. Somogyi. "REVEAL, a General Reverse Engineering Algorithm for Inference of Genetic Network Architectures". In: *Pacific Symposium on Biocomputing.* (1998), pp. 18–29. ISSN: 2335-6928.

[13] J. C. W. Locke et al. "Experimental Validation of a Predicted Feedback Loop in the Multi-Oscillator Clock of Arabidopsis Thaliana". In: *Molecular Systems Biology* 2.1 (2006), p. 59. DOI: 10.1038/msb4100102.

[14] R. S. Malik-Sheriff et al. "BioModels—15 Years of Sharing Computational Models in Life Science". In: *Nucleic Acids Research* 48.D1 (2020), pp. D407–D415. DOI: 10.1093/nar/gkz1055.

[15] C. Müssel, M. Hopfensitz, and H. A. Kestler. "BoolNet—an R Package for Generation, Reconstruction and Analysis of Boolean Networks". In: *Bioinformatics* 26.10 (2010), pp. 1378–1380. DOI: 10.1093/bioinformatics/btq124.

[16] M. Ostrowski, L. Paulevé, T. Schaub, A. Siegel, and C. Guziolowski. "Boolean Network Identification from Perturbation Time Series Data Combining Dynamics Abstraction and Logic Programming". In: *Biosystems* 149 (2016), pp. 139–153. DOI: 10.1016/j.biosystems.2016.07.009.

[17] J. D. Schwab, S. D. Kühlwein, N. Ikonomi, M. Kühl, and H. A. Kestler. "Concepts in Boolean Network Modeling: What Do They All Mean?" In: *Computational and Structural Biotechnology Journal* 18 (2020), pp. 571–582. DOI: 10.1016/j.csbj.2020.03.001.

[18] P. T. Spellman et al. "Comprehensive Identification of Cell Cycle-Regulated Genes of the Yeast Saccharomyces Cerevisiae by Microarray Hybridization". In: *Molecular Biology of the Cell* 9.12 (1998), pp. 3273–3297. DOI: 10.1091/mbc.9.12.3273.

Reinforcement Learning-Based Adaptive Operator Selection

Rafet Durgut[1][0000−1111−2222−3333] and Mehmet E. Aydin[2][0000−0002−4890−5648]

[1] Karabuk University, Engineering Faculty, Computer Engineering Dept., Turkey
rafetdurgut@karabuk.edu.tr
[2] UWE Bristol, Dept. of Computer Science and Creative Technologies, Bristol, UK
mehmet.aydin@uwe.ac.uk

Abstract. Metaheuristic and swarm intelligence approaches require devising optimisation algorithms with operators to let produce neighbouring solutions to conduct a move. The efficiency of algorithms using single operator remains recessive in comparison with those with multiple operators. However, use of multiple operators require a selection mechanism, which may not be always as productive as expected; therefore an adaptive selection scheme is always needed. In this study, an experience-based, reinforcement learning algorithm has been used to build an adaptive selection scheme implemented to work with a binary artificial bee colony algorithm in which the selection mechanism learns when and subject to which circumstances an operator can help produce better and worse neighbours. The implementations have been tested with commonly used benchmarks of uncapacitated facility location problem. The results demonstrates that the selection scheme developed based on reinforcement learning, which can also be named as smart selection scheme, performs much better that state-of-art adaptive selection schemes.

Keywords: Adaptive Operator Selection · Reinforcement learning · Artificial Bee Colony · Uncapacitated Facility Location Problem (UFLP).

1 Introduction

Metaheuristic and swarm intelligence algorithms have gained a deserved popularity through the success accomplished over last few decades. Although they do not guarantee globally optimal solutions within a reasonable time, the success in offering useful near-optimum solutions within an affordable time has helped gain such credit. This does not mean that metaheuristic and swarm intelligence algorithms can be seamlessly implemented for a productive algorithmic solution. The main shortcoming arises in handling local optima capabilities, which enforces researchers to build a balance in exploration for new and fresh solutions while exploiting the gained success level within the search space. That is known as Exploration versus Exploitation (*EvE*) rate in the literature [5]. *EvE* rate guides to search through as many neighbourhoods as possible while retaining exploitation of achieved success and gained experience for a better performance,

where weaker exploration causes falling in local optima while weaker exploitation would cause higher fluctuations in performance [13].

Metaheuristic approaches, especially population-based ones, use neighbourhood functions, also known as operators, to let the search process identify next solutions to move to. It is conceivable that search with single operators would have higher likelihood to stick in a local optima than multiple operators. Many hybridisation approaches and memetic algorithms have been designed to help diversify the search through a balanced *EvE*, which usually appear in the form of using multiple operators subject to a selection scheme. The idea an operator to apply after another would prevent the search falling in local optima contributing to diversification of the search. It appears that the nature of the operators to be applied in an order and the order managed in use play very important role in the success level of the algorithms. Adaptive operator selection schemes have been studied for a while to achieve a useful balance in *EvE* and level of diversification in search [12].

Adaptive operator selection is a process of two phases; (i) *credit assignment* in which the selected operators are credited based on the level of success measured, or (ii) *operator selection* in which an operator is identified to run based on the credit level in order to produce a neighbour [11]. The amount of credit to assign is decided using either the positive difference achieved in fitness values or the categories of success or fail [10]. Credit assignment phase also covers the calculation of the time window in which the amount of credit to assign to selected operators is estimated [4]. On the other hand, operator selection phase imposes prioritisation/rank of operators within a pool of functions/operators. *Probability Matching* (PM), *Adaptive Pursuit* (AP) and *Upper Confidence Bound* (UCB) are known to be among state-of-art operators selection schemes [4].

Adaptive operator selection schemes have been used in the literature with evolutionary algorithms and swarm intelligence. Failho et. al [9] uses a multi-armed bandits approach with genetic algorithms, while Durgut and Aydin [7] comparatively studied the success of PM, AP, and UCB schemes to supply a binary artificial bee colony algorithm. Yue et. al. [19] proposes a self-adaptive particle swarm optimisation algorithm adaptively selecting among 5 operators to solve large scale feature selection problems.

Adaptive operator selection schemes estimate likelihood of each operator within the pool relying on credits gained to the time. The selection happens through the estimated likelihoods irrespective of the problem state in hand. It is clear that the success of selected operator is not sensitive to the problem state; whether it is in a harsh neighbourhood or trapped in a difficult local optima or not. Reinforcement learning (RL) gains more and more popularity day-by-day to solve dynamic problems progressively, gaining experiences through problems solving process [3, 17]. There are renown powerful RL algorithms let map input sets to outputs through experiencing the the problems states and collecting environmental responses to the actions taken [20].

In this study, an artificial bee colony (ABC) algorithm has been implemented for solving uncapacitated facility location problems (UFLP) represented in bi-

nary form. ABC algorithms have been implemented to solve many real-world engineering problems. Among them are combinatorial optimisation problems, which formulated as binary optimisation problems. ABC can be viewed as multi-start hill-climbing algorithms in optimisation, where new neighbouring solutions are generated with operators as discussed above. In this study, the ABC algorithm is furnished with multiple operators selected with reinforcement learning-based selection scheme.

The rest of this paper is organised as follows; Adaptive operator selection schemes are introduced in Section 2, the operator selection scheme developed based on reinforcement algorithm is explained in Section 3. Experimentation and results are presented and discussed in Section 4 while conclusions are briefed in Section 5.

2 Adaptive Operator Selection

One of the common problem of heuristic-based optimisation algorithms is that search is inevitably driven into local optima, which sometimes remains as the offered final solution. The aim of use multiple operator is to help rescue the search from local optima by the means of diversifying search using different neighbourhood functions/operators interchangeably or systematically. Operator selection schemes are used for this purpose.

Operator selection is not necessarily to be adaptive by nature, but, most of recent studies have been developed as adaptive to insert smartness in the process of selection. Metaheuristic and evolutionary approaches can come up with self-imposing operator selection. Evolutionary algorithms such as genetic algorithms and genetic programming have self-contained probabilistic operator selection while metaheuristics such as variable neighbourhood search imposes a systematic count-based operator change mechanism to achieve diversity in search and manage neighbourhood change. Operator selection built-in algorithms do not offer much flexibility in working with multiple operators, while memetic algorithms, hill-climbing style heuristic algorithms and modern swarm intelligence algorithms allow customising operator selection mechanism to engineer bespoke efficient optimisation algorithms.

Adaptive operator selection is the process of prioritisation of the operators based on merits, which can be imported in the algorithms via crediting each operator based on achievements gained. Although there are a number of adaptive operator selection schemes studied, the general mechanism is depicted in Fig. 1 in which a two phase process is run; (i) *operator selection* and (ii) *credit assignment*. As suggested, the pool of operators holds a finite number of operators to select an operator from in order to produce neighbours to move to, while the selected operators is credited upon its action and success level it achieves in producing new solutions. The credit level to assign to the selected operator is estimated based on preferred rules.

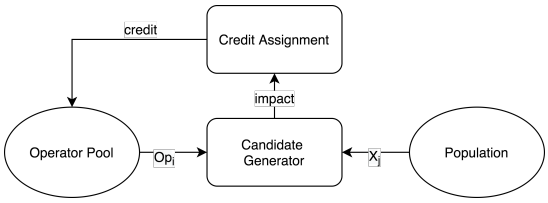


Fig. 1. General overview of adaptive operator selection process with support of population and pool of operators

2.1 Operator Selection

The first phase of operator selection process is to execute the selection rule imposed by operator selection scheme in order to produce neighbouring solutions to move to. The main aim is to keep a *EvE* rate as balanced as possible so that the search to be intensified within the neighbourhood as long as it produces positively and to be diversified as soon as it turns to negative productivity. Literature reports a number of operator selection schemes; random selection, merit-based selection, probability matching, adaptive pursuit and multi-arm bandit approaches, e.g. upper confidence bound (UCB). Random selection chooses an operator from the pool completely randomly, Roulette-wheel takes the success counts of each operator into account to calculate a probability-based prioritisation, while probability matching (PM) approach accounts the success as merits and lets to increase the selectability of non-chosen operators using the following rule:

$$p_{i,t} = p_{min} + (1 - Kp_{min})\frac{q_{i,t}}{\sum_{j=1}^{K} q_{j,t}}, \quad i = 1, 2..K \tag{1}$$

where K is the number of operators in the pool, $p_{min} \in [0, 1]$ represents the minimum probability of being selected, and $q_{i,t}$ is the credit level/value of operation i at time t. Both PM and AP use p_{min} to set a base probability for each operator, which would help address the *EvE* dilemma with allocating a minimum chance to every operators to be selected. PM imposes to calculate the probabilities of being selected per operation, while AP uses the strategy of "*winner takes all*" approach that credits more to promising options. adaptive pursuit (AP) calculates the probabilities with Eq. 2.

$$p_{i,t} = \begin{cases} p_{i,t-1} + \beta(p_{max} - p_{i,t-1}), & \text{if } i = i_t* \\ p_{i,t-1} + \beta(p_{min} - p_{i,t-1}), & \text{otherwise} \end{cases} \tag{2}$$

Both of PM and AP impose higher dominance for exploitation, which is aimed to decrease by UCB using the following rule, which selects the operator with highest probability.

$$p_{i,t} = \begin{cases} 1 - p_{min} * (K - 1) & \text{if } i = i_t* \\ p_{min}, & \text{otherwise} \end{cases} \tag{3}$$

where K is the number of operators in the pool, $p_{min} \in [0, 1]$ represents the minimum (base) probability for being selected, i_t* is calculated with 4.

$$i_t* = \arg \max_{i=1,..,K} \{q_{i,t} + C \times \sqrt{\frac{2 \log \sum_{j=1}^{K} n_{j,t}}{n_{i,t}}}\} \tag{4}$$

where op_t represents the selected operator, C works as a scaling factor, n is number of times the operator selected while $q_{i,t}$ and $n_{i,t}$ on the right-hand-side of equation help control *EvE* dilemma, respectively.

2.2 Credit Assignment

The next phase of adaptive operator selection process is to estimate a credit to be assigned to the operator just used. This involves how to estimate the amount of reward to assign and what to be the base for estimate of a credit. Literature suggests that mainly two classes of approaches have been implemented; whether a success has been achieved or not, or how much positive difference accomplished. The former approach considers if the result is "success" or "fail", while the latter processes the amount of achievement in quantity to estimate the level of reward to assign.

The process of credit assignment entails clarifying the time window with which the reward level is to be estimated. The time window can span from last single step to a pre-defined number of previous steps in which the credit level and/or the achievement level can be averaged. This reveals that a credit can be decided as *instant* credit, an *averaged* credit or the *maximum* credit.

3 Proposed Approach: Adaptive selection with Reinforced-Clusters

Operator selection adaptively developed and used for higher efficiency in diversification of search process. The operator selection schemes, even the adaptive ones, propose choosing an operator based on credits gained over the success counts through out the search, but, regardless of the input sets, the problem state, and search circumstances. The merit-based schemes usually select operators through a blind process, where the total gained credit is relied on regardless of the status of search etc. It is known that operators do not always produce success due to their limitations; each performs better under some circumstances, while does worse in other circumstances. Once the fruitful circumstances are ascertained for each operator, a complementary policy can be customised for deliberative selection to achieve success.

This study aims to propose a more conscious selection process developed based on reinforcement learning approach implemented into a distance-based clustering algorithm in which the distance in between the input set and the fine-tuned cluster centres is estimated and made reference index in operator selection. The idea of setting up a selection scheme based on clusters is discussed

and implemented in machine learning studies. Reinforcement learning is known to be very useful in handling dynamically changing environment and for solving dynamic problems, particularly for operating within unknown dynamic environments. One of earlier studies proposes embedding reinforcement learning in a distance-based clustering algorithm, namely hard-c-means algorithm, to train agents to select the best scheduling operator subject to dynamic production environments to solve dynamic scheduling problems [2]. Inspiring of this study, a reinforced-clustering algorithm is put together to optimise the cluster centres so that the problem states can be classified with optimised clusters, where each cluster will correspond to an operator. The algorithm will impose selecting the cluster centre, operator, closer to the input set in distance. This will facilitates a selection scheme conscious with problem state.

Operators are selected based on probabilities, $p_{i,t}$, calculated as in Eq. 3, where the best operator is determined using Eq. 5. The other operators are also prioritised based on the distance in between the problem state at time t, $\mathbf{x_t}$, and the cluster centres, $\mathbf{c_t}$ - corresponding to the operators. Here, the distance metric used in this study is *hamming* distance due to the binary representation of the problem and the operators.

$$i_t* = \arg \min_{i=1,..,K} \{\beta q_{i,t} + \gamma e_i(x_t)\} \tag{5}$$

where $q_{i,t}$ is the credit level/value of operation i at time t, while $e_i(x_t) = \|\mathbf{x_t} - c_i\|$, the estimated distance between an input set and cluster c_i, β and γ are coefficients to balance between credit and distance metrics. Note that unlike other methods, the reward value of good solutions is reflected as negative.

4 Experimental Results

The reinforced-clustering-based operator selection scheme has been tested with a binary ABC algorithm to solve uncapacitated facility location problem (UFLP) instances, which is one of well-known NP-Hard combinatorial problem. The details of UFLP benchmarking instances taken form OR-Library can be found in many articles [1, 8].

The problem solving algorithm to use reinforced-clustering-based operator selection scheme is chosen as the standard artificial bee colony (ABC) algorithm reported in [14]. The standard ABC is designed for continuous numerical optimisation problems, while UFLP is a combinatorial optimisation problem represented in binary form [18]. The algorithm has been rearranged to work with state-of-art binary operators; *binABC* [16] and *ibinABC* [6] work on the basis of XOR logical operator and *disABC* [15] uses a hamming distance-based binary logic.

Algorithm 1 presents a pseudo code of ABC algorithm embedded with reinforced-cluster-based operator selection scheme implemented for UFL problems. As seen, ABC imposes a three-phase process to evolve a swarm (population) of solutions. The first phase exploits *employed* bees to generate new solutions with selected

Algorithm 1 The pseudo code of binary ABC embedded with reinforced-cluster based operator selection scheme

```
1: Initialisation phase:
2: Set algorithm parameters
3: Create initial population
4: while Termination criteria is not met do
5:     Employed bee phase:
6:     Select operators and assign to bees
7:     for i=1 to N do
8:         Select neighbour, apply operator and obtain candidate solution (v_i)
9:         if f(v_i) is better than f(x_i) then
10:            Replace v_i with x_i
11:            Get reward and add to r_{op,t} and update centroid of c_{op,t}
12:            Reset trial counter
13:        else
14:            Increment trial counter
15:        end if
16:    end for
17:    Onlooker bee phase:
18:    Calculate probabilities for food sources
19:    Select operators and assign to bees
20:    Increment operator counter, t=0
21:    for i=1 to N do
22:        Determine current solution according to probability
23:        Select neighbour food source
24:        Apply operator and obtain candidate solution (v_c)
25:        if f(v_c) is better than f(x_c) then
26:            Replace v_c with x_c
27:            Get reward and add to r_{op,t} and update centroid of c_{op,t}
28:            Reset trial counter
29:        else
30:            Increment trial counter
31:        end if
32:    end for
33:    Update Phase:
34:    Credit assignment
35:    Memorisation
36:    Scout bee phase:
37:    if Limit is exceed for any bee then
38:        Create random solution for the first exceeding bee and evaluate it
39:    end if
40: end while
```

binary operators applying to the materials taken from a selected solutions and one of its neighbours. The generated solution is added to the swarm if it is better than the parents, the amount of reward to allocate to the operators is estimated and the position of centre for selected and used operator is updated. If the the generated new solution is not better than the parent solution no reward is generated and the trail counter is incremented.

The *onlooker* bees conduct the second phase of ABC in which the solutions are selected with a probabilistic approach to let randomness contribute the diversity of the swarm. Similar to the first phase, the operator selection, the reward estimation and crediting are performed and the corresponding cluster centres are updated. The *scout* bees follow up the *onlookers* to replace from non-improvable solutions with randomly generated ones to keep the swarm further divers.

The experimentation has started with parametric study to fine-tune parameters used in both the algorithm and within the mechanics of the operator se-

lection scheme. The experimentation for parametric study has been conducted using the hardest benchmarking instance of UFL problem, which is known as CapC. The parameters configured for best fit are tabulated in Table 4 and averaged over 30 repetitions.

Table 1. Parameter configurations tested

Parameter	Values		
Reward	Inst	Avrg	Max
P_{min}	0.10	0.20	0.30
W	10.00	25.00	50.00
β	0.01	0.05	0.10
γ	0.10	0.50	0.90

Table 2 presents the *hit* metric, which is the number of trails attained the optimum. The best performance so far is 25 hits out of 30 trails, where $\gamma = 0.5$, $\beta = 0.01$ and $P_{min} = 0.1$ are found and setup. Next, the reward estimation across a time/iteration window is fine-tuned, where the parametric study results obtained for *average* and *extreme* rewards are tabulated in Table 3. The best hit values are obtained *25* and *27* out of *30* trails for *average* and *extreme* reward cases. respectively.

Table 2. Parameter tuning for Instant reward measured with *hit* metric

γ	P_{min}								
	0.1			0.2			0.3		
	β			β			β		
	0.01	0.05	0.1	0.01	0.05	0.1	0.01	0.05	0.1
0.1	24	16	20	18	24	24	24	24	24
0.5	25	21	19	19	19	21	24	19	14
0.9	16	21	14	21	21	14	17	21	17

The window size (W) of *25* and *50* produce best results, while all trails are tested with $P_{min} = 0.1$, $\beta = 0.05$ and $\gamma = 0.1$. The averaged achievements conclude that $W = 25$ produces the best configuration.

The best configuration concluded out of parametric study has been run with hardest benchmark instances, CapC, to trace the operator selection through timeline, where the progress of operation selection is plotted in Fig. 2. The plot demonstrates that *disABC* operates best over the first 200 iterations and then *ibinABC* takes over the best delivery. *binABC* doesn't perform well in comparison to other two as suggested in the plot.

The results by the proposed approach have been tabulated in Table 4 alongside of other adaptive operator selection methods explained above for comparative purposes. As seen, all adaptive methods embedded in binary ABC algorithm have assisted solve all UFLP benchmark instances with 100% success except CapC, where the *Gap* and *St. Dev* metrics are *0* and the *hit* measure is 30 out of 30 for all instances except CapC. It is paramount to define the *gap* as the average difference in between the optimum value and the fitness/cost value found,

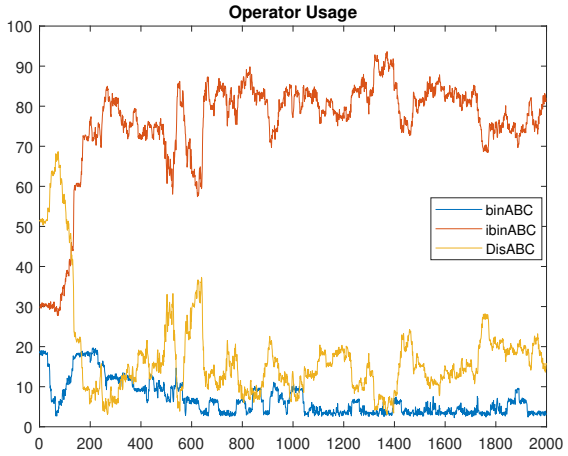


Fig. 2. Operator usage rates through search process

Table 3. Parametric fine-tuning results in *hit* metric for both *average* and *extreme* rewards

			Average Reward			Extreme Reward						Average Reward			Extreme Reward		
			γ			γ						γ			γ		
W	P_{min}	β	0.1	0.3	0.9	0.1	0.3	0.9	W	P_{min}	β	0.1	0.3	0.9	0.1	0.3	0.9
5	0.1	0.01	16	21	16	24	25	24	25	0.1	0.01	22	20	22	22	23	24
		0.05	23	19	17	23	24	23			0.05	25	17	19	27	21	23
		0.1	21	19	18	23	24	19			0.1	24	21	18	22	19	21
	0.2	0.01	24	21	23	19	19	23		0.2	0.01	23	21	21	17	25	23
		0.05	22	20	18	21	19	17			0.05	15	19	21	21	22	22
		0.1	20	21	19	19	18	19			0.1	22	20	21	17	25	23
	0.3	0.01	20	21	23	19	18	20		0.3	0.01	21	20	20	20	16	26
		0.05	20	23	20	21	20	19			0.05	20	18	18	22	23	17
		0.1	22	15	18	21	16	19			0.1	21	18	19	25	22	16
10	0.1	0.01	25	21	24	22	16	25	50	0.1	0.01	23	19	19	21	18	19
		0.05	21	23	20	20	18	22			0.05	21	22	19	27	21	19
		0.1	24	15	23	22	19	22			0.1	20	20	18	21	18	23
	0.2	0.01	25	19	20	21	18	13		0.2	0.01	18	20	25	21	22	19
		0.05	21	22	20	14	23	17			0.05	19	21	19	21	18	18
		0.1	24	21	20	15	21	21			0.1	23	14	22	22	17	19
	0.3	0.01	24	20	20	24	20	19		0.3	0.01	16	25	20	21	18	20
		0.05	21	14	16	20	24	19			0.05	22	17	21	22	16	20
		0.1	23	21	19	20	20	22			0.1	16	19	16	21	14	18

while *St. Dev.* is the standard deviation calculated over 30 repeated trails. CapC seems to be the hardest benchmark instance, which helps fine-tuning the hyper parameters and comparing the results produced by each rival approaches. The proposed method, labelled as "C-BABC" in the tables, produces the lowest *gap* and *st. dev* and the highest *hit* in comparisons to "PM-BABC", "AP-BABC" and "UCB-BABC", which are the binary ABC algorithms embedded with *PM*, *AP* and *UCB* as explained above.

Table 4. The comparative results obtained; the proposed operator selection scheme vs alternatives

Benchmarks	PM-ABC			AP-BABC			UCB-BABC			C-BABC		
	Gap	Std. Dev.	Hit	Gap	Std. Dev.	Hit	Gap	Std. Dev.	Hit	Gap	Std. Dev.	Hit
Cap71	0	0	30	0	0	30	0	0	30	0	0	30
Cap72	0	0	30	0	0	30	0	0	30	0	0	30
Cap73	0	0	30	0	0	30	0	0	30	0	0	30
Cap74	0	0	30	0	0	30	0	0	30	0	0	30
Cap101	0	0	30	0	0	30	0	0	30	0	0	30
Cap102	0	0	30	0	0	30	0	0	30	0	0.	30
Cap103	0	0	30	0	0	30	0	0	30	0	0	30
Cap104	0	0	30	0	0	30	0	0	30	0	0	30
Cap131	0	0	30	0	0	30	0	0	30	0	0	30
Cap132	0	0	30	0	0	30	0	0	30	0	0	30
Cap133	0	0	30	0	0	30	0	0	30	0	0	30
Cap134	0	0	30	0	0	30	0	0	30	0	0	30
CapA	0	0	30	0	0	30	0	0	30	0	0	30
CapB	0	0	30	0	0	30	0	0	30	0	0	30
CapC	0.0055	1428.003	25	0.0043	1302.539	26	0.0087	1694.457	22	0.0033	1149.5	27

The success of proposed method has been comparatively tested with a number of recently published studies, which can be considered as state-of-art works. The comparative results have been picked up form corresponding articles [1] and tabulated with the results produced by the proposed approach. As clearly seen on Table 5, the proposed method, C-BABC, outperforms all the algorithms known to be the state-of-the-art with a 100% success of solving all benchmark instances except CapC, which is solved with the highest score, while binAAA and JayaX solve all instances except CapB and CapC. Due to level of hardness in solving CapB and CapC approaches are tested with, so is the proposed approach in comparative way. The difference between the results by the proposed approach and other competitor algorithms have been tested statistically with Wilcoxon signed rank and the results are presented in Table 6, where C-BABC, the proposed method is significantly performed better.

Table 5. Comparative results; The proposed method (C-BABC) versus some state-of-art approaches

Benchmark	GA-SP			BPSO			binAAA			JayaX			C-BABC		
	Gap	Std. Dev.	Hit	Gap	Std. Dev.	Hit	Gap	Std. Dev.	Hit	Gap	Std. Dev.	Hit	Gap	Std. Dev.	Hit
Cap71	0	0	30	0	0	30	0	0	30	0	0	30	0	0	30
Cap72	0	0	30	0	0	30	0	0	30	0	0	30	0	0	30
Cap73	0.066	899.65	19	0.024	634.625	26	0	0	30	0	0	30	0	0	30
Cap74	0	0	30	0.0088	500.272	29	0	0	30	0	0	30	0	0	30
Cap101	0.068	421.655	11	0.0432	428.658	18	0	0	30	0	0	30	0	0	30
Cap102	0	0	30	0.00989	321.588	28	0	0	30	0	0	30	0	0	30
Cap103	0.063	505.036	6	0.04939	521.237	14	0	0	30	0	0	30	0	0	30
Cap104	0	0	30	0.040	1432.239	28	0	0	30	0	0	30	0	0	30
Cap131	0.068	720.877	16	0.171	1505.749	10	0	0	30	0	0	30	0	0	30
Cap132	0	0	30	0.058	1055.238	21	0	0	30	0	0	30	0	0	30
Cap133	0.091	685.076	10	0.082	690.192	10	0	0	30	0	0	30	0	0	30
Cap134	0	0	30	0.195	2594.211	18	0	0	30	0	0	30	0	0	30
CapA	0.046	22451.21	24	1.69	319855.4	8	0	0	30	0	0	30	0	0	30
CapB	0.58	66658.65	9	1.40	135326.7	5	0.24	39224.74	15	0.07	27033.02	26	0	0	30
CapC	0.70	51848.28	2	1.62	115156.4	1	0.29	29766.31	1	0.021	5455.94	17	0.0033	1149.5	27

Table 6. Statistical test results for state-of-art methods compared with proposed approach

	binAAA		JayaX		BPSO		GA-SP	
Benchmarks	p-value	H	p-value	H	p-value	H	p-value	H
Cap71	1	0	1	0	1	0	1	0
Cap72	1	0	1	0	1	0	1	0
Cap73	1	0	1	0	1.E-01	0	1.E-03	1
Cap74	1	0	1	0	3.E-06	1	4.E-08	1
Cap101	1	0	1	0	2.E-01	0	4.E-04	1
Cap102	1	0	1	0	5.E-01	0	1	0
Cap103	1	0	1	0	1.E-06	1	1.E-06	1
Cap104	1	0	1	0	5.E-01	0	1	0
Cap131	1	0	1	0	1.E-06	1	1.E-06	1
Cap132	1	0	1	0	1.E+00	0	4.E-08	1
Cap133	1	0	1	0	2.E-06	1	1.E-06	1
Cap134	1	0	1	0	5.E-04	1	1	0
CapA	1	0	1	0	5.E-05	1	1.E-01	0
CapB	6.E-05	1	2.E-07	1	2.E-06	1	2.E-06	1
CapC	4.E-06	1	1.E-04	1	3.E-06	1	4.E-06	1

5 Conclusion

This study has been done to investigate how machine learning can help adapt a dynamically updating scheme for operator selection within ABC algorithms as one of recently developed swarm intelligence approaches in solving binary problems. The research has been done embedding an online learning mechanism into binary ABC to learn which operator performs better in given circumstances. The main contribution of this research is that the adaptive operator selection has been achieved through reinforcement learning which is implemented with Hard-C-means clustering algorithm converted its unsupervised nature into reinforcement learning. Unlike the previously suggested adaptive selection schemes, this approach maps the binary input set into corresponding operators, hence, each time the hamming distance between both binary sets is used to make the selection, while the centres of the clusters are optimised/fine-tuned with estimated rewards per operator selection. The optimised cluster centres remain as the basis of operator selection. The proposed algorithm is tested with solving UFL problems, and statistically verified that the proposed approach significantly outperforms the state-of-art approaches in solving the same benchmark instances. It is also demonstrated that other existing adaptive approaches are also outperformed.

References

1. Aslan, M., Gunduz, M., Kiran, M.S.: Jayax: Jaya algorithm with xor operator for binary optimization. Applied Soft Computing **82**, 105576 (2019)
2. Aydin, M.E., Öztemel, E.: Dynamic job-shop scheduling using reinforcement learning agents. Robotics and Autonomous Systems **33**(2-3), 169–178 (2000)

3. Coronato, A., Naeem, M., De Pietro, G., Paragliola, G.: Reinforcement learning for intelligent healthcare applications: A survey. Artificial Intelligence in Medicine **109**, 101964 (2020)
4. DaCosta, L., Fialho, A., Schoenauer, M., Sebag, M.: Adaptive operator selection with dynamic multi-armed bandits. In: Proceedings of the 10th annual conference on Genetic and evolutionary computation. pp. 913–920 (2008)
5. Dokeroglu, T., Sevinc, E., Kucukyilmaz, T., Cosar, A.: A survey on new generation metaheuristic algorithms. Computers & Industrial Engineering **137**, 106040 (2019)
6. Durgut, R.: Improved binary artificial bee colony algorithm. Frontiers of Information Technology & Electronic Engineering **(in press)** (2020)
7. Durgut, R., Aydin, M.E.: Adaptive binary artificial bee colony algorithm. Applied Soft Computing **101**, 107054 (2021)
8. Durgut, R., Aydin, M.E.: Adaptive binary artificial bee colony algorithm. EasyChair Preprint no. 4687 (EasyChair, 2020)
9. Fialho, Á.: Adaptive operator selection for optimization. Ph.D. thesis, Université Paris Sud-Paris XI (2010)
10. Fialho, Á., Da Costa, L., Schoenauer, M., Sebag, M.: Extreme value based adaptive operator selection. In: International Conference on Parallel Problem Solving from Nature. pp. 175–184. Springer (2008)
11. Fialho, Á., Da Costa, L., Schoenauer, M., Sebag, M.: Analyzing bandit-based adaptive operator selection mechanisms. Annals of Mathematics and Artificial Intelligence **60**(1-2), 25–64 (2010)
12. Hussain, A., Muhammad, Y.S.: Trade-off between exploration and exploitation with genetic algorithm using a novel selection operator. Complex & Intelligent Systems pp. 1–14 (2019)
13. Hussain, K., Salleh, M.N.M., Cheng, S., Shi, Y.: On the exploration and exploitation in popular swarm-based metaheuristic algorithms. Neural Computing and Applications **31**(11), 7665–7683 (2019)
14. Karaboga, D., Basturk, B.: On the performance of artificial bee colony (abc) algorithm. Applied soft computing **8**(1), 687–697 (2008)
15. Kashan, M.H., Nahavandi, N., Kashan, A.H.: Disabc: A new artificial bee colony algorithm for binary optimization. Applied Soft Computing **12**(1), 342–352 (2012)
16. Kiran, M.S., Gündüz, M.: Xor-based artificial bee colony algorithm for binary optimization. Turkish Journal of Electrical Engineering & Computer Sciences **21**(Sup. 2), 2307–2328 (2013)
17. Moerland, T.M., Broekens, J., Jonker, C.M.: Model-based reinforcement learning: A survey. arXiv preprint arXiv:2006.16712 (2020)
18. Ozturk, C., Hancer, E., Karaboga, D.: Dynamic clustering with improved binary artificial bee colony algorithm. Applied Soft Computing **28**, 69–80 (2015)
19. Xue, Y., Xue, B., Zhang, M.: Self-adaptive particle swarm optimization for large-scale feature selection in classification. ACM Transactions on Knowledge Discovery from Data (TKDD) **13**(5), 1–27 (2019)
20. Yang, T., Zhao, L., Li, W., Zomaya, A.Y.: Reinforcement learning in sustainable energy and electric systems: A survey. Annual Reviews in Control (2020)

Time-indexed formulations for the Order Acceptance Scheduling problem considering energy aspects with sequence-dependent setup-times

M. Bouzid, O. Masmoudi and A. Yalaoui

ICD, LOSI, Université de Technologie de Troyes,
12 rue Marie Curie CS 42060, 10004 Troyes, France

mariam.bouzid@utt.fr, oussama.masmoudi@utt.fr, alice.yalaoui@utt.fr

Keywords : Order Acceptance Scheduling, Time-indexed formulation, Energy.

1 Introduction

In this period of economic recession [1] coupled with growing environmental concerns, the development of effective policies and tools remains crucial to tackle current and future challenges. In response, governments has been implementing energy taxes and/or subsidies to reduce energy consumption and greenhouse gases (GHG) emission. Consequently, the industrial sector has been adapting to the newest regulations and energy prices increase while maintaining competitiveness. This past decade, the impact of energy awareness in the supply chain have been widely studied [2].

Energy sobriety is beneficial for both economic and environmental reasons. First, energy cost is a shortfall for heavy energy-using industry as the energy supplies are getting expensive. For that purpose, energy providers have designed preferential tariffs rate such as TOU, real-time or critical peak pricing. TOU rate incite manufacturers to shift their production to cheaper off-peak hours. Second, depending on the energy mix used (e.g. coal or gas based), reducing energy consumption or costs is a direct way to reduce GHG emission [3].

Early industry focused on mass production with high volumes of few products. Yet, these past decades, major changes in industry have been occurring [4]. Product variety and demand for tailor-made products force manufacturers to make a compromise between their available production capacity and their generated profit. This global tendency appears in a plethora of manufacturing sectors and Order Acceptance Scheduling (OAS) is an abstraction to model this particular trend. OAS is a particular scheduling problem where the decision covers the selection of a subset of n orders and their sequencing in a capacity-constrained production system with the objective of maximizing total profit. Literature on OAS considering energy aspects is very sparse. In fact, only three papers have been reported [5]–[7].

In this vein, this paper investigates a single machine OAS problem with release date and sequence dependent setup-times under TOU tariffs and taxed carbon emission with the objective of maximizing total profit. Chen et al. are the first to introduce [5] this problem while proposing a disjunctive Mixed Integer Linear Program (MILP). Bouzid et al. consider [6] an arc-time-indexed (ATI) MILP to cope with the high complexity of this NP-hard problem and successfully solve some large instances. Without sequence-dependent setup-times, time-indexed formulations described in [8] are shown to be outperforming a classical MILP. Based on this, an adaptation of these formulations is undertaken in order to solve the considered OAS problem. Performances of the formulations against existing models in the literature are shown and discussed in terms of solving time, average gap and optimal solutions found. Spatial complexity of each model is also provided in order to grasp the behavior of the proposed models.

2 Problem statement

The OAS with sequence-dependent setup-times, release date under TOU costs and taxed carbon periods is investigated. In this problem, the objective is to maximize the revenues minus tardiness penalties and energy costs. Each order $j = 1, \dots, n$ is completely defined by its processing time p_j, release date r_j, due date d_j,

deadline $\bar{d}_j$, revenue e_j, power consumption Ω_j and tardiness penalties w_j. In addition, a sequence dependent setup-time s_{ij} is defined between any pair of orders i and j. A dummy order 0 is introduced in order to start the sequence, each of its properties are set to zero except its setup-time s_{0j} between any order j. An order j is accepted when it is sequenced in the span ranging from its release date r_j to its deadline $\bar{d}_j$ and rejected otherwise. A proportional tardiness penalty w_j is subtracted to an order revenue e_j when the planning exceeds its due date d_j (**Figure 1**).

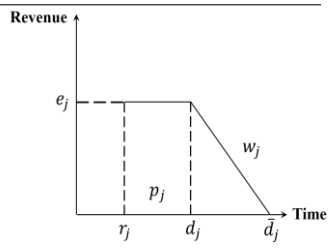


Figure 1 Revenue minus tardiness calculation of an order j adapted from [13]

Moreover, the planning horizon is divided into periods of fluctuating TOU tariffs and CO_2 emission, characterized by an electricity cost, an amount of CO_2 per kg and a tax per emitted kg of carbon.

3 Solution approach

The mathematical formulations provided in this work are time-indexed, *i.e.* they both rely on the discretization of the time horizon into unitary slots $t = 0, \dots, T$. Since the setup-times are sequence dependent, the binary decision variables u_{ij} are employed in each model for any pair of order i and j to determine the sequencing. The binary variable u_{ij} takes value 1 if and only if order i precedes directly order j in the sequence, 0 otherwise.

An example is given in order to better understand the proposed formulations. **Table 1** and **Table 2** presents the optimal solution of an example with $n = 3$ orders with their processing times $p = [5,3,2]$, release dates $r = [1,2,1]$, due dates $d = [6,5,12]$, deadlines $\bar{d} = [9,10,14]$, revenues $e = [10,10,6]$, power consumption $\Omega = [1,2,1]$ and weight penalties $w = [2,1,3]$. The horizon is $T = 14$. The setup-times between orders are defined by $s = [[0,1,2,3],[0,0,3,2],[0,1,0,3],[0,1,1,0]]$. Finally, the starting times of TOU and carbon emission intervals $b = g = [0,5,8]$, the electricity price $EC = [2,10,2]$ and the amount of CO_2 emitted $q = [4,1,4]$ are defined.

3.1. On/off formulation

Each binary decision variable $x_{jt} = 1$ indicates whether the order j is processed at time $t = r_j, \dots, \bar{d}_j$, or not $x_{jt} = 0$. In the same way, the binary decision variables $y_{jt} = 1$ corresponds to a unit of processed setup of an order $j = 1, \dots, n$ at time $t = r_j, \dots, \bar{d}_j$. Finally, for any order $j = 0, \dots, n$, the binary decision variable a_j takes value 1 if order j is accepted, 0 otherwise.

Table 1 Example of an integral solution represented by the values of the decision variables x_{jt}, sequence is $0 - 3 - 2$ and order 1 is rejected.

j/t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
3	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

An accepted order j is produced during the totality of its processing time, counting the number of x_{jt} variables in the range of its release date r_j to its deadline $\bar{d}_j$. Non-preemption is guaranteed by forcing the contiguity of the x_{jt} variables. Units of setup operations are denoted by the y_{jt} variables and in the same manner as the x_{jt} variables, they must be adjacent. Finally, the y_{jt} variables shall appear before the processing of an order j, observing the sequencing determined by the u_{ij} variables and the right amount of setup-times.

3.2. Pulse formulation

The decision variables z_{jt} refer to the possible instants $t = r_j, \dots, \bar{d}_j - p_j + 1$ when the order $j = 1, \dots, n$ starts. Meaning that $z_{jt} = 1$ if and only if order j begins its production at time period t, and 0 otherwise.

Table 2 Example of an optimal solution represented by the values of the decision variables z_{jt}

j/t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
3	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

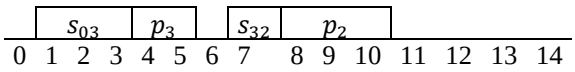


Figure 2 Corresponding Gantt diagram relative to the example.

In the pulse model, an order j is accepted if it starts between its release date and its deadline, meaning that a single z_{jt}variable is expected between r_j and $\bar{d}_j - p_j + 1$ if the order is processed. A precedence constraint guarantees that if an order j is processed after an order i, the order j shall starts at least after its release date, its setup operation and the production of order i.

4. Conclusion and perspectives

In this paper the OAS problem under energy aspects is studied. Two distinct time-indexed formulations are used to solve this NP-hard problem. Since the objective is the maximization of total profit with energy costs and tardiness penalties, these formulations exploit to their benefits the strong time-dependency of the investigated problem. The under-going work is dedicated to the development of fix-and-relax heuristics on these models with various approximation strategies including setup-times and variable relaxation. This solution approach and its preliminaries results will be presented on this occasion.

References

[1] International Monetary Fund, *World Economic Outlook, April 2020*. S.l.: Intl Monetary Fund, 2020.

[2] K. Gao, Y. Huang, A. Sadollah, et L. Wang, « A review of energy-efficient scheduling in intelligent production systems », *Complex Intell. Syst.*, vol. 6, nº 2, p. 237-249, juill. 2020, doi: 10.1007/s40747-019-00122-6.

[3] H. Zhang, F. Zhao, K. Fang, et J. W. Sutherland, « Energy-conscious flow shop scheduling under time-of-use electricity tariffs », *CIRP Annals*, vol. 63, nº 1, p. 37-40, 2014, doi: 10.1016/j.cirp.2014.03.011.

[4] E. Asadollahi-Yazdi, P. Couzon, N. Q. Nguyen, Y. Ouazene, et F. Yalaoui, « Industry 4.0: Revolution or Evolution? », *American Journal of Operations Research*, p. 28.

[5] S.-H. Chen, Y.-C. Liou, Y.-H. Chen, et K.-C. Wang, « Order Acceptance and Scheduling Problem with Carbon Emission Reduction and Electricity Tariffs on a Single Machine », *Sustainability*, vol. 11, nº 19, p. 5432, sept. 2019, doi: 10.3390/su11195432.

[6] M. Bouzid, O. Masmoudi, et A. Yalaoui, « Order Acceptance Scheduling on a Single Machine with Energy Aspects », in *Intelligent and Fuzzy Techniques: Smart and Innovative Solutions*, vol. 1197, C. Kahraman, S. Cevik Onar, B. Oztaysi, I. U. Sari, S. Cebi, et A. C. Tolga, Éd. Cham: Springer International Publishing, 2021, p. 1573-1580.

[7] M. Kong, J. Pei, X. Liu, P.-C. Lai, et P. M. Pardalos, « Green manufacturing: Order acceptance and scheduling subject to the budgets of energy consumption and machine launch », *Journal of Cleaner Production*, vol. 248, p. 119300, mars 2020, doi: 10.1016/j.jclepro.2019.119300.

[8] M. Bouzid, O. Masmoudi, et A. Yalaoui, « Time-indexed formulations for the order acceptance scheduling problem under energy aspects », 2020.

Mathematical model and metaheuristics for scheduling production processes in hospital catering

F. Abderrabi[1,2], M. Godichaud[1], A. Yalaoui[1], F. Yalaoui[1], L. Amodeo[1] and A. Qerimi[2]

[1] University of Technology of Troyes, Logistics Industrial Systems Optimization, Troyes, France
fatima.abderrabi, matthieu.godichaud, alice.yalaoui, farouk.yalaoui, lionel.amodeo@utt.fr
[2] Hospital Center of Troyes, Troyes, France
fatima.abderrabi, ardian.qerimi@hcs-sante.fr

Abstract. The problem addressed in this paper was motivated by a real case optimization problem of the supply chain of the hospital center of Troyes (HCT). The HCT is currently involved in the review and improvement of its logistics processes, and in the implementation of operational research techniques in order to provide effective solutions to better optimize the activities of its logistics chain. In this work, the considered problem focuses on the catering component of hospital logistics. A new mathematical model and different metaheuristics for the production scheduling of multi-products and multi-stages food processes are developed. These resolution methods have been implemented by using the solver Cplex and the Java programming language. The computation results of the developed methods have proven their effectiveness for the scheduling of production processes and allowed significant improvements in the current organization and in the performance of the studied production system.

Keywords : hospital catering, production scheduling, flexible job shop problem, sequence-dependent setup time, job-splitting, mathematical model, genetic algorithms, iterated local search algorithms.

1 Introduction

Nowadays, hospitals are facing the challenges regarding quality of care and performance. In the management and organization of hospitals, there is still much progress to be made to improve the quality of care while reducing costs. To respond effectively to the patients needs and to improve the working conditions and well-being of their employees, hospitals are looking for tools and new ways of organization and management. In the present work, a mathematical model and metaheuristics for the scheduling production processes in hospital catering are developed. This work is part of an industrial thesis in partnership with HCT and LOSI. The logistics platform of the HCT is composed of the central food production unit, a laundry, and a store that can satisfy, in addition to its needs, requests of meals, washings and pharmaceutical products of its partners. The objective of the addressed problem is to find solutions for scheduling production processes to satisfy the demands of meals of the customers while optimizing certain criteria and taking into account several specific industrial constraints.

2 Problem description

The problem of food production process scheduling considered in this study aims to schedule the operations from the pre-treatment of raw materials to the stock of finished products of a meal manufacturing process in hospital catering or more generally in collective catering. This problem is considered as a flexible job shop scheduling with sequence-dependent setup time, since each job has its own order of operations and each operation has to be affected to one among a set of alternative machines. The problem of food production process scheduling can be described by a set of jobs, where each job corresponds to the preparation of a dish characterized by a number of portions (quantity), and a set of operations for the preparation of the dish (from raw material to finished product). For each job, there is a due date to respect. It is worth to highlight that the dishes to be prepared do not have the same operating ranges (set of operations necessary for the preparation of dish). In this study, we identified ten possible operating ranges for all the dishes to be prepared and it is possible that several dishes may have the same operating range. For each operation of an operating range there is a set of material resources able to realize it, such as : ovens, packaging machines, cooling cells, etc. These material resources can be classified into three categories : material resources with a capacity of one portion and that can not process several jobs at the same time (material resources that can perform preprocessing and cold production operations), material resources with a capacity greater than one portion and which can not process several jobs

at the same time (ovens,...) and material resources with a capacity greater than one portion and that can process several jobs at the same time (cooling cells). For each material resource, there is a setup time to take into account which corresponds to the preparation time of the machine before carrying out an operation and the cleaning time of the machine between two consecutive operations. A time window of availability is known for each material resource. Note that the corresponding machines may not be identical, involving different processing times according to the chosen machine. The setup times of machines are sequence dependent because it depends on the preceding operation on the same machine. The food production process scheduling involves two steps : (i) assignment of operations to machines, (ii) sequencing of operations on machines. In order to respect the production capacity of material resources, a job can be splitted into smaller sub-lots, in such a way that the operations of sub-lots of a job can be performed simultaneously on different machines. This strategy, which is useful when machine capacity does not allow the treatment of the whole job, also enables a more efficient processing scheme. The criterion to minimize in the present study is the total of flow time of jobs in the production system. The choice of this criterion is based on the fact that the respect of the cold chain at each stage of the product life cycle must be ensure. It aims to constantly maintain a low temperature (positive or negative depending on the product) to ensure the maintenance of all the food qualities (hygienic, nutritional and gustatory).

3 Resolution methods

To respond effectively to this new industrial problem, we first developed mathematical models for small instances of the problem. The results of these works are the subject of scientific publications ([1], [2]). The developed mathematical models have been implemented by using the solver Cplex and they have been tested on 150 instances of different types : real instances of HCT, randomly generated instances of HCT type, randomly generated instances and adapted instances of literature ([6], [7], etc). The implementation results (Table 1) of the mathematical models show that these latter are able to find solutions in less than three hours of execution for the small instances. The execution times of the mathematical models for these instances vary according to the number of jobs, sub-lots, and operations. The computational results of the proposed mathematical models on different types of instances show the limits of an exact resolution for the studied problem. To solve the large instances in reasonable resolution times, two genetic algorithms (GAs) and two iterated local search algorithms (ILSs) are developed. The results of these works are the subject of a scientific publication ([5]). The developed metaheuristics have been implemented in Java programming language and have been tested on the same instances (instances presented below) as the mathematical models. From the computation results presented in Table 1, we remark that the developed metaheuristics are efficient in terms of quality and rapidity. The performance depend on the type of instances and their sizes, and it depends also on the choice of heuristics for the generation of initial solutions. By comparing the different resolution methods on all the tested instances, we found that for some instances, the two genetic algorithms succeeded in finding the optimal solutions in very short computation times compared to the mathematical models. For the instances for which the optimality has not been reached, the gaps between the solutions obtained with the algorithms and the optimal solutions are very small. For the large instances for which the mathematical models have failed to find solutions after more than three hours of execution, the developed metaheuristics have found feasible solutions within reasonable computation times. By comparing the results obtained with the genetic and iterated local search algorithms on all the tested instances, we find that the ILSs are less good than the GAs in terms of quality of solutions obtained. Whereas, in terms of rapidity, the ILSs methods are faster compared to the GAs. Table 2 represents the results of the genetic algorithm on some examples of real production days with comparison between the real solutions as these production days were organized and the solutions proposed by the genetic algorithm. The performance indicators between solutions are based on the total flow time and the gaps between the solutions. From these results, we remark that the gaps between the real solutions and those of the genetic algorithm are very importants and significants. For example, for the instance having : 62 dishes, 68 sub-lots, 218 operations, 29 machines, we have brought about a considerable improvement of $18,72$ % on the real solution. The proposed metaheuristics make it possible also to improve the solutions of the adapted instances of the literature. For example, for the instance of Lee et al. [7], the genetic algorithms have brought about an improvement of 3.92 % on the solutions obtained by the methods proposed in [7], which shows the quality of the developed metaheuristics.

J	SL	O	M	F0 (h)	T0 (s)	F1 (h)	T1 (s)	F2 (h)	T2 (s)	F3 (h)	T3 (s)	F4 (h)	T4 (s)
5	5	24	29	39.9	240	39.9	0.4	39.9	1	39.9	0.1	39.9	0.28
6	6	29	29	49.2	1200	49.2	0.6	49.2	1.4	49.2	0.16	49.2	0.40
8	10	39	29	69.4	9000	69.4	1	69.4	2	69.4	0.46	69.4	0.3
9	11	44	29	-	> 10800	77.7	1.5	77.7	3	77.7	0.71	77.7	1.2
10	12	48	29	-	> 10800	85.0	2	85.0	4	86.4	0.96	85.9	1.4
20	22	93	29	-	> 10800	162.4	30	162.4	60	166.5	5	165.8	11
40	42	179	29	-	> 10800	339.7	90	339.7	180	354.9	23	352.3	49
50	58	227	29	-	> 10800	471.6	138	478.6	276	490.6	30	485.6	60
60	68	271	29	-	> 10800	590.4	180	593.9	360	624.4	43	616.7	87
82	92	370	29	-	> 10800	798.3	300	788.6	600	846.5	70	840.8	141

J: nomber of jobs, SL: number of sub-lots, O: number of operations, M: number of machines, F0: total flow time of the mathematical model, T0: computational time of the mathematical model, F1: total flow time of GA1, T1: computational time of GA1, F2: total flow time of GA2, T2: computational time of GA2, F3: total flow time of ILS1, T3: computational time of ILS1, F4: total flow time of ILS2, T4 : computational time of ILS2

Table 1: Computational results of the developed resolution methods on real instances of HCT.

	Instance 1	Instance 2	Instance 3
- Number of dishes	82	110	62
- Number of sub-lots of dishes	92	115	68
- Number of operations	370	392	218
- Number of material resources	29	29	29
- Average number of meals produced	4 800	4 800	4 800
- Real solutions	901,97 h	1062,66 h	278,23 h
- Genetic algorithm solutions	788,64 h	952,48 h	26,12 h
- Gaps between real and genetic algorithm solutions	-12,56 %	-10,36 %	-18,72 %

Table 2: Comparison between real and genetic algorithm solutions on some examples of production days.

4 Conclusion

This article presents the results of a study on a new industrial problem. Different resolution methods for scheduling production processes in hospital catering were developed. A mathematical model integrating all the constraints of the studied problem was developed. The computational results of the mathematical model on different types of instances show the limits of an exact resolution for the problem of scheduling production processes. To solve the large instances of this problem, different metaheuristics have been developed and tested on several types of instances. The computational results of these metaheuristics have proven their effectiveness for scheduling operations of the food production processes and allowed significant improvements in current real solutions and system performance. The present work opens the way to different perspectives such as the study of the production planning problem over several days and our future works will focus on the development of resolution methods for this problem.

References

1. F. Abderrabi, M. Godichaud, A. Yalaoui, F. Yalaoui, L. Amodeo, A. Qerimi, E. Thivet. Mathematical model for food production scheduling in hospital catering, OLA 2020.
2. F. Abderrabi, M. Godichaud, A. Yalaoui, F. Yalaoui, L. Amodeo, A. Qerimi, E. Thivet. Mathematical model for scheduling food production in hospital catering, IFAC World Congress 2020.
3. F. Abderrabi, M. Godichaud, A. Yalaoui, F. Yalaoui, L. Amodeo, A. Qerimi, E. Thivet. Optimization in hospital logistics : food production scheduling in hospital catering, SHEIC 2020.
4. F. Abderrabi, M. Godichaud, A. Yalaoui, F. Yalaoui, L. Amodeo, A. Qerimi, E. Thivet. Optimization in hospital logistics : improving working conditions and employees well-being, SHEIC 2020.
5. F. Abderrabi, M. Godichaud, A. Yalaoui, F. Yalaoui, L. Amodeo, A. Qerimi, E. Thivet. Flexible job shop scheduling problem with sequence dependent setup time and job splitting : hospital catering case study, Journal of Applied Sciences 2021.
6. Sriboonchandr, Prasert; Kriengkorakot, Nuchsara; Kriengkorakot, Preecha. Improved Differential Evolution Algorithm for Flexible Job Shop Scheduling Problems, Math. and Comp. Applications 2019.
7. S. Lee, I. Moon, H. Bae, J. Kim. Flexible job-shop scheduling problems with 'AND'/'OR' precedence constraints, International Journal of Production Research 2012.

Theoretical analysis of a dynamic pricing problem with linear and isoelastic demand functions

M. Terzi[1,2] Y.Ouazene[1,2] A.Yalaoui[1,2] and F. Yalaoui[1,2]

[1] Chaire Connected Innovation, Université de Technologie de Troyes, 12 rue Marie Curie, CS 42060 10004 Troyes CEDEX, France
[2] Université de Technologie de Troyes, CNRS, 12 rue Marie Curie, CS 42060 10004 Troyes CEDEX, France
mourad.terzi@utt.fr
yassine.ouazene@utt.fr
alice.yalaoui@utt.fr
farouk.yalaoui@utt.fr

Abstract. Dynamic pricing strategies are usually adopted to dynamically adjust the products' prices taking into account demand function characteristics to maximize the revenue. This paper addresses the problem in which a firm has to make decisions about its selling prices in each period to maximize the total profit over the whole horizon. We propose a theoretical analysis of this problem from which we show that: first, when the demand function is linear, the problem can be formulated as a quadratic programming problem. We also present the Karush-Kuhn-Tucker system, which can be used to find the optimal pricing policy when the objective function is concave. Then, when the demand is isoelastic, we also show that the problem can be reduced to the maximization of N independent functions in bounded intervals. Some numerical examples are provided to illustrate the results obtained for both the linear and isoelastic cases.

Keywords : Revenue maximization, Dynamic pricing, Linear and isoelastic demand, Quadratic programming, KKT conditions

1 Introduction

Dynamic pricing is a pricing strategy where the firms adjust dynamically the prices of the products and services according to the perceived demand at different times (Narahari et al. (2005)). One of the key elements when dealing with a dynamic pricing problem is the demand function which characterizes the relation between different factors like (selling price, advertising, seasonality,...) and the demand. In the paper of. Huang et al. (2013), a survey on the demand functions was presented. The factors considered are price, rebate, lead time, space, quality, and advertising. The authors observed that: 1) the linear and isoelastic demand functions are the two widely used in the literature, and 2) the majority of publications consider the price and quality factors.

Initially, the dynamic pricing has been applied to the service industries such as airline (Smith et al. (1992)) and hotels (Bitran, Mondschein (1995)). According to Elmaghraby, Keskinocak (2003), factors like 1) the availability of demand data and decision-support system to track the changes in prices and, 2) the simplicity of prices adjusting due to the recent developments in technologies lead to several works on dynamic pricing on a wide range of industries like retails (Chen et al. (2016)).

Several studies dealing with the coordination of dynamic pricing and production decisions with the discrete-time horizon and multiple products are conducted. The work of Bajwa, Sox (2015) presented a joint pricing, production, and advertising decisions model for a firm that produces and sells multiple products as different brands. The authors assumed that the demand is a function of the price and advertising money and demonstrated that coordinating the marketing and operational decisions leads the firm to increase its profitability. The paper of Bajwa et al. (2016) considered a manufacturer with a limited production capacity. They proposed a model that allows lost sales under a price-dependent demand function. Ouazene et al. (2017) studied the problem in

which the products can be sold through multiple channels and the demand is a price-dependent function. The authors compared the dynamic and constant pricing strategies. The paper of Couzon et al. (2019) presented an extension of the classical capacitated lot-sizing problem by considering a production system with variable capacity under a price-dependent demand function. The work of Couzon et al. (2020) improved the model studied in Bajwa et al. (2016) by introducing new lower and upper bounds that reduced the search space. They also proposed new constructive efficient heuristics to solve the model. All the papers cited above assumed that the demand in a given period is a function of the price of the product in the same period and can take the linear or isoelastic form.

Several surveys on dynamic pricing have been published, Bitran, Caldentey (2003), Elmaghraby, Keskinocak (2003), and Chen, Simchi-Levi (2012) reviewed the literature on dynamic pricing with the presence of inventory considerations. A survey on dynamic pricing and learning was conducted in Boer den (2015). The authors reviewed the literature on dynamic pricing with demand uncertainty.

The presented work investigates the problem in which a firm has to make decisions about its selling prices in each period to maximize the total profit over the whole horizon. This problem has been initially tackled by Shakya et al. (2012) and solved by combining neural networks and evolutionary algorithms. Their study is based on linear, exponential, and multinomial logit demand functions. In the presented paper, a theoretical analysis in which we consider the linear and isoelastic demand functions will be conducted. The mathematical properties of the problem will be studied and some theoretical results that lead to finding the optimum pricing policy will be provided.

The remainder of this paper is organized as follows. Section 2 presents the dynamic pricing problem assumptions and mathematical formulation. Section 3 describes the resolution approach under the linear and isoelastic demand. A numerical experiments are presented in section 4. A conclusion is to be found in section 5.

2 Problem description

The dynamic pricing problem addressed in this study is the same as the model presented in Shakya et al. (2012). The model is denoted as (P_0). The problem considers a firm that produces and sells its product. The goal is to find the product's price in each period to maximize the firm's total profit over a given horizon. Following the notations used to describe the model:

N	Number of periods in the horizon
t	Time index, $t = 1, ..., N$
Q_t	Number of production (sales) at period t
P_t	Price of a product at period t (decision variable)
C_t	Cost of one unit production in period t
Π	Total profit during the entire planning horizon
$\overline{P_t}$	Upper bound for the price at period t
$\underline{P_t}$	Lower bound for price at period t
$\overline{K_t}$	Upper bound for the capacity at period t
M_t	Lower bound for the capacity at period t

The initial mathematical model P_0 is detailed below:

$$\max \Pi = \sum_{t=1}^{N} (P_t Q_t - C_t Q_t) \tag{1}$$

$$st: \quad M_t \leq Q_t \leq K_t, \qquad t = 1, 2, ..., N \tag{2}$$

$$\underline{P_t} \leq P_t \leq \overline{P_t}, \qquad t = 1, 2, ..., N \tag{3}$$

$$P_t > 0, \qquad t = 1, 2, ..., N \tag{4}$$

The objective function represents the total profit over all the horizon to maximize. P_tQ_t is the total revenues in period t and C_tQ_t is the cost per production in period t. Constraints (2) consider the production capacity, the objective is to regulate the use of the available capacity in each period (machines, labor, etc ...) by considering production values that are at least equal to the minimum available capacity and don't exceed the maximum production capacity. Constraints (3) bound the selling price of each period by $\underline{P_t}$ and $\overline{P_t}$ to avoid a lower profit value, and a lower demand. Finally, constraints (4) are the non-negativity constraints. Not that the decisions variables P_t $(t = 1, ..., N)$ are a strict positive real numbers.

3 Resolution approach

In the presented work, the linear and isoelastic demand functions are considered. Both are price-dependent demand. The linear demand is adopted from Shakya et al. (2012) and the isoelastic function is the same as the demand studied in Couzon et al. (2020). The following notations will be considered in the presented work. Some new notations will be introduced throughout the analytical study.

$P^T = (P_1, P_2, ..., P_N)$	The pricing policy's vector
M	The total constraints' number

3.1 Case with linear demand function

Following the same assumption as in Shakya et al. (2012), the demand in period t (equation 5) is linear and depends on the price of the product in the same period and on the price of the product in other periods. $a_t(> 0)$ is defined as the intercept parameter, it represents the number of customers willing to buy the product at period t. $b_{t't}$ are the slope parameters. They represent the impact of price in period t' on the demand in period t. b_{tt} is generally assumed to be negative because when the product's price in a period t increases, the corresponding demand in the same period decreases.

$$Q_t = \psi(P_1, P_2, ..., P_N) = a_t + \sum_{t'=1}^{N} b_{t't} P_{t'} \tag{5}$$

Replacing Q_t by its value from (5), P_0 can be written as:

$$\max_{P_1, P_2, .., P_N} \Pi = \sum_{t=1}^{N} \left(a_t + \sum_{t'=1}^{N} b_{t't} P_{t'} \right) (P_t - C_t) \tag{6}$$

$$st: \qquad M_t \leq a_t + \sum_{t'=1}^{N} b_{t't} P_{t'} \leq K_t, \qquad t = 1, 2, ..., N \tag{7}$$

$$\underline{P_t} \leq P_t \leq \overline{P_t}, \qquad t = 1, 2, ..., N \tag{8}$$

$$P_t > 0, \qquad t = 1, 2, ..., N \tag{9}$$

Proposition 1. : *The total profit function Π is quadratic and its expression is given in equation (10). W is a $(N \times N)$ symmetric matrix and V^T is a $(1 \times N)$ vector of real numbers. D is a real constant number and it is independent from the selling price vector.*

$$\Pi = \frac{1}{2} P^T W P + V^T P + D \tag{10}$$

Proof of proposition 1: Π can be rewritten as follows:

$$\Pi = \sum_{t=1}^{N} a_t P_t - \sum_{t=1}^{N} a_t C_t + \sum_{t=1}^{N} P_t \sum_{t'=1}^{N} b_{t't} P_{t'} - \sum_{t=1}^{N} C_t \sum_{t'=1}^{N} b_{t't} P_{t'}$$

$$\Pi = S_1 + S_2 + S_3$$

$$With \quad S_1 = -\sum_{t=1}^{N} a_t C_t = D$$

$$S_2 = \sum_{t=1}^{N} a_t P_t - \sum_{t=1}^{N} C_t \sum_{t'=1}^{N} b_{t't} P_{t'}$$

$$S_2 = (a_1 P_1 - C_1(b_{11}P_1 + b_{21}P_2 + ... + b_{N1}P_N)) + (a_2 P_2 - C_2(b_{12}P_1 + b_{22}P_2 + ... + b_{N2}P_N)) + \ldots +$$
$$(a_N P_N - C_N(b_{1N}P_1 + b_{2N}P_2 + ... + b_{NN}P_N))$$

$$S_2 = P_1(a_1 - C_1 b_{11} - C_2 b_{12} - \ldots - C_N b_{1N}) + P_2(a_2 - C_1 b_{21} - C_2 b_{22} - \ldots - C_N b_{2N}) + \ldots +$$
$$P_N(a_N - C_1 b_{N1} - C_2 b_{N2} - \ldots - C_N b_{NN})$$

$$S_2 = P_1(a_1 - \sum_{t=1}^{N} C_t b_{1t}) + P_2(a_2 - \sum_{t=1}^{N} C_t b_{2t}) + \ldots + P_N(a_N - \sum_{t=1}^{N} C_t b_{Nt})$$

$$S_2 = V^T P$$

$With$

$$V_{1,N}^T = \left(a_1 - \textstyle\sum_{t=1}^{N} b_{1t}C_t, \;\; a_2 - \sum_{t=1}^{N} b_{2t}C_t, \;\; \ldots, \;\; a_N - \sum_{t=1}^{N} b_{Nt}C_t\right)$$

and

$$P_{N,1} = \begin{pmatrix} P_1 \\ P_2 \\ \vdots \\ P_N \end{pmatrix}$$

$$S_3 = \sum_{t=1}^{N} P_t \sum_{t'=1}^{N} b_{t't} P_{t'} = P_1(b_{11}P_1 + b_{21}P_2 + \ldots + b_{N1}P_N) + P_2(b_{12}P_1 + b_{22}P_2 + \ldots + b_{N2}P_N) + \ldots +$$
$$P_N(b_{1N}P_1 + b_{2N}P_2 + \ldots + b_{NN}P_N)$$

Let consider :

$$S_4 = \frac{1}{2} P^T W P$$

with P *is the same vector as defined for* S_2 *and*

$$W_{N,N} = \begin{pmatrix} 2b_{11}, & b_{12}+b_{21}, & b_{13}+b_{31}, & \ldots, b_{1N}+b_{N1} \\ b_{12}+b_{21}, & 2b_{22}, & b_{23}+b_{32}, & \ldots, b_{2N}+b_{N2} \\ \vdots & \vdots & \vdots & \vdots \quad \vdots \\ b_{1N}+b_{N1}, & b_{2N}+b_{N2}, & b_{3N}+b_{N3}, \ldots & , 2b_{NN} \end{pmatrix}$$

$$S_4 = \frac{1}{2}(P_1, P_2, .., P_N) \begin{pmatrix} 2b_{11}, & b_{12}+b_{21}, & b_{13}+b_{31}, & \ldots, b_{1N}+b_{N1} \\ b_{12}+b_{21}, & 2b_{22}, & b_{23}+b_{32}, & \ldots, b_{2N}+b_{N2} \\ \vdots & \vdots & \vdots & \vdots \quad \vdots \\ b_{1N}+b_{N1}, & b_{2N}+b_{N2}, & b_{3N}+b_{N3}, \ldots & , 2b_{NN} \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ \vdots \\ P_N \end{pmatrix}$$

$$S_4 = \frac{1}{2}\Big(2b_{11}P_1 + (b_{12}+b_{21})P_2 + ... + (b_{1N}+b_{N1})P_N), (b_{12}+b_{21})P_1 + 2b_{22}P_2 + ... + (b_{2N}+b_{N2})P_N,$$

$$\ldots, (b_{1N}+b_{N1})P_1 + (b_{2N}+b_{N2})P_2 + ... + 2b_{NN}P_N)\Big) \begin{pmatrix} P_1 \\ P_2 \\ \vdots \\ P_N \end{pmatrix}$$

$$S_4 = \frac{1}{2}\Big(P_1\Big(2b_{11}P_1 + (b_{12}+b_{21})P_2 + ... + (b_{1N}+b_{N1})P_N\Big) + P_2\Big((b_{12}+b_{21})P_1 + 2b_{22}P_2 + ... + (b_{2N}+b_{N2})P_N\Big)$$

$$+ \ldots + P_N\Big((b_{1N}+b_{N1})P_1 + (b_{2N}+b_{N2})P_2 + ... + 2b_{NN}P_N\Big)\Big)$$

$$S_4 = \frac{1}{2}\Big(P_1\Big(2b_{11}P_1 + 2b_{21}P_2 + \ldots + 2b_{N1}P_N\Big) + P_2\Big(2b_{12}P_1 + 2b_{22}P_2 + \ldots + 2b_{N2}P_N\Big) + \ldots +$$

$$P_N\Big(2b_{1N}P_1 + 2b_{2N}P_2 + \ldots + 2b_{NN}P_N\Big)\Big)$$

$$S_4 = \Big(P_1\Big(b_{11}P_1 + b_{21}P_2 + \ldots + b_{N1}P_N\Big) + P_2\Big(b_{12}P_1 + b_{22}P_2 + \ldots + b_{N2}P_N\Big) + \ldots +$$

$$P_N\Big(b_{1N}P_1 + b_{2N}P_2 + \ldots + b_{NN}P_N\Big)\Big)$$

$S_4 = S_3$

Then, the following relation is tune :

$$\Pi = \frac{1}{2}P^T W P + V^T P + D$$

Considering the constraints of P_0, they can be rewritten as:

$$\sum_{t'=1}^{N} b_{t't}P_{t'} \leq K_t - a_t \qquad t = 1, 2, .., N \tag{11}$$

$$-\sum_{t'=1}^{N} b_{t't}P_{t'} \leq a_t - M_t \qquad t = 1, 2, .., N \tag{12}$$

$$P_t \leq \overline{P_t} \qquad t = 1, 2, .., N \tag{13}$$

$$-P_t \leq -\underline{P_t} \qquad t = 1, 2, .., N \tag{14}$$

From equations (10), (11), (12), (13) and, (14) we have a quadratic objective function and linear constraints, as a result, P_0 is a quadratic programming problem and it can be represented as:

$$\begin{aligned} &\max_{P_1, P_2, .., P_N} \Pi = \frac{1}{2}P^T W P + V^T P \\ &st: \qquad\qquad AP \leq E \end{aligned} \tag{15}$$

The last term D is omitted from the objective function because it's a constant and it doesn't have any influence on the optimal pricing policy. The matrix $A_{M\times N}$ is defined from the M constraints and the vector E contains the right side of each constraint. The values of A and E are given in the following two equations.

$$A_{M,N} = \begin{pmatrix} b_{11}, & b_{21}, & \ldots & , b_{N1} \\ -b_{11}, & -b_{21}, & \ldots & , -b_{N1} \\ 1, & 0, & \ldots & , 0 \\ -1, & 0, & \ldots & , 0 \\ \vdots & \vdots & \vdots & \vdots \\ b_{1N}, & b_{2N}, & \ldots & , b_{NN} \\ -b_{1N}, & -b_{2N}, & \ldots & , -b_{NN} \\ 0, & 0, & \ldots & , 1 \\ 0, & 0, & \ldots & , -1 \end{pmatrix} \tag{16}$$

$$E_{M,1} = \begin{pmatrix} K_1 - a_1 \\ a_1 - M_1 \\ \overline{P_1} \\ -\underline{P_1} \\ \vdots \\ K_N - a_N \\ a_N - M_N \\ \overline{P_N} \\ -\underline{P_N} \end{pmatrix} \tag{17}$$

Since the problem P_0 is quadratic and all the constraints are linear, two cases are distinguished regarding the convexity of the objective function Π. When Π is not concave i.e the matrix W is not definite or semi-definite negative, the problem is not convex and it can be solved using non-linear programming algorithms such as interior-point method, gradient methods, etc. However, all these methods reach in generally a local optimum. When Π is concave i.e the matrix W is definite or semi-definite negative, P_0 is a convex programming problem and it can be solved optimally.

First, let consider the case when Π is concave, one way to find the optimal solution of P_0, is the resolution of Karush-Kuhn-Tucker or KKT system related to P_0. The KKT conditions generally aren't sufficient i.e if a point P^* is a solution for the KKT system, then P^* can be a local optimum, a global optimum, or saddle point. However, when dealing with a convex programming problem, the KKT conditions became sufficient and any solution of the KKT system is a global optimum of the considering problem. In the rest of the section, the KKT system for the problem P_0 when this later is a convex programming problem is presented.

Let Π' = -Π and P_0' the problem presented as follows:

$$\begin{aligned} &\min_{P_1,P_2,..,P_N} \Pi' = \frac{1}{2} P^T W' P + V'^T P \\ &st: \qquad\qquad AP \leq E \end{aligned} \tag{18}$$

Note that the matrix W' = -W and the vector V'^T = -V^T. The resolution of P_0 to optimality is equivalent to the resolution of P_0' to optimality. Furthermore, P_0' is considered to define the KKT system. Before the presentation of the KKT system, some new notations are introduced:

$A_j P - E_j = g_j(P) \qquad j = 1, 2, .., M$
$\lambda^T = (\lambda_1,\lambda_2,..,\lambda_M) \qquad \lambda_j$ is the jth KKT multiplier with $j = 1, 2, .., M$

The KKT system related to P_0' is detailed below:

$$\nabla \Pi'(P) + \sum_{j=1}^{M} \lambda_j \nabla g_j(P) = 0 \qquad N \text{ equations} \tag{19}$$

$$g_j(P) \leq 0 \qquad j = 1, 2, .., M \qquad M \text{ equations} \tag{20}$$

$$\lambda_j g_j(P) = 0 \qquad j = 1, 2, ..., M \qquad M \text{ equations} \tag{21}$$

$$\lambda_j \geq 0, \qquad j = 1, 2, ..., M \tag{22}$$

After computing the gradients related to the first N equations, they are represented as $P^T W' + V'^T + \lambda^T A = 0$. Regarding the equations (20), they can be replaced by $(AP - E) \leq 0$. The value of $\lambda_j g_j(P) \leq 0 \ \forall j$, since from the equations (20) and (22) we have $g_j(P) \leq 0 \ \forall j$ and $\lambda_j \geq 0 \ \forall j$, respectively. As a result, the M equations related to $\lambda_j g_j(P) = 0 \ \forall j$ are replaced by the constraint $\lambda_1 g_1(P) + \lambda_2 g_2(P) + ... + \lambda_M g_M(P) = 0$ which corresponds to $\lambda^T(AP - E) = 0$. The last sum is equal to 0 if and only if each term $\lambda_j g_j(P) = 0 \ \forall j$. The KKT system for P_0' can be represented

as:

$$P^T W' + V'^T + \lambda^T A = 0 \qquad N \text{ equations} \tag{23}$$
$$AP - E \leq 0 \qquad M \text{ equations} \tag{24}$$
$$\lambda^T (AP - E) = 0 \qquad 1 \text{ equation} \tag{25}$$
$$\lambda_j \geq 0, \qquad j = 1, 2, ..., M \tag{26}$$

Regarding the first N equations, they are represented as a one line vector $(1 \times N)$. The constraints remain the same if we consider the transpose of $P^T W' + V'^T + \lambda^T A = 0$ which is equal to $W'P + V' + A^T\lambda$. Now, considering the equations $AP - E \leq 0$ and $\lambda^T(AP - E) = 0$, the vector $S = (s_1, s_2, ..., s_M)^T$ is added, with $s_i \geq 0$ such that $AP - E + S = 0$ and $\lambda^T(-S) = 0$. The last term $\lambda^T(-S) = 0$ is equal to $-\lambda_1^T s_1 - \lambda_2^T s_2 - ... - \lambda_M^T s_M = 0$. As each term $-\lambda_j^T s_j \leq 0 \; \forall j$, the term $\lambda^T(-S) = 0$ can be replaced by $\lambda^T(S) = 0$. The KKT system is represented as follows:

$$W'P + V' + A^T\lambda = 0 \qquad N \text{ equations} \tag{27}$$
$$AP - E + S = 0 \qquad M \text{ equations} \tag{28}$$
$$\lambda^T(S) = 0 \qquad 1 \text{ equation} \tag{29}$$
$$\lambda_j \geq 0, \qquad j = 1, 2, ..., M \tag{30}$$

Finally, the matrix representation of the KKT system of P_0' is:

$$\begin{pmatrix} W', & A^T & 0 \\ A, & 0 & I_M \end{pmatrix} \begin{pmatrix} P \\ \lambda \\ S \end{pmatrix} = \begin{pmatrix} -V' \\ E \end{pmatrix}$$
$$\lambda_j s_j = 0, \qquad j = 1, 2, ..., M$$
$$\lambda_j \geq 0, \qquad j = 1, 2, ..., M$$
$$s_j \geq 0, \qquad j = 1, 2, ..., M$$

3.2 Case with Isoelastic demand function

The isoelastic demand function also called the constant elasticity function is the simplest nonlinear demand function. One of its advantages is that it does not require a finite upper limit of the price Huang et al. (2013). The same demand as the one studied in Bajwa et al. (2016), and Couzon et al. (2020) is considered. The demand in a period t depends only on the price of the product in the same period (equation 31). $(\gamma \geq 0)$ is the seasonality factor. β is the price elasticity of demand, it measures the percentage change in the quantity demanded for a product in relation to percentage change in its price. According to the authors in Phillips (2005), the price elasticity is defined as $\beta = \frac{-pD(p)}{d'(p)}$. Since $d'(p) \leq 0$ (downward-slopping of the demand), the value of $\beta \geq 0$.

$$Q_t = \alpha \gamma_t P_t^{-\beta} \tag{31}$$

Replacing Q_t its value from (31), the total profit function Π is equal to:

$$\Pi = \sum_{t=1}^{N} \alpha \gamma_t P_t^{-\beta} (P_t - C_t) \tag{32}$$
$$\Pi = \alpha \Big[\gamma_1 P_1^{-\beta} (P_1 - C_1) + \gamma_2 P_2^{-\beta} (P_2 - C_2) + \ldots + \gamma_N P_N^{-\beta} (P_N - C_N) \Big] \tag{33}$$
$$\Pi = \alpha \sum_{t=1}^{N} f_t(P_t) \tag{34}$$
$$with \; f_t(P_t) = \gamma_t P_t^{-\beta} (P_t - C_t) \tag{35}$$

Regarding the capacity constraints $M_t \leq Q_t \leq K_t \ \forall t$, they are represented as:

$$M_t \leq \alpha\gamma_t P_t^{-\beta} \leq K_t \tag{36}$$

$$\frac{M_t}{\alpha\gamma_t} \leq P_t^{-\beta} \leq \frac{K_t}{\alpha\gamma_t} \tag{37}$$

$$\left(\frac{K_t}{\alpha\gamma_t}\right)^{-\frac{1}{\beta}} \leq P_t \leq \left(\frac{M_t}{\alpha\gamma_t}\right)^{-\frac{1}{\beta}} \tag{38}$$

We define I_t as : $I_t = [\underline{P_t}, \overline{P_t}] \cap \left[\left(\frac{K_t}{\alpha\gamma_t}\right)^{-\frac{1}{\beta}}, \left(\frac{M_t}{\alpha\gamma_t}\right)^{-\frac{1}{\beta}}\right] = [a_t, b_t]$, then, P_0 can be written as:

$$\max_{P_1,P_2,..,P_N} \Pi = \alpha \sum_{t=1}^{N} f_t(P_t)$$
$$s.t\ P_t \in\ [a_t, b_t]$$

Finding the optimal pricing policy is equivalent to find the value P_{t_max} which maximizes f_t, i.e $P^* = (\max f_1(P_1), \max f_2(P_2)..., \max f_N(P_N))$. The optimal selling price for each f_t is obtained analytically through the study of f_t's variation. The derivative of f_t, and the value P_{0t} for which $f_t'(P_t) = 0$ are presented in equations (39) and (40) respectively. When $\beta \leq 1$, $f_t'(P_t) > 0 \ \forall\ P_t \geq 0$, which implies that f_t is increasing in $[0, +\infty[$ specially in $[a_t, b_t]$, then $P_{t\_max} = b_t$. When $\beta > 1$, $P_{0t} > 0$ and $0 < C_t < P_{0t}$. As a result, $f_t' \geq 0$ for P_t in $]0, P_{0t}]$ and $f_t' \leq 0$ for $P_t \geq P_{0t}$. This means that, f_t is increasing in $]0, P_{0t}]$ and decreasing in $[P_{0t}, +\infty[$. Regarding the order between P_{0t}, a_t and b_t, we can consider the following three cases:

1. if $a_t \leq P_{0t} \leq b_t$ then $P_{t\_max} = P_{0t}$
2. if $b_t \leq P_{0t}$ then $P_{t\_max} = b_t$
3. if $a_t \geq P_{0t}$ then $P_{t\_max} = a_t$

$$f_t' = \alpha\gamma_t\Big(P_t^{-(\beta+1)}\big(P_t(1-\beta) + \beta.C_t\big)\Big) \tag{39}$$

$$P_{0t} = \frac{\beta}{\beta - 1} C_t \tag{40}$$

4 Numerical Experiments

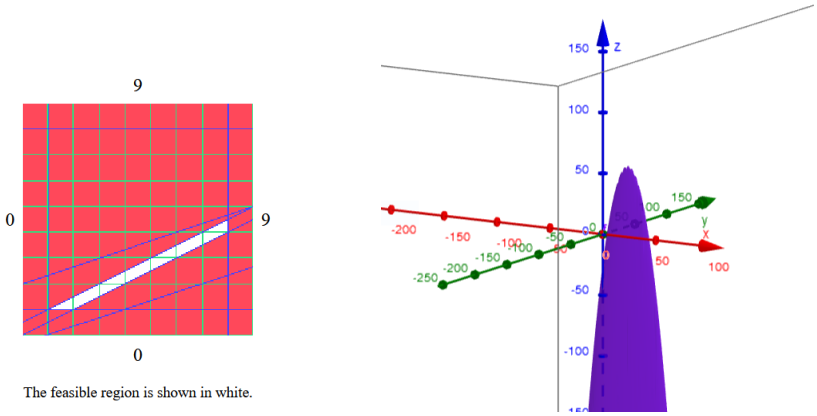
In this section, two numerical examples are presented to illustrate the proposed approach.

Example 1:

The linear demand function is considered, the instance's parameters are generated randomly (table 1). The number of periods is fixed to $N = 2$. The total profit function Π is concave since ($2b_{11} = -2 < 0$, $2b_{22} = -6 < 0$ and $2b_{11}.2b_{22} - (b_{12} + b_{21})^2 = 3 > 0$), therefore the optimization problem P_0 is a convex programming problem. The feasible region X and the function Π are shown in figures (1a) and (1b) respectively.

Parameters	$t = 1$	$t = 2$
C_t	2	3
a_t	3	8
M_t	3	2
K_t	4	9
$\overline{P_t}$	8	8
$\underline{P_t}$	1	1
b_{tj}	$b_{11} = -1, b_{12} = 1$	$b_{21} = 2, b_{22} = -3$

Table 1: Instance 1 Parameters values



(a) feasible region (X) (b) Profit function Π

Fig. 1: Total profit function and feasible area

Before the resolution, the nature of the optimal pricing policy P^* is considered. Since the total profit function is concave and the global maximum of Π in $\mathbb{R}^2$ is $P_{max} = (17, \frac{32}{3}) \notin X$, any interior point of X is not an optimal solution for the problem P_0. As a result, P^* belongs to the boundary of X.

We apply the KKT system as defined in the section 3.1 (see **Appendix A** for details). The KKT system is implemented and resolved using **gekko** package on Python3. The optimal pricing policy is $P^* = (P_1^* = 8, P_2^* = \frac{9}{2})$ and the optimal total profit value is $\Pi^* = 27.75$. One can remark that $P_1^* = \overline{P_1} = 8$ and $-P_1^* + 2P_2^* = 1$, which confirm that P^* belongs to the boundary of X.

Example 2:
The isoelastic demand is considered with $\beta = 2$ and, $\alpha = 100$. The values of N, C_t, M_t, K_t, $\underline{P_t}$ and $\overline{P_t}$ are the same as for Example 1. The seasonality parameters are fixed to $\gamma_1 = \gamma_2 = 0.5$. The red and blue curves in the following figure represent $f_1(P_1)$ and $f_2(P_2)$ respectively.

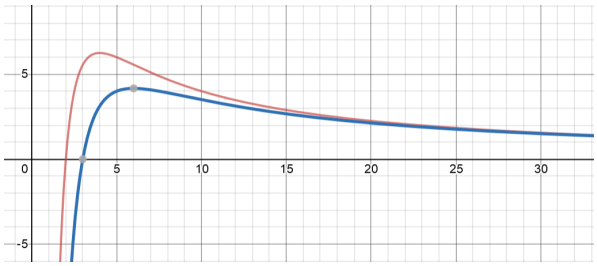


Fig. 2: Curves of f_1 and f_2

Table 2 shows the values of a_t, b_t, P_{0t}, and P_t^* which are computed by following the steps described in the section 3.2. For f_1, the value of P_{01} belongs to $[a_1, b_1]$, as a result $P_1^* = P_{01} = 4$. Regarding the function f_2, the value of $P_{02} \geq b_2$, thus, $P_2^* = b_2 = 5$. The total profit function $\Pi = f_1(P_1^*) + f_2(P_2^*) = 6.25 + 4 = 10,25$.

5 Conclusion

This paper investigated the dynamic pricing problem adopted from Shakya et al. (2012), in which a firm produces and sells its product over a finite horizon. The problem considers constraints such as limited production capacity and production costs. The firm has to set its selling

Parameters	$t = 1$	$t = 2$
a_t	3.53	2.35
b_t	4.08	5
P_{0t}	4	6
P_t^*	4	5

Table 2: Prices' intervals and optimal pricing policy

prices such that the total profit is maximized.

As a first contribution, the case when the demand at a period t is a linear function of the price in the same period and the prices of the other periods is studied. It has been shown that under these assumptions the problem can be formulated as a quadratic programming problem. The Karush-Kuhn-Tucker system to obtain the optimal pricing policy when the total profit function is concave is presented.

The second contribution consists of the consideration of the isoelastic demand function which is commonly used in the literature. It has been proven that when dealing with this demand function, the objective function is the sum of N univariate functions over N bounded intervals. As a result, the optimal pricing policy is resumed to find the maximum of each function.

The presented work assumes that the selling price is the only factor that influences demand. However, consumers are generally sensitive to other parameters like the lead time, rebate, and competitor prices. One extension of this work is the incorporation of these parameters to the demand function to achieve a more accurate representation of the real market behavior.

Acknowledgment

The authors would like to acknowledge the support from the European Regional Development Fund (FEDER) and the Industrial Chair Connected-Innovation (https://chaire-connected-innovation.fr/)

A Appendix A

A.1 KKT system for Example 1

From the parameters values presented in the table 1, the problem optimization problem is formulated as:

$$\min_{P_1,P_2} \Pi' = \frac{1}{2}(P_1,P_2)\begin{pmatrix}2 & -3\\ -3 & 6\end{pmatrix}\begin{pmatrix}P_1\\P_2\end{pmatrix} + (-2,-13)\begin{pmatrix}P_1\\P_2\end{pmatrix}$$

$$s.t: \begin{pmatrix}-1 & 2\\ 1 & -2\\ 1 & 0\\ -1 & 0\\ 1 & -3\\ -1 & 3\\ 0 & 1\\ 0 & -1\end{pmatrix}\begin{pmatrix}P_1\\P_2\end{pmatrix} \leq E_{8,1} = \begin{pmatrix}1\\0\\8\\-1\\1\\6\\8\\-1\end{pmatrix}$$

The KKT system is defined as:

$$\begin{pmatrix}2 & -3\\ -3 & 6\end{pmatrix}\begin{pmatrix}P_1\\P_2\end{pmatrix} + \begin{pmatrix}-2\\-13\end{pmatrix} + \begin{pmatrix}-1 & 1 & 1 & -1 & 1 & -1 & 0 & 0\\ 2 & -2 & 0 & 0 & -3 & 3 & 1 & -1\end{pmatrix}\begin{pmatrix}\lambda_1\\\lambda_2\\\lambda_3\\\lambda_4\\\lambda_5\\\lambda_6\\\lambda_7\\\lambda_8\end{pmatrix} = \begin{pmatrix}0\\0\end{pmatrix}$$

$$\begin{pmatrix}-1 & 2\\ 1 & -2\\ 1 & 0\\ -1 & 0\\ 1 & -3\\ -1 & 3\\ 0 & 1\\ 0 & -1\end{pmatrix}\begin{pmatrix}P_1\\P_2\end{pmatrix} - \begin{pmatrix}1\\0\\8\\-1\\1\\6\\8\\-1\end{pmatrix} + \begin{pmatrix}s_1\\s_2\\s_3\\s_4\\s_5\\s_6\\s_7\\s_8\end{pmatrix} = \begin{pmatrix}0\\0\\0\\0\\0\\0\\0\\0\end{pmatrix}$$

$$(\lambda_1,\lambda_2,\lambda_3,\lambda_4,\lambda_5,\lambda_6,\lambda_7,\lambda_8)\begin{pmatrix}s_1\\s_2\\s_3\\s_4\\s_5\\s_6\\s_7\\s_8\end{pmatrix} = 0$$

$$\lambda_j \geq 0, \qquad j = 1,2,...,8$$
$$s_j \geq 0, \qquad j = 1,2,...,8$$

Bibliography

Bajwa Naeem, Fontem Belleh, Sox Charles R. Optimal product pricing and lot sizing decisions for multiple products with nonlinear demands // Journal of Management Analytics. 2016. 3, 1. 43–58.

Bajwa Naeem, Sox Charles R. Coordination of pricing, advertising, and production decisions for multiple products // International Journal of Services and Operations Management. 2015. 22, 4. 495–521.

Bitran Gabriel, Caldentey René. An overview of pricing models for revenue management // Manufacturing & Service Operations Management. 2003. 5, 3. 203–229.

Bitran Gabriel R, Mondschein Susana V. An application of yield management to the hotel industry considering multiple day stays // Operations research. 1995. 43, 3. 427–443.

Boer Arnoud V den. Dynamic pricing and learning: historical origins, current research, and new directions // Surveys in operations research and management science. 2015. 20, 1. 1–18.

Chen Le, Mislove Alan, Wilson Christo. An empirical analysis of algorithmic pricing on amazon marketplace // Proceedings of the 25th International Conference on World Wide Web. 2016. 1339–1349.

Chen Xin, Simchi-Levi David. Pricing and inventory management // The Oxford handbook of pricing management. 2012. 1. 784–824.

Couzon Paulin, Ouazene Yassine, Yalaoui Farouk. Joint pricing and lot-sizing problem with variable capacity // IFAC-PapersOnLine. 2019. 52, 13. 106–111.

Couzon Paulin, Ouazene Yassine, Yalaoui Farouk. Joint optimization of dynamic pricing and lot-sizing decisions with nonlinear demands: Theoretical and computational analysis // Computers & Operations Research. 2020. 115. 104862.

Elmaghraby Wedad, Keskinocak Pınar. Dynamic pricing in the presence of inventory considerations: Research overview, current practices, and future directions // Management science. 2003. 49, 10. 1287–1309.

Huang Jian, Leng Mingming, Parlar Mahmut. Demand functions in decision modeling: A comprehensive survey and research directions // Decision Sciences. 2013. 44, 3. 557–609.

Narahari Y, Raju CVL, Ravikumar K, Shah Sourabh. Dynamic pricing models for electronic business // Sadhana. 2005. 30, 2-3. 231–256.

Ouazene Yassine, Yalaoui Farouk, Kelly Russel, Idjeraoui Tayeb. Coordination and optimization of dynamic pricing and production decisions // 2017 IEEE Symposium Series on Computational Intelligence (SSCI). 2017. 1–6.

Phillips Robert Lewis. Pricing and revenue optimization. 2005.

Shakya Siddhartha, Kern Mathias, Owusu Gilbert, Chin Choong Ming. Neural network demand models and evolutionary optimisers for dynamic pricing // Knowledge-Based Systems. 2012. 29. 44–53.

Smith Barry C, Leimkuhler John F, Darrow Ross M. Yield management at American airlines // interfaces. 1992. 22, 1. 8–31.

Embedding Simulated Annealing within Stochastic Gradient Descent*

Matteo Fischetti[0000−0001−6601−0568] and Matteo Stringher

Department of Information Engineering
University of Padova
via Gradenigo 6/A, I-35100 Padova, Italy
matteo.fischetti@unipd.it
stringher.matteo@gmail.com

Abstract. We propose a new metaheuristic training scheme for Machine Learning that combines Stochastic Gradient Descent (SGD) and Discrete Optimization in an unconventional way. Our idea is to define a discrete neighborhood of the current SGD point containing a number of "potentially good moves" that exploit gradient information, and to search this neighborhood by using a classical metaheuristic scheme borrowed from Discrete Optimization. In the present paper we investigate the use of a simple Simulated Annealing (SA) metaheuristic that accepts/rejects a candidate new solution in the neighborhood with a probability that depends both on the new solution quality and on a parameter (the *temperature*) which is modified over time to lower the probability of accepting worsening moves.
Computational results on image classification (CIFAR-10) are reported, showing that the proposed approach leads to an improvement of the final validation accuracy for modern Deep Neural Networks such as ResNet34 and VGG16.

Keywords: Simulated Annealing · Stochastic Gradient Descent · Deep Neural Networks · Machine Learning · Training Algorithm

1 Introduction

Machine Learning (ML) is a fundamental topic in Artificial Intelligence. Its growth in the research community has been followed by a huge rise in the number of projects in the industry leveraging this technology.

Deep learning is a subset of ML, based on learning data representation through the use of neural network architectures, specifically Deep Neural Networks (DNNs). Inspired by human processing behavior, DNNs have set new state-of-art results in speech recognition, visual object recognition, object detection, and many other domains.

* Work supported by MiUR, Italy (project PRIN). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

Stochastic Gradient Descent (SGD) is *de facto* the standard algorithm for training Deep Neural Networks (DNNs). Leveraging the gradient, SGD allows one to rapidly find a good solution in the very high dimensional space of weights associated with modern DNNs. Moreover, the use of minibatches allows one to exploit modern GPUs and to achieve a considerable computational efficiency.

In the present paper we investigate the use of an alternative training method, namely, the Simulated Annealing (SA) algorithm [8]. The use of SA for training is not new, but previous proposals are mainly intended to be applied for non-differentiable objective functions for which SGD is not applied due to the lack of gradients; see, e.g., [15,10]. Instead, our SA method requires differentiability of (a proxy of) the loss function, and leverages on the availability of a gradient direction to define local moves that have a large probability to improve the current solution.

Our approach is computationally evaluated in an implementaion leveraging hyper-parameters. Assume some hyper-parameter values (e.g., learning rates for SGD) are collected in a discrete set H. At each SGD iteration, we randomly pick one hyper-parameter from H, temporarily implement the corresponding *move* as in the classical SGD method (using the gradient information) and evaluate the new point on the current minibatch. If the loss function does not deteriorate too much, we *accept* the move as in the classical SGD method, otherwise we *reject* it: we step back to the previous point, change the minibatch, randomly pick another hyper-parameter from H, and repeat. The decision of accepting/rejecting a move is based on the classical SA criterion, and depends of the amount of loss-function worsening and on a certain parameter (the *temperature*) which is modified over time to lower the probability of accepting worsening moves.

A distinctive feature of our scheme is that hyper-parameters are modified within a *single* SGD execution (and not in an external loop, as customary) and evaluated on the fly on the current minibatch, i.e., their tuning is fully embedded within the SGD algorithm.

Computational results are reported, showing that the proposed approach leads to an improvement of the final validation accuracy for modern DNN architectures (ResNet34 and VGG16 on CIFAR-10).

2 Simulated Annealing

The basic SA algorithm for a generic optimization problem can be outlined as follows. Let S be the set of all possible feasible solutions, and $f : S \to \mathbb{R}$ be the objective function to be minimized. An optimal solution s^* is a solution in S such that $f(s^*) \leq f(s)$ holds for all $s \in S$.

SA is an iterative method that constructs a trajectory of solutions $s^{(0)}, \cdots, s^{(k)}$ in S. At each iteration, SA considers moving from the current feasible solution $s^{(i)}$ (say) to a candidate new feasible solution s_{new} (say). Let $\Delta(s^{(i)}, s_{new}) = f(s_{new}) - f(s^{(i)})$ be the objective function *worsening* when moving from $s^{(i)}$ to s_{new}—positive if s_{new} is strictly worse than $s^{(i)}$. The hallmark of SA is that worsening moves are not forbidden but accepted with a certain *acceptance*

probability $p(s^{(i)}, s_{new}, T)$ that depends on the amount of worsening $\Delta(s^{(i)}, s_{new})$ and on a parameter $T > 0$ called *temperature*. A typical way to compute the acceptance probability is through *Metropolis' formula* [11]:

$$p(s, s_{new}, T) = \begin{cases} e^{-\Delta(s^{(i)}, s_{new})/T} & \text{if} \quad \Delta(s^{(i)}, s_{new}) > 0 \\ 1 & \text{if} \quad \Delta(s^{(i)}, s_{new}) \leq 0 \ . \end{cases} \tag{1}$$

Thus, the probability of accepting a worsening move is large if the amount of worsening $\Delta(s^{(i)}, s') > 0$ is small and the temperature T is large. Note that the probability is 1 when $\Delta(s^{(i)}, s') \leq 0$, meaning that improving moves are always accepted by the SA method.

Temperature T is a crucial parameter: it is initialized to a certain value T_0 (say), and iteratively decreased during the SA execution so as to make worsening moves less and less likely in the final iterations. A simple update formula for T is $T = \alpha \cdot T$, where $\alpha \in (0, 1)$ is called *cooling factor*. Typical ranges for this parameter are $0.95 - 0.99$ (if cooling is applied at each SA iteration) or $0.7 - 0.8$ (if cooling is only applied at the end of a "computational epoch", i.e., after several SA iterations with a constant temperature).

The basic SA scheme is outlined in Algorithm 1; more advanced implementations are possible, e.g., the temperature can be restored multiple times to the initial value.

Algorithm 1 : SA

Input: function f to be minimized, initial temperature $T_0 > 0$, cooling factor $\alpha \in (0, 1)$, number of iterations $nIter$
Output: the very last solution $s^{(nIter)}$

1: Compute an initial solution $s^{(0)}$ and initialize $T = T_0$
2: **for** $i = 0, \ldots, nIter - 1$ **do**
3: Pick a new tentative solution s_{new} in a convenient neighborhood $\mathcal{N}(s^{(i)})$ of $s^{(i)}$
4: $worsening = f(s_{new}) - f(s^{(i)})$
5: $prob = e^{-worsening/T}$
6: **if** $random(0, 1) < prob$ **then**
7: $s^{(i+1)} = s_{new}$
8: **else**
9: $s^{(i+1)} = s^{(i)}$
10: **end if**
11: $T = \alpha \cdot T$
12: **end for**

At Step 6, $random(0, 1)$ is a pseudo-random value uniformly distributed in [0,1]. Note that, at Step 5, the acceptance probability *prob* becomes larger than 1 in case $worsening < 0$, meaning that improving moves are always accepted (as required).

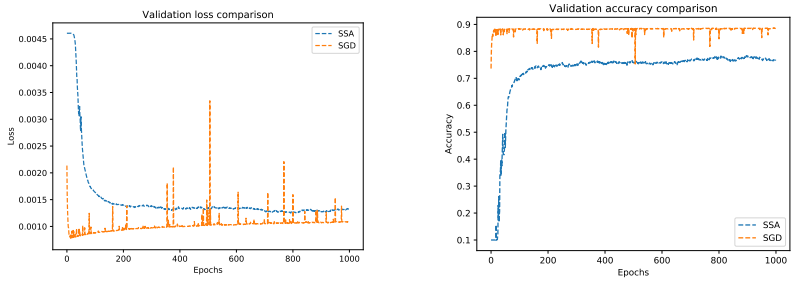
2.1 A naive implementation for training without gradients

In the context of training, one is interested in minimizing a *loss function* $L(w)$ with respect to a large-dimensional vector $w \in \Re^M$ of so-called *weights*. If $L(w)$ is differentiable (which is not required by the SA algorithm), there exists a gradient $\nabla(w)$ giving the steepest increasing direction of L when moving from a given point w.

Here is a very first attempt to use SA in this setting. Given the current solution (i.e., set of weights) w, we generate a random move $\Delta(w) \in \Re^M$ and then we evaluate the loss function in the nearby point $w' := w - \epsilon\Delta(w)$, where ϵ is a small positive real number. If the norm of $\epsilon\Delta(w)$ is small enough and L is differentiable, due to Taylor's approximation we know that

$$L(w') \simeq L(w) - \epsilon \, \nabla^T(w)\Delta(w) \; . \tag{2}$$

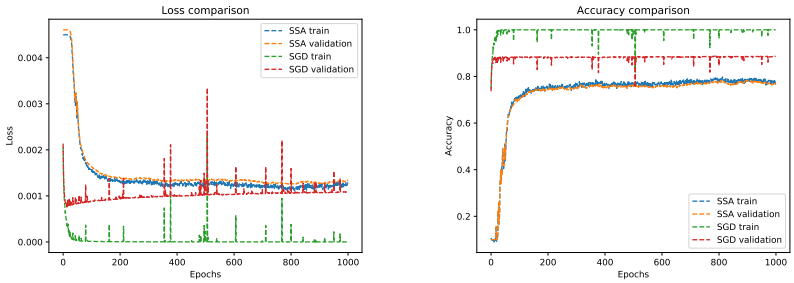
Thus the objective function improves if $\nabla(w)^T\Delta(w) > 0$. As we work in the continuous space, in the attempt of improving the objective function we can also try to move in the opposite direction and move to $w'' := w + \epsilon \, \Delta(w)$. Thus, our actual move from the current w consists of picking the best (in terms of objective function) point w_{new}, say, between the two nearby points w' and w'': if w_{new} improves $L(w)$, then we surely accept this move; otherwise we accept it according to the Metropolis' formula (1). Note that the above SA approach is completely derivative free: as a matter of fact, SA could optimize directly over discrete functions such as the accuracy in the context of classification.



(a) Validation loss (b) Validation accuracy

Fig. 1: Performance on the validation set of our naive SA implementation (SSA) for VGG16 on Fashion-MNIST. SGD: learning rate $\eta = 0.001$, no momentum/Nesterov acceleration. `SSA`: $\epsilon = 0.01$, $\alpha = 0.97$, $T_0 = 1$.

In a preliminary phase of our work we implemented the simple scheme above in a stochastic manner, using minibatches when evaluating $L(w')$ and $L(w'')$, very much in the spirit of the SGD algorithm. Figures 1–2, compare the performance of the resulting Stochastic SA algorithm, called `SSA`, with that of a straightforward



(a) Loss comparison (b) Accuracy comparison

Fig. 2: Comparison of our naive SA implementation (SSA) vs SGD for VGG16 on Fashion-MNIST. SGD: learning rate $\eta = 0.001$, no momentum/Nesterov acceleration. `SSA`: $\epsilon = 0.01$, $\alpha = 0.97$, $T_0 = 1$. Subfigure (b) clearly shows that `SSA` has no overfitting but is not able to exploit the full capacity of VGG16, resulting into an unsatisfactory final accuracy.

SGD implementation with constant learning rate and no momentum [20] nor Nesterov [13] acceleration, using the Fashion-MNIST [21] dataset and the VGG16 [16] architecture. Figure 2(b) reports accuracy on both the training and the validation sets, showing that `SSA` does not suffer from overfitting as the accuracy on the training and validation sets are almost identical—a benefit deriving from the derivative-free nature of `SSA`. However, `SSA` is clearly unsatisfactory in terms of validation accuracy (which is much worse than the SGD one) in that it does not exploit well the VGG16 capacity.

We are confident that the above results could be improved by a more advanced implementation. E.g., one could vary the value of ϵ during the algorithm, and/or replace the loss function by (one minus) the accuracy evaluated on the current minibatch—recall that `SSA` does not require the objective function be differentiable. However, even an improved `SSA` implementation is unlikely to be competitive with SGD. In our view, the main drawback of the `SSA` algorithm (as stated) is that, due the very large dimensional space, the random direction $\pm\Delta(w)$ is very unlikely to lead to a substantial improvement of the objective function as the effect of its components tend to cancel out randomly. Thus, a more clever definition of the basic move is needed to drive `SSA` in an effective way.

3 Improved SGD training by SA

We next introduce an unconventional way of using SA in the context of training. We assume the function $L(w)$ to be minimized be differentiable, so we can compute its gradient $\nabla(w)$. From SGD we borrow the idea of moving in the anti-gradient direction $-\nabla(w)$, possibly corrected using momentum/Nesterov

acceleration techniques. Instead of using a certain *a priori* learning rate η, however, we randomly pick one from a discrete set H (say) of possible candidates. In other words, at each SA iteration the move is selected randomly in a discrete neighborhood $\mathcal{N}(w^{(i)})$ whose elements correspond to SGD iterations with different learning rates. An important feature of our method is that H can (actually, should) contain unusually large learning rates, as the corresponding moves can be discarded by the Metropolis' criterion if they deteriorate the objective function too much.

A possible interpretation of our approach is in the context of SGD hyper-parameter tuning. According to our proposal, hyper-parameters are collected in a discrete set H and sampled *within a single SGD execution*: in our tests, H just contains a number of possible learning rates, but it could involve other parameters/decisions as well, e.g., applying momentum, or Nesterov (or none of the two) at the current SGD iteration, or alike. The key property here is that any element in H corresponds to a reasonable (non completely random) move, so picking one of them at random has a significant probability of improving the objective function. As usual, moves are accepted according to the Metropolis' criterion, so the set H can also contain "risky choices" that would be highly inefficient if applied systematically within a whole training epoch.

Algorithm 2 : `SGD-SA`

Parameters: A set of learning rates H, initial temperature $T_0 > 0$
Input: Differentiable loss function L to be minimized, cooling factor $\alpha \in (0, 1)$, number of epochs $nEpochs$, number of minibatches N
Output: the best performing $w^{(i)}$ on the validation set at the end of each epoch

Divide the training dataset into N minibatches
Initialize $i = 0$, $T = T_0$, $w^{(0)}$ = random_initialization()
for $t = 1, \ldots, nEpochs$ **do**
  for $n = 1, \ldots, N$ **do**
    Extract the n-th minibatch (x, y)
    Compute $L(w^{(i)}, x, y)$ and its gradient $v =$ backpropagation$(w^{(i)}, x, y)$
    Randomly pick a learning rate η from H
    $w_{new} = w^{(i)} - \eta\, v$
    Compute $L(w_{new}, x, y)$
    $worsening = L(w_{new}, x, y) - L(w^{(i)}, x, y)$
    $prob = e^{-worsening/T}$
    if $random(0, 1) < prob$ **then**
      $w^{(i+1)} = w_{new}$
    else
      $w^{(i+1)} = w^{(i)}$
    end if
    $i = i + 1$
  end for
  $T = \alpha \cdot T$
end for

Our basic approach is formalized in Algorithm 2, and will be later referred to as `SGD-SA`. More elaborated versions using momentum/Nesterov are also possible but not investigated in the present paper, as we aim at keeping the overall computational setting as simple and clean as possible.

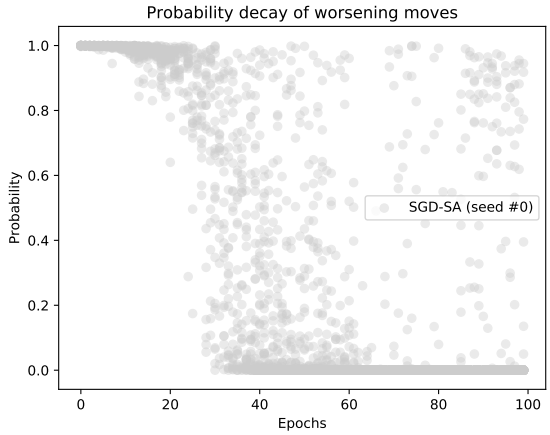
4 Computational analysis of `SGD-SA`

We next report a computational comparison of SGD and `SGD-SA` for a classical image classification task involving the CIFAR-10 [9] dataset. As customary, the dataset was shuffled and partitioned into 50,000 examples for the training set, and the remaining 10,000 for the test set. As to the DNN architecture, we tested two well-known proposals from the literature: VGG16 [16] and ResNet34 [5]. Training was performed for 100 epochs using PyTorch, with minibatch size 512. Tests have been performed using a single NVIDIA TITAN Xp GPU.

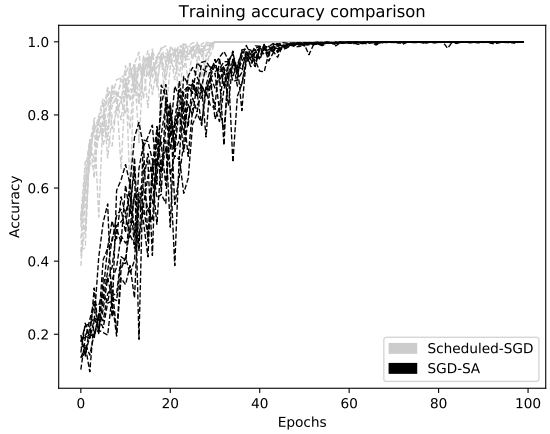
Our `Scheduled-SGD` implementation of SGD is quite basic but still rather effective on our dataset: it uses no momentum/Nesterov acceleration, and the learning rate is set according the following schedule: $\eta = 0.1$ for first 30 epochs, 0.01 for the next 40 epochs, and 0.001 for the final 30 epochs. As to `SGD-SA`, we used $\alpha = 0.8$, initial temperature $T_0 = 1$, and learning-rate set $H = \{0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.09, 0.08, 0.07, 0.06, 0.05\}$.

Both `Scheduled-SGD` and `SGD-SA` use pseudo-random numbers generated from an initial random seed, which therefore has some effects of the search path in the weight space and hence on the final solution found. Due to the very large number of weights that lead to statistical compensation effects, the impact of the seed on the initialization of the very first solution $w^{(0)}$ is very limited—a property already known for SGD that is inherited by `SGD-SA` as well. However, random numbers are used by `SGD-SA` also when taking some crucial "discrete" decisions, namely: the selection of the learning rate $\eta \in H$ (Step 7) and the acceptance test (Step 12). As a result, as shown next, the search path of `SGD-SA` is very dependent on the initial seed. Therefore, for both `Scheduled-SGD` and `SGD-SA` we decided to repeat each run 10 times, starting with 10 random seeds, and to report results for each seed. In our view, this dependency on the seed is in fact a *positive* feature of `SGD-SA`, in that it allows one to treat the seed as a single (quite powerful) hyper-parameter to be randomly tuned in an external loop.

Our first order of business is to evaluate the convergence property of `SGD-SA` on the training set—after all, this is the optimization task that SA faces directly. In Figure 3 we plot the average probability *prob* (clipped to 1) of accepting a move at Step 12, as well as the training-set accuracy as a function of the epochs. Subfigure 3a shows that the probability of accepting a move is almost one in the first epochs, even if the amount of worsening is typically quite large in this phase. Later on, the probability becomes smaller and smaller, and only very small worsenings are more likely to be accepted. As a result, large learning rates are automatically discarded in the last iterations. Subfigure 3b is quite interesting: even in our simple implementation, `Scheduled-SGD` quickly converges to the best-possible value of one for accuracy, and the plots for the various seeds (gray lines) are



(a) Probability of accepting worsening moves



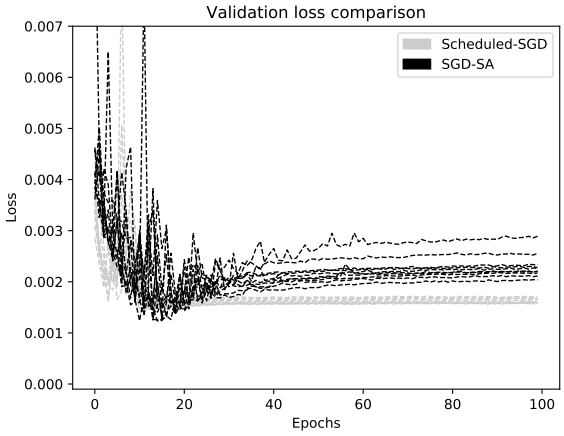
(b) Training accuracy (10 runs with different random seeds)

Fig. 3: Optimization efficiency over the training set (VGG16 on CIFAR-10)

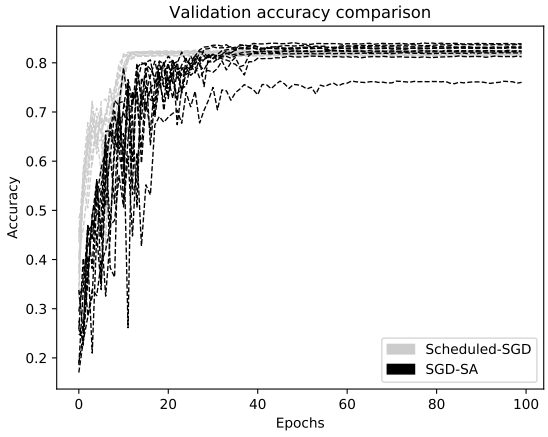
almost overlapping—thus confirming that the random seed has negligible effects of `Scheduled-SGD`. As to `SGD-SA` (black lines), its convergence to accuracy one is slower than `Scheduled-SGD`, and different seeds lead to substantially different curves—a consequence of the discrete random decisions taken along the search path.

Figure 4 shows the performance on the validation set of `Scheduled-SGD` and `SGD-SA` (both with 10 runs with different random seeds) when using the ResNet34 architecture—results with VGG16 are very similar, hence they are not reported.

As expected, the search path of `SGD-SA` is more diversified (leading to accuracy drops in the first epochs) but the final solutions tend to generalize better than `Scheduled-SGD`, as witnessed by the better performance on the validation set.



(a) Validation loss



(b) Validation accuracy

Fig. 4: ResNet34 on CIFAR-10 (validation set)

Table 1 gives more detailed results for each random seed, and reports the final validation accuracy and loss reached by `Scheduled-SGD` and `SGD-SA`. The results show that, for all seeds, `SGD-SA` always produces a significantly better (lower) validation loss than `Scheduled-SGD`. As to validation accuracy, `SGD-SA`

Method	Seed	VGG16		ResNet34	
		Loss	Accuracy	Loss	Accuracy
Scheduled-SGD	0	0.001640	85.27	0.001519	82.18
	1	0.001564	84.94	0.001472	82.58
	2	0.001642	84.84	0.001467	82.27
	3	0.001662	84.93	0.001468	82.37
	4	0.001628	84.92	0.001602	81.69
	5	0.001677	85.37	0.001558	81.80
	6	0.001505	84.91	0.001480	82.24
	7	0.001480	85.28	0.001532	82.07
	8	0.001623	85.26	0.001574	81.52
	9	0.001680	85.41	0.001499	82.41
`SGD-SA`	0	0.001127	86.44	0.001306	82.55
	1	0.001206	86.18	0.001231	84.11
	2	0.001121	86.04	0.001238	83.32
	3	0.001133	86.76	0.001457	81.39
	4	0.001278	85.17	0.001585	76.31
	5	0.001112	86.30	0.001276	83.74
	6	0.001233	85.71	0.001405	82.07
	7	0.001130	86.59	0.001261	82.57
	8	0.001167	86.14	0.001407	83.12
	9	0.001084	86.28	0.001240	83.19
Best `Scheduled-SGD`		0.001480	85.41	0.001467	82.58
Best `SGD-SA`		0.001084	86.76	0.001240	84.11

Table 1: Best validation accuracy and loss, seed by seed.

outperforms `Scheduled-SGD` for all seeds but seeds 3, 4 and 6 for ResNet34. In particular, `SGD-SA` leads to a significantly better (1-2%) validation accuracy than `Scheduled-SGD` if the best run for the 10 seeds is considered.

5 Conclusions and future work

We have proposed a new metaheuristic training scheme that combines Stochastic Gradient Descent and Discrete Optimization in an unconventional way.

Our idea is to define a discrete neighborhood of the current solution containing a number of "potentially good moves" that exploit gradient information, and to search this neighborhood by using a classical metaheuristic scheme borrowed from Discrete Optimization. In the present paper, we have investigated the use of a simple Simulated Annealing metaheuristic that accepts/rejects a candidate new solution in the neighborhood with a probability that depends both on the new solution quality and on a parameter (the temperature) which is varied over time. We have used this scheme as an automatic way to perform hyper-parameter

tuning within a single training execution, and have shown its potentials on a classical test problem (CIFAR-10 image classification using VGG16/ResNet34 deep neural networks).

In a follow-up research we plan to investigate the use of two different objective functions at training time: one differentiable to compute the gradient (and hence a set of potentially good moves), and one completely generic (possibly black-box) for the Simulated Annealing acceptance/rejection test—the latter intended to favor simple/robust solutions that are likely to generalize well.

Replacing Simulated Annealing with other Discrete Optimization metaheuristics (tabu search, variable neighborhood search, genetic algorithms, etc.) is also an interesting topic that deserves future research.

References

1. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**, 281–305 (Feb 2012), http://dl.acm.org/citation.cfm?id=2188385.2188395
2. Claesen, M., Moor, B.D.: Hyperparameter search in machine learning. CoRR **abs/1502.02127** (2015), http://arxiv.org/abs/1502.02127
3. Franzin, A., Stützle, T.: Revisiting simulated annealing: A component-based analysis. Computers & Operations Research **104** (12 2018). https://doi.org/10.1016/j.cor.2018.12.015
4. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Gordon, G., Dunson, D., Dudík, M. (eds.) Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 15, pp. 315–323. PMLR, Fort Lauderdale, FL, USA (11–13 Apr 2011), http://proceedings.mlr.press/v15/glorot11a.html
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. ArXiv e-prints (Dec 2015)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR **abs/1512.03385** (2015), http://arxiv.org/abs/1512.03385
7. Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P.T.P.: On large-batch training for deep learning: Generalization gap and sharp minima. CoRR **abs/1609.04836** (2016), http://arxiv.org/abs/1609.04836
8. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220 4598**, 671–80 (1983)
9. Krizhevsky, A., Nair, V., Hinton, G.: CIFAR-10 (Canadian Institute for Advanced Research). http://www.cs.toronto.edu/~kriz/cifar.html
10. Ledesma, S., Torres, M., Hernández, D., Aviña, G., García, G.: Temperature cycling on simulated annealing for neural network learning. In: Gelbukh, A., Kuri Morales, Á.F. (eds.) MICAI 2007: Advances in Artificial Intelligence. pp. 161–171. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
11. Metropolis, N.C., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equation of state calculation by fast computing machines. Journal of Chemical Physics **21**, 1087–1092 (1953)
12. Neelakantan, A., Vilnis, L., Le, Q.V., Sutskever, I., Kaiser, L., Kurach, K., Martens, J.: Adding gradient noise improves learning for very deep networks. CoRR **abs/1511.06807** (2015), https://arxiv.org/abs/1511.06807

13. Nesterov, Y.: A method of solving a convex programming problem with convergence rate O(1/sqr(k)). Soviet Mathematics Doklady **27**, 372–376 (1983), http://www.core.ucl.ac.be/~{}nesterov/Research/Papers/DAN83.pdf
14. Nøkland, A., Hiller Eidnes, L.: Training Neural Networks with Local Error Signals. arXiv e-prints arXiv:1901.06656 (Jan 2019)
15. Sexton, R., Dorsey, R., Johnson, J.: Beyond backpropagation: Using simulated annealing for training neural networks. Journal of End User Computing **11** (07 1999)
16. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. ArXiv e-prints (Sep 2014)
17. Smith, L.N.: Cyclical learning rates for training neural networks (2015), http://arxiv.org/abs/1506.01186, cite arxiv:1506.01186Comment: Presented at WACV 2017; see https://github.com/bckenstler/CLR for instructions to implement CLR in Keras
18. Smith, L.N.: A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay. CoRR **abs/1803.09820** (2018), http://arxiv.org/abs/1803.09820
19. Smith, S.L., Kindermans, P., Le, Q.V.: Don't decay the learning rate, increase the batch size. CoRR **abs/1711.00489** (2017), http://arxiv.org/abs/1711.00489
20. Sutskever, I., Martens, J., Dahl, G., Hinton, G.: On the importance of initialization and momentum in deep learning. In: Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28. pp. III–1139–III–1147. ICML'13, JMLR.org (2013), http://dl.acm.org/citation.cfm?id=3042817.3043064
21. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms (2017)
22. Zhang, H., Dauphin, Y.N., Ma, T.: Fixup initialization: Residual learning without normalization. CoRR **abs/1901.09321** (2019), http://arxiv.org/abs/1901.09321

Incorporating User Preferences in Multi-objective Feature Selection in Software Product Lines using Multi-Criteria Decision Analysis

Takfarinas Saber[1,2], Malika Bendechache[1,3], and Anthony Ventresque[1,2]

[1] Lero – the Irish Software Research Centre
[2] School of Computer Science, University College Dublin, Ireland
{takfarinas.saber, anthony.ventresque}@ucd.ie,
[3] School of Computing, Dublin City University, Ireland
malika.bendechache@dcu.ie

Abstract *Software Product Lines Engineering* has created various tools that assist with the standardisation in the design and implementation of clusters of equivalent software systems with an explicit representation of variability choices in the form of *Feature Models*, making the selection of the most ideal software product a *Feature Selection* problem. With the increase in the number of properties, the problem needs to be defined as a multi-objective optimisation where objectives are considered independently one from another with the goal of finding and providing decision-makers a large and diverse set of non-dominated solutions/products. Following the optimisation, decision-makers define their own (often complex) preferences on how does the ideal software product look like. Then, they select the unique solution that matches their preferences the most and discard the rest of the solutions—sometimes with the help of some Multi-Criteria Decision Analysis technique. In this work, we study the usability and the performance of incorporating preferences of decision-makers by carrying-out Multi-Criteria Decision Analysis directly within the multi-objective optimisation to increase the chances of finding more solutions that match preferences of the decision-makers the most and avoid wasting execution time searching for non-dominated solutions that are poor with respect to decision-makers' preferences.

Keywords: Feature Selection, Software Product Line, Multi-Objective Evolution Algorithm, Multi-Criteria Decision Analysis.

1 Introduction

Software Engineering is divided into multiple domains [1]. One of these domains is Software Product Lines (SPL) which considers groups of related software systems as a whole, rather than dealing with every single one of them separately [2]. Feature Models (FMs) is the most recurrent representation of SPLs. Furthermore,

the FM holds a listing of all the possible feature configurations/combinations which could be viewed as constraints. Therefore, making the FM a representation of all valid software products that could be made out the features in the SPL. Building a software product out of a particular SPL requires the selection of features that respect the desired software configuration. With the multiple characteristics/objectives that are interesting to decision-makers in practice (e.g., cost, technical feasibility, or reliability), the problem of finding the 'best' feature configuration is seen as an instance of a *multi-objective optimisation problem* [3,4].

Evolutionary algorithms have long been used to efficiently optimise problems in various domains from Computer Networks (e.g., [5–7]) to Intelligent Transport Systems (e.g., [8]), to Software Engineering, based on analytical/mathematical (e.g., [5,6]) or simulated (e.g., [8,9]) models. Evolutionary algorithms are particularly effective when dealing with multi-objective optimisation problems in software engineering (e.g., [10–13]). This is also the case for multi-objective feature selection in SPL for which the state-of-the-art SATIBEA [3] is an Indicator-Based Evolutionary Algorithm (IBEA) that uses a SAT solver as a mutation operator to correct infeasible solutions.

Multi-objective optimisation techniques result in a set of non-dominated products/solutions from which decision-makers select the product that fits their preferences the most. Given that the number of solutions in the set of non-dominated solutions is often large and that preferences of decision-makers are often complex, decision-makers are usually assisted by Multi-Criteria Decision Analysis (MCDA) tools to accomplish this task [14]. There exist multiple MCDA techniques that take decision-makers' preferences (each of them with its degree of preference expressibility) and return the product that match them the most. We show in this paper that: (i) some MCDA techniques are simplistic and can only handle a limited number of preference types (e.g., only take weights into accounts such as ELECTRE-IV), but they are fast, whereas (ii) other more elaborate MCDA techniques handle larger preference variations (e.g., they enable the use of different utility functions such as PROMETHEE-II), but they are slower and more time-consuming.

In this paper, we aim to include preferences of the decision-makers directly in the multi-objective search process to avoid spending a precious execution time searching for solutions that are (despite being non-dominated) far from decision-makers' preferences. In this paper, we study the effects of using MCDA techniques in the selection process of SATIBEA instead of the Indicator-Based technique (i.e., based on the contribution in Hypervolume of each solution). Particularly, we would like to evaluate the impact in terms of both: (i) the execution time overhead that it would induce, and (ii) quantity of non-dominated solutions matching preferences of decision-makers missed by SATIBEA.

This paper makes the following contributions:

– We propose SAT_MCDA_EA, a hybrid algorithm that includes decision-makers preferences in an MCDA form directly in the evolutionary search process.

– We show that using MCDA techniques as a selection operator has an insignificant impact in terms of execution time overhead in comparison to the execution time taken by one generation of SATIBEA.
– We also show that using MCDA techniques (particularly PROMETHEE-II) enables finding a large number of solutions which better match preferences of decision-makers and that are missed by SATIBEA (despite not outperforming SATIBEA on most of the multi-objective performance metrics).

Combining MCDA techniques with multi-objective evolutionary algorithms has already been attempted in a few recent works (e.g., [15–17]). However, to the best of our knowledge, this is the first time it is attempted in the Software Engineering domain in general and on the multi-objective feature selection in FM in particular.

The remainder of this paper is organised as follows: Section 2 presents the background of our study. Section 3 describes some common MCDA techniques and details our SAT_MCDA_EA approach. Section 4 provides our overall set-up and benchmark for multi-objective feature selection in SPL. Section 5 reports the results of our evaluation in terms of execution time overhead and performance of SAT_MCDA_EA against SATIBEA. Finally, Section 6 concludes the paper.

2 Background

In this section, we detail two aspects that make up the background of our work.

2.1 Software Product Line Engineering

Software Product Line Engineering is the paradigm that attempts to manage software variations more systematically and provide tools that cover the domain engineering and the application engineering processes with their multiple phases/activities [18]. In SPL, all software artefacts (i.e., variations of the same feature) could be picked and put together to form a particular product as long as they are compatible.

Feature Models is a way to represent an SPL. FMs represent the set of all available features with their variations and incompatibilities (i.e., constraints). Figure 1 shows a toy FM example with ten inter-connected features. It shows, for example, that the final product requires a 'Screen'. It also shows that there exist three 'Screen' types (i.e., 'Basic', 'Colour' or 'High Resolution') and only one of them could be selected for the final product. To build a software product from the SPL, we need to select a subset of features $\mathcal{S} \subseteq \mathcal{F}$ such that constraints of the FM $\mathcal{F}$ are satisfied. Constraints of the FM can be modelled as a satisfiability (SAT) problem for instantiating Boolean variables to true or false (in our case, every variable represents a feature) in a way that satisfies all the constraints. A variable $f_i \in \{\text{true, false}\}$ is set to true if the feature $F_i \in \mathcal{F}$ is picked to be part of $\mathcal{S}$, and false otherwise.

An FM can be represented in a conjunctive normal form (CNF). Therefore, searching for a valid software product in the SPL is equivalent to searching

for a feasible solution to the SAT problem. For instance, the FM in Figure 1 describes the screen alternatives in its SAT model with these clauses: $(Basic \vee Colour \vee High\ resolution) \wedge (\neg Basic \vee \neg Colour) \wedge (\neg Basic \vee \neg High\ resolution) \wedge (\neg Colour \vee \neg High\ resolution)$.

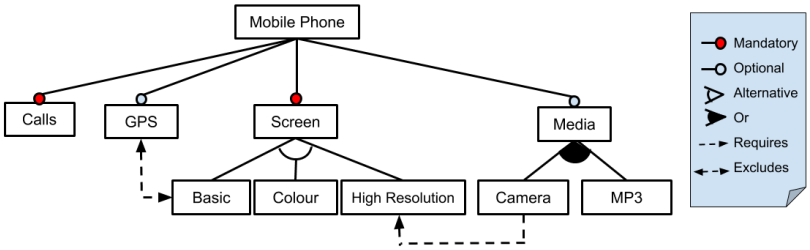


Figure 1: Example of a Feature Model

2.2 Multi-Objective Optimisation

Multi-Objective Optimisation (MOO) considers the optimisation of more than two objective functions at the same time. Software products can be seen from various perspectives (e.g., development cost, reliability, performance). Therefore, by considering each of the perspectives as independent objectives, feature selection in SPL is a suitable candidate for MOO [14].

As a meaningful sample case, we use a set of commonly used optimisation objectives in the literature [19–21]:

- *Correctness* – reduce the number of violated constraints.
- *Richness of features* – increase the number of picked features (have products with more functionality, minimisation of its negative value is considered).
- *Features used before* – reduce the number of picked features that were not used before.
- *Known defects* – reduce the number of known defects in picked features.
- *Cost* – reduce the cost of the picked features.

3 State-of-the-Art and Proposed Approach

In this section, we describe the state-of-the-art algorithm SATIBEA and our proposed approach.

3.1 SATIBEA

SATIBEA [3] is an extension to the Indicator-Based Evolutionary Algorithm (IBEA) which guides the optimisation through a quality indicator selection

process (in this case, the Hypervolume); a SAT solver has been introduced as a mutation operator to assist IBEA.

Note that there are multiple algorithms designed to address the multi-objective feature selection in SPL problem. Most of these algorithms perform in a similar fashion as SATIBEA (evolutionary algorithm + exact algorithm such as SMT [20] or MILP [21, 22]). In this work, we do not compare to them as we do not aim to design an algorithm that is better in terms of multi-objective metrics (even if we report the performance with respect to those metrics below). Instead, our goal is to showcase the fact that including preferences of the decision-makers in the evolutionary search process is worth considering when decision-makers have complex preferences as: (i) it only adds a marginal execution time overhead, and (ii) it finds solutions that are interesting with respect to decision-makers' preferences, but missed by particular IBEA algorithms (in our case SATIBEA).

3.2 Multi-Criteria Decision Analysis

Providing a set of non-dominated solutions, decision-makers explore them to find their preferred one. Given the large size of the non-dominated sets that are obtained after performing the multi-objective optimisation, decision-makers take advantage of MCDA techniques to select the ideal solution with respect to their preferences.

MCDA deals with decision-making constrained by multiple and often conflicting criteria (or objectives or goals). MCDA has been broadly divided into two categories [14]: (i) Outranking Methods: builds a preference relation, and (ii) Multiple Attribute Utility and Value Theory: the 'utility' of every action is scored based on its utility.

In this work, we select three commonly used MCDA techniques: two outranking methods (ELECTRE-IV [23] and PROMETHEE-II [24]) and one Multiple Attribute Utility and Value Theory method (MAUT [25]).

We propose in this paper to substitute the Indicator-Based selection operator in the original SATIBEA algorithm by one of the aforementioned MCDA techniques (i.e., ELECTRE-IV, PROMETHEE-II or MAUT) to create what we call SAT_MCDA_EA. Therefore, we are creating three distinct algorithms under the same umbrella of SAT_MCDA_EA: (i) SAT_ELECTRE-IV_EA, where we use ELECTRE-IV as the selection operator, (ii) SAT_PROMETHEE-II_EA, where we use PROMETHEE-II as the selection operator, and (iii) SAT_MAUT_EA, where we use MAUT as the selection operator.

4 System Set-up

This section presents the different elements that we have used in our experiments: the dataset, the multi-objective performance metrics, the parameters of the genetic algorithms (i.e., SATIBEA and SAT_MCDA_EA), the parameters we use for the MCDA techniques, and the hardware configuration.

4.1 System and Algorithms Set-up

We use the implementation of SATIBEA that is made available to us by its creators (implemented in Java) and implement our approach on top of it. We conduct our experiments on a machine with a 4 core CPU (our algorithms use a core at a time though) and 16 GB of RAM. We ran all our algorithms and determined the average results over 30 runs for each instance.

We use the same parameters for SATIBEA as those defined by its authors (e.g., population size: 300, crossover rate: 0.8, mutation rate of each feature selection: 0.001, and solver mutation rate: 0.02). We also use the same parameters as SATIBEA for our SAT_MCDA_EA approach. Furthermore, we define addition parameters for the MCDA techniques to simulate preferences of decision-makers. Note that the chosen preferences are only selected to showcase different capabilities of each MCDA method. Therefore, it will be worth performing a more robust analysis with different kinds of preferences and a full parameters sweeping for each of these MCDA methods in a future work.

- ELECTRE-IV: requires a parameter triplet (optimisation threshold, preference threshold, and indifference threshold) for every objective. We set these triplets to (5,6,5), (3,4,3), (0.1,0.3,0.1), (1,2,1) and (3,4,3) for Correctness, Richness of features, Feature used before, Known defects, and Cost.
- PROMETHEE-II: requires a parameter pair (weight and preference function) for each objective. We set equal weights for all objectives and set their preference functions to Level, Linear, Linear, Level, and Gaussian for Correctness, Richness of features, Feature used before, Known defects, and Cost.
- MAUT: only requires one parameter per objective (weight) that we set equally for all the objectives.

Based on the parameters that each of the MCDA techniques requires, we see that PROMETHEE-II is the most expressive between them as it enables decision-makers to design their own custom utility function for each objective and feed it to the MCDA.

4.2 Dataset

For our experiments, we use the five of the largest open source FMs we could find [20]. Table 1 shows the version and the size of each of the FMs that we consider in our experiments. The table also reports the number of features and the size of the SAT problem necessary to represent the FM in a conjunctive normal form (in terms of number of variables and number of clauses). Similarly to the SATIBEA paper [3], we set the execution time on the Linux Kernel to 1,200s. For the other datasets, we use smaller execution times based on the convergence time of SATIBEA [19, 26].

4.3 Multi-Objective Performance Metrics

To assess the performance of our algorithms we use 5 multi-objective performance metrics: 4 quality metrics (Hypervolume, Epsilon, Generation Distance, and Inverted Generation Distance) and 1 diversity metric (Spread).

Table 1: Versions and characteristics of the feature models used in our experiments.

Dataset	Version	#Features	#Variables	#Clauses	Time (s)
Linux kernel	2.6.28.6	5,701	6,888	343,944	1,200
eCos	20100825	1,244	1,244	3,146	50
Fiasco	2011081207	300	1,638	5,228	200
FreeBSD	8.0.0	1,396	1,396	62,183	200
μClinux	3.0	616	1,850	2,468	100

- Hypervolume (HV): computes the volume (measured in k dimensions of the problem's search space) that is dominated by the Pareto front (to maximise).
- Epsilon (ϵ): evaluates the smallest distance that is needed for every solution in Pareto front to dominate the Reference front (to minimise).
- Generation Distance (GD): evaluates the smallest distance needed for every solution in Pareto front to dominate the Reference front (to minimise).
- Inverted Generation Distance (IGD): evaluates average distance between every solution in Reference front and its closest solution in Pareto front (to minimise).
- Spread (S): computes the solutions' distribution to evaluate their extent spread in Pareto front (to maximise).

5 Evaluation

5.1 Execution Time Overhead

One of the major issues that kept designers of evolutionary algorithms away from using MCDA techniques within the search process is the excessive execution time that these techniques require. More researchers and practitioners favour less time-consuming indicator-based methods. This is even more true with problems that are only given a few seconds as a total optimisation time budget. In this section, we evaluate the overhead execution time that is introduced by the use of MCDA techniques. We compare the execution time of MCDA techniques to the execution time needed to evolve a full generation and also to the execution time of the default indicator-based method (in our case, the Hypervolume).

Table 2 shows the average execution time in millisecond over 30 iterations of the second generation of SATIBEA (the generation following the evolution of the randomly generated initial population) using the default indicator-based (Hypervolume). The table also shows the average execution time of each particular selection technique from Indicator-Based, to the three considered MCDA techniques (i.e., ELECTRE-IV, MAUT, and PROMETHEE-II).

We clearly see that the execution time of a full SATIBEA generation is very large in comparison to the execution time of the different selection operators (148 times larger on average than the largest selection time per instance). A single generation takes on average 531, 11, 84, 100, and 12 times larger execution times than the most time-consuming selection process (in this case, using

Table 2: Average execution time (ms) of the second generation of SATIBEA, indicator-based selection, and MCDA selection methods.

Dataset	Generation	Indicator-Based	ELECTRE-IV	MAUT	PROMETHEE-II
Linux Kernel	53,788	30.50	1.71	62.75	101.23
eCos	1,235	30.22	1.93	60.33	114.82
Fiasco	12,477	44.49	1.42	59.68	149.04
FreeBSD	12,742	29.57	1.56	71.28	127.30
uClinux	1,197	31.6	1.55	58.09	96.44

PROMETHEE-II) on the instances Linux Kernel, eCos, Fiasco, FreeBSD and uClinux respectively. This is a clear indication that using any of the studied MCDA techniques is less likely to add a significant execution time overhead. The execution time of the section process is particularly insignificant when dealing with the large instances (Linux Kernel, Fiasco and FreeBSD).

We see that with the exception of ELECTRE-IV, MCDA techniques (i.e., MAUT and PROMETHEE-II) necessitate a larger execution time than the default Indicator-Based selection. This is one of the main reasons why the simplistic weighted-sum is the de-facto go to in absence of a pure multi-objective objective optimisation (keeping objectives separate with no aggregation). However, we notice in our usecase that the order by which the execution time of these MCDA techniques exceed the Indicator-Based selection is rather small ($\sim$0.9 and $\sim$2.5 more execution time on average for MAUT and PROMETHEE-II respectively).

Therefore, we could claim that from an execution time perspective and in the context of multi-objective feature selection in large software product lines such as the ones studied in our paper, decision-makers should no longer be reluctant to provide their preferences in advance to be embedded in the multi-objective optimisation process.

5.2 Multi-Objective Performance Metrics

Knowing that using MCDA techniques in the multi-objective optimisation process does not add a significant execution time overhead is good, but obtaining improved results is better –despite not being the most important in our case as our goal is to find more solutions that match decision-makers' preferences. Therefore, we would like to evaluate the impact of our approach in terms of performance and quantify it using the different multi-objective metrics seen in Section 4.

Table 3 shows the average performances achieved by SATIBEA and SAT_-MCDA_EA techniques (i.e., SAT_ELECTRE-IV_EA, SAT_MAUT_EA, SAT_-PROMETHEE-II_EA) with respect to the quality metrics HV, IGD, GD, Epsilon and Spread. We put in bold the best achieved performances per instance and per metric. We also put (*) when results are not statistically significant between SATIBEA and the best performing SAT_MCDA_EA technique (p-value ¡ 0.05 when evaluated using the non-parametric two-tailed Mann-Whitney U test).

Table 3 clearly shows that SATIBEA achieves the best performances on the metrics HV and IGD on all instances. SATIBEA also achieves the best performances on Epsilon in 4 out of 5 instances on average. This is a clear indication that SATIBEA maintains its supremacy with regards to very important multi-objective performance metrics. This is quite understandable as SATIBEA's aim by design is to cover most of the search space, which yields better multi-objective quality metrics performances. However, SAT_MCDA_EA algorithms target solutions that better match the predefined preferences of the decision-makers and leave large parts of the search space unprobed, which yields low multi-objective quality metrics performances.

Table 3 also shows that SATIBEA does not always achieve the best results with respect to the Spread metric. SAT_ELECTRE-IV_EA achieves the best performance on Spread on 3 out of 5 instances on average. Although, Spread is a secondary metrics and should not be interpreted alone without the other quality metrics. Looking at SAT_ELECTRE-IV_EA's performance in terms of HV, we see that it is poor, which reduces the importance of its Spread performance.

Table 3 also shows that SATIBEA is not achieving the best GD on any instance (achieved by SAT_PROMETHEE-II_EA). This is an indication that most of the solutions that are found by SAT_PROMETHEE-II_EA are non-dominated by the solutions found by the other algorithms. However, given that the performance of SAT_PROMETHEE-II_EA in terms of HV is poor, we can deduce that its solutions are not diverse enough. While this might seem negative, we believe that this is a good characteristic. Decision-makers would rather be provided with several non-dominated solutions that are similar and better match their preferences, rather than a set of non-dominated solutions covering a larger space, but match their preferences less. Furthermore, SAT_MAUT_EA also achieves a better performance than SATIBEA in terms of GD on 3 out of 5 instances on average.

5.3 SAT_MCDA_EA's Strictly Non-Dominated Solutions

With SAT_PROMETHEE-II_EA and SAT_MAUT_EA achieving good GD performances, we would like to measure the ratio of non-dominated solutions found by SAT_MCDA_EA algorithms, but missed by SATIBEA. We gather all non-dominated solutions found over all iterations by each algorithm and perform a pairwise non-dominance comparison. Table 4 shows the ratio (in percentage) of solutions found by each SAT_MCDA_EA that are strictly non-dominated (neither equal nor dominated) by any solution found by SATIBEA.

Table 4 confirms our assumption that many solutions found by SAT_MAUT_EA and SAT_PROMETHEE-II_EA are strictly non-dominated by those found by SATIBEA. We see that SAT_PROMETHEE-II_EA finds the largest number of solutions non-dominated by those found by SATIBEA (∼83% non-dominated solutions on average, and 94% on Fiasco). Therefore, if decision-makers have a prior knowledge of what makes a good software, they are better off using PROMETHEE-II as a selection operator. While this will not yield optimal multi-objective metrics, it will yield more solutions matching their preferences.

Table 3: Comparison of the average performances achieved by SATIBEA and the various SAT_MCDA_EA algorithms.

Dataset	Metric	SATIBEA	SAT_ELECTRE-IV_EA	SAT_MAUT_EA	SAT_PROMETHEE-II_EA
Linux Kernel	HV	**0.136**	0.124	0.123	0.134
	IGD	**0.010**	0.016	0.016	0.012
	GD	0.030	0.130	0.012	**0.007**
	ϵ	**1982**	2047	2051	1991
	S	1.16	**1.24**	1.21	1.19
eCos	HV	**0.252**	0.206	0.188	0.085
	IGD	**0.0071**	0.0072	0.008	0.016
	GD	0.0722	3.8714	0.0935	**0.0031**
	ϵ	**147**	260	217	149
	S	**1.51***	1.30	1.33	1.55
Fiasco	HV	**0.195**	0.133	0.132	0.124
	IGD	**0.009**	0.022	0.024	0.018
	GD	0.065	0.237	0.076	**0.008**
	ϵ	277	917	950	**171**
	S	**1.58**	1.14	1.16	1.27
FreeBSD	HV	**0.24**	0.18	0.18	0.08
	IGD	**0.006**	0.011	0.012	0.018
	GD	0.091	0.156	0.066	**0.004**
	ϵ	**133**	303	308	498
	S	1.21	**1.23***	1.20	1.21
uClinux	HV	**0.893**	0.89	0.891	0.805
	IGD	**0.054**	0.055	0.056	0.060
	GD	0.043	0.016	0.015	**0.012**
	ϵ	**598***	611	604	1199
	S	1.067	**1.229**	1.198	1.003

Table 4: Ratio (in per cent) of strictly non-dominated solutions found over the 30 iterations by SATIBEA using one of the MCDA methods in comparison with the solutions found by SATIBEA when using the default Indicator-Based method.

Dataset	SAT_ELECTRE-IV_EA vs SATIBEA	SAT_MAUT_EA vs SATIBEA	SAT_PROMETHEE-II_EA vs SATIBEA
Linux Kernel	40	41	66
eCos	33	42	90
Fiasco	27	59	94
FreeBSD	26	48	92
uClinux	5	34	73

6 Conclusion and future work

In this paper, we proposed using MCDA techniques directly within the multi-objective search process by employing them as the selection operator. We have evaluated their impact both in terms of induced execution time overhead and in terms of quality of the obtained solutions. We have seen that using the MCDA techniques introduces a non-significant overhead execution time with respect to the execution time of the other operators that make up the evolution. However, we have also seen that using the MCDA techniques within the search process impacts negatively the performance of the algorithm with respect to various multi-objective performance metrics with the exception of GD. We have confirmed that the SAT_MCDA_EA algorithms perform particularly well with respect to GD as they find a large number of solutions that match their preferences but that are not dominated by the solutions found by SATIBEA. The insight obtained from this study encourages us to deepen the investigation of combining MCDA techniques with the multi-objective feature selection in SPL.

Acknowledgement: This work was supported, in part, by Science Foundation Ireland grants No. 13/RC/2094_P2 (Lero) and 13/RC/2106_P2 (ADAPT).

References

1. Ramirez, A., Romero, J.R., Ventura, S.: A survey of many-objective optimisation in search-based software engineering. Journal of Systems and Software **149** (2019) 382–395
2. Metzger, A., Pohl, K.: Software product line engineering and variability management: achievements and challenges. In: FSE, ACM (2014) 70–84
3. Henard, C., Papadakis, M., Harman, M., Le Traon, Y.: Combining multi-objective search and constraint solving for configuring large software product lines. In: ICSE. (2015) 517–528
4. Yadav, H., Chhikara, R., Kumari, A.C.: A novel hybrid approach for feature selection in software product lines. Multimedia Tools and Applications **80**(4) (2021) 4919–4942
5. Lynch, D., Saber, T., Kucera, S., Claussen, H., O'Neill, M.: Evolutionary learning of link allocation algorithms for 5g heterogeneous wireless communications networks. In: GECCO. (2019) 1258–1265
6. Saber, T., Fagan, D., Lynch, D., Kucera, S., Claussen, H., O'Neill, M.: A hierarchical approach to grammar-guided genetic programming: the case of scheduling in heterogeneous networks. In: TPNC. (2018) 225–237
7. Saber, T., Fagan, D., Lynch, D., Kucera, S., Claussen, H., O'Neill, M.: A multi-level grammar approach to grammar-guided genetic programming: the case of scheduling in heterogeneous networks. Genetic Programming and Evolvable Machines **20**(2) (2019) 245–283
8. Saber, T., Wang, S.: Evolving better rerouting surrogate travel costs with grammar-guided genetic programming. In: IEEE CEC. (2020) 1–8
9. Bendechache, M., Svorobej, S., Endo, P.T., Mario, M.N., Ares, M.E., Byrne, J., Lynn, T.: Modelling and simulation of elasticsearch using cloudsim. In: DS-RT. (2019) 1–8

10. Saber, T., Gandibleux, X., O'Neill, M., Murphy, L., Ventresque, A.: A comparative study of multi-objective machine reassignment algorithms for data centres. Journal of Heuristics **26**(1) (2020) 119–150
11. Saber, T., Delavernhe, F., Papadakis, M., O'Neill, M., Ventresque, A.: A hybrid algorithm for multi-objective test case selection. In: CEC. (2018) 1–8
12. Saber, T., Ventresque, A., Brandic, I., Thorburn, J., Murphy, L.: Towards a multi-objective vm reassignment for large decentralised data centres. In: UCC, IEEE (2015) 65–74
13. Saber, T., Ventresque, A., Gandibleux, X., Murphy, L.: Genepi: A multi-objective machine reassignment algorithm for data centres. In: International workshop on hybrid metaheuristics, Springer (2014) 115–129
14. Mjeda, A., Wasala, A., Botterweck, G.: Decision spaces in product lines, decision analysis, and design exploration: an interdisciplinary exploratory study. In: VaMoS, ACM (2017) 68–75
15. Mohammed, A., Harris, I., Soroka, A., Nujoom, R.: A hybrid mcdm-fuzzy multi-objective programming approach for a g-resilient supply chain network design. Computers & Industrial Engineering **127** (2019) 297–312
16. Kapsoulis, D., Tsiakas, K., Trompoukis, X., Asouti, V., Giannakoglou, K.: Evolutionary multi-objective optimization assisted by metamodels, kernel pca and multi-criteria decision making techniques with applications in aerodynamics. Applied Soft Computing **64** (2018) 1–13
17. Jafarian-Namin, S., Kaviani, M.A., Ghasemi, E.: An integrated moea and mcdm for multi-objective optimization (case study: control chart design). In: IEOM. (2016)
18. Horcas, J.M., Pinto, M., Fuentes, L.: Software product line engineering: a practical experience. In: SPLC. (2019) 164–176
19. Saber, T., Brevet, D., Botterweck, G., Ventresque, A.: Is seeding a good strategy in multi-objective feature selection when feature models evolve? Information and Software Technology **95** (2018) 266–280
20. Guo, J., Liang, J.H., Shi, K., Yang, D., Zhang, J., Czarnecki, K., Ganesh, V., Yu, H.: Smtibea: a hybrid multi-objective optimization algorithm for configuring large constrained software product lines. Software & Systems Modeling **18**(2) (2019) 1447–1466
21. Saber, T., Brevet, D., Botterweck, G., Ventresque, A.: Reparation in evolutionary algorithms for multi-objective feature selection in large software product lines. SN Computer Science **2**(3) (2021) 1–14
22. Saber, T., Brevet, D., Botterweck, G., Ventresque, A.: Milpibea: Algorithm for multi-objective features selection in (evolving) software product lines. In: EvoCOP. (2020) 164–179
23. Govindan, K., Jepsen, M.B.: Electre: A comprehensive literature review on methodologies and applications. European Journal of Operational Research **250**(1) (2016) 1–29
24. Brans, J.P., De Smet, Y.: Promethee methods. In: Multiple criteria decision analysis. Springer (2016) 187–219
25. Allah Bukhsh, Z., Stipanovic, I., Klanker, G., O'Connor, A., Doree, A.G.: Network level bridges maintenance planning using multi-attribute utility theory. Structure and infrastructure engineering **15**(7) (2019) 872–885
26. Brevet, D., Takfarinas, S., Goetz, B., Anthony, V.: Preliminary study of multi-objective features selection for evolving software product lines. SSBSE (2016)

An Exploratory Analysis on a Disinformation Dataset

Matheus Marinho[1], Carmelo J. A. Bastos-Filho[1], and Anthony Lins[2]

[1] Universidade de Pernambuco, Recife/PE, Brazi
{mblm,carmelofilho}@ecomp.poli.br
[2] Universidade Catolica de Pernambuco, Recife/PE, Brazi
anthony.lins@unicap.br

Abstract. Understanding the phenomenon of disinformation and its spread through the internet has been an increasingly challenging task, but it is necessary since the effects of this type of content have their impacts in the most diverse areas and generate more and more impacts within society. Automated fact-checking systems have been proposed by applying supervised machine learning techniques to assist in filtering fake news. However, two challenges are still present, the first related to understanding disinformation in its subgroups. The second challenge is related to the availability of datasets containing news classified between true and false. This article proposes an exploratory analysis through unsupervised algorithms and the t-SNE technique to visualize data with high dimensionality, identify the subgroups present in the disinformation, and the identification of possible outsiders between the classes. We also propose a new Corpus in Portuguese containing 19446 news, classified as true and false, and 15 linguistic features extracted from this dataset. Finally, we propose to use two classification models using the Random Forest techniques, with and without intruders. In the end, the model without intruder achieved superior performance, reaching an accuracy of 97.33%

Keywords: Disinformation · Unsupervised Learning · Clustering

1 Introduction

Disinformation can change political opinions, influencing the results of elections, contributing to the spread of diseases, causing problems in public health, and even causing deaths with hate campaigns and generating extremist groups [13]. The growth of "Fake News" was stimulated by the change in society, which started to live more and more in a network [3], organized around interconnected universes of digital communication driven by the internet. The network society has generated a reduction in complexity and agility in the dispersion of content, thus allowing news, whether true or false, written by any individual, to reach an audience of hundreds of millions of readers [1].

One of the solutions that emerged in this fight against the disinformation ecosystem is fact-checking, a service proposed by journalists that seek to identify

evidence, understand the context of the information and what can be inferred from that evidence through a fact verification process [14]. However, due to the complexity linked to this solution, the conclusion of the analyzed fact's veracity can take days [7]. This complexity creates a challenge, as human fact-checkers are unable to keep up with the amount of disinformation and the speed with which they spread, and in this way, the opportunity arises for the creation of automated fact-checking systems (AFC).

Typically, AFCs are developed to assist in the classification activity, where the news is labeled as true or false. In this type of activity, supervised machine learning techniques are used in most cases, with a model previously trained from a set of data containing the types of news evaluated. However, for the successful construction of these models used in the AFC, two things need to be taken into account: (i) identify the main characteristics that need to be taken into account when evaluating the news and the strengths of these features in the process [12] and identify the relationships between the variables input and output, as well as the relationships between features and; (ii) the use of a diverse dataset since the precision of the model generated by supervised machine learning is directly related to the quality of the data contained in the classifier training dataset.

This paper aims to make an exploratory analysis of Fake.br Corpus [9], a dataset related to disinformation, through the generation of clusters, to understand: how the features are distributed in each cluster; the main characteristics of these groups and; the relations of the features with the type of news (true/fake). In this paper, we also present a new set of disinformation data with news in Portuguese, increase the existing corpus of Fake.br, and perform analysis of possible noisy data, which may hinder the process of generating the classification model. In the end, we present the characteristics of these noisy data and compare the results with and without noisy data.

Although the data already have a classification, the present paper seeks to identify possible intrusive data existing in the data sets and understand the characteristics existing in the subgroups related to disinformation and true news.

The rest of the paper is organized as follows. Section 2 reviews the related work. Section 3 offers the background theory about Hierarchical Clustering and t-SNE. Section 4 describes the methodology. Section 5 shows the results. Finally, Section 6 presents conclusions and future work.

2 RELATED WORK

In [11], the authors analyzed news features: n-grams, punctuation, psycholinguistics, legibility, and syntax. For each class, they proposed different classifiers using a linear support vector machine (SVM). The authors observed that depending on the dataset analyzed, a different category has the best performance. In this case, the legibility and punctuation classes had a better performance. The authors also considered only semantic characteristics and can conclude that actual news tends to have more function words, negations, and express relativity. On the other hand, the language used to report fake content uses more social and

positive words and focuses on the present and future actions. Also, fake news authors use more adverbs, verbs, and punctuation characters than legitimate news authors.

Reis et al. [12] listed 11 categories for the evaluation of news and proposed 294,292 models, each with 20 features randomly selected from the existing classes. The work concluded that features related to the social media data class, such as the number of shares or reactions related to the news, the credibility of the domain, and characteristics that indicate political bias, are the most present in the models. In these cases, the models obtained better performance in the separation of trustworthy news and disinformation.

For the application of machine learning techniques, a fundamental requirement is a dataset capable of representing the problem in question. However, on the theme of Disinformation, it was only in 2018 that the authors [9] were the first to build a Corpus containing true and false news in Portuguese. Fake.br Corpus, contains 7200 news items, divided equally between true news and disinformation, collected between the years 2016 and 2018. Among the themes present in the data set, are politics and economics. The authors also made available a set containing 25 characteristics, 21 of which related to linguistic issues.

3 Background Theory

In this section, we present the relevant concepts to allow the reader to understand the proposal properly.

3.1 Hierarchical Clustering

The process of unsupervised learning in the generation of clusters consists of dividing the data into groups so that the data present in the same cluster are as similar as possible, and the difference between groups is as significant as possible [6]. Clustering techniques can be divided into two categories: hierarchical and non-hierarchical. Unsupervised techniques are distinguished, as they do not constitute a specific number of groupings; however, they generate groups through an increasing sequence of divisions or continuous group connections.

Hierarchical methods are composed of two classes of algorithms for generating clusters: agglomerative clustering and divisive clustering [5].

Agglomerative clustering is a conventional clustering method that can produce an informative hierarchical structure of clusters. The algorithm starts with a large number of small initial clusters. The agglomerative cluster iteratively merges a pair of clusters with the highest affinity under a given criterion until some stop condition is reached [4]. There are many conventional methods for calculating the affinity between a pair of clusters, such as single linkage, full linkage, medium linkage, and ward [2]. In this work, we use the ward link to reduce the variance between the merged clusters. We used agglomerative clustering in this work because we aim to analyze the purity of the formed clusters in different aggregation levels.

3.2 t-SNE

T-Distributed Stochastic Neighbor Embedding is a method for exploring high-dimensional data, proposed by Maaten and Hinton [8]. Visual exploration is an essential component of data analysis, as it allows the development of intuitions and hypotheses for the processes that generated the data. Stochastic neighbor embedding techniques calculate an N x N similarity matrix both in the original data space and in the small dimension embedding space so that similarities form a probability distribution over pairs of objects, these probabilities are usually provided by a kernel Normalized Gaussian or Student-t calculated from the input data [15].

T-SNE minimizes the divergence of the distribution that measures similarities between pairs of the input objects and a distribution that measures similarities between pairs of the corresponding low-dimension points in the embedding, through a function that calculates the distance between a pair of objects, typically uses the Euclidean distance. Another critical parameter is perplexity, which is used to compare probability models, the performance of the SNE is quite robust to changes in perplexity, and the typical values are between 5 and 50.

4 Methodology

In this section, we present the Fake.br Corpus, how the data preprocessing methods and the use of agglomerative clustering. After that, we discuss the dataset proposed in this paper, how it was built, its final configuration, and the existing features. Finally, the process of applying the t-SNE technique to identify regions of conflict between true and false news will be discussed, as well as how we apply the supervised technique to verify if there was a better convergence of the classifier after removing this news and applying the statistic tests in the validation process between the existing classifiers.

4.1 Fake.br Corpus

This paper aims to understand the construction of disinformation in Brazil. In this context, we found that there is a wide variety of datasets in the English language. However, it is not easy to obtain this type of information in the Portuguese language. The Fake.Br dataset [9] is the first with this purpose. The language of the data set is Portuguese and contains 7200 news collected from websites. The dataset is divided equally between fake and true news and grouped into six categories: politics; tv and celebrities; society and daily news; science and technology; economy and; religion.

The authors also made available a set with 25 features regarding each news. Among them, we can cite the number of words, the number of verbs, and the average sentence size. In the data set, the features' types are distributed as follows: 21 features are numeric with linguistic information, and four are categorical, which are related to the date of publication, author, news link, and category.

4.2 Pre-processing

After analyzing the distribution of features, we discarded the categorical features. Since we observed that the information in them is irrelevant and does not influence the model's response, for example, taking the author as a parameter could bias the proposed model. Thus, instead of having clusters that group true and false news, groups divided the authors or the categories of each news item. Because of this, we used the 21 numerical features related to linguistic characteristics.

Finally, we normalized the data using a Min-Max algorithm that outputs a value in the range between 0 and 1. It allows an easy way to compare measured values using different scales or measures.

4.3 Agglomerative Clustering

The clustering algorithm was applied using the scikit-learn toolkit [10]. We used the following configurations: the amount of cluster equal to 16; the affinity is the ward, and; the metric is the Euclidean distance. Then, we calculate each cluster's purity according to the majority of news belonging to a specific class, which, in turn, labels that cluster. For example, suppose that we have a cluster containing 100 news, 80 of which are true, and 20 are fake. Then, we label the cluster as true with a purity of 80%.

We propose an evaluation of the purity of each cluster to understand: (i) what are the characteristics of the purest, that is, that it contains only news of the same classification; and (ii) identify which subgroups exist within each class of classification, in order to understand the different behaviors that may exist within disinformation and true news.

4.4 Brazilian Disinformation Corpus

To increase diversity and update Fake.br Corpus, this article proposed an update with 12246 more news, collected in the following ways: (i) for a part of the true news, three web crawlers were proposed, to automate the extraction of content from the Globo, Sistema Jornal do Commercio and Diário de Pernambuco. Among the types of news, those related to politics and health were extracted, as they are the most relevant topics at the moment due to the state elections in November 2020 and the amount of disinformation generated in the health area related mainly to treatments and ways of propagation of COVID-19. These three communication vehicles were chosen because they are companies that do not publish disinformation. (ii) For collecting a part of the disinformation news, a web crawler was developed to extract content from Boatos.org, a service managed by journalists to compile the fake content that circulates on the internet. Finally, (iii) texts were extracted through Monitor do Whatsapp, a project that provides texts extracted from public groups within the communication platform, with themes related to politics. The classification of these texts was carried out by a team of journalists with experience in the fact verification process, and only

news related to disinformation was used in this work, as it was the class with the least amount of examples. At the end of this collection process, the new data set configuration can be seen in Table 1, containing 12127 true news and 7319 disinformation.

Table 1. Brazilian Disinformation Corpus configuration

Source	Quantity
Fake.br Corpus - Fake	3600
Fake.br Corpus - True	3600
Globo	935
Sistema Jornal do Commercio	5661
Diário de Pernambuco	1931
Monitor do Whatsapp	887
Boatos.org	2832

Regarding the extracted characteristics, we initially attempted to replicate those existing in Fake.br Corpus, however of the 21 numerical characteristics, 6 were not capable of replication because they did not beat their value with those existing in the original data set, leaving the new set of data with 15 characteristics, they were: number of tokens; words without punctuation; the number of types; the number of uppercase; the number of verbs; the number of nouns; the number of adjectives; the number of adverbs; the number of pronouns; pausality; the number of characters; average sentence length; average word length; emotiveness; and diversity.

4.5 Outsiders Analysis

We proposed two analyzes using the technique for data exploration and visualization of high-dimensional data, t-SNE, at the Brazilian Disinformation Corpus. In the first evaluation, we used the Corpus classifications, making it possible to identify conflicting regions, with intruders both from the real news inside the fake news region, and the opposite. For the second evaluation, a new classification was proposed using an unsupervised technique, called k-means. This analysis aims to visualize the subgroups existing within each class and understand which groups are more reliable to their class: purer, and which groups contain that intrusive data. For this, different values were tested in the cluster quantity, seeking to find the quantity that best represented the problem, which in this case, is a high purity value in all groups, and this value in the Brazilian Disinformation Corpus was nine groups.

After identifying the intruders and an analysis of their characteristics, two supervised classifiers were proposed: one with all Corpus data and; another without the intruding elements. We aim to validate whether the removal of such news would increase the classification accuracy, as the models would not be trained with data that may be noisy. For this last stage, classifiers were generated using

the Random Forest technique, as we have presented the latest work using Fake.br Corpus, this technique performed well in the disinformation classification process. For this evaluation, we used precision, recall, and F1. Besides, we used the t-student test to verify if there was a statistically significant improvement among the models.

5 Results

We carried out the initial analysis using the unsupervised technique at Fake.br Corpus, to identify and understand the subgroups present in the truthful news and the disinformation. We emphasize that it is not our intention to carry out a classification process, as the data are already previously classified. Analyzing the dendrogram (Figure 1) generated by the hierarchical clustering algorithm, it is possible to observe that this technique was able to separate the two groups contained in the dataset (fake and true news). In the first level (see Table 2), we obtained a cluster with 93.82% purity with its data mostly belonging to the class of true news and a second cluster with 93.70% purity, this being composed mostly by false news. Thus, in general analysis, only 449 news items were grouped erroneously, and this value corresponds to only 6.24% of the total data. In contrast, 93.76% of the news items were correctly separated between the generated clusters.

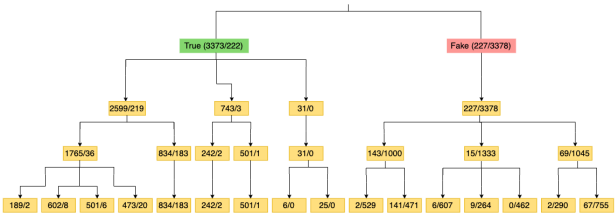


Fig. 1. Dendrogram

Thirty clusters were formed, of which 18 clusters have the most accurate news, and 12 clusters contain mostly fake news. This information reveals more considerable variability in the characteristics belonging to the data of this class within the true news. However, the true clusters ended up being repeated in 4 cases; that is, the lower level cluster is the same as the upper-level cluster. This repetition may be evidence that these clusters contain strong characteristics and therefore did not go through changes in the merges performed by the agglomerative technique.

In general, of the 30 clusters proposed (Figure 1, 26 are more than 90% pure, of these 13 clusters are more than 99%, and five are 100% pure. Analyzing the perspective of classes: from the total of 26 clusters, 15 have the class of the true ones, and 11 are fake clusters; of the 13 clusters, nine are true, and only four are

fake and; of the five clusters with 100% purity, four are true, and only one is fake. It reinforces the initial conclusion that the true clusters have a greater variety in the characteristics' values, so it has a larger number of clusters. However, this information is different, consequently allowing a good separation between the true news.

Analyzing clusters with 100% purity, these are the characteristics present in each cluster: (i) in the only fake cluster, the feature that stands out from the others is the number of links, this value is higher than in all others groups, that is, the 462 news items in this group have a large number of links in the text; (ii) the first of the four true clusters, consists of only six news items and these have a high value in 13 of the 21 characteristics analyzed; (iii) the second cluster has 25 true news items, these have the values of 14 characteristics above the average, with 13 characteristics with lower values than the previous cluster and with the characteristic of the quantity of first and second personal pronouns with the highest value between clusters and; (iv) the other two true clusters with 100% purity is generated from the (ii) and (iii) cluster, that is, they have the same characteristics previously presented.

Table 2. Purity from k = 2

Cluster	Class	True News	Fake News	Purity (%)
0	True	3373	222	93.82
1	Fake	227	3378	93.70

For the analysis of subgroups, we chose to analyze the dendrogram sheet containing 16 groups, as it contains the largest number of divisions, thus generating a more detailed (specific) level among the news within each macro class (fake and true). In the disinformation group, it is possible to observe that there are not many differences between the groups. The group with the most significant difference is the one that has a high value in the number of links, and the rest has a small variation in the characteristics of average word size, average sentence size, and diversity. This was already expected, as stated earlier, were more true than false groups were generated.

After the analysis the subgroups existing in the set of true news, we observed that they are initially divided by the number of tokens, the number of words without punctuation and the number of types. Another thing that distinguishes them is the number of pronouns and the number of characters. Finally, the higher the values in these characteristics mentioned above, the greater their purity related to the true groups.

The second analysis was related to applying the t-SNE technique in the dataset generated by this paper, the Brazilian Disinformation Corpus. In Figure 2, in blue, we have the points related to true news and, in red, disinformation. Note that there is a separation zone between the groups; however, intruders are in both classes. To understand, the characteristics of these news items that do not belong to their zone, a cut was proposed in the reduced plan, where the

Fig. 2. t-SNE Brazilian Disinformation Corpus

Table 3. Outsiders characteristics

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	class
0,46	0,49	0,49	0,25	0,46	0,42	0,38	0,26	0,33	0,27	0,46	0,35	0,28	0,20	0,60	true
0,57	0,60	0,60	0,33	0,58	0,53	0,51	0,43	0,51	0,29	0,57	0,38	0,29	0,24	0,51	fake

region of each class of news was determined, and in this way, the news that did not belong to their zone was classified as outsiders. The characteristics of these news items can be seen in Table 3, where the class indicates which one it belongs to and therefore, if the class is true, it means that it is the set of real news that was in the region of fake news and vice versa and columns 1 until 15 represent the features.

Analyzing the table of intruders, it is possible to understand the reason why these news invaded regions that did not belong, because in the case of true intruders, they are news that has a high value of diversity, and as seen earlier, this is a characteristic of related news disinformation. On the other hand, the disinformation classified as an intruder presented high values in most of the features, which was mapped as a characteristic of the real news, for example, the features of number of tokens, number of types, and numbers of nouns. In total, 670 intruders were identified, grouped into 429 false and 241 true.

The third analysis was carried out by applying an unsupervised k-means technique to find the subgroups existing in the Brazilian Disinformation Dataset. In this case, the number of groups that best represented the set was 9, and from that identification, we applied the technique of dimensionality reduction t-SNE again. The result can be seen in Figure 3. Groups 3, 5, and 8, respectively, dark green, light blue, and pink, are related to disinformation, and the rest related to true news. Again, out of 9 proposed groups, the class with the largest number of groups is true.

From the image, it is possible to notice that three groups are well separated from the others, they are: 2; 3 and; 7, and 3 groups have their data in a conflict

Fig. 3. t-SNE Brazilian Disinformation Corpus with k-means and k = 9

zone, the groups: 1; 5 and; 6. Making a more detailed analysis in groups 3 and 7, which are the groups of the right and left extremities, and they have the following characteristics: group 3, which is true news, has a high value in the first three features (number tokens, word number without punctuation and number types); and group 7, in turn, related to disinformation has a high value in the diversity characteristic and a low value in the others.

Table 4. Supervised Results Brazilian Disinformation Dataset

Dataset	Class	Precision	Recall	F1	Accuracy
Raw	Fake	93.72	92.72	93.07	94.85
	True	95.76	96.14	95.90	
Outsiders	Fake	96.42	96.37	96.35	97.33
	True	97.93	97.89	97.89	

The last analysis was done by applying the supervised machine learning technique to assess the impact of generating a classifier using the intrusive data. For this purpose, the Random Forest algorithm was used in the Brazilian Disinformation Dataset, and the results can be seen in Table 4. In all metrics, the removal of intruders increased the accuracy from 94.85% to 97.33%. In a particular way, in each class, intruders' absence contributed to an increase in precision and recall metrics. Analyzing the F1 measure of the true class, the best result reached 97.89%, and the disinformation class obtained 96.35%.

6 Conclusion

Society is increasingly connected through the internet, and news has been spreading in a massive and fast way. The absence of filters has allowed an increase in

the circulation of fake news. Automated fact-checking systems have gained global attention to combat disinformation. It can be implemented by combining computational intelligence techniques and journalistic concepts. Thus, it is possible to create models capable of identifying probable false news and alerting the network readers.

This article had the following objectives. The first is constructing a new Corpus, entitled Brazilian Disinformation Corpus, containing 19446 news items and 15 language features. This new data set allowed for mapping more news related to disinformation. The second objective is to apply unsupervised techniques to identify existing subgroups in the categories of true and false news, reinforcing once again that we did not intend to carry out a classification activity but rather to have a deeper understanding of the Corpus. From the obtained results, we observed that there are more subgroups related to true news than related to disinformation, and it can also be concluded that within the pure subgroups, the features that stood out the most were a high value in the number of links in fake class and in the true news a high value in 13 out of 21 features proposed in [9].

The third objective is the identification of the existing subgroups in the Brazilian Disinformation Dataset. For this, we applied the k-means technique 9 clusters. Despite the difference in the datasets' size, the number of true clusters was greater than that of false ones and what differentiated the subgroups was also the different values in the features of the number of tokens, word number without punctuation, and number types. The true news had a high value, and the fake news groups had a low value, and in the diversity feature, this behavior was reversed. The fourth assessment was related to identifying intrusive news through a technique of visualization of data of high dimensionality. We identified the regions that characterized the real news and the fake news, and finally, we extracted the intruder news. In the end, 670 news items were classified as intruders, and their characteristics show behavior that diverged from the pattern found in each macro group (true and false). The true intruders had a high diversity value and fake content a high value of token quantity features and number of types.

Lastly, two classification models were proposed, generated from the Random Forest algorithm, for the disinformation and true news classification process. The first model contained all the news from the Brazilian Disinformation Corpus and the second model was generated without the outsiders elements identified through the analysis of the regions generated by the t-SNE technique. In the end, the model without the existence of outsiders obtained statistically better performance. Thus, we can conclude that removing the outsiders improved the classifier efficiency, reaching an accuracy of 97.33% and F1 measure of 96.35% for fake news and 97.89% for true news.

We plan to analyze entropy, purity, and statistical tests to discard the features for this work highlighted as irrelevant in the clustering process. We believe we can obtain even more concise results in the formation of supervised models. We will also perform a more detailed analysis of the intruder data to understand

whether or not there is a possible miss classification of this data. Finally, all of this will contribute to constructing an automated fact verification system using computational intelligence techniques.

References

1. Allcott, H., Gentzkow, M.: Social media and fake news in the 2016 election. Journal of economic perspectives **31**(2), 211–36 (2017)
2. Barreto, S., Ferreira, C., Paixao, J., Santos, B.S.: Using clustering analysis in a capacitated location-routing problem. European Journal of Operational Research **179**(3), 968–977 (2007)
3. Castells, M.: A Galáxia Internet: reflexões sobre a Internet, negócios e a sociedade. Zahar (2003)
4. Cimiano, P., Hotho, A., Staab, S.: Comparing conceptual, divise and agglomerative clustering for learning taxonomies from text. In: Proceedings of the 16th European Conference on Artificial Intelligence, ECAI'2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004 (2004)
5. Fahad, A., Alshatri, N., Tari, Z., Alamri, A., Khalil, I., Zomaya, A.Y., Foufou, S., Bouras, A.: A survey of clustering algorithms for big data: Taxonomy and empirical analysis. IEEE transactions on emerging topics in computing **2**(3), 267–279 (2014)
6. Frigui, H., Krishnapuram, R.: Clustering by competitive agglomeration. Pattern recognition **30**(7), 1109–1119 (1997)
7. Hassan, N., Adair, B., Hamilton, J.T., Li, C., Tremayne, M., Yang, J., Yu, C.: The quest to automate fact-checking. In: Proceedings of the 2015 Computation+ Journalism Symposium (2015)
8. Maaten, L.v.d., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**(Nov), 2579–2605 (2008)
9. Monteiro, R.A., Santos, R.L., Pardo, T.A., de Almeida, T.A., Ruiz, E.E., Vale, O.A.: Contributions to the study of fake news in portuguese: New corpus and automatic detection results. In: International Conference on Computational Processing of the Portuguese Language. pp. 324–334. Springer (2018)
10. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
11. Pérez-Rosas, V., Kleinberg, B., Lefevre, A., Mihalcea, R.: Automatic detection of fake news. arXiv preprint arXiv:1708.07104 (2017)
12. Reis, J.C., Correia, A., Murai, F., Veloso, A., Benevenuto, F., Cambria, E.: Supervised learning for fake news detection. IEEE Intelligent Systems **34**(2), 76–81 (2019)
13. Silva, R.M., Santos, R.L., Almeida, T.A., Pardo, T.A.: Towards automatically filtering fake news in portuguese. Expert Systems with Applications p. 113199 (2020)
14. Thorne, J., Vlachos, A.: Automated fact checking: Task formulations, methods and future directions. arXiv preprint arXiv:1806.07687 (2018)
15. Van Der Maaten, L.: Accelerating t-sne using tree-based algorithms. The Journal of Machine Learning Research **15**(1), 3221–3245 (2014)

Solving Inventory Routing Problems with the Gurobi Branch-and-Cut Algorithm

Danny Meier[1], Benjamin Keller[1]. Markus Kolb[1] and Thomas Hanne[1,2] [0000-0002-5636-1660]

[1] University of Applied Sciences and Arts Northwestern Switzerland, Olten, Switzerland
[2] thomas.hanne@fhnw.ch

Abstract. This study is about the implementation and test of a branch-and-cut algorithm for an Inventory-Routing Problem (IRP). The considered mathematical model and the dedicated problem instances have been published by C. Archetti and others. The performance of the implementation on present day computer hardware has been compared to results published in literature several years ago. Also, the influence of the configuration of the used optimizing software package as well as experiences with the publication of the mentioned researchers are documented in this paper.

Keywords: Inventory Routing Problem, Logistics, Numerical Optimization, branch-and-cut, Gurobi.

1 Introduction

The supply chain facilitates the procurement of materials and transforms them into a finished product when it can be distributed to the customers. A supply chain interconnects the supplier, the distributor, the manufacturer, the logistic, the retailer and the end customer. This relation between suppliers till end customers involves a lot of processes and the involvement of the electronic supply chain management (e-scm). Supply chain management is nowadays a crucial factor for success. The role of IT is becoming more and more important and to stay competitive it is inevitable to continuously improve the information system. Larger international companies are facing a lot of challenges like the growing complexity of electronic supply chain integration, the increasing customer demands and global competition (Jitpaiboon 2005).

One part of the supply chain is the distribution of goods. In this study we will focus on the outbound logistic. Products have to be distributed between manufacturer, retailers and customers. In the oil industry gas has to be distributed to the gas stations. In the retailer market products have to be shipped to the retailer stores. What these market segments have in common is that customers will eventually buy from a competitor if products are not available. Depending on market segments products should never run out of stock. But costs incur by storing a product. Depending on the storing condition the product storage can be cheaper in the main warehouse than in the retailer's warehouse. Larger companies, especially global operating corporations, have recognized that optimizing their logistic can yield tremendous cost reductions. In the industry today new ways of relationships have been established in the supply chain. The regular scenario for inventory management is that each retailer will place orders to the supplier to prevent stock out (Retailer Managed Inventory). One of the new scenarios is the so-called Vendor Managed Inventory (VMI). This system moves the inventory planning responsibility to the supplier. The supplier has to monitor the inventory of each retailer and will replenish according to a defined replenishment policy to prevent stock out as well as holding inventory costs as low as possible. The applied model is explained in more detail in Section 3.

Due to its significance to the global industry the research sector has started to address these issues more than half a century ago. One of the first problems that came up in the 1930s was the mathematical optimization problem of visiting all cities on a map with respect to find the shortest tour where each city must be visited only once. This problem is called Traveling Salesman Problem (TSP) and a solution represents a Hamiltonian Circle. The TSP is one of the most basic problems and has its limitations. For traveling to the destinations only one vehicle is available. The vehicle has unlimited capacity and there is no time limitation when a destination should be delivered (Bertazzi & Speranza 2013). Before the algorithm can start calculating all destinations must be known. Solving this simple model is already NP-hard and is classified in most literature as NP-complete shown by Papadimitriou (Papadimitriou 1977).

On top of the TSP another subproblem is in between to the Inventory Routing Problem (IRP). The Vehicle Routing Problem (VRP) extends the raw routing problem by introducing at least one or more vehicles, demanding customers, and other constraints such as multiple suppliers, capacity constraints or time windows, which increases the complexity dramatically. Every vehicle starts and returns after a defined route to the same depot. Between the depot and every customer as well as between all customers is one link that has a specified cost. The

VRP algorithm minimizes the cost to fulfill the customers demand while satisfying the defined constraints (Bertazzi & Speranza 2013). Constraints can be vehicle capacity limitation or the retailer opening hours.

In addition to the VRP the Inventory Routing Problem (IRP) considers time horizon. The algorithm can run over a period of time, for example three days. The calculation can be extended by the decision, if it is better to deliver a customer on day one, two or three. This increases the problem significantly because the quantity to be delivered can vary day by day. To simplify the problem and depending on the products to be delivered a constraint can be defined for example when a customer is visited the stock has to be filled up to the maximum level. The IRP is deciding day by day, which customer has to be served with how much of the specified product (Bertazzi & Speranza 2013). The results of the algorithm are optimized routes and costs. Daily consumption is most of the time an assumption and has to be predicted. This implies in some scenarios a calculation cannot be made in advance. In some environments time can be essential. There are many ways to calculate the companies IRP. The simplest way is the brute-force method which tries out every permutation. This solution needs a lot of computing power and is not solvable for larger instances even not with distributed computing.

The branch-and-cut method is a popular technique to solve many variations of integer linear problems (ILPs) and can be tailored to a specific integer programming problem (Mitchell 1999). There are different libraries or solvers available to find solutions in linear programming. The paper of Archetti et al. (Archetti, Bertazzi, Laporte & Speranza 2007) has used the CPLEX solver from IBM. This paper will give new benchmark results - produced with up-to-date consumer hardware - for the Vendor-Managed Inventory Routing Problem with a deterministic order-up-to level policy. In this implementation we have used the Gurobi solver package. The implemented exact branch-and-cut approach is explained in the paper of Archetti et al. (Archetti et al. 2007). These benchmarks are necessary to have a sound baseline for other solution approaches. They can be compared against that.

This paper is structured as follows. Section 2 gives a literature survey of research papers which has been consulted and read to become acquainted with the topic of inventory routing problems and the understanding of optimization algorithms. Section 3 introduces the reader into the chosen model for what an efficient algorithm is researched. Explanations and finding to certain algorithms have been given in Section 4 whereas Section 5 will list the received results and compare them to existing approaches.

2 Literature Survey

This is not meant to be a comprehensive review of literature in the field of inventory routing but rather give an overview of activity in the field and especially put in context some papers which are relevant for the specific model dealt with in this paper. The first studies on inventory routing problems were conducted in the eighties. Mentionable pioneering work was done by Bell, Dalberto, Fisher, Greenfield, Jaikumar, Kedia, Mack & Prutzman (1983), Federgruen and Zipkin (1984) or Blumenfeld, Burns, Diltz & Daganzo (1985). Since then numerous studies about different variants of inventory routing problems have been discussed in literature. A standard version of the problem does not exist. The discussed problems differ in several criteria. There are several literature surveys which make a classification of the problems presented in published research papers. Among them is Coelho, Cordeau & Laporte (2013) which applies these criteria to about 30 published inventory routing problems: time horizon (finite, infinite), structure (one-to-one, one-to-many, many-to-many), routing (direct, multiple, continuous), inventory policy (maximum level, order-up-to level), inventory decisions (lost sales, backorder, non-negative), fleet composition (homogeneous, heterogeneous), fleet size (single, multiple, unconstrained). Another survey by Andersson, Hoff, Christiansen, Hasle & Løkketangen (2010) classifies about 70 publications in similar categories and elaborates on industrial aspects of combined inventory management and routing.

The problem considered in this study was developed and solved with an exact algorithm by Archetti et al. (Archetti et al. 2007). This paper describes the first attempt to solve an inventory routing problem exactly. The authors developed a branch-and-cut algorithm and a set of problem instances. The instances have been published and used by other researchers in the field for benchmarking. The algorithm of Archetti et al. has later been analyzed and improved by Solyalı and Süral (2011). They use an a priori tour heuristic to simplify the routing decision and found their algorithm to perform better than the one initially proposed by Archetti et al. Also Coelho and Laporte (2013) reference the same model. They extended the model to a multi vehicle problem and developed an algorithm for solving multi vehicle instances to optimality. Archetti, Bertazzi, Hertz & Speranza (2012) also developed a hybrid heuristic based on the problem defined by Archetti et al. (2007).

3 Model

The treated model - Vendor Managed Inventory Routing Problem with a deterministic order-up-to level policy - was introduced first by Bertazzi, Paletta & Speranza (2002). This kind of problem is still very hard to solve exactly for bigger instances even though the computation power of a single consumer computer has increased massively. The defined model behind this paper is reduced to its main characteristics for simplification and comparability to other benchmarks.

A product is shipped from one common supplier to several retailers, where each retailer has a deterministic consumption in each time frame over a given finite time horizon. The goal is to find tours starting from the supplier to only these retailers which have to be served at the respective time frame. The total costs, consisting of inventory and transportation costs, have to be minimized.

3.1 Problem Description and Formulation

A single product is shipped from a supplier s_0 to a set of retailers $M = \{1, 2, ..., n\}$ over a time horizon H. At each discrete time $t \in T = \{1, ..., H\}$ a single product of quantity r_{0t} is produced at the supplier and a quantity of r_{st} is consumed at each retailer $s \in M$. A starting inventory level B_0 at the supplier and I_{s0} at the retailers is given. The maximum inventory level at the supplier is not restricted whereas a retailer has a given maximum inventory level $U_s \geq I_{s0}$. The current inventory level at time t for the supplier s_0 is $B_t \geq 0$ and for each retailer s the current inventory level at time t is $I_{st} \geq 0$. The inventory cost h_0 for the supplier is charged in $T' = T \cup \{H + 1\}$. The extended time horizon $H + 1$ considers that for the time frame before start, the inventory cost must be considered. Inventory costs are always calculated at the end of a time frame that means after replenishment and consumption took place. The retailer s, $s \in M$, has inventory cost h_s of and the total inventory cost over time horizon is then $\sum_{t \in T'} h_s I_{st}$. The inventory level after the time horizon is not relevant, which means that after the last time frame an unfeasible status can occur. Shipments can be performed at any time frame $t \in T$ by one vehicle with capacity C. The vehicle's capacity must not be exceeded at any time. The transportation cost c_{ij} where $c_{ij} = c_{ji}$, $i, j \in M' = M \cup \{0\}$ is calculated by the Euclidean distance $NINT = \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2}$. The inventory policy at the retailers is denoted by order-up-to level (VMIR-OU) that means if a retailer is visited $y_t^{ij} \in \{0,1\}$ at time t the inventory must be replenished up to maximum inventory level U_s. In each time frame $t \in T'$ the supplier replenishes its inventory by r_{0t}. The objective function is to minimize the total transportation and inventory costs without violating any constraint.

$$minimize \sum_{t \in T'} h_o B_t + \sum_{s \in M} \sum_{t \in T'} h_s I_{st} + \sum_{i \in M'} \sum_{j \in M', j \neq i} \sum_{t \in T'} c_{ij} y_{ij}^t \qquad (1)$$

3.2 Mathematical Formulation

Variable	Description
M	Set of retailers
$M^I = M \cup \{0\}$	Set of supplier and retailers
H	Time horizon
T	Set of discrete time frames
$T^I = T \cup \{H + 1\}$	Set of discrete time frames including one time frame before delivery takes place
C	Vehicle capacity
r_{0t}	Production at supplier at time t
r_{st}	Consumption at retailer s at time t
h_0	Unit inventory cost at supplier
h_s	Unit inventory cost at retailer s
B_0	Starting inventory at supplier
B_t	Inventory at supplier at time t
I_{s0}	Starting inventory at retailer s
I_{st}	Inventory at retailer s at time t
U_s	Maximum inventory at retailer s
c_{ij}	Transportation cost from retailer i to retailer j
x_{st}	Shipped quantity to the visited retailer s at time t
z_{st}	1 if retailer s at time t is visited. Otherwise, 0

y_{ij}^{t} Arc from retailer i to retailer j traveled at time t

MIP formulation. An exact definition of the model with all constraints and inequalities has been given by (Archetti et al. 2007). Next to the VMIR-OU policy they also consider other inventory policies such as VMIR and VMIR-ML.

Constraints. The following constraints have been assessed as useful and efficient.

Inventory and stock-out constraints

$$B_t = B_{t-1} + r_{0t-1} - \sum_{s \in M} x_{st-1}, \quad t \in T' \tag{2}$$

The supplier's inventory level is calculated at time *t* by the level at time *t*-1 plus the produced quantity of products r_{0t-1} at time *t*-1, minus the sum of quantity shipped x_{st-1} to the retailers at time *t*-1.

$$B_t \geq \sum_{s \in M} x_{st} \tag{2}$$

Supplier stock-out constraint: This constraint ensures that the inventory level at the supplier has sufficient quantity in stock for the upcoming deliveries at time *t*.

$$I_{st} = I_{st-1} + x_{st-1} - r_{st-1}, s \in M, t \in T' \text{ where } x_{s0} = r_{s0} = 0, s \in M \tag{4}$$

Retailer inventory definition: The retailer's inventory level is given a time *t* by the level at time *t*-1, plus the quantity x_{st-1} delivered at time *t*-1, minus the used quantity r_{st-1} at time *t*-1.

$$I_{st} \geq 0, s \in M, t \in T' \tag{5}$$

Replenishment constraints

$$x_{st} \geq U_s z_{st} - I_{st}, s \in M, t \in T \tag{6}$$

$$x_{st} \leq U_s - I_{st}, s \in M, t \in T \tag{7}$$

$$x_{st} \leq U_s z_{st}, s \in M, t \in T \tag{8}$$

Order-up-to level: These constraints define that if a retailer *s* is served the quantity x_{st} shipped to the retailers will fill up the inventory to the maximum U_s. If a retailer *s* is served at time *t* the variable z_{st} is 1 and otherwise equals 0.

Capacity constraint

$$\sum_{s \in M} x_{st} \leq C, t \in T \tag{9}$$

Capacity constraint: This constraint ensures that the delivered product quantity does not exceed the load capacity *C* of the transportation vehicle at any time $t \in T'$.

Routing constraints

$$\sum_{j \in M', j<i} y_{ij}^{t} + \sum_{j \in M', j>i} y_{ij}^{t} = 2z_{it} \tag{10}$$

Routing constraint: When a retailer *s* is served at time *t* the vehicle must use a route leaving from retailer *s* at time *t*.

(a) Subtour elimination constraint according to Fischetti, Gonzalez & Toth (1998) and (Gendreau, Laporte & Semet (1998)

$$\sum_{i \in S} \sum_{j \in S, j<i} y_{ij}^{t} \leq \sum_{i \in S} z_{it} - z_{kt}, S \subseteq M, t \in T \text{ for some } k \in S \tag{11}$$

(b) Traditional subtour constraint according to (Dantzig, Fulkerson & Johnson 1954)

$$\sum_{i \in S} \sum_{j \in S, j<i} y_{ij}^{t} \leq |S| - 1, S \subseteq M, t \in T \tag{12}$$

(c) Improved subtour constraint (11)

$$\sum_{t\in T'}\left(\sum_{i\in S}\sum_{j\in S, j<i} y_{ij}^{t'} \right) \leq \sum_{i\in S} z_{it'} - z_{kt'}),\ S \subseteq M, t \in T, k \in S \text{ where } S \text{ is subtour of } t \quad (13)$$

$$x_{st} \geq 0,\ s \in M,\ t \in T \quad (14)$$

Non-negativity constraint: When a retailer *s* is visited at time *t* the delivered good x_{st} cannot be below zero.

$$y_{ij}^{t} \in \{0,1\},\ i,j \in M,\ \mathrm{j} < \mathrm{i},\ t \in T \quad (15)$$

Path constraint: When the path between retailer i to retailer j is traveled at time t the value is equal 1 otherwise 0.

$$y_{i0}^{t} \in \{0,1,2\},\ i \in M,\ t \in T \quad (16)$$

Path constraint: When the path between supplier and retailer *i* is traveled at time *t* and continued to another retailer the value y_{i0}^{t} is equal 1 and when same path is taken back to the supplier the value y_{i0}^{t} is equal 2. If the path has not been taken yet the value y_{i0}^{t} is zero.

$$z_{it} \in \{0,1\},\ i \in M',\ t \in T \quad (17)$$

Retailer visit constraint: When the retailer z_{it} is visited at time *t* the value is equal 1 otherwise zero.

Inequalities
The following inequalities have been assessed as useful and efficient.

$$I_{st} \geq (1 - z_{st}) r_{st},\ s \in M,\ t \in T \quad (18)$$

When the retailer s is not visited (z_{st} is equal zero) at time *t,* then the inventory level I_{st} must be at least equal to the product quantity r_{st} consumed at time *t*.

$$I_{st-k} \geq \left(\sum_{j=0}^{k} r_{st-j}\right)\left(1 - \sum_{j=0}^{k} z_{st-j}\right),\ s \in M,\ t \in T, k = 0, \dots, t-1 \quad (19)$$

This inequality extends the inequality (18) by the value *k*. The period of time retailers are not served is calculated by $t - k$.

$$I_{st} \geq U_s z_{st-k} - \sum_{j=t-k}^{t-1} r_{st},\ s \in M,\ t \in T, k = 1, \dots, t-1 \quad (20)$$

The inventory at a retailer I_{st} at time *t* must be equal or greater than the inventory at the last delivery minus the sum of consumed goods since then.

$$z_{st} \leq z_{0t},\ s \in M,\ t \in T \quad (21)$$

When any retailer *s* is visited at time *t,* z_{st} is equal 1. Additionally, the supplier has to be included in the route at time *t*, z_{0t} equals 1. This inequality is to prevent that a retailer can be visited without ever having visited the supplier.

$$y_{i0}^{t} \leq 2 z_{it},\ s \in M,\ t \in T \quad (22)$$

If the arc between supplier and retailer *i* has been traveled at time *t* then retailer *i* must be visited.

$$y_{ij}^{t} \leq z_{it},\ i,j \in M,\ t \in T \quad (23)$$

If the arc between retailer i and j has been traveled, then both retailers must be visited.

3.3 Test instances

Archetti et al. (2007) produced a large set of 160 test instances classified into three groups and five test sets. These instances are specific to the Vendor-Managed Inventory Routing Problem and are the most common and most used instances to benchmark. The three parameters which identify the test instance and its respective properties are the time horizon *h* = {3, 6} that describes how many discrete time frames should be taken into account. The inventory costs *c* = {low, high} are grouped into two ranges of values. Each retailer and the supplier are in this inventory cost range. The low group has inventory costs between 0.01and 0.05 and the high group has inventory costs multiplied by 10 in the range of 0.1 to 0.5. The third parameter describes how many retailers s are involved. A test set *i* = {1, …, 5} contains every combination of the parameters mentioned before. To identify the file the following naming convention

has been applied: h[h][c][i]n[s]. For example, a test instance from test set 2 with 6 time frames, 30 retailers and low inventory costs is named as h6low2n30. A complete test set with all combinations is listed at Table 2 on page 10. The whole collection could be obtained from (Coelho n.d.). Therefore, one test set consists of 32 test instances. Table 1 summarizes the properties within one test set.

Table 1. Content of a test set provided by Archetti et al.

Inventory cost *c*	**Time frames** *h*	**Retailers** *s*
Low [0.01, 0.05]	3	$5 \leq x \leq 50, x \mod 5 = 0$
Low [0.01, 0.05]	6	$5 \leq x \leq 30, x \mod 5 = 0$
High [0.1, 0.5]	3	$5 \leq x \leq 50, x \mod 5 = 0$
High [0.1, 0.5]	6	$5 \leq x \leq 30, x \mod 5 = 0$

In this paper only the first two test sets have been fully benchmarked and compared. The results of the other three test sets are quite similar. Test set three and four have the same characteristics as test set two. And test set five has the same characteristics as test set one.

4 Algorithm

To solve the inventory routing problem the commercial solver package from Gurobi has been examined. The development has been done in C# and .NET 4.0. The developed implementation fully concentrates on the branch-and-cut algorithm and its limited options to fine tune the behavior of the solving process. The evaluated parameters are specific to the used solver package and do not allow further in-depth controlling.

- CPU cores: The default setting is set to the amount of available physical cores. Intel based CPUs with Hyper-Threading support offer the double number of cores to use.
- Heuristic ratio: The percentage of heuristic influence as a ratio of the total computation time to increase finding feasible solutions.
- Presolve capabilities: How much effort should be invested to try to tighten the model.
- Solving strategy: Sets the focus with which strategy a feasible solution should be found. Strategy can be set to balanced, focus on finding fast feasible solutions, focus on optimality or focus on best bounds.
- Branching priority: Without configuration the branching priority is set automatically and can change during the execution. Setting the branching priority for certain decision variables overrides the automatic behavior and considers the set value with priority.

Next to the configurable properties, optimization has been tried to apply through finding stronger constraints for the subtour elimination (see 4.1). Adding each subtour elimination constraint at the beginning is not feasible because of the number of available permutations to form the constraint. Next to time issues this will blow up computer memory at the time. According to best practice this kind of constraints will be added as soon as a violated intermediate solution will occur. Adding the additional constraint just after the solving process and resolve the model again has been deemed as insufficient. This has been proven by some simple runs of more complex test instances. Thus, the approach using a callback method is even more efficient. If the callback method recognizes an optimal solution with a violated subtour, a lazy constraint according to (11) will be added. In Applegate, Bixby, Chvátal & Cook (2007) different approaches for subtour elimination are examined. For the test series in this paper the two most promising algorithms have been implemented. The proposed subtour elimination algorithm by (Archetti et al. 2007) is the separation algorithm of (Padberg & Rinaldi 1991) with the constraint definition of (11) that was introduced by (Fischetti et al. 1998) and (Gendreau et al. 1998). The side constraint $k = \arg\max_j \{z_{jt}\}$ has been also considered but simplified. Then z_{jt} is a binary decision variable that is only 1 if $j \in M$ has been visited in time $t \in T$. Under the condition that $j \in S$ and $S \subseteq M$, every k is part of the subtour and has been visited. But we found a more efficient way to find an optimal solution faster. If a violated constraint will not only be added for the violated time frame but also be added to all time frames of the test instance, then an average time saving of factor 1.5 can be expected. The traditional subtour elimination algorithm (12) mostly used in TSP related problems has been proven as inefficient.

Unfortunately, the consumed literature did not provide any exact implementation details even for the most critical part - namely the subtour elimination constraint. Based on a personal interpretation of the mathematical

equation the first implementations and all further deviates have been implemented as best possible. Also, source code was not available - neither on request - from the paper authors.

4.1 Implementation of Subtour Elimination Constraints

When the solver finds a feasible solution the callback method checks its routes for subtours. For this the algorithm starts at the supplier ($i = 0$) and transits to a retailer j through an arc which is part of the tour ($y_{oj}^t \geq 0$). From this retailer j the tour proceeds through an arc to another retailer k ($y_{jk}^t = 1$). In this way the arcs forming the route are followed from retailer to retailer until the supplier is reached again. Subtours exist when not every served retailer ($z_{st} = 1$) has been visited by following these adjacent arcs. The retailers visited in this tour form a subtour but this subtour is not considered a real subtour because it contains the supplier.

From the remaining retailers which are served but have no connection to the supplier a retailer is selected as the starting point of the first real subtour. From there adjacent arcs are followed until this retailer is reached again. The visited retailers are then part of the first subtour. If there are still retailers which are served but not part of the first subtour and also are not connected to the supplier again one is selected as the starting point for the second subtour. This procedure is repeated until all served retailers are part of a subtour.
For every real subtour (subtour that does not contain the supplier) the following constraints are added to the model as lazy constraints:

$$\sum_{i \in S} \sum_{j \in S, j<i} y_{ij}^t \leq \sum_{i \in S} z_{it} - z_{kt}, t \in T \text{ for every } k \in S \qquad (24)$$

where S is the set of retailers forming a subtour. Because similar subtours were likely to appear in multiple time frames the constraints (24) were not only constructed for the time frame t in which the specific subtour was found but for all $t \in T$.

5 Results

5.1 Algorithm Optimization Results

Experiments showed that subtle adjustments to some of the settings of the Gurobi solver sometimes resulted in noticeable change of the calculation time. Also experimenting with attributes of variables resulted in unexpected changes of performance.

With Gurobi the type attribute of variables can be defined as integer, binary or continuous. Variables also have attributes for upper and lower bound. The following two examples illustrate what was observed when experimenting with these attributes. Changing the attribute type of the y_{ij}^t (vertex i to j is part of the route at time t) variables from binary to integer with a lower and upper bound of 0 and 1 respectively, improved the performance of the program. Another example of the influence of variable attributes was observed with the x_{st} variables (delivered quantity). Initially the upper bound attribute of these variables was set to the maximum inventory level Us which is valid for the VMIR-OU and VMIR-ML policy. Decreased calculation time was observed when the upper bound attribute was set to infinity. The maximum value for x_{st} was restricted by the replenishment constraints (6) to (8) which were added to the Gurobi model.

Calculations have been performed for several different combinations of settings for the Gurobi solver. A comparison of the program performance with different settings for the number of used CPUs was made. A selection of instances was calculated using four and eight CPUs. The total time used for calculating the test set instances was lower when using eight CPUs. However, some instances were calculated faster with only four CPUs. A pattern relating the computation times to the instance size by number of retailers or time frames was not recognizable, although instances with a rather long computation time were more likely to perform better with eight CPUs.

To determine the influence of the branching priority settings of the solver, several computations were compared. For each computation over a set of problem instances a different variable group was prioritized. That means the instance set was solved setting the branching priority attribute of for example all z_{st} to 1 while for all other variables the attribute was 0 by default. Then the calculation was repeated for other variable groups, e.g., x_{st} or y_t with increased branching priority.
The conclusion from these experiments was that the branching priority did not have any noticeable impact on the overall performance of the solving process. The usage of different solving strategies by alteration of the MIPFocus parameter of the solver did not result in faster computation compared to the default setting. With the default setting the solver automatically balances between finding new feasible solutions and proving that the current solution is optimal. Neither emphasizing the first nor the later nor a focus on the bound improved the perfor-

mance of the calculation. The increase of time the solver should spend for feasibility heuristics by raising the according parameter had a negative impact on the computation time.

5.2 Computation Results

Computation results have been calculated on a system with a Microsoft Windows 8.1 64bit operating system and a quad core Intel i7 3770k CPU with 16GB of memory. The source code is written in C# .NET 4.0 and compiled with enabled code optimization. The MIP solving package is provided by Gurobi in version 6.0.4. Maximum running time is limited to 1800s (30min). Source code is available on request to the authors.

The computation results are structured into two different benchmark categories. The first category compares the different implemented solutions discussed in section 4. The second category shows the evolution of computation power. The results are compared to the initial results from Archetti et al. (2007) and the newer result from (Coelho & Laporte 2014).

The test runs in this paper have been executed with four different settings. At first the difference of computation time has been examined if different thread counts have been used (4T or 8T). The CPU in this test environment has four physical cores and the ability to use the Hyper-Threading technology from Intel which doubles the count of virtual cores. Other settings than four or eight threads has been proven to be not efficient. Lower counts don't use the full power of the machine and higher counts have the disadvantage of rescheduling overhead within the process execution. Between the two chosen settings no overall winner has been evaluated. For some test instances the setting with four threads performs better than using eight threads. And for some others the opposite is the case. But no prediction is possible which performs better according to retailer count or involved time frames. Regarding to all tests that have been done, the average computation time of the four thread count settings is slightly lower. The second setting was the introduction of the more efficient subtour elimination constraint (13) (SEE1 - Extended subtour elimination constraint (13) used / SEE0 - Original subtour elimination constraint (11) used). Implementation details of the new constraint are described in Section 4.1. This extension has been proven as more efficient. In most cases the computation time could be reduced. In some special cases the algorithm was able to find an optimal solution within the time limit for test instances, which are not solvable with the original subtour elimination constraint in time.

Unfortunately, the implementation in this paper was not able to produce similar or better results in case of computation time as the results from Coelho et al. Even after this study it is not evident why the implementation is worse. Reasons could be the implementation itself or the chosen solver package from Gurobi.

Table 2. Computation results for the self-implemented branch-and-cut algorithm

Instance	Costs	8T-SI E1		4T-SEE1		8T-SEE0		4T-SEE0	
		t [s]	G [%]	t [s]	G [%]	t [s]	G [%]	t [s]	G [%]
h3high1n10	4'970.62	0		0		0		0	
h3high1n15	5'713.84	0		0		0		0	
h3high1n20	7'353.82	1		1		1		1	
h3high1n25	8'657.70	2		1		1		1	
h3high1n30	12'635.55	232		324		1'007		850	
h3high1n35	11'984.69	28		6		21		35	
h3high1n40	14'006.60	*1'800	2.7%	*1'800	0.3%	*1'800	1.6%	*1'800	1.6%
h3high1n45	14'661.20	40		83		342		213	
h3high1n5	2'149.80	0		0		0		0	
h3high1n50	15'235.80	*1'800	0.8%	*1'800	0.6%	*1'800	0.5%	*1'800	1.8%
h3low1n10	2'167.37	0		0		0		0	
h3low1n15	2'236.53	0		0		0		0	
h3low1n20	2'793.29	1		1		1		1	
h3low1n25	3'309.64	1		1		1		1	
h3low1n30	3'918.76	229		199		1'032		529	
h3low1n35	3'694.48	7		5		18		13	
h3low1n40	4'263.43	*1'800	0.9%	*1'800	7.6%	*1'800	1.9%	*1'800	10.0%
h3low1n45	4'369.38	67		70		134		145	
h3low1n5	1'281.68	0		0		0		0	
h3low1n50	4'629.92	*1'800	8.2%	*1'800	1.3%	*1'800	4.9%	*1'800	5.6%
h6high1n10	8'870.15	1		1		4		3	
h6high1n15	12'118.83	2		1		2		12	
h6high1n20	14'702.95	329		141		134		159	
h6high1n25	15'581.47	16		7		51		75	5.0%
h6high1n30	23'184.00	*1'800	4.8%	*1'800	3.9%	*1'800	n/a	*1'800	

h6high1n5	5'942.82	0		0		0		0	
h6low1n10	4'499.25	1		1		1		2	
h6low1n15	5'462.68	1		1		2		2	
h6low1n20	6'490.18	466		128		353		159	
h6low1n25	7'095.86	92		35		138		69	
h6low1n30	8'319.59	*1'800	19.4%	*1'800	10.8%	*1'800	7.1%	*1'800	1.9%
h6low1n5	3'335.24	0		0		0		0	
Average [s]		385		369		439		225	
Solved		26 / 32		26 / 32		26 / 32		**30** **32**	

* = Not able to solve test instance to optimality within time limit

Table 3. Computation result comparison - Current implementation vs. Archetti vs. Coelho et al.

Instance	**Costs**	**4T-SS E1**		**Archetti**		**Coelho et al.**	
		t [s]	**G [%]**	**t [s]**	**G [%]**	**t [s]**	**G [%]**
h3high2n10	4'803.17	0		0		0	
h3high2n15	5'821.04	0		0		1	
h3high2n20	7'385.03	1		10		6	
h3high2n25	9'266.87	1		14		9	
h3high2n30	11'351.36	324		164		18	
h3high2n35	10'706.91	6		199		24	
h3high2n40	11'722.58	*1'800	2.7%	1'003		74	
h3high2n45	13'675.96	83		1'205		76	
h3high2n5	1'959.05	0		0		0	
h3high2n50	15'453.80	*1'800	0.8%	*1'800	n/a	148	
h3low2n10	2'510.13	0		0		2	
h3low2n15	2'506.21	0		0		0	
h3low2n20	2'799.90	1		12		7	
h3low2n25	3'495.97	1		25		7	
h3low2n30	3'737.11	199		84		13	
h3low2n35	3'796.80	5		173		27	
h3low2n40	4'166.95	*1'800	0.9%	1'500		74	
h3low2n45	4'226.82	70		1'133		38	
h3low2n5	1'176.63	0		0		0	
h3low2n50	4'919.75	*1'800	8.2%	*1'800	n/a	235	
h6high2n10	8'569.73	1		11		7	
h6high2n15	11'932.10	1		22		6	
h6high2n20	14'646.96	141		1'536		470	
h6high2n25	16'823.20	7		578		35	
h6high2n30	20'090.29	*1'800	4.8%	*1'800	n/a	802	
h6high2n5	5'045.91	0		0		1	
h6low2n10	5'236.98	1		7		16	
h6low2n15	5'494.74	1		39		10	
h6low2n20	6'082.54	128		*1'800	n/a	688	
h6low2n25	7'484.84	35		548		27	
h6low2n30	7'761.53	*1'800	19.4%	*1'800	n/a	1'190	
h6low2n5	2'722.33	0		0		1	
Average [s]		369		539		125	
Solved		26 / 32		27 / 32		32 / 32	

* = Not able to solve test instance to optimality within time limit

Table 4. Computation results for the self-implemented branch-and-cut algorithm

Instance	Costs	8T-SI E1		4T-SEE1		8T-SEE0		4T-SEE0	
		t [s]	G [%]	t [s]	G [%]	t [s]	G [%]	t [s]	G [%]
h3high2n10	4'803.17	0		0		0		0	
h3high2n15	5'821.04	1		0		1		1	
h3high2n20	7'385.03	1		1		1		1	
h3high2n25	9'266.87	85		69		89		85	
h3high2n30	11'351.36	8		50		31		14	
h3high2n35	10'706.91	3		2		3		4	
h3high2n40	11'722.58	128		187		570		456	
h3high2n45	13'675.96	338		119		262		244	
h3high2n5	1'959.05	0		0		0		0	
h3high2n50	15'453.80	702		1'191		1'250		422	
h3low2n10	2'510.13	0		0		0		0	
h3low2n15	2'506.21	0		0		0		1	
h3low2n20	2'799.90	1		1		1		1	
h3low2n25	3'495.97	60		48		126		262	
h3low2n30	3'737.11	67		11		41		35	
h3low2n35	3'796.80	23		3		3		3	
h3low2n40	4'166.95	90		257		1'119		783	
h3low2n45	4'226.82	428		303		232		349	
h3low2n5	1'176.63	0		0		0		0	
h3low2n50	4'919.75	246		462		386		408	
h6high2n10	8'569.73	1		0		1		0	
h6high2n15	11'932.10	2		2		10		11	
h6high2n20	14'646.96	16		5		79		46	
h6high2n25	16'823.20	1'352		1'432		1'204		*1'800	0.3%
h6high2n30	20'090.29	287		417		215		191	
h6high2n5	5'045.91	0		0		0		0	
h6low2n10	5'236.98	0		0		1		1	
h6low2n15	5'494.74	1		2		2		16	
h6low2n20	6'082.54	12		9		41		32	
h6low2n25	7'484.84	*1'800	0.9%	419		*1'800	1.1%	*1'800	1.0%
h6low2n30	7'761.53	711		182		437		240	
h6low2n5	2'722.33	0		0		0		0	
Average [s]		199		162		247		225	
Solved		31 / 32		32 / 32		31 / 32			
								31	**32**

* = Not able to solve test instance to optimality within time limit

Table 5. Computation result comparison - Current implementation vs. Archetti vs. Coelho et al.

Instance	Costs	4T-SSE1		Archetti		Coelho et al.	
		t [s]	G [%]	t [s]	G [%]	t [s]	G [%]
h3high2n10	4'803.17	0		0		0	
h3high2n15	5'821.04	0		1		3	
h3high2n20	7'385.03	1		8		5	
h3high2n25	9'266.87	69		47		7	
h3high2n30	11'351.36	50		130		20	
h3high2n35	10'706.91	2		97		21	
h3high2n40	11'722.58	187		449		29	
h3high2n45	13'675.96	119		553		145	
h3high2n5	1'959.05	0		0		0	
h3high2n50	15'453.80	1'191		1'782		251	
h3low2n10	2'510.13	0		0		0	
h3low2n15	2'506.21	0		1		2	
h3low2n20	2'799.90	1		6		7	
h3low2n25	3'495.97	48		53		16	
h3low2n30	3'737.11	11		128		18	
h3low2n35	3'796.80	3		74		23	
h3low2n40	4'166.95	257		369		38	
h3low2n45	4'226.82	303		928		158	
h3low2n5	1'176.63	0		0		0	
h3low2n50	4'919.75	462		1'235		133	
h6high2n10	8'569.73	0		7		4	
h6high2n15	11'932.10	2		31		6	
h6high2n20	14'646.96	5		354		37	
h6high2n25	16'823.20	1'432		1'732		113	
h6high2n30	20'090.29	417		*1'800	n/a	197	
h6high2n5	5'045.91	0		0		1	
h6low2n10	5'236.98	0		4		2	
h6low2n15	5'494.74	2		22		8	
h6low2n20	6'082.54	9		282		52	
h6low2n25	7'484.84	419		1'710		222	
h6low2n30	7'761.53	182		*1'800	n/a	220	
h6low2n5	2'722.33	0		0		1	
Average [s]		162		425		54	
Solved		32 / 32		30 / 32		32 / 32	

* = Not able to solve test instance to optimality within time limit

6 Conclusions

The initial idea of this research project was to do experiments with a heuristic to solve a problem in the field of inventory routing. Probably a heuristic, ideally with problem instances for bench- marking, could be found and improved. However, the search in literature for such a heuristic was not successful, but problem instances for an inventory routing model could be found. The problem instances were created during the development of an exact branch-and-cut algorithm for solving an inventory routing problem by (Archetti et al. 2007).

With the existence of an optimal solution a heuristic could be benchmarked against it. For a benchmark to be valid the computing environment would have to be similar. Because the results by Archetti et al. were calculated some years ago and the algorithms implementation was not available, the work concentrated on re-implementing the branch-and-cut algorithm. Due to the limited time horizon for this research project the work concentrated on the exact algorithm and its implementation.

The algorithm was implemented using the Gurobi Optimizer software package. It was then run against the mentioned problem instances and the outcomes were compared to results available in literature. In general, we found the optimal solution faster, which is undoubtedly due to the better performance of the computer hardware and maybe, also to the advancement of the solver software (although not the same products were used than in other papers). However, for bigger instances the computation time significantly exceeded earlier results if they were at all solvable in the given time limit. It is not quite clear why this is the case. Probable causes could lie in the internals of the used solver software. We also assume that the detailed handling of the subtour elimination constraints could be different. The description of these constraints in the referred literature was not detailed enough for us to be clear on how exactly these were constructed and implemented.

Several experiments were made with solver specific settings and variable arguments. These tests showed that the performance of the calculation depends a lot on the configuration of the underlying solver software and on the specific problem instance.

References

Andersson, H., Hoff, A., Christiansen, M., Hasle, G. & Løkketangen, A. (2010), 'Industrial aspects and literature survey: Combined inventory management and routing', Computers & Operations Research 37(9), 1515–1536.

Applegate, D. L., Bixby, R. E., Chvátal, V. & Cook, W. J. (2007), The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics), Princeton University Press.

Archetti, C., Bertazzi, L., Hertz, A. & Speranza, M. G. (2012), 'A hybrid heuristic for an inventory routing problem', INFORMS Journal on Computing 24(1), 101–116.

Archetti, C., Bertazzi, L., Laporte, G. & Speranza, M. G. (2007), 'A branch-and-cut algorithm for a vendor-managed inventory-routing problem', Transportation Science 41(3), 382–391.
Bell, W. J., Dalberto, L. M., Fisher, M. L., Greenfield, A. J., Jaikumar, R., Kedia, P., Mack, R. G. & Prutzman, P. J. (1983), 'Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer', Interfaces 13(6), 4–23.
Bertazzi, L., Paletta, G. & Speranza, M. G. (2002), 'Deterministic order-up-to level policies in an inventory routing problem', Transportation Science 36(1), 119–132.
URL: http://dx.doi.org/10.1287/trsc.36.1.119.573
Bertazzi, L. & Speranza, M. G. (2013), 'Inventory routing problems with multiple customers', EURO J Transp Logist 2(3), 255–275.
URL: http://dx.doi.org/10.1007/s13676-013-0027-z
Blumenfeld, D. E., Burns, L. D., Diltz, J. D. & Daganzo, C. F. (1985), 'Analyzing trade-offs between transportation, inventory and production costs on freight networks', Transportation Research Part B: Methodological 19(5), 361–380.
Coelho, L. (n.d.), 'Collection of test instances for the inventory routing problem', Website URL: http://www.leandro-coelho.com/instances/inventory-routing/
Coelho, L. C., Cordeau, J.-F. & Laporte, G. (2013), 'Thirty years of inventory routing', Transportation Science 48(1), 1–19.
Coelho, L. C. & Laporte, G. (2013), 'The exact solution of several classes of inventory-routing problems', Computers & Operations Research 40(2), 558–565.
Coelho, L. C. & Laporte, G. (2014), 'An optimised target-level inventory replenishment policy for vendor-managed inventory systems', International Journal of Production Research 53(12), 3651–3660.
URL: http://dx.doi.org/10.1080/00207543.2014.986299
Dantzig, G., Fulkerson, R. & Johnson, S. (1954), 'Solution of a large-scale traveling-salesman problem', Journal of the Operations Research Society of America 2(4), pp. 393–410. URL: http://www.jstor.org/stable/166695
Federgruen, A. & Zipkin, P. (1984), 'A combined vehicle routing and inventory allocation problem', Operations Research 32(5), 1019–1037.
Fischetti, M., Gonzalez, J. J. S. & Toth, P. (1998), 'Solving the orienteering problem through branch-and-cut', INFORMS Journal on Computing 10(2), 133–148.
URL: http://dx.doi.org/10.1287/ijoc.10.2.133
Gendreau, M., Laporte, G. & Semet, F. (1998), 'A branch-and-cut algorithm for the undirected selective traveling salesman problem', Networks 32(4), 263–273.
URL: http://dx.doi.org/10.1002/(sici)1097-0037(199812)32:4¡263::aid-net3¿3.3.co;2-h
Jitpaiboon, T. (2005), 'The roles of information systems integration in the supply chain integration context- firm perspective', Supply Chain.
Mitchell, J. E. (1999), 'Branch-and-cut algorithms for combinatorial optimization problems', Mathematical Sciences.
Padberg, M. & Rinaldi, G. (1991), 'A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems', SIAM Rev. 33(1), 60–100.
URL: http://dx.doi.org/10.1137/1033004

Papadimitriou, C. H. (1977), ‘The Euclidean travelling salesman problem is NP-complete’, Theoretical Computer Science 4(3), 237–244.
URL: http://dx.doi.org/10.1016/0304-3975(77)90012-3
Solyalı, O. & Süral, H. (2011), ‘A branch-and-cut algorithm using a strong formulation and an a priori tour-based heuristic for an inventory-routing problem’, Transportation Science 45(3), 335–345.
URL: http://dx.doi.org/10.1287/trsc.1100.0354

A gradient-based path-relinking algorithm for bi-objective optimization problems[1]

N. Aslimani, E-G. Talbi, R. Ellaia

University of Lille, France. e-mail: nassime.aslimani@inria.fr, el-ghazali.talbi@univ-lille.fr

Abstract

This paper presents a new approach for bi-objective continuous optimization based on the exploitation of available gradient information. Given any two Pareto solutions, the proposed algorithm integrates the multiple gradient descent algorithm (MGDA) to carry out a path-relinking in the objective space using a retroactive process. It can be used as an intensification procedure for any multi-objective metaheuristic. Using the anchor solutions as input Pareto solutions, the algorithm can approximate the whole Pareto front.

The proposed PR-MGDA (Path-Relinking MGDA) algorithm shows its efficiency in the generation of evenly distributed solutions in the Pareto front connecting two given Pareto solutions in the objective space. Considering the two anchor solutions, the algorithm outperforms popular evolutionary algorithms both in terms of the quality of the obtained Pareto fronts (convergence, diversity and spread) and specially the search time.

Keywords: Path-relinking, Bi-objective Optimization, Gradient methods, MGDA algorithm, Anchor points.

1. Introduction

Optimization problems encountered in practice are seldom single-objective. In general, there are many conflicting objectives to handle: for instance, to minimize the cost and to maximize the quality (e.g. physic, mechanic, service). Indeed, many diverse areas (networking, computational biology, logistics and transportation, engineering design, finance, environment among others) are concerned by multi-objective optimization problems (MOPs). The optimal solution for MOPs in not a single solution as in single-objective optimization problems, but a set of solutions defined as Pareto optimal solutions. A solution is Pareto optimal if it is not possible to improve a given objective without deteriorating at least another objective. This set of solutions represents the compromise solutions between the different conflicting objectives.

[1]The ELSAT2020 project is co-financed by the European Union with the European Regional Development Fund, the French state and the Hauts de France Region Council

Gradient based methods are widely used for solving single-objective optimization problems, with or without constraints. However, for MOPs, this is less well established. In order to solve MOPs using gradient based methods, a popular strategy is the weighted sum formulation (WSF), which consists in combining all the objectives into a single objective using a linear agregation method. Multiple solutions can be obtained by varying the weight coefficients among the objective functions to hopefully generate different Pareto solutions. Many other scalarization functions have been used in the literature, such as the normal boundary intersection (NBI) [1], normal constraint method (NC) [2], physical programming method (PP) [3], goal programming (GP) [4], ϵ-constraint method [5], and the directed search domain (DSD) [6, 7]. Another gradient based strategies are used in memetic algorithms in which we hybridize multi-objective evolutionary algorithms (MOEAs) with local search strategies where gradient information is used to built a Pareto descent direction [8]. For instance, Harada et al. proposed a new gradient based local search method called the Pareto Descent Method, based on random selection of search directions among Pareto descents [9]. Kim et al. presented a directional operator to further enhance convergence of any MOEAs by introducing a local gradient search method to multi-objective global search algorithms [10]. Recently, Lara et al. compared hybrid methods with a new local search strategy without explicit gradient information and showed that using the gradient information was beneficial [11]. Many other works propose pure gradient based methods for MOPs. In [12][13][14], a common descent direction is computed along all objectives. For instance, the multiple gradient descent algorithm (MGDA) was developed as an extension of the steepest descent method to MOPs [12]. In [15], an analytical description of the entire set of descent directions, that can be integrated in a new gradient based method for MOPs, have been presented.

Without loss of generality, we assume that all objectives are to be minimized, then we consider a MOP of the form:

$$\min_{X \in S} \ \mathbf{F}(X) = (f_1(X), \cdots, f_m(X))^{\mathrm{T}} \tag{1}$$

where:

$f_k : \mathbb{R}^n \longrightarrow \mathbb{R}$, for $k \in \{1, \cdots, m\}$, denotes the objective functions, S is the decision space: $S = \prod_{i=1}^{n} [l_i, u_i]$, X is the decision vector with n decision variables: $X = (x_1, \cdots, x_n) \in \mathbb{R}^n$,.

For the convenience of later discussion, we introduce some basic concepts:

Definition 1. *Pareto dominance*: let $X = (x_1, x_2, ..., x_n)$ and $Y = (y_1, y_2, ..., y_n)$ be decision vectors (solutions). Solution X is said to dominate solution Y, denoted as $X \preceq Y$, if and only if :

$$\forall i \in [1, m] \ : \ f_i(X) \leq f_i(Y)) \wedge (\exists j \in [1, m] \ : \ f_j(X) < f_j(Y) \tag{2}$$

Definition 2. *Pareto optimal solution*: a solution X is Pareto optimal if it is not dominated by any other solution which means there is no other solution

$Y \in S$ such that $Y \preceq X$.

Definition 3. *Pareto Set & Pareto front*: the set of all Pareto optimal solutions is called Pareto set (PS). The corresponding set of Pareto optimal objective vectors is called Pareto front (PF).

In this paper, we propose a new gradient based approach for path-relinking in bi-objective optimization problems using two complementary dynamics: The MGDA gradient approach combined to a backtracking approach. Starting with a couple of Pareto solutions (e.g. anchor points or any potential Pareto solutions) the proposed algorithm alternates the two approaches in a way that generates a zigzag move along the PF connecting the two Pareto solutions. Starting from a Pareto solution, a MGDA descent is first applied locally and then a reverse move is applied to get a new solution from which we operates again a new MGDA descent (toward the PF) and so on until meeting the second Pareto solution.

The remainder of this paper is organized as follows. Section 2 is a brief presentation of the MGDA algorithm. Section 3 describes our proposed path-relinking approach. Section 4 reports the computational results. Finally, Section 5 presents the conclusions and future research perspectives.

2. The MGDA algorithm

The Multi-Gradient Descent Algorithm (MGDA) is an extension of the classical Gradient Descent Algorithm to MOPs [16]. The main idea of MGDA is to identify a direction common to all criteria along which the value of every objective improves. For a MOP minimization problem we seek a descent direction common to all criteria. This algorithm is proved to converge towards a Pareto stationary solution. The common descent vector is defined as the unique element minimizing the norm in the convex hull $\overline{U}$ of the gradients of each objective, where $\overline{U}$ is defined for a given $x \in \mathbb{R}^n$ as:

$$\overline{U} = \{\omega \in \mathbb{R}^n, \omega = \sum_{i=1}^{n} \alpha_i \nabla f_i(x), \sum_{i=1}^{n} \alpha_i = 1\} \tag{3}$$

Indeed, since $\overline{U}$ is a closed, bounded and convex set associated in the affine space $\mathbb{R}^n$, then $\overline{U}$ admits a unique element of minimum norm, say δ. Two cases are possible:

- $\delta = 0$, and then x represents a Pareto-stationary solution, a necessary condition for Pareto-optimality.
- $\delta \neq 0$, and the directional derivatives of the objective functions satisfy the inequalities:
$$(\nabla f_i, \delta) \geq \|\omega\|^2 \tag{4}$$
Hence, $-\delta$ is a descent direction common to all the objective functions.

The MGDA procedure is given in the following algorithm:

Algorithm 1 : **MGDA** procedure

1: Solution initialization $X = X^0$
2: Evaluate n objective $f_i(X)$; $\forall i = 1 \cdots n$; Compute the normalised gradient vectors $\delta_i = \nabla f_i(X)/Si$; $\forall i = 1 \cdots n$;
3: Determine the minimal-norm element δ in the convex hull $\overline{U}$;
4: If $\delta = 0$ (or under a given tolerance) , stop ;
5: Else perform line search to determine the optimal step size ρ
6: Update solution X to $X - \rho\delta$

3. The proposed PR-MGDA path-relinking algorithm

3.1. Principle and motivation

Starting from a giving solution (e.g. Potential Pareto solution), the MGDA allows to converge to a stationary Pareto point following a direction of common descent constructed by linear combination of the gradients of the involved objectives. The major drawback of the MGDA algorithm is that it can easily be trapped on a local Pareto front because of Pareto stationarity which exhausts the common direction of descent as we approach a Pareto (local) front. Therefore this method is ineffective on MOPs presenting a multi-modality, as for the the gradient descent in the presence of multi-modality in the single-objective context.

On the other hand, if the starting solution is close to the Pareto front then the MGDA succeeds in reaching the FP. If we move back locally from this solution using a direction other than the direction of descent used by MGDA, one can obtain a new starting solution which is different from the first. Hence, by applying the MGDA to reach a new solution on the Pareto front, one can iterate the process again to carry out a path-relinking between any two Pareto solutions, as shown in Fig.1.

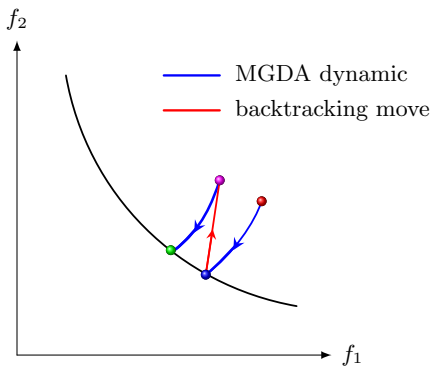


Figure 1: Alternation of MGDA/backtracking dynamic

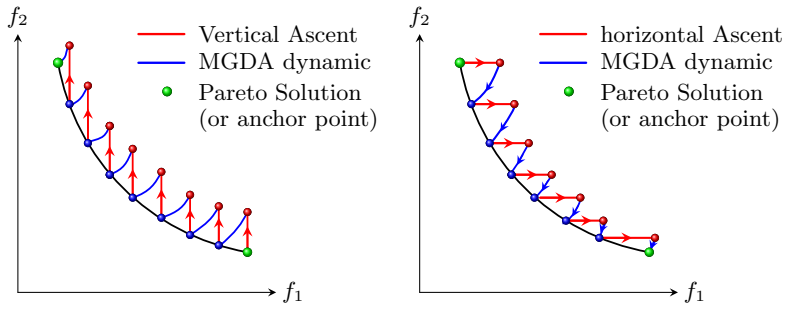


Figure 2: Illustration of Retro-MGDA mechanism

Our proposed path-relinking strategy is able to capture the Pareto front which connects any pair of Pareto solutions. It can be used as an intensification procedure for any multi-objective metaheuristic. To approximate the whole Pareto front, one can use the Anchor solutions as the starting Pareto solutions (Fig.2).

3.2. Our gradient-based path-relinking strategy

The path-relinking strategy is based on controlling moves on the objective space by using the gradient information. Consider a solution X such as $(\nabla f_1(X), \nabla f_2(X)) \neq (0,0)$. Let $\eta_i = \|\nabla f_i(X)\|$, and consider the normalized gradients: $\delta_i = \nabla f_i(X)/\eta_i$ Let $\alpha =< \delta_1, \delta_2 >$ and $t \in \mathbb{R}$. Consider the combined gradient direction δ_t built on the two gradients and defined as:

$$\delta_t = t\delta_1 + (1-t)\delta_2 \tag{5}$$

Due to the linear approximation, for a small enough step λ:

$$F(X + \lambda\delta_t) - F(X) \simeq \lambda\delta_t \nabla F(X) \tag{6}$$

But:
$$\delta_t \nabla F(X) = \delta_1(\eta_1\delta_1, \eta_2\delta_2)^T = (\eta_1\delta_t\delta_1, \eta_2\delta_t\delta_2)^T \tag{7}$$

Thus:
$$F(X + \lambda\delta_t) - F(X) \simeq \lambda(\eta_1\delta_t\delta_1, \eta_2\delta_t\delta_2)^T \tag{8}$$

In the objective space, we say that the displacement is vertical if:

$$f_1(X + \lambda\delta_t) - f_1(X) = 0. \tag{9}$$

Likewise, we say that the displacement is horizontal if:

$$f_2(X + \lambda\delta_t) - f_2(X) = 0 \tag{10}$$

Hence, to get locally a vertical displacement, one should move along δ_{t_v} such that $\eta_1\delta_t\delta_1 = 0$ or $\delta_t\delta_1 = 0$. But:

$$\delta_t\delta_1 = (t\delta_1 + (1-t)\delta_2)\delta_1 = t\delta_1^2 + (1-t)\delta_1\delta_2 = t + (1-t)\alpha \tag{11}$$

Thus δ_{t_v} corresponds to

$$t_v + (1 - t_v)\alpha = 0, \tag{12}$$

which means:

$$t_v = \frac{\alpha}{\alpha - 1} = \frac{< \delta_1, \delta_2 >}{< \delta_1, \delta_2 > -1} \tag{13}$$

Likewise, to get locally a horizontal displacement, one should move along δ_{t_h} such that $\eta_1 \delta_t \delta_1 = 0$ or $\delta_t \delta_1 = 0$. But:

$$\delta_t \delta_2 = (t\delta_1 + (1-t)\delta_2)\delta_2 = t\delta_1\delta_2 + (1-t)\delta_1^2 = t\alpha + 1 - t = t(\alpha - 1) + 1 \tag{14}$$

Thus δ_{t_h} corresponds to

$$1 + (1 - \alpha)t_h = 0, \tag{15}$$

which means:

$$t_h = \frac{1}{\alpha - 1} = \frac{1}{< \delta_1, \delta_2 > -1} \tag{16}$$

Besides the two directions δ_h and δ_v, we can also consider the transverse direction $\delta_\perp = \frac{1}{2}(\delta_2 - \delta_1)$ which has the particularity of being orthogonal to the direction of descent $\delta = \frac{1}{2}(\delta_1 + \delta_2)$. Indeed:

$$\delta \cdot \delta_\perp = \frac{1}{2}(\delta_2 - \delta_1) \cdot \frac{1}{2}(\delta_1 + \delta_2) = \frac{1}{4}(\|\delta_2\|^2 - \|\delta_1\|^2) = 0 \tag{17}$$

The proposed path-relinking gradient-based algorithm (PR-MGDA) is detailed below:

Algorithm 2 : **PR-MGDA** pseudo code

Input : F, λ, (MGDA stepsize), α (backtracking stepsize)
Output: RND : Approximation of the PF
determine the anchor points X_1^*, X_2^*, which represent the solutions of single-objective problems $X_i^* = \underset{X \in S,\ l_i \leq x_i \leq u_i}{Argmin} f_i(X)$ (see Fig. 2).
Set $X = X_2^*$ (or $X = X_1^*$ with horizontal backtracking scenario)
Set $X_p = X$; $Y = F(X)$; $Ys = F(X_1^*)$
Set $\alpha = 0.005 \times \|F(X_2^*) - F(X_1^*)\|$;
while $Y_1 > Ys_1$ **do**
    Compute the normalized gradients of the two objectives: $\delta_i = \nabla f_i(X)/\eta_i$
    Compute the vertical direction using equations (5,13): $\delta_t = t_v\delta_1 + (1 - t_v)\delta_2$.
    Normalize the vertical direction $\delta_t = \delta_t / \|\delta_t\|$
    Compute the transverse displacement according to : $X_d = X + \alpha\delta_\perp$
    ▷ We can also consider a variant with a vertical displacement involving δ_v
    Update the current solution X by applying the MGDA algorithm to X_d : $X_d = MGDA(X, \lambda)$
    Update the current objective: $Y = F(X)$
end while

4. Computational experiments

4.1. Parameters setting and performance indicators

The input of the PR-MGDA algorithm is a pair of any two potential Pareto solutions. In this paper, we have used the anchor solutions to approximate the whole Pareto front using path-relinking. Hence, one can use any single-objective algorithm to approximate the anchors points. In our experiments, we use a genetic algorithm (GA)[2] to handle this issue.

The step size of the MGDA move is set to $\lambda = 0.01$. This parameter configuration was adopted for all the experiments. The algorithm have been run on each test problem for 10 times.

In order to evaluate the performance of the proposed PR-MGDA algorithm, seven complementary test problems are selected from the literature: ZDT1, ZDT2, ZDT3, ZDT4, ZDT6, Pol and Kur. These problems are covering different type of difficulties and are selected to illustrate the capacity of the algorithm to handle diverse type of Pareto fronts. In fact, all these test problems have different levels of complexity in terms of convexity and continuity. For instance, the test problems KUR and ZDT3 have disconnected Pareto fronts; ZDT4 has too many local optimal Pareto solutions, whereas ZDT6 has non convex Pareto optimal front with low density of solutions near Pareto front.

Three performance measures were adopted in this study: the generational distance (GD) to evaluate the convergence, the Spacing (S) and the Spread (Δ) to evaluate the diversity and cardinality.

- The convergence metric (GD) measure the extent of convergence to the true Pareto front. It is defined as:

$$GD = \frac{1}{N}\sum_{i=1}^{N} d_i, \tag{18}$$

  where N is the number of solutions found and d_i is the Euclidean distance between each solution and its nearest point in the true Pareto front.

- The Spread Δ, beside measuring the regularity of the obtained solutions, also quantifies the extent of spread in relation to the true Pareto front. The Spread is defined as:

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N} |d_i - \overline{d}|}{d_f + d_l + d_i + (N-1)\overline{d}}. \tag{19}$$

  where d_i is the Euclidean distance between two consecutive solutions in the obtained set, d_f and d_l denotes the distance between the boundary solutions of the true Pareto front and the extreme solutions in the set of obtained solutions, $\overline{d}$ denotes the average of all distances d_i, $i = 1, 2, \cdots, N-1$ under assumption of N obtained non-dominated solutions.

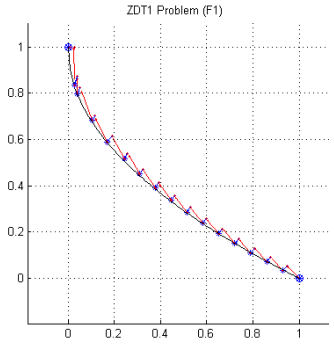
[2] Available in the yarpiz library www.yarpiz.com.

- The Spacing metric S indicates how the solutions of an obtained Pareto front are spaced with respect to each other. It is defined as:

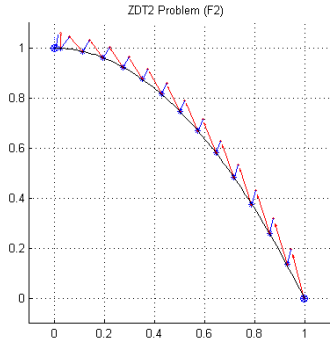
$$S = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(d_i - \overline{d})^2} \tag{20}$$

4.2. Computational results

The proposed algorithm PR-MGDA is compared with three popular evolutionary algorithms: MOEA/D [17], NSGA-II [18] and PESA-II [19] [3]. The obtained computational results are summarized in Table 1 in term of the mean and the standard deviation (Std) of the used metrics (GD, S, Δ), the average number of Pareto solutions found (NS), the average number of function evaluations (FEs), the average execution time in seconds (Time).

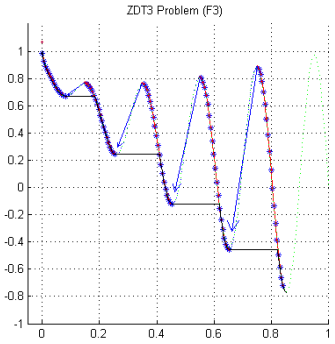


(a) *PF captured by PR-MGDA for ZDT1 problem*

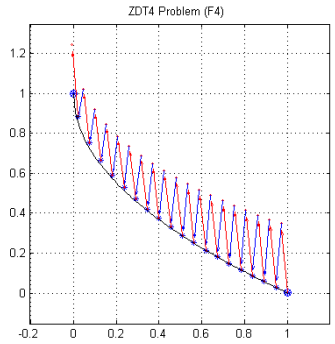


(b) *PF captured by PR-MGDA for ZDT2 problem*

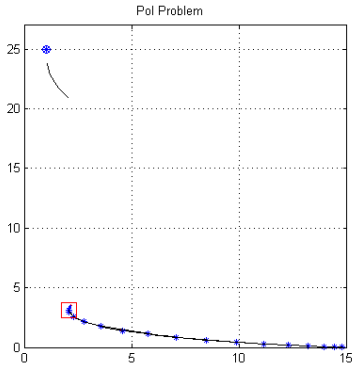
[3] MATLAB implementation obtained for the yarpiz library available at www.yarpiz.com.



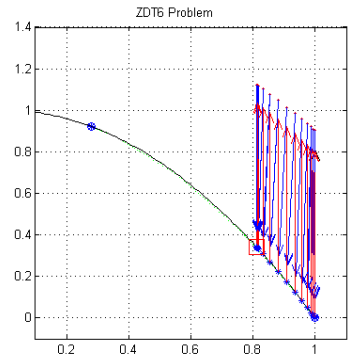
(c) *PF captured by PR-MGDA for ZDT3 problem*



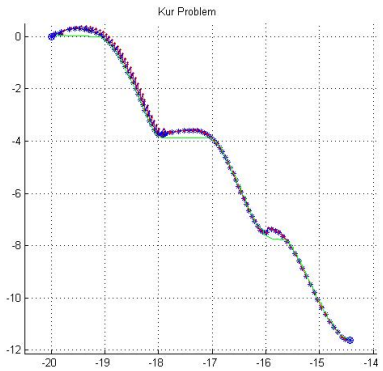
(d) *PF captured by PR-MGDA for ZDT4 problem*



(e) *PF captured by PR-MGDA for Pol problem*



(f) *PF partially captured by PR-MGDA for ZDT6 problem*



(g) *PF captured by PR-MGDA for Kur problem*

Table 1: Comparison of MOEA/D, NSGA-II, PESA-II and PR-MGDA for some considered test problems.

		GD		S		Δ			
Fct	Method	Mean	Std	Mean	Std	Mean	Std	NS	CPU(s)
	MOEA/D	2,81e-02	2,92e-02	2,42e-02	6,83e-03	9,95e-01	7,98e-02	100	86,67
ZDT1	NSGA-II	9,17e-02	1,03e-02	2,17e-02	1,56e-03	7,95e-01	2,86e-02	100	105,08
	PESA-II	5,38e-02	5,31e-03	3,73e-01	2,68e-02	8,82e-01	6,01e-02	100	40,97
	PR-MGDA	**8,36e-04**	**2,56e-07**	**9,16e-03**	**1,03e-06**	**7,52e-01**	**9,71e-07**	**116**	**9,58**
	MOEA/D	1,32e-01	4,55e-02	2,66e-02	3,87e-03	1,13e+00	8,99e-03	100	61,61
ZDT2	NSGA-II	1,49e-01	1,74e-02	1,65e-02	2,31e-03	9,10e-01	1,69e-02	100	113,18
	PESA-II	9,04e-02	3,42e-03	5,40e-01	4,61e-02	8,32e-01	2,73e-02	100	32,82
	RGDA	**2,65e-04**	**2,14e-05**	**3,34e-03**	**9,50e-05**	**7,34e-01**	**6,54e-06**	**115**	**10,61**
	MOEA/D	2,08e-02	8,00e-03	6,01e-02	1,52e-02	1,19e+00	3,43e-02	100	82,50
ZDT3	NSGA-II	6,27e-02	4,74e-03	3,75e-02	1,23e-02	8,25e-01	2,12e-02	100	112,91
	PESA-II	4,45e-02	3,42e-03	3,54e-01	3,22e-02	8,48e-01	1,04e-01	100	33,93
	PR-MGDA	**4,70e-03**	**3,34e-04**	**5,38e-02**	**6,26e-04**	**9,62e-01**	**1,65e-03**	**33**	**8,48**
	MOEA/D	8,28e-01	5,75e-01	1,70e-01	1,45e-01	1,09e+00	5,77e-02	100	62,65
ZDT4	NSGA-II	4,46e-01	1,33e-01	3,03e-01	1,79e-01	8,85e-01	9,56e-02	100	129,75
	PESA-II	1,12e+01	5,16e-01	2,72e+01	4,86e+00	1,11e+00	5,16e-02	100	18,00
	PR-MGDA	**1,03e-03**	**4,93e-04**	**1,11e-02**	**3,00e-03**	**7,52e-01**	**3,05e-04**	**60**	**8,54**
	MOEA/D	5,40e-01	1,08e-01	1,35e+00	1,01e+00	1,25e+00	1,54e-01	100	91,11
Pol	NSGA-II	2,48e+00	5,84e-02	1,75e+00	2,09e-03	9,72e-01	3,55e-03	100	123,86
	PESA-II	1,52e+01	6,02e+00	1,01e+01	1,16e+00	9,77e-01	9,85e-02	100	45,64
	PR-MGDA	**1,48e-03**	**9,34e-06**	**6,03e-01**	**2,13e-02**	**9,59e-01**	**3,70e-08**	**22**	**5,24**
	MOEA/D	4,22e-01	1,18e-01	4,63e-02	2,02e-02	1,09e+00	5,98e-02	100	57,86
ZDT6	NSGA-II	3,08e-01	2,38e-02	1,57e-01	4,15e-02	8,73e-01	2,93e-02	100	121,89
	PESA-II	4,42e-01	4,69e-03	2,00e+00	6,31e-01	1,03e+00	4,97e-02	100	29,17
	PR-MGDA	**7,83e-03**	**6,31e-03**	**1,04e-01**	**4,14e-03**	**7,20e-01**	**2,05e-02**	**35**	**5,65**
	MOEA/D	5,51e-03	5,51e-03	6,76e-01	4,35e-01	1,42e+00	1,80e-01	100	80,83
Kur	NSGA-II	5,31e-04	2,64e-05	1,22e-01	4,15e-03	4,10e-01	3,10e-02	100	117,86
	PESA-II	1,78e-01	1,28e-02	4,15e+00	4,86e-01	9,04e-01	5,75e-02	100	33,76
	PR-MGDA	**3,71e-03**	**3,05e-04**	**1,50e-01**	**8,56e-03**	**9,03e-01**	**9,92e-06**	**24**	**12,02**

In terms of the quality of the Pareto front approximation, by analyzing the *GD* metric statistics, we can see that the proposed PR-MGDA is well converging to the true Pareto front for all tested problems. Furthermore, the Spacing and Spread measures indicate that the proposed PR-MGDA has the ability of generating uniform and diverse solutions. Except for the ZDT6 problem whose PF presents folds (Pareto many-to-one problems) [20] which leads to the presence of several points with a zero gradient according to one of the two objectives. In this case the PR-MGDA algorithm can only get a partial capture of the FP. To prevent this problem we have been led to slightly shift the points where this occurs, which has partially overcome this problem. Indeed, despite this precaution, it happens that the PR-MGDA does not capture the whole of the FP for this problem (Fig.3(f)).

In terms of the execution time, the proposed PR-MGDA largely outperforms MOEAs. For instance, the PR-MGDA is at least 10 times faster than NSGA-II, even including the time to generate the anchor solutions. Note that all the

simulations for the considered problems did not exceed 70,000 FEs of which 50,000 FEs were reserved for the resolution of the anchors points. However, the number of solutions obtained strongly depends on the step of the retreat which was previously set to 0.5% of the difference D_a between the two anchor solutions. Thus, if we decide to reduce this step for example to 0.1% of D_a, we can easily multiply the number of solutions obtained while respecting the limit of the FEs imposed for the comparison.

5. Conclusion and perspectives

In this paper, we have proposed a Gradient-based approach for path-relinking in bi-objective non-constrained continuous optimization problems. The algorithms is based on the MGDA approach combined to a backtracking approach.

The computational results for many test functions were compared with popular MOEAs: MOEA/D, NSGA-II and PESA-II. The results indicate that the PR-MGDA algorithm outperforms the algorithms both in quality of the approximations and specially in the execution time.

Our proposed approach can be integrated in any multi-objective optimization algorithm as an intensification search procedure. Indeed, PR-MGDA can be applied to any pair of solutions in order to generate evenly solutions between them. We will also investigate an extension of our PR-MGDA approach to many-objective optimization problems. Moreover, we are working strategies to handle constrained multi-objective optimization problems using projected gradient descent.

References

[1] I. Das and J. Dennis. Normal-boundary intersection: a new method for generating pareto optimal point in nonlinear multicriteria optimization problems. siam journal on optimization. *Journal of Glob Optim*, 3(8):631–657, 1998.

[2] A. Messac and C. Mattson. Normal constraint method with guarantee of even representation of complete pareto frontier. *AIAA journal*, 42(10):2101–2111, 2004.

[3] A. Messac. Physical programming-effective optimization for computational design. *AIAA journal*, 34(1):149–158, 1996.

[4] K. Deb. Nonlinear goal programming using multi-objective genetic algorithms. *Journal of the Operational Research Society*, 3(52):291–302, 2001.

[5] Y. Haimes. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man, and Cybernetics*, 1(3):296–297, 1971.

[6] S. Utyuzhnikov, P. Fantini, and M. Guenov. Numerical method for generating the entire pareto frontier in multiobjective optimization. In *Proceedings of EUROGEN*, pages 12–14, 2005.

[7] S. Utyuzhnikov, P. Fantini, and M. Guenov. A method for generating a well-distributed pareto set in nonlinear multiobjective optimization. *Journal of Computational and Applied Mathematics*, 223(2):820–841, 2009.

[8] O. Schütze, C. Coello, S. Mostaghim, E-G. Talbi, and M. Dellnitz. Hybridizing evolutionary strategies with continuation methods for solving multi-objective problems. *Engineering Optimization*, 40(5):383–402, 2008.

[9] K. Harada, J. Sakuma, and S. Kobayashi. Local search for multiobjective function optimization:pareto descent method. In *Genetic and Evolutionary Computation Conference GECCO*, number 659-666, 755-762, 2006.

[10] H. J. Kim and M. S. Liou. New multi-objective genetic algorithms for diversity and convergence enhancement. *AIAA journal*, 2009.

[11] A. Lara, G. Sanchez, C. Coello, and O. Schutze. Hcs: A new local search strategy for memetic multiobjective evolutionary algorithms. volume 14, pages 112–132, 2009.

[12] J-A. Desideri. Mutiple-gradient descent algorithm for multiobjective optimization. In *J. Berhardsteiner, editor, European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS)*, 2012.

[13] J. Fliege and B. F. Svaiter. Steepest descent methods for multicriteria optimization. *Mathematical Methods of Operations Research*, 51(3):479–494, 2000.

[14] S. Schäffler, R. Schultz, and K. Weinzierl. Stochastic method for the solution of unconstrained vector optimization problems. *Journal of Optimization Theory and Applications*, 114(1):209–222, 2002.

[15] P. Bosman. On gradients and hybrid evolutionary algorithms. In *IEEE Transactions on Evolutionary Computation*, volume 1, pages 51–69, 2012.

[16] J-A. Désidéri. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5-6):313–318, 2012.

[17] Q. Zhang and H. Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731, 2007.

[18] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[19] D. Corne, N. Jerram, J. Knowles, and M. Oates. Pesa-ii: Region-based selection in evolutionary multiobjective optimization. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, pages 283–290, 2001.

[20] S. Huband, P. F. Hingston, L. Barone, and L. While. A review of multiobjective test problems and a scalable test problem toolkit. In *IEEE Transactions on Evolutionary Computation*, volume 5, pages 477–506, 2006.

Lift rise spots exploration of a yaw-wise rotational flap by enhanced-Kriging-based efficient global optimization

K. Chiba[1] and M. Kanazaki[2]

[1] The University of Electro-Communications, Tokyo, Japan
kazchiba@uec.ac.jp
[2] Tokyo Metropolitan University, Tokyo, Japan
kana@tmu.ac.jp

1 Introduction

Makers design commercial airplanes in a form optimized for cruise conditions; therefore, they lack the lift during takeoff and landing. Many research and developments have been implemented on high-lift devices to solve this problem [1]. Our previous study advocated the idea of adding a yaw-wise rotational degree of freedom to a trailing-edge slotted flap to raise the lift in its landing condition without appending any new devices and proved that the yaw-wise rotation grows the lift [2]. The previous research conducted only a parametric study for five positions for one variable of the yaw-wise rotation angle; it did not wholly explore the flap position. Hence, by raising the number of parameters that determine the flap position and exploiting an optimization method to probe the design space exhaustively, we will pursue the potential of the simple idea of adding yaw-wise rotational degrees of freedom to the flap to gain the lift further.

2 Problem settings

Reference [2] provides details of the target geometry. The objective function is to maximize the lift coefficient C_L of the computational model shown in Fig. 1. We ponder two design variables: the yaw-wise rotational angle ψ (defined in Fig. 2) and the flap gap between the wing and the flap (height in the z-direction) $\mathrm{gap}_{\mathrm{flap}}$. Initially, the overlap and flap deflection angle would also be design variables. However, due to this problem's time-consuming nature, we would rise with a small number of design variables and use them as a touchstone to discern if we can expand the issue to a large-scale one. The applicable range of each design variables is $-4.0 \leq \psi^{\circ} \leq +4.0$ and $0.01c \leq \mathrm{gap}_{\mathrm{flap}} \leq 0.03c$, where c denotes the chord length of the wing.

3 Evaluation system

Metaheuristics are suitable for engineering product design optimizations with extreme nonlinearity, such as fluid phenomena, because they need no gradient information for objective functions. Real-world design problems often use global design space search approaches descending from metaheuristics such as evolutionary computations and swarm optimizations. These optimizers enable global explorations for nondominated solutions that are candidates of compromise solutions for large-scale design problems with multiple objectives and many design variables. They likewise can play a role in creating an efficient database for obtaining various design knowledge. However, metaheuristics requires at least $\mathcal{O}(10^4)$ evaluations before convergence. Time-consuming to assess objectives by computational fluid dynamics is a severe bottleneck in fluid machinery design. Hence, problems involving fluid evaluation cannot generally converge in a realistic period.

In contrast, some researchers proposed surrogate optimizations using the Kriging model [3]/the radial basis function [4]. Small-scale design problems, which have the number of design variables $\mathcal{O}(10^0)$, can diminish the evaluation frequency significantly than with metaheuristics. The Kriging model uses no problem-dependent parameters; past studies have provided derivative Kriging models as a general-purpose method [5]. The original Kriging model appends only one additional sample point. However, our Kriging model is an enhanced algorithm that permits an arbitrary number of sample points to be augmented based on expected improvement (EI) values on the approximate

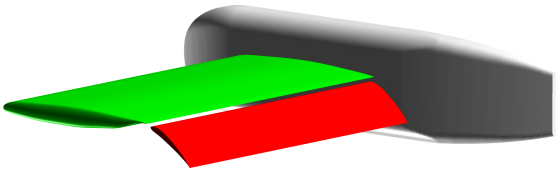


Fig. 1. Configuration of the computational model consisting of a fuselage, a wing, and a single slotted flap. This figure visualizes the flap at the neutral condition without any yaw-wise rotation.

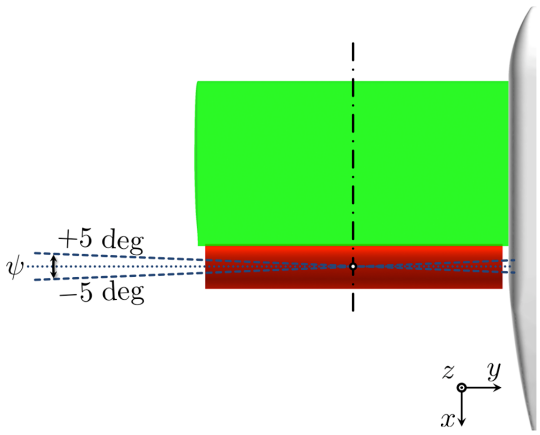


Fig. 2. The definition of yaw-wise rotation angle ψ and its acceptable range on top view. The figure depicts the wing translucently to see the overlap of the wing for the flap.

surface [6]. This study set the number of samples added at one time to 3, considering the C_L evaluation time.

We use a compressible flow solver FaSTAR [7], which solves the 3D Raynolds-averaged Navier-Stokes equations by an unstructured MUSCL-type cell-vertex finite volume method [8] with Hishida's differentiable limiter for keeping second-order spatial accuracy. FaSTAR utilizes schemes: Harten-Lax-van Leer-Einfeldt-Wada method [9] for numerical flux computations; lower-upper symmetric Gauss-Seidel implicit method [10] for time integration. This study adopts the explicit algebraic Reynolds stress model [11] as a turbulence model because [12] stated being suitable for estimating high lift devices' transonic aerodynamic performance. Computations run on the part of the cluster with Intel Xeon E5-2670v2 2.50 GHz in our laboratory. It takes approximately 100 hours to evaluate one individual in 100 parallel runs.

4 Result

We have evaluated the ten initial and the first three additional sample points; we are assessing the second extra sample point. Figure 3 depicts the response surface created from the results procured so far. As for $\mathrm{gap}_{\mathrm{flap}}$, the narrower the passage, the more C_L tends to increase; we can physically explain this by a simple mechanism: narrowing $\mathrm{gap}_{\mathrm{flap}}$ grows the pressure and accelerates the flow, raising the pressure on the upper surface of the flap and the wing.

In contrast, ψ is asymmetric for 0°; we can discuss ψ quantitatively that can improve C_L over 0°. Moreover, there seems to be a multi-modal response to 0° on both sides. We would further consider the physics of this point in future results.

5 Conclusion

This research explored appropriate spots of a trailing-edge slotted flap with a yaw-wise rotational degree of freedom to grow the lift in an airplane's landing configuration without affixing new devices. Our previous research conducted a parametric study on a single variable: yaw-wise rotation angle, and found that yaw-wise rotation gains lift. Therefore, we would expand the number of parameters that fix the flap position and penetrate the potential of the simple idea of adding yaw-wise rotational degrees of freedom to the flap to improve the lift further. Hence, we define a two-design-variable problem by adding the flap gap to the yaw-wise rotational angle as design variables. Since the number of design variables is small, this study adopted enhanced Kriging-based global optimization and efficiently explored design space. The increase in the degree of freedom of positioning has enabled the lift's further growth and specified the physical mechanism. Therefore,

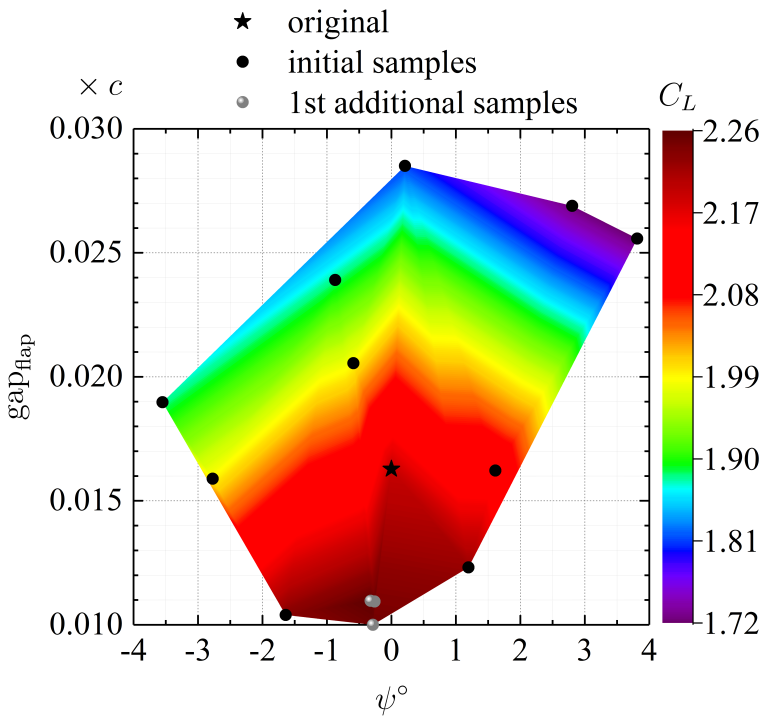


Fig. 3. response surface.

we would conduct a four-design-variable problem, including the flap overlap and the flap angle, to reveal the relationship between the lift performance and the relative position between the main wing and the slotted flap.

References

1. van Dam, C.P.: The aerodynamic design of multi-element high-lift systems for transport airplanes. Progress in Aerospace Science **38** (2002) 101–144
2. Chiba, K., Komatsu, T., Ito, T.: Aerodynamic efficacy of adding yaw-wise rotational degree of freedom to an airplane flap. Aerospace Systems **3** (2020) 207–217
3. Simpson, T.W., Mauery, T.M., Korte, J.J., Mistree, F.: Kriging models for global approximation in simulation-based multidisciplinary design optimization. AIAA Journal **39** (2001) 2233–2241
4. Buhmann, M.D.: Radial basis functions. Acta Numerica **9** (2000) 1–38
5. Kleijnen, J.P.C.: Kriging metamodeling in simulation: A review. European Journal of Operational Research **192** (2009) 707–716
6. Kanazaki, M., Imamura, T., Matsuno, T., Chiba, K.: Multiple additional sampling by expected improvement maximization in efficient global optimization for real-world design problems. In G. Leu, H. Singh, S.E., ed.: Intelligent and Evolutionary Systems. Volume 8 of Proceedings in Adaptation, Learning and Optimization., Springer International Publishing AG (2017) 183–194
7. Hashimoto, A., Murakami, K., Aoyama, T., Ishiko, K., Hishida, M., Sakashita, M., Lahur, P.R.: Toward the fastest unstructured CFD code "FaSTAR". AIAA Paper 2012-1075 (2012)
8. Burg, C.: Higher order variable extrapolation for unstructured finite volume RANS flow solvers. AIAA Paper 2005-4999 (2005)
9. Obayashi, S., Guruswamy, G.P.: Convergence acceleration of an aeroelastic Navier-Stokes solver. AIAA Journal **33** (1994) 1134–1141
10. Sharov, D., Nakahashi, K.: Reordering of hybrid unstructured grids for Lower-Upper Symmetric Gauss-Seidel computations. AIAA Journal **36** (1998) 484–486
11. Wallin, S., Johansson, A.V.: An explicit algebraic Reynolds stress model for incompressible and compressible turbulent flows. Journal of Fluid Mechanics **403** (2000) 89–132
12. Hellsten, A.K.: New advanced k-ω turbulence model for high-lift aerodynamics. AIAA Journal **43** (2005) 1857–1869

Space Partitioning with multiple models for Parallel Bayesian Optimization

Maxime Gobert[1], Jan Gmys[2], Nouredine Melab[3] and Daniel Tuyttens[1]

[1] Mathematics and Operations Research Department, University of Mons, Belgium
{maxime.gobert,daniel.tuyttens}@umons.ac.be
[2] Inria Lille - Nord Europe, France
jan.gmys@univ-lille.fr
[3] Univ. Lille, CNRS, Centrale Lille, Inria, UMR 9189 - CRIStAL - Centre de Recherche en Informatique Signal et Automatique de Lille, F-59000 Lille, France
nouredine.melab@univ-lille.fr

1 Context and motivations

Bayesian Optimization (BO) is a global optimization framework that uses bayesian surrogate models such as Gaussian Processes (GP) to address black-box problems [1], [2] with costly-to-evaluate objective functions. Bayesian models and especially GPs are attractive for their ability to provide the uncertainty over their predictions. Using this information, one can build an indicator of utility for a point to be simulated. This indicator, named Infill Criterion (IC) or Acquisition Function (AF), is used to guide the optimization process and find valuable new point(s) to be exactly evaluated. Based on this procedure, Jones *et al.* [3] introduce the Efficient Global Optimization (EGO) algorithm that uses the Expected Improvement (EI) AF. Many approaches emerged from the idea of making EGO parallel. In particular, Ginsbourger *et al.* [4] introduced the q-dimensional EI criterion (q-EI), able to provide q candidate points when optimized. In [5], Ginsbourger *et al.* write that even though q-EI and its optimization methods are mathematically founded, it is not "mathematically tractable beyond a few dimensions". This motivates the introduction of the Kriging Believer (KB) and Constant Liar (CL) heuristics also presented in [5]. The two heuristics allow to approximate the optimization of q-EI at a much lower time cost. EGO using q-EI is called q-EGO in the following. Many other approaches based on EI or q-EI are constructed. For example, in [6] the authors use Infinitesimal Perturbation Analysis (IPA) to construct a gradient estimator of the q-EI surface, and a multi-start heuristic to find the set of points to evaluate. Zhan *et al.* [7] use a niching strategy to locate several optimal areas of the single point EI. Marmin *et al.* [8] write the analytical form of the multi-point EI gradient to be able to optimize the function with gradient information and reduce the computational cost compared to sequential heuristics or Monte Carlo sampling of [5]. However, none of the methods are efficient on parallel architectures including dozens of processing units. The creation of the batch of candidates is often time consuming, and the GP model fitting cost increases fast. The reference q-EGO algorithm has been experimented and driven to its limits in Briffoteaux *et al.* [9]. The analysis revealed that q-EGO performs well with small budgets (i.e. number of calls to the objective function) and small batches ($q \leq 8$). However, it suffers from early stagnation, poor scalability, and budget seems misspent since increasing q does not necessarily improve the final target for a given number of algorithm iterations (called cycles). The limits of q-EGO comes from at least two aspects: (1) we need to update q times the model to provide q candidate and the Kriging model becomes extremely time consuming to fit as the size of the learning set is increased; (2) the way the candidates are selected is not suited for large batches.

2 The proposed approach

From the previous assessment, our approach called Binary Space Partitioning EGO (BSP-EGO) revisits the way the candidates are selected - i.e. the Acquisition Process (AP). BSP-EGO uses EI as it is known to work consistently well on various problems. To preserve diversity in the candidate selection process BSP-EGO recursively splits the search space $\mathcal{D}$ into sub-domains. The partition can be seen following a binary tree where each node is a sub-domain and the leaves form a cover of $\mathcal{D}$ without overlap. One sub-AP is performed for each leaf using the same surrogate model fitted

Algorithm 1 multi-model BSP-EGO pseudo-code

Input
f: function to optimize, $\mathcal{D}$: search space
n_{init}: initial sample size, n_{cycle}: number of cycles, n_{learn}: learning size
depth: tree depth

1: $\boldsymbol{X},\ \boldsymbol{y} \leftarrow$ initial_sampling$(\mathcal{D}, n_{init}, f)$
2: $\mathcal{T} \leftarrow$ build_tree(*depth*)
3: **for** i in $1 : n_{cycle}$ **do**
4: $\quad\mathcal{B}_{candidates} \leftarrow \varnothing$
5: $\quad$**for** *leaf* in $\mathcal{T}$ **do** ▷ parallel loop
6: $\quad\quad\boldsymbol{X}_{leaf}, \boldsymbol{y}_{leaf} \leftarrow gather\_points(leaf, n_{learn})$
7: $\quad\quad model_{leaf} \leftarrow$ learn_model$(\boldsymbol{X}_{leaf}, \boldsymbol{y}_{leaf})$
8: $\quad\quad\boldsymbol{c} \leftarrow$ find_best_candidate$(\mathcal{D}_{leaf}, model_{leaf}, Infill_criterion)$
9: $\quad\quad\mathcal{B}_{candidates} \leftarrow \mathcal{B}_{candidates} \cup \boldsymbol{c}$
10: $\quad$**end for**
11: $\quad\mathcal{B}_{candidates} \leftarrow$ selection$(\mathcal{B}_{candidates})$
12: $\quad\mathcal{T} \leftarrow$ update_tree$(\mathcal{T}, \mathcal{B}_{candidates})$
13: $\quad\boldsymbol{X} \leftarrow \boldsymbol{X} \cup \mathcal{B}_{candidates}$
14: $\quad\boldsymbol{y} \leftarrow \boldsymbol{y} \cup f(\mathcal{B}_{candidates})$ ▷ parallel evaluation
15: **end for**
16: **return** $\boldsymbol{x}^*,\ y^*$

over all known points, and the best candidates regarding the chosen AF are selected. Then, the partition evolves according to the performance of each sub-domain. If the sub-domain is attractive (in terms of AF), it is split once more while the *worst* is merged to maintain a fixed number of sub-domains. The full description is available in [10]. BSP-EGO is designed to improve parallelism by providing an arbitrary big number of candidates from its AP, and to improve the balance in exploration/exploitation. Indeed, being able to dig deep into a sub-domain allows to sample more this area without neglecting others by forcing the sampling in sub-domains of *a priori* poor interest. Results of [10] show that even though we are able to increase the degree of parallelism, and improve the optimization in restricted time, the fast increasing time of GP fitting is still a bottleneck. Actually, the database is filling even faster as the number of cores increases since n_{batch} points are added per cycle.

This work investigates the use of multiple surrogate models (on per sub-domain) to limit the size of the learning set and lower the computational cost. For each sub-domain, the closest n_{learn} points the closest from the center of the sub-domain are collected to fit a GP model over them. Besides considerably lowering the fitting cost, this approach has several benefits. First, all models are distinct and independent, favoring diversity as experienced in Villanueva *et al.* [11] where space partitions and distinct surrogate models in sub-regions are used to locate several local optima. Second, the whole AP can be performed in parallel, meaning that each worker performs one fast GP fitting. Moreover, search space partitioning is reported as a viable method to deal with non stationary objective functions [12][13]. Algorithm 1 displays the pseudo-code of the multi-model BSP-EGO. It starts with the creation of an initial learning set, and the initialization of the tree $\mathcal{T}$. Then, until the budget runs out, BSP-EGO goes to every leaves (line 6 of Alg. 1) to create a GP model. Note that if the total number of points is smaller than n_{learn}, the model is fitted with all available points. Given an infill criterion, the local AP proposes a candidate for the considered leaf. Finally, a sub-set of candidates is exactly evaluated to be incorporated in the learning set.

3 Results and conclusions

Preliminary experiments have been conducted on three 6-dimensions benchmark functions to measure the walltime and solution quality as a function of the batch size n_{batch} and the size of the learning set n_{learn}. The functions are chosen with very distinct landscapes: the rosenbrock function is regular and unimodal with a large valley, the ackley function is noisy and multimodal, and the alpine function is highly multimodal but not noisy. The two firsts are optimized on the domain $\mathcal{D} = [-32, 32]^6$ and the last one on $\mathcal{D} = [0, 10]^6$. Experiments are performed on Lille site of the

Grid'5000 testbed [14]. Up to 4 nodes using 2 Intel Xeon E5-2630 v4 of 10 cores each are used. For a total of 48 cycles allocated to each experiment (whatever the batch size) we observe that the multi-model approach total execution time does not increase much with the batch size, which indicates a good scalability. With a learning size of 128, and a batch size of 32, this approach is up to 300 times faster compared to the global model one with only an acceptable reduction of the final target quality. To achieve equivalent results in terms of solution quality, we can use a more precise model learnt over 256 points, still with $n_{batch} = 32$, and have a speed up of approximately 54. This approach also makes possible the use of more than 64 computing cores that increases considerably the global evaluation budget (and the final objective value) in a restricted time. For example, running 48 cycles with $n_{batch} = 64$ and $n_{learn} = 128$ takes less than 3 minutes for non-expensive benchmark functions evaluations, which is barely more than the same experiment with $n_{batch} = 8, 16, 32$. The partitioning approach also seems to perform well when the landscape to optimize presents many local optima and needs more exploration. Indeed, each model focuses on its specific sub-region and is not influenced by data of distant regions. On the other side, it makes it difficult to compare the value of the infill criteria coming from different models and can have a negative effect when exploitation must be favored. The BSP-EGO with local-models algorithm is a promising step towards scalable EGO-like approaches. However, increasing the degree of parallelism, the second problem is even more present: how can one take advantage of this scalability to improve the candidate selection procedure (i.e. the AP) ? Recent developments in parallel BO (batch BO) seems to indicate that using different infill criteria working cooperatively is promising. Whereas the local-models BSP-EGO algorithm needs extensive testing and tuning, its framework is highly compatible with the idea of multiple AF operating cooperatively to select a good batch of candidates to exactly evaluate.

References

1. H. J. Kushner. A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise. *Journal of Fluids Engineering*, 86(1):97–106, 03 1964.
2. J. Močkus. On bayesian methods for seeking the extremum. In G. I. Marchuk, editor, *Optimization Techniques IFIP Technical Conference Novosibirsk, July 1–7, 1974*, pages 400–404, Berlin, Heidelberg, 1975. Springer Berlin Heidelberg.
3. Donald Jones. A taxonomy of global optimization methods based on response surfaces. *J. of Global Optimization*, 21:345–383, 12 2001.
4. David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. A multi-points criterion for deterministic parallel global optimization based on kriging. 03 2008.
5. David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. Kriging is well-suited to parallelize optimization. 2010.
6. Jialei Wang, Scott Clark, Eric Liu, and Peter Frazier. Parallel bayesian global optimization of expensive functions. 02 2016.
7. Dawei Zhan, Jiachang Qian, and Yuansheng Cheng. Balancing global and local search in parallel efficient global optimization algorithms. *Journal of Global Optimization*, 67(4):873–892, Apr 2017.
8. Sébastien Marmin, Clément Chevalier, and David Ginsbourger. Differentiating the multipoint expected improvement for optimal batch design. pages 37–48, 2015.
9. Guillaume Briffoteaux, Maxime Gobert, Romain Ragonnet, Jan Gmys, Mohand Mezmaz, Nouredine Melab, and Daniel Tuyttens. Parallel surrogate-assisted optimization: Batched bayesian neural network-assisted ga versus q-ego. *Swarm and Evolutionary Computation*, 57:100717, 2020.
10. Maxime Gobert, Jan Gmys, Nouredine Melab, and Daniel Tuyttens. Adaptive Space Partitioning for Parallel Bayesian Optimization. working paper or preprint, January 2021.
11. Diane Villanueva, Rodolphe Le Riche, Gauthier Picard, and Raphael Haftka. Dynamic design space partitioning for optimization of an integrated thermal protection system. 04 2013.
12. B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, Jan 2016.
13. Robert B Gramacy and Herbert K H Lee. Bayesian treed gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, 103(483):1119–1130, 2008.
14. Daniel Balouek, Alexandra Carpen Amarie, Ghislain Charrier, Frédéric Desprez, Emmanuel Jeannot, Emmanuel Jeanvoine, Adrien Lèbre, David Margery, Nicolas Niclausse, Lucas Nussbaum, Olivier Richard, Christian Pérez, Flavien Quesnel, Cyril Rohr, and Luc Sarzyniec. Adding virtualization capabilities to the Grid'5000 testbed. In Ivan I. Ivanov, Marten van Sinderen, Frank Leymann, and Tony Shan, editors, *Cloud Computing and Services Science*, volume 367 of *Communications in Computer and Information Science*, pages 3–20. Springer International Publishing, 2013.

Comparing local search initialization for k-means and k-medoids clustering in a planar Pareto Front, a computational study

Jiangnan Huang[1], Zixi Chen[1], Nicolas Dupin[1]

Université Paris-Saclay, 91405, Orsay, France
{jiangnan.huang}{zixi.chen},{nicolas.dupin}@universite-paris-saclay.fr

Abstract. Having N points in a planar Pareto Front (2D PF), k-means and k-medoids are solved by dynamic programming algorithms running in $O(N^3)$ time. Standard local search approaches, PAM and Lloyd's heuristics, are investigated in the 2D PF case to solve faster large instances. Specific initialization strategies related to the 2D PF case are implemented with the generic ones (Forgy's, Hartigans, k-means ++). The local minimums obtained with different initialization algorithms after the PAM and Lloyd's iteration are compared to the optimal values. This study highlights that local minimums of a poor quality exist in the 2D PF case. A parallel or multi-start heuristic using four initialization strategies avoids accurately poor local optimums.

Keywords : Clustering algorithms ; K-means; K-medoids; Heuristics ; Local Search; bi-objective optimization ; Pareto front

1 Introduction

K-means clustering is one of the most famous unsupervised learning problem, and is widely studied in the literature [12, 14]. The k-medoids problem, the discrete variant of the k-means problem, maximizes the dissimilarity around a representative solution [13]. If k-medoids clustering is more combinatorial than k-means clustering, it is known to be more robust on noises and outliers [12]. Both k-medoids and k-means problem are NP hard in the general and the planar case, [11, 15]. One dimensional (1D) case of k-means is solvable in polynomial time, with Dynamic Programming (DP) algorithms [19, 9].

Facing the NP-hard complexity, a seminal heuristic to solve k-means problems was provided by Lloyd in [14], it is a steepest descent heuristic converging to local minimums. A careful initialization has also an important impact to solve k-means, many initialization strategies were proposed and discussed [1, 8, 10, 12]. A comparative analysis was provided for general instances of k-means to analyze the impact and efficiency of initialization strategies [2] Lloyd's local search was extended to solve heuristically k-medoids problems, this adaptation is named PAM (Partitioning Around Medoids) [13]. Comparing empirically the efficiency of initialization strategies for PAM local search is also of interest for k-medoids.

The specific case of k-means and k-medoids clustering in a planar Pareto Front (2D PF) holds for an application to bi-objective optimization problems and algorithms [4, 7]. In both cases, there is a polynomial complexity with an exact DP algorithm running in $O(N^3)$ time [4, 7]. However, this cubic complexity may be a bottleneck to solve quickly large instances, which may be required for the real-world application. PAM and Lloyd's local search algorithms are still useful. Similarly with [2, 17], this paper aims to compare the impact and efficiency of local search initialization, in the case of a 2D PF instances. Contrary to [2, 17], local minimums can be compared to known optimal values thanks to the DP exact algorithm. Another open question is to determine if the 2D PF case induce also good properties for PAM and Lloyd's local search heuristics.

This paper is organized as follows. In section 2, we define formally the problem and fix the notation. In section 3, initialization heuristics are presented. In section 4, the experimental conditions are presented before describing and analyzing the computational results. In section 5, our contributions are summarized, discussing also future directions of research.

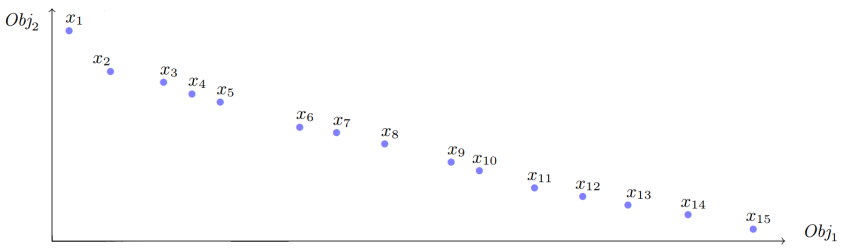


Fig. 1. Illustration of a 2d Pareto Front and its indexation

2 Problem statement and notations

We suppose in this paper having a set $E = \{x_1, \dots, x_N\}$ of N elements of $\mathbb{R}^2$ in a 2D PF. As illustrated in Figure 1, the 2D PF E can be re-indexed such that $E = \{x_k = (y_k, z_k)\}_{k \in [\![1,N]\!]}$ such that $k \in [\![1, N]\!] \mapsto x_k$ is increasing and $k \in [\![1, N]\!] \mapsto y_k$ is decreasing, it is proven formally in [4]. Defining the binary relation $y = (y^1, y^2) \prec z = (z^1, z^2)$ with $y^1 < z^1$ and $y^2 > z^2$, $\prec$ is a total order in E and $x_1 \prec \cdots \prec x_N$.

We define $\Pi_K(E)$, as the set of all the possible partitions of E in K subsets. Defining a cost function f for each subset of E to measure the dissimilarity, K-sum-clustering problems are combinatorial optimization problems, minimizing the sum of the measure f for all the K partitioning clusters:

$$\min_{\pi \in \Pi_K(E)} \sum_{P \in \pi} f(P) \tag{1}$$

K-medoids and *K-means* problems are in the shape of (1). K-medoids problem considers the minimal sum of the squared distances from one point of P, the *medoid*, to the other points of P:

$$\forall P \subset E, \quad f_{medoids}(P) = \min_{c \in P} \sum_{x \in P} \|x - c\|^2 \tag{2}$$

Note that $\|$ denotes the standard Euclidean norm in this paper, $|$ is used to denote the cardinal of a subset of E. K-means clustering considers the sum of the squared distances from any point of P to the *centroid*:

$$\forall P \subset E, \quad f_{means}(P) = \min_{c \in \mathbb{R}^2} \sum_{x \in P} \|x - c\|^2 = \sum_{x \in p} \left\| x - \frac{1}{|p|} \sum_{y \in p} y \right\|^2 \tag{3}$$

The first equality in (3) shows the similarity with k-medoids, k-medoids is the discrete version of k-medoids. The second equality in (3), proven with convex optimization in [16], is used in this paper to compute $f_{means}(P)$ in $O(|P|)$ time, whereas $f_{medoids}(P)$ is computed in $O(|P|^2)$ time using (2). This difference is crucial in the computational efficiency to solve k-means and k-medoids.

PAM and Lloyd's algorithm are similar local search algorithms iterating over solutions encoded as partitioning subsets $P_1, \ldots, P_K$ and their respective centroids $c_1, \ldots, c_K$ (or medoids for PAM). Having a current solution encoded like previously, two steps are processed to improve the solution:

- The partitioning subsets $P_1, \ldots, P_K$ are modified, assigning each point to the closest centroid or medoid c_i. This step runs in $O(NK)$ time.
- The centroids or medoids c_i are recomputed with formulas (3) or (2) considering the updated partitioning subsets $P_1, \ldots, P_K$. This step runs in $O(N)$ time for k-means in $O(N^2)$ time for k-medoids.

Each of the previous step improves the current clustering solution if any modification is operated. Else, the current solution is a *local minimum* related to PAM or Lloyd's local search, that can be different from a *global minimum*, optimal for optimization problems (1). The initial solution given to the local searches has an influence to the quality of local minimums [2]. In the following we compare the efficiency of specific and generic initialization strategies for the 2D PF case. Considering the 2D PF case allows have the value of optimal solutions thanks to the specific DP algorithm, event for some large instances [4, 7].

3 Initialization heuristics

In this section, we introduce an overview of the different initialization heuristics which are implemented in the following computational experiments, distinguishing the generic ones which are commonly used for any clustering instances, to specific strategies related to specific properties of the 2D PF case. Initialization

strategies were more studied for the k-means problem. We adapt such strategies for the k-medoids problem defining the same clusters than the k-means initialization, ie keeping the assignment of the points into clusters, and recomputing the medoids with (2) instead of considering the centroids with (3). Hence, we only describe how to define partitioning clusters, giving the complexity results of this phase and keeping in mind that the computations of centroids and medoids run respectively in $O(N)$ and $O(N^2)$ time.

3.1 Generic initialization strategies

Firstly, we present generic initialization designed for the k-means problem :

- **Random selection (RAND)**: this algorithm is one of the most naive algorithms, selecting K points randomly (and uniformly) among N points in the sample space to former an initialization of the centers. Clusters are defined assigning each point to the cluster of its closest randomly selected point. Defining such clusters runs in $O(N)$ time.
- **Furthest Point (FP)** (max-min distance): FP selects randomly the first point. Once $k < K$ points are selected, the $k+1$ new one maximizes the minimal distance from this point to the previously k selected point. Considering that outliers are always the furthest from other points, this algorithm is therefore easily affected by outliers. Selecting the K points runs in $O(K^2N)$ time.
- **K-means++**: K-means++ is an upgraded and randomized version of FP. K-m++ selects randomly the first point, and use different selection probabilities instead of the deterministic max-min distance used by FP initialization [1]. If a point is far from the already selected points, its selection probability is higher. Selecting K points for K-means++ runs in $O(K^2N)$ time.
- **Forgy's method**: Forgy's initialization uniformly and randomly assigns each point to one of the K clusters, it runs in $O(N)$ time [8].
- **Hartigan's method**: Hartigan's initialization firstly sorts the points according to their distances from the centroid of the N initial points. The i -th $i \in \{1, 2, \ldots, K\}$ center is then chosen to be the $(1 + (i-1)N/K)$ - point [10]. The computation cost is given by the sorting complexity, Hartigan's method runs in $O(N \log N)$ time to define partitioning clusters.

3.2 Initialization using 2D PF indexation

The 2D PF indexation, illustrated in Figure 1, allows to provide variants of the random and Hartigan's initialization, using this specific indexation. The following strategies select initial centroids or medoids in $O(N \log N)$ time, the time complexity of the sorting re-indexation:

- $\mathbf{N/(K+1) - Uniform}$: using the 2d PF indexation, $N/(K+1) - Uniform$ selects the K points at indexes round$\left(\frac{iN}{K+1}\right)$ for $i \in \{1, 2 \ldots, K\}$.

- **N/(2K) − Uniform**: using the 2d PF indexation, $N/(2K) - Uniform$ selects the K points at indexes round$\left(\frac{iN}{2K}\right)$ for $i \in \{1, 2 \ldots, K\}$.
- **N/K-Random selection (N/K-RAND)** N/K-RAND is a tailored variant of RAND selection, the K points points for initial centroids or medoids in the sample space are randomly chosen in each of the N/K-size intervals dividing uniformly the indexes of the 2D PF.

3.3 Initialization using p-dispersion for 2D PF

The p-dispersion problems select $p \geqslant 2$ points among N initial points, maximizing diversity measures, some variants are solvable in polynomial time in a 2D PF [3]. Similarly with Hartigan's heuristic, one may select among diversified points the ones to be defined as initial centroids or medoids. We use the standard p-dispersion (Max-Min p-dispersion) problem, having a time complexity in $O(pn \log n)$, other variants are too much time consuming for our application, at least in $O(pn^2)$ [3]. Selecting directly K points with (Max-Min) K-dispersion selects the two extreme points. Two slight adaptations are thus provided to avoid both extreme points, keeping the time complexity in $O(pn \log n)$:

- **K+2-dispersion** ($K + 2$-disp): a $K + 2$-dispersion is solved using the tailored DP algorithm [3], and the two extreme points (x_1 and x_N after reindexation) are removed.
- **2K + 1 dispersion** ($2K + 1$-disp):a $2K + 1$-dispersion is solved using the tailored DP algorithm [3], and we keep the even indexes reindexing the $2K+1$ points from 1 to $2K + 1$.

Note that the standard p-center problems have also time complexity allowing quick initialisation [5]. However, such dissimilarity measures are based on the most extreme points, that would include the outliers, which is a dangerous property for k-means and k-medoids initialization. K-center initialization strategies are thus not in our benchmark of initialization strategies.

3.4 Initialization using 1D dynamic programming

One dimensional (1D) k-means is solvable in polynomial time, with an implementation available in a R package [19]. A first DP algorithm runs in $O(KN^2)$ time and using $O(KN)$ memory space [19]. An improvement was proposed in $O(KN)$ time and using $O(N)$ memory space [9]. Such complexity result allows for our application to use this 1D DP algorithm as an initialization heuristic to define first clusters, using two heuristics to reduce the 2D PF into a 1D case:

- **1D-DP-reduc**: using the specific shape of a 2D PF illustrated in Figure 1, one may approximate define a 1D structure, associating to each point x_i of the 2D PF the scalar $z_i = \sum_{j=0}^{i-1} \|x_j - x_{j+1}\|$. The 1D interval clustering define the indexation for the 2D PF clustering, computing the costs of clusters in the 2D PF requires respectively $O(N)$ and $O(N^2)$ time computations for k-means and k-medoids (which computes also the initial centroids or medoids).

- **1D-DP-linReg**: this initialization is similar to 1D-DP-reduc, the difference being in the reduction from a 2D PF case to 1D. A linear regression is used, 1D points z_i are now the orthogonal projection of points x_i in the linear regression, as illustrated Figure 2.

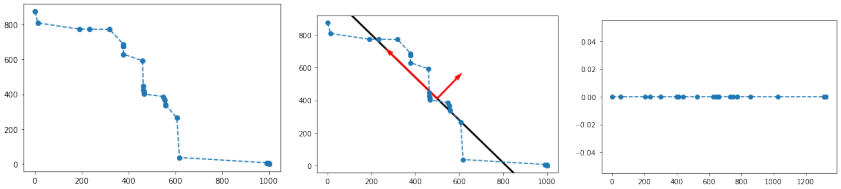


Fig. 2. From a 2dPF to a 1D case with 1D-DP-linReg approach

4 Computational experiments and results

4.1 Data generation

To the best of our knowledge, no specific datasets for 2D PF are available for our study. Starting from any decreasing function $f : [0, 1] \to \mathbb{R}$, one may generate 2D PF with N random values in $(y_n) \in [0, 1]^N$ and considering the 2d points $(y_n, f(y_n))$ for $n \in [\![1; N]\!]$. With a uniform distribution in the random generation and a linear function f, such generation would be too regular, and the naive $N/(2K) - Uniform$ initialization would be very close to the trivial optimal solutions. Hence, a new data generator, described in Algorithm 1, was provided to furnish 2D PF without regularity, like the one given in Figure 2:

Algorithm 1: random generation of a 2d-PF

Input: N the size of the wished 2D PF
  initialize $n = 2$, $\mathcal{L} = \{(0, 1000); (1000, 0)\}$, $\mathcal{B} = \{(0, 1000, 1000, 0)\}$
  for $i = 3$ to N
    select randomly $(x_1, y_1, x_2, y_2) \in \mathcal{B}$ and remove it from $\mathcal{B}$
    select randomly $x_3 \in]x_1, x_2[$ and $y_3 \in]y_1, y_2[$
    add (x_3, y_3) in $\mathcal{L}$ and add (x_1, y_1, x_3, y_3) and (x_3, y_3, x_2, y_2) in $\mathcal{B}$
  end for
return $\mathcal{L}$

In Algorithm 1, the loop invariant is that the list $\mathcal{L}$ contains non-dominated points, and $\mathcal{B}$ contains the couple of neighbor points in $\mathcal{L}$ with the indexation of Figure 1. Starting from a point (x_1, y_1, x_2, y_2) and having $x_3 \in]x_1, x_2[$ and $y_3 \in]y_1, y_2[$, we have $(x_1, y_1) \prec (x_3, y_3) \prec (x_2, y_2)$. Hence, adding (x_3, y_3) in $\mathcal{L}$ keep the two by two incomparability in $\mathcal{L}$, and the neighboring properties in $\mathcal{B}$.

4.2 Computational experiments and conditions

Instances used in our computational results were generated by Algorithm 1 with five values of N, $N \in \{50; 100; 500; 1000; 5000\}$, generating 10 instances for each value of N. For each generated 2D PF, we experimented K-clustering with five values of K, $K \in \{2; 3; 5; 10; 200\}$. This defines a total of $250 = 5 \times 5 \times 10$ instances for k-medoids and k-means problems. For the reproducibility of the results, these instances and the results instance by instance are available in https://github.com/ndupin/Pareto2d. For these 250 instances, following results are provided for each initialization strategy of k-means and k-medoids:

- the initial value of the heuristic solution;
- the value of the local minimum using PAM or Lloyd's local search
- the number of iterations to converge to a local minimum

Having the optimal solutions for K-medoids, it allows to present the quality of solutions in terms of over-cost gap related to the optimal solutions proven by the DP algorithm [4]. For k-means, the similar DP algorithm is only a heuristic, it can be proven that is is a lower bound of any local minimum of the previous local searches [7]. Our experiments did not show any counter-example where a better solution that the k-means 2D PF DP algorithm is found.

Note that we do not provide computation times. The heuristics converge very quickly, in order of second, which is visible in the number of iterations. Comparing the solving time with heuristics and the exact DP could be interesting. Optimal values with DP algorithms were provided using $K = 20$ computations, and storing the values in the DP matrix. Note that RAND, N/K-RAND, Forgy, Hartigan, k-means++ are randomized initialization strategies, the results are given in average using 25 runs with different seeds. Other initialization heuristics are deterministic, one run is enough to provide the results.

Similarly to [6], a parallel local search uses a portfolio of several initialization strategies, and computes independently the local search algorithms fr each initial solution. It allows to have the best local minimum among the selected initialization strategies. Having two threads, the 1D DP initialization were selected, the results are reported as **Paral_1D_DP**. Allowing four threads, both previous initialization were selected with k-means++ and FP initialization, this is denoted **Parall4**.

4.3 Analyses of computational results

The evolution of the average number of iterations to converge to a local minimum when N is increasing are illustrated for k-means and k-medoids in Figures 3 and 4 respectively. It illustrates that the number of iteration is very small, and is also slightly increasing. Keeping in mind that each iteration runs in $O(N)$ or $O(N^2)$ for k-means and k-medoids respectively, the initialization strategy is prominent it the total computation time to apply PAM and Lloyd's local searches. This property is specific for 2D PF sub-cases, this does not hold for general isntances of k-means and k-medoids. The 2D PF structure with non-nested optimality

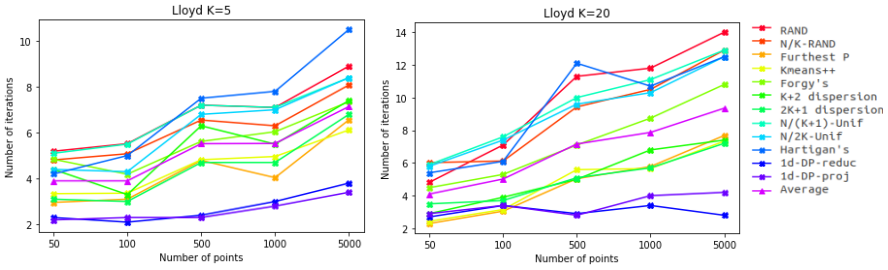


Fig. 3. Number of iterations for K=5 and 20 and Lloyd's local search applied to the initialization strategies, with N varying from 50 to 5000

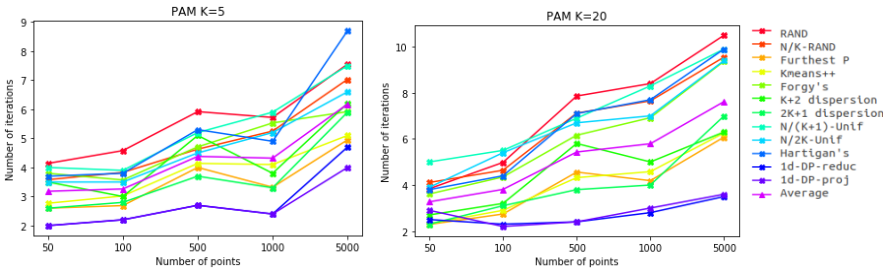


Fig. 4. Number of iterations for K=5 and 20 and PAM local search applied to the initialization strategies, with N varying from 50 to 5000

property allows to have exact polynomial DP algorithm, this is also useful for the generic algorithms to solve k-means and k-medoids in the 2D PF sub-case. We note also that the hierarchy among initialization strategies in Figures 3 and 4 is rather stable. It is actually related to the quality of the initial solutions, that are measured in the exhaustive table of results online, and also to the final quality of local minimums, comparing their hierarchy with Tables 1 and 2.

Tables 1 and 2 present statistical indicators for the quality of local minimums induced by the different initialization strategies. The 250 instances provide for each initialization strategy a sequence of 250 local minimums, that can be compared to the values of the exact DP algorithms [7, 4]. Statistics are provided based on the over-cost percentages, with average values and variance and quartiles to appreciate the dispersion. Generally, local minimums exist with a very poor quality, few such local minimums degrade dramatically the average values, and the medians are much better than the average. Variance computations are in such cases non significative. Note that Tables 1 and 2 provide results which are very similar considering k-medoids or k-means clustering in a 2D PF.

RAND and Forgy's initialization are the worst strategies in terms of quality of local minimums. N/K-RAND improves the results obtained by the RAND initialization, but the results are still unsatisfactory. FP, k-means ++ are the

Table 1. Comparison of statistical indicators for the relative over-costs of local minimums induced by the different initialization strategies for the k-means problem, comparison to the solutions from [7].

Init.	Time	Average	Q1	Median	Q3	variance
RAND	$O(N)$	964,64 %	15,35 %	135,15 %	400,80 %	-
Furthest P	$O(K^2N)$	41,75 %	1,90 %	18,80 %	50,43 %	57,21 %
Kmeans++	$O(K^2N)$	74,67 %	14,03 %	48,05 %	93,10 %	94,38 %
Forgy	$O(N)$	3224,63 %	31,48 %	330,15 %	2245,43 %	-
Hartigan	$O(N \log N)$	1783,30 %	0,00 %	75,70 %	322,90 %	-
N/(K+1)-Unif.	$O(N \log N)$	1280,03 %	0,00 %	43,95 %	253,60 %	-
N/2K-Unif.	$O(N \log N)$	1221,94 %	0,00 %	42,40 %	249,25 %	-
N/K-RAND	$O(N \log N)$	1142,18 %	6,83 %	50,30 %	236,43 %	-
K+2 disp	$O(KN \log N)$	99,69 %	0,00 %	22,40 %	93,33 %	857,72 %
2K+1 disp	$O(KN \log N)$	399,72 %	0,00 %	12,35 %	72,20 %	-
1D-DP-reduc	$O(KN)$	336,50 %	0,00 %	0,15 %	115,18 %	-
1D-DP-proj	$O(KN)$	158,45 %	0,00 %	1,90 %	38,78 %	3677,36 %
best Parall		2,90 %	0,00 %	0,00 %	0,90 %	0,83 %
Paral_1D_DP		154,77 %	0,00 %	0,00 %	10,95 %	3685,79 %
Parall4		3,90 %	0,00 %	0,00 %	1,90 %	1,70 %

Table 2. Comparison of statistical indicators for the relative over-costs of local minimums induced by the different initialization strategies for the k-medoids problem,comparison to the optimal solution from [4].

Init	Time	Average	Q1	Median	Q3	var
RAND	$O(N^2)$	1661,16 %	27,70 %	162,85 %	499,75 %	-
Furthest P	$O((N+K^2)N)$	43,38 %	3,73 %	19,60 %	54,38 %	66,13 %
Kmeans++	$O((N+K^2)N)$	79,54 %	15,78 %	53,25 %	109,48 %	118,93 %
Forgy	$O(N^2)$	3687,94 %	39,25 %	336,15 %	2304,70 %	-
Hartigan	$O(N^2)$	2214,39 %	0,15 %	102,70 %	387,18 %	-
N/(K+1)-Unif.	$O(N \log N)$	1921,86 %	0,00 %	72,55 %	358,65 %	-
N/2K-Unif.	$O(N \log N)$	1723,85 %	0,00 %	83,20 %	355,08 %	-
N/K-RAND	$O(N \log N)$	1526,19 %	11,38 %	84,20 %	330,25 %	-
K+2 disp	$O(KN \log N)$	340,92 %	0,00 %	31,85 %	131,43 %	-
2K+1 disp	$O(KN \log N)$	820,99 %	0,00 %	21,95 %	90,68 %	-
1D-DP-reduc	$O(N^2)$	677,31 %	0,00 %	0,15 %	74,45 %	-
1D-DP-proj	$O(N^2)$	617,97 %	0,00 %	0,60 %	23,40 %	-
best Parall		5,20 %	0,00 %	0,00 %	1,20 %	2,39 %
Paral_1D_DP		615,29 %	0,00 %	0,00 %	16,53 %	-
Parall4		5,87 %	0,00 %	0,00 %	2,78 %	2,58 %

best generic approaches, and also the best approaches to avoid local minimums of a poor quality. These initialization approaches are also the most time consuming, as given by the complexities underlined in Tables 1 and 2.

Naive initialization using 2D PF indexation, N/(K+1)-Uniform and N/(2K)-Uniform have very similar results, with more than a quartile of optimal solutions found, and a degradation of the results for the other quartiles, with a quartile of poor solutions. Initialization using p-dispersion improves the previous uniform ones, with more than a half of excellent solutions, but the last quartile is mainly composed of poor solutions. Initialization based in the 1D DP algorithm may he seen as the best individual approaches to find optimal solutions, with more than a half of excellent solutions. However, very poor local minimums still exist for the resulting local search heuristic.

Combining in parallel both 1D DP initialization allows to improve significantly the quality of quartiles, with more than a half of optimal solutions, but this is still not enough to avoid the bad local minimums. This induced to consider in the pool of initialization FP and k-means ++, these approaches having less dispersed results. Table 3 presents aggregated results for the 10 instances for several values of K and N, and illustrates the complementarity of FP and k-means ++ with the 1D DP initialization. The resulting Parall4 heuristic provides accurately solutions of an excellent quality, close to the best reachable ones with local search as shown in the row BestParall of Tables 1 and 2. Parall4 heuristic does not require a lot of computation times as in [7, 4], and requiring only 4 threads for parallel computations, or one thread in a multi-start local search with four sequential local searches.

5 Conclusions and perspectives

Standard local search approaches, PAM and Lloyd's heuristics, are investigated to solve faster large instances of k-means and k-medoids clustering in the 2D PF case. Generally, local minimums with a poor quality are found using any initialization heuristic. Two heuristic reductions to the 1D cases, allowing to use the specific 1D DP algorithm provided the most optimal solutions. The generic initialization methods, k-means ++ and Furthest Point, were the best to avoid local minimums of a poor quality. Combining these four approaches in a parallel or multi-start local search heuristic allows to have accurately solutions of a very good quality. Such approach is much faster than the exact DP algorithms [7, 4], for an application to large instance sizes.

This work offers several new research perspectives. In the applications to evolutionary algorithms of 2D PF clustering already discussed by [7, 4, 5], it allows to have faster solutions of a very good quality. Perspectives may be to improve the local search for the specific 2D PF case instead of using the generic PAM and Lloyd's neighborhoods, to improve the accuracy. Lastly, one may try to generalize such study and results to dimension 3 and more. A challenging issue would be to extend an efficient projection to the 1D PF case, it was easier considering the 2D PF case.

References

1. Arthur, D., Vassilvitskii, S.: k-means++: The advantages of careful seeding. In: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, pp. 1027–1035. Society for Industrial and Applied Mathematics (2007)
2. Celebi, M., Kingravi, H., Vela, P.: A comparative study of efficient initialization methods for the k-means clustering algorithm. Expert Systems with Applications **40**, 200–210 (2013). DOI 10.1016/j.eswa.2012.07.021
3. Dupin, N.: Polynomial algorithms for p-dispersion problems in a 2d pareto front. arXiv preprint arXiv:2002.11830 (2020)
4. Dupin, N., Nielsen, F., Talbi, E.: k-medoids clustering is solvable in polynomial time for a 2d Pareto front. In: World Congress on Global Optimization, pp. 790–799. Springer (2019)
5. Dupin, N., Nielsen, F., Talbi, E.: Clustering a 2d Pareto Front: p-center problems are solvable in polynomial time. In: International Conference on Optimization and Learning, pp. 179–191. Springer (2020)
6. Dupin, N., Talbi, E.: Parallel matheuristics for the discrete unit commitment problem with min-stop ramping constraints. International Transactions in Operational Research **27**(1), 219–244 (2020)
7. Dupin, N., Talbi, E., Nielsen, F.: Clustering in a 2-dimensional pareto front: p-median and p-center are solvable in polynomial time. 7th International Conference on Metaheuristics and Nature Inspired Computing META18 (2018)
8. Forgy, E.: Cluster analysis of multivariate data: Efficiency vs. interpretability of classification. Biometrics **21**(3), 768–769 (1965)
9. Grønlund, A., et al.: Fast exact k-means, k-medians and Bregman divergence clustering in 1d. arXiv preprint arXiv:1701.07204 (2017)
10. Hartigan, J., Wong, M.: Algorithm AS 136: A k-means clustering algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics) **28**(1), 100–108 (1979)
11. Hsu, W., Nemhauser, G.: Easy and hard bottleneck location problems. Discrete Applied Mathematics **1**(3), 209–215 (1979)
12. Jain, A.: Data clustering: 50 years beyond k-means. Pattern recognition letters **31**(8), 651–666 (2010)
13. Kaufman, L., Rousseeuw, P.: Clustering by means of medoids. North-Holland (1987)
14. Lloyd, S.: Least squares quantization in PCM. IEEE transactions on information theory **28**(2), 129–137 (1982)
15. Mahajan, M., Nimbhorkar, P., Varadarajan, K.: The Planar K-Means Problem is NP-Hard. Theor. Comput. Sci. **442**, 13–21 (2012)
16. Nielsen, F.: Introduction to HPC with MPI for Data Science. Springer (2016)
17. Pena, J., Lozano, J., Larranaga, P.: An empirical comparison of four initialization methods for the k-means algorithm. Pattern recognition letters **20**(10), 1027–1040 (1999)
18. Slonim, N., Aharoni, E., Crammer, K.: Hartigan's K-Means Versus Lloyd's K-Means-Is It Time for a Change? In: IJCAI, pp. 1677–1684 (2013)
19. Wang, H., Song, M.: Ckmeans. 1d. dp: optimal k-means clustering in one dimension by dynamic programming. The R journal **3**(2), 29–33 (2011)

Appendix: Results by instance size for k-medoids

Table 3. Comparison of average values on the datasets with $N = 100, 1000, 5000$ and $K = 3, 10$ of the initial overcost gap to the optimal solutions after initialization and after local search convergence for K-medoids. Bold values indicate the best average values for a given size of instances.

For $K = 3$:

	N=100 init	N=100 init+LS	N=1000 init	N=1000 init+LS	N=5000 init	N=5000 init+LS
RAND	642,6%	39,5%	785,6%	34,7%	616,7%	50,7%
N/K-RAND	273,4%	29,1%	237,6%	28,5%	283,8%	30,5%
Furthest P	181,9%	37,4%	202,5%	**23,0%**	**129,9%**	18,4%
Kmeans++	228,7%	55,5%	327,4%	56,3%	206,0%	15,0%
Hartigan's	457,6%	54,3%	497,2%	24,2%	485,9%	84,1%
Forgy's	1040,8%	97,0%	1247,0%	67,2%	1048,8%	73,6%
K+2 disp	372,8%	16,4%	562,8%	78,2%	262,4%	21,2%
2K+1 disp	165,1%	32,2%	321,8%	29,2%	161,1%	**12,7%**
N/(K+1)-Unif	307,2%	39,9%	259,2%	38,9%	373,0%	33,9%
N/2K-Unif	233,6%	41,1%	156,1%	39,1%	254,0%	36,6%
1D-DP-reduc	**11,2%**	**0,2%**	594,8%	134,8%	737,0%	154,6%
1D-DP-proj	13,1%	1,3%	**136,2%**	84,6%	737,0%	154,6%

For $K = 10$:

	N=100 init	N=100 init+LS	N=1000 init	N=1000 init+LS	N=5000 init	N=5000 init+LS
RAND	2624,14 %	501,14 %	3386,84 %	375,79 %	3633,05 %	418,77 %
N/K-RAND	822,59 %	298,63 %	1306,63 %	182,64 %	927,22 %	314,39 %
Furthest P	206,07 %	34,10 %	355,41 %	63,35 %	348,31 %	53,23 %
Kmeans++	300,26 %	100,13 %	554,78 %	103,51 %	592,68 %	137,74 %
Hartigan's	851,03 %	378,84 %	2646,14 %	249,99 %	1903,80 %	347,85 %
K+2 disp	1108,43 %	162,79 %	1170,57 %	125,64 %	1185,58 %	130,94 %
2K+1 disp	128,35 %	55,56 %	481,00 %	76,65 %	565,40 %	92,58 %
N/(K+1)-Unif	873,48 %	475,54 %	2122,56 %	212,33 %	1449,22 %	366,43 %
N/2K-Unif	733,31 %	271,75 %	1799,85 %	185,13 %	577,15 %	300,36 %
1D-DP-reduc	47,56 %	**39,55 %**	**13,67 %**	**5,63 %**	**11,76 %**	**1,08 %**
1D-DP-proj	**47,17 %**	41,58 %	23,47 %	6,87 %	19,89 %	5,06 %

A comparison of Learnheuristics using different Reward Functions to solve the Set Covering Problem

Broderick Crawford[1 [0000−0001−5500−0188]], Ricardo Soto[1 [0000−0002−5755−6929]], Felipe Cisternas-Caneo[1 [0000−0001−7723−7012]], Diego Tapia[1 [0000−0002−0603−3722]], Hanns de la Fuente-Mella[1 [0000−0003−2564−8770]], Wenceslao Palma[1 [0000−0002−7232−0412]], José Lemus-Romani[1 [0000−0001−5379−0315]], Mauricio Castillo[1 [0000−0001−7804−6381]], and Marcelo Becerra-Rozas[1 [0000−0003−0426−0144]]

Pontificia Universidad Católica de Valparaíso, Valparaíso, Chile
{broderick.crawford,ricardo.soto,hanns.delafuente,wenceslao.palma}@pucv.cl
{felipe.cisternas.c,diego.tapia.r,jose.lemus.r}@mail.pucv.cl
{mauricio.castillo.d,marcelo.becerra.r}@mail.pucv.cl

Abstract. The high computational capacity that we have thanks to the new technologies allows us to communicate two great worlds such as optimization methods and machine learning. The concept behind the hybridization of both worlds is called Learnheuristics which allows to improve optimization methods through machine learning techniques where the input data for learning is the data produced by the optimization methods during the search process. Among the most outstanding machine learning techniques is Q-Learning whose learning process is based on rewarding or punishing the agents according to the consequences of their actions and this reward or punishment is carried out by means of a reward function. This work seeks to compare different Learnheuristics instances composed by Sine Cosine Algorithm and Q-Learning whose different lies in the reward function applied. Preliminary results indicate that there is an influence on the quality of the solutions based on the reward function applied.

Keywords: Learnheuristic · Sine Cosine Algorithm · Q-Learning · Reward Function · Reinforcement Learning

1 Introduction

Optimization problems are very recurrent in the real world being very complex to solve and it is necessary to obtain results in reasonable times. Approximate optimization techniques such as metaheuristics provide good results in reasonable times but when solving increasingly complex industry problems the quality of the solutions worsen.

Nowadays there is a high computational capacity thanks to new technologies allowing machine learning techniques to process large volumes of data in short

times. During the optimization process a large amount of data is generated serving as input for some machine learning techniques to improve the optimization process and obtain better quality solutions.

This hybridization is called Learnheuristic [1], where it is composed of an optimization module and a machine learning module. The optimization module solves an optimization problem and in each iteration generates a quantity of data that is delivered to the machine learning module so that it learns and makes decisions that affect the optimization technique in order to improve the quality of the solutions.

This work applies the concept of Learnheuristic, where the optimization module is composed of the metaheuristic Sine Cosine Algorithm (SCA) and the machine learning module is composed of Q-Learning. Q-Learning is used to learn how to select binarization schemes when SCA solves the Set Covering Problem. In particular, we seek to demonstrate the impact generated by different reward functions applied in Q-Learning.

The paper is organized as follows: Section 2 presents the Set Covering Problem, Section 3 presents the reinforcement learning, Q-Learning and the reward functions found in the literature, Section 4 presents Sine Cosine Algorithm, why it is necessary to use binarization schemes and how the Learnheuristic is generated, Section 5 presents the experimental results to finish with the conclusions in Section 6.

2 Set Covering Problem

The Set Covering Problem (SCP) is a classic optimization problem, which can be used to model in different applications and various domains, such as assignment problems, transport networks, among others. This problem is class NP-Hard [8] and consists of finding the set of elements with the minor cost that meets a certain amount of needs. The mathematical model of the Set Covering Problem is available at [10].

The SCP allows modeling of real-life optimization problems such as the location of gas detectors for industrial plants [22], the location of electric vehicle charging points in California [25], the optimal UAV locations for the purpose of generating wireless communication networks in disaster areas [17] and airline and bus crew scheduling [18]. These studies allow us to appreciate the importance of solving this problem with optimization techniques that guarantee good results.

3 Reinforcement Learning

Reinforcement learning is a subcategory of Machine Learning whose learning process consists of an agent performing different actions in different states and the objective of the agent is to learn what is the best action for each of the states by judging the consequence of performing each of the actions. Some of the classic examples are Q-Learning [23], Monte Carlo RL [11] and SARSA [19].

3.1 Q-Learning

Watkins et. al. [23] proposed Q-Learning in 1992 and it is one of the best known reinforcement learning techniques. The main objective is to maximize the accumulated reward of an action in a particular state, in other words, to find the best action for a state.

An agent travels in different states and in each one of them an action is experienced immediately obtaining a reward or a punishment and the moment when an action is taken in a particular state is called a episode.

The agent should learn to select the best action for each one of the possible states. As the episodes pass the agent performs all possible actions for a state and the best action is the one that obtains the best accumulated reward [23].

The Q-Learning algorithm tries to learn how much accumulated reward the agent will get in the long run for each pair of action-state. The action-state function is represented as $Q(s, a)$ which returns the reward that the agent will get when performing the action a in the state s and assumes that it will follow the same policy obtained by the Q function until the end of the episode, this value is called Q-Value.

These Q-Values are stored in the Q-Table, which is a matrix where the rows correspond to the states and the columns correspond to the actions.

The Q-Value obtained for action a in state s when the n-th episode occurs is calculated as follows:

$$Q_n(s,a) = \begin{cases} (1-\alpha_n)Q_{n-1}(s,a) \ + \ \alpha_n[r_n \ + \ \gamma V_{n-1}(s_{n+1})] \ if \ \ s = s_n \ \ and \ \ a = a_n \\ Q_{n-1}(s,a) \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad otherwise, \end{cases} \tag{1}$$

where,

$$V_{n-1}(s_{n+1}) \ \ = \ \ max \ \ Q_{n-1}(s_{n+1}, b) \tag{2}$$

s_n corresponds to the current state, a_n corresponds to the action selected and performed for the n-th episode, $max \ \ Q_{n-1}(s_{n+1}, b)$ corresponds to the highest Q-Value obtained for episode $n-1$ for the following state s_{n+1}, in other words, the best action for the following state s_{n+1}, r_n corresponds to the reward function that allows rewarding or punishing the action based on its consequence, α_n corresponds to the learning factor and γ corresponds to the discount factor.

3.2 Reward Function

The big question when using Q-Learning is: How to reward or punishment the consequences of carrying out an action? A good balance of reward and penalty allows an equitable variation of the selection of actions so the best action found is more reliable.

In the literature different Learnheuristics were found where metaheuristics incorporate Q-Learning as a machine learning technique. The reward function used by these Learnheuristics are diverse and adapted to the behavior of the metaheuristic.

For example, in [15] proposes 3 ways of reward or penalty for Ant Colony Optimization where all are associated to the best ant tour and, in some cases, they are calculated with respect to a predefined W constant.

On the other hand, other ways of rewarding or penalizing were found where they are oriented to the performance of Metaheuristics and can be applied to any of them. Such are the cases of [24] where the reward is 1 when the fitness is improved or -1 otherwise and as a result of this the reward or penalty visible in equation (4) is born where only the reward is given. The different reward functions are shown in Table 1.

Reference	Reward Function
[24]	$\mathrm{r}_n = \begin{cases} 1, & if\ the\ current\ action\ improves\ fitness \\ -1, & otherwise. \end{cases}$ (3)
-	$\mathrm{r}_n = \begin{cases} 1, if\ the\ current\ action\ improves\ fitness \\ 0, & otherwise. \end{cases}$ (4)
[15]	$\mathrm{r}_n = \begin{cases} W \cdot f_{best}, if\ (r,s) \in\ tour\ done\ by\ the\ best\ agent \\ 0, & otherwise. \end{cases}$ (5)
[15]	$\mathrm{r}_n = \begin{cases} \frac{W}{f_{best}}, if\ (r,s) \in\ tour\ done\ by\ the\ best\ agent \\ 0, & otherwise. \end{cases}$ (6)
[15]	$\mathrm{r}_n = \begin{cases} \sqrt{f_{best}}, if\ (r,s) \in\ tour\ done\ by\ the\ best\ agent \\ 0, & otherwise. \end{cases}$ (7)

Table 1: Reward Function

4 Sine Cosine Algorithm

Sine Cosine Aalgorithm (SCA) is a population-based metaheuristic where initial solutions are randomly generated and altered during the search process [13]. The equations of movements proposed for both phases are as follows:

$$X_{i,j}^{t+1} = \begin{cases} X_{i,j}^{t} + r_1 \cdot \sin(r_2) \cdot |r_3 P_j^t - X_{i,j}^t|, & r_4 < 0.5 \\ X_{i,j}^{t} + r_1 \cdot \cos(r_2) \cdot |r_3 P_j^t - X_{i,j}^t|, & r_4 \geq 0.5 \end{cases} \tag{8}$$

$$r_1 = a - t\frac{a}{T} \tag{9}$$

Where r_1 is a parameter by equation 9, r_2, r_3 and r_4 are random numbers, $X_{i,j}^t$ is the position of the $i-th$ solution in the $j-th$ dimension in $t-th$ iteration and P_j^t it is the position of the best solution in the $j-th$ dimension in the $t-th$ iteration.

SCA is a metaheuristic that was built to solve problems in continuous domains, that is why to solve combinatorial optimization problems such as the Set Covering Problem it is necessary to transform the solutions from the continuous domain to the discrete domain [4].

One of the most used mechanisms to transfer solutions from the continuous domain to the binary domain is the Two-Step Technique. This technique consists of first transferring the values of the solutions from the continuous domain to the $[0, 1]$ domain by means of transfer functions and then taking the values of the solutions in the $[0, 1]$ domain and discretizing them by means of discretization functions.

4.1 Learnheuristic framework

Recent studies [20, 2, 21] built a general Learnheuristics framework that incorporates Q-Learning as a machine learning module for operator selection. In particular, they select binarization schemes derived from combinations between transfer functions and discretization functions of the Two-Step Technique.

In the present work, modifications were made to the proposal presented by the authors in [2]. The **actions** to be taken by the agents are the **binarization schemes**, the **states** are the **phases of the metaheuristic**, i.e. exploration or exploitation, the **episodes** where an action is selected and applied in a particular state will be the **iterations** and the **agents** will be the **individuals** of the Sine Cosine Algorithm.

4.2 Balancing Exploration and Exploitation

To design a good metaheuristic is to make a proper trade-off between two forces: exploration and exploitation. It is one of the most basic dilemmas that both individuals and organizations constantly face. Exploration consists of looking for alternatives different from those already found while exploitation consists of exploiting a previously known alternative. This translates into selecting a new product for an individual, innovating or not for an organization and for metaheuristics into exploring the search space or exploiting a limited region of it. Ambidextrous Algorithms [12, 3] aim at balancing exploration and exploitation oriented to decision making.

Before solving the balancing problem between exploration and exploitation, one must first have the ability to measure these indicators and then make a decision. Diversity metrics allow the measurement of exploration and exploitation because they quantify the dispersion of the individuals of the population.

There are different ways to quantify this diversity where the metrics based on central measurements and others based on frequency stand out. Metrics based on central measurements quantify diversity levels only in population algorithms. This is because the best search agents tend to attract the other solutions to them. In other words, the distance between solutions increases in exploration search processes and decreases in exploitation search processes [14].

In the literature there are different ways to quantify diversity and for this work the Dimensional-Hussain Diversity [9] was used. Dimensional-Hussain Diversity is a diversity based on central measurements and is defined as:

$$D_{dh}(X) = \frac{1}{l \cdot n} \sum_{d=1}^{l} \sum_{i=1}^{n} |mean(x^d) - x_i^d| \tag{10}$$

Where $D_dh(X)$ corresponds to the Dimensional-Hussan Diversity of the population X, $mean(x^d)$ is average of the d-th dimension, n is the number of search agents in the population X and l is the number of dimension of the optimization problem.

Morales-Castañeda et. al. in [14] propose some equations which can obtain a percentage of exploration (XPL%) and a percentage of exploitation (XPT%) based on the diversity of the population. The particularity of these equations is that they are generic, that is, any diversity metric can be used, since the percentages are calculated around the diversity that the population has in a given iteration in relation to the maximum diversity obtained during the search process. These equations are as follows:

$$XPL\% = \left(\frac{Div_t}{Div_{max}} \right) \times 100 \ , \ XPT\% = \left(\frac{|Div_t - Div_{max}|}{Div_{max}} \right) \times 100 \tag{11}$$

Where Div_t corresponds to the current diversity in the t-th iteration and Div_{max} corresponds to the maximum diversity obtained in the entire search process.

By obtaining these percentages of exploration and exploitation we can determine the current state to be used in Q-Learning. This determination is done in the following way:

$$next\ state = \begin{cases} Exploration\ if\ XPL\% \geq XPT\% \\ Exploitation\ if\ XPL\% < XPT\% \end{cases} \tag{12}$$

All of the above are summarized in Algorithm 1.

5 Experimental results

To evaluate the impact of the reward function in Q-Learning is that 5 different instances of BQSCA were implemented where the main change lies in the reward function to be applied. Table 2 shows the applied functions and their respective names.

Each of the instances was run 31, with a population of 40 individuals and 1000 iterations performed in each run. As mentioned in Section 3.1, Q-Learning has 2 additional parameters and they are the learning factor (α) and the discount factor (γ). The 5 versions of BQSCA were configured with the same learning factor whose value is $\alpha = 0.1$ proposed in [5] and the same discount factor whose value is $\gamma = 0.4$ proposed in [5]. On another hand, the value of the constant a of the parameter r_1 of equation 9 is 2 [13].

Reference	Reward Type	Equation	Name
[24]	Penalty -1	eq. (3)	BQSCA-QL1
-	with Out Penalty	eq. (4)	BQSCA-QL2
[15]	global Best	eq. (7)	BQSCA-QL3
[15]	root Adaptation	eq. (6)	BQSCA-QL4
[15]	escalating Multiplicative Adaptation	eq. (5)	BQSCA-QL5

Table 2: Q-Learning Implementation

The Set Covering Problem instances proposed by OR-Library, which are betchmarck instances where the different authors make their comparisons, have been solved.

Table 4 shows the details obtained for the algorithms. The first column refers to each evaluated instance, the second column refers to the best known optimum for each instance, the fourth column indicates the best optimum obtained for each algorithm, the fifth column indicates the average of the results obtained for each algorithm and the sixth column indicates the Relative Percetage Deviation (RPD%) for each algorithm. These last 3 columns are repeated for each algorithm under analysis.

The RPD measures the percentage deviation of the best result obtained Z in relation to the best known result Z_{opt} for each instance. The measure is calculated as follows:

$$RPD = \frac{Z - Z_{opt}}{Z_{opt}} \times 100 \tag{13}$$

The results indicate that the 5 instances of BQSCA obtain good results, reaching optimal results in some instances. It should be noted that BQSCA-QL1 obtains preliminarily better results than the other instances.

Additionally, a statistical test was performed to validate the results obtained. Since the data does not come from nature, it does not have a normal distribution. On the other hand, since the data are not independent of each other, a nonparametric statistical test was performed. The Wilcoxon-Mann-Whitney test was applied. The hypothesis used for this statistical test is the following:

$$H_0 = \text{Algorithm A} \geq \text{Algorithm B} \quad , \quad H_1 = \text{Algorithm A} < \text{Algorithm B}$$

If the result of the statistical test is obtained a p-value < 0.05, we cannot assume that Algorithm A has worse performance than Algorithm B, rejecting H_0.

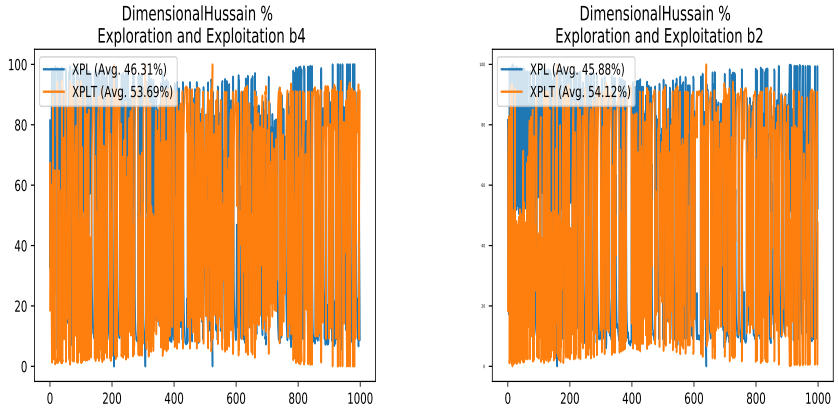
The results indicate that the 5 instances of BQSCA obtain good results, reaching optimal results in some instances. It should be noted that BQSCA-QL1 obtains preliminarily better results than the other instances.The results are shown in Table 3.

By being able to quantify the diversity of the population in a particular iteration, we can analyze the behavior of the Binary Q-Sine Cosine Algorithm in

	QL1	QL2	QL3	QL4	QL5
QL1	-	>0.005	**0.0484**	**0.0360**	>0.005
QL2	>0.005	-	>0.005	>0.005	>0.005
QL3	>0.005	>0.005	-	>0.005	>0.005
QL4	>0.005	>0.005	>0.005	-	>0.005
QL5	>0.005	>0.005	>0.005	>0.005	-

Table 3: p-value average BQSCA

terms of exploration and exploitation. This information is shown in Figures 1a and Figure 1b. It can be seen that the Learnheuristics on average reach a balance close to 50% of exploration and exploitation but it is not a smooth transition as there are sharp jumps from exploration to exploitation or from exploitation to exploitation. In conclusion, the proposal perturb exploration and exploitation but fails to control it.



(a) BQSCA-QL1 solving instance scpb4 (b) BQSCA-QL3 solving instance scpb2

Fig. 1: BQSCA-QL1 Exploration-Exploitation

By analyzing the results obtained from both Table 4 and the statistical test in Table 3, we can see that BQSCA-QL1 stands out over some proposals. A peculiarity of this instance is that it is the only one that considers negative reward [6, 7, 16]. Experimental results show that considering negative reward "deters" exploration in reinforcement learning algorithm.

6 Conclusion

Learnheuristics improve the balance between exploration and exploitation to obtain high quality solutions. In particular, they perturb the solutions by generating an exploration and exploitation balance but it is not a tenuous and controlled balance. This evidences the great interest of researchers in Learnheuristics or Ambidextrous Algorithms and the great field of research they open due to the two interacting worlds.

Algorithm 1 Binary Q-Sine Cosine Algorithm

Input: The population $X = \{X_1, X_2, ..., X_n\}$
Output: The updated population $X' = \{X'_1, X'_2, ..., X'_n\}$ and X_{best}

Initialize Q-Table with q_0
Initialize random population X
Set initial r_1
Calculate Initial Population Diversity (X)
Define the initial state using equation (12)
for *iteration* (t) **do**
    a **: Select action from Q-Table**
    for *solution* (i) **do**
        Evaluate solution X_i in the objective function
        for *dimension* (j) **do**
            Update P_j^t, where $P_j^t = X_{best,j}$
            Randomly generate the value of r_2, r_3, r_4
            Update the position of $X_{i,j}$
        end for
    end for
    Binarization X **with action** a **and apply reward function**
    Calculate Population Diversity (X)
    Define the next state using equation (12)
    Update Q-Table using equation (1)
    Update r_1
    Update X_{best}
end for
Return the updated population X where X_{best} is the best result

Regarding the reward function, it can be concluded from the statistical evaluation that, to solve the Set Covering Problem with the metaheuristic Sine Cosine Algorithm applying Q-Learning as a binarization scheme selector, there is an influence of these but there is not a better one than the others.

In the future, this research can be extended to solve another optimization problem, apply another metaheuristic or apply another reinforcement learning technique to demonstrate the impact of the reward function in other work contexts.

7 Acknowledgements

Felipe Cisternas-Caneo and Marcelo Becerra-Rozas are supported by Grant DI Investigación Interdisciplinaria del Pregrado/VRIEA/PUCV/039.324/2020. Broderick Crawford and Wenceslao Palma are supported by Grant CONICYT /FONDECYT/REGULAR/1210810. Ricardo Soto is supported by Grant CONICYT/FONDECYT /REGULAR/1190129. Broderick Crawford, Ricardo Soto and Hanns de la Fuente-Mella are supported by Grant Núcleo de Investigación en Data Analytics/VRIEA /PUCV/039.432/2020. José Lemus-Romani is supported by National Agency for Research and Development (ANID)/Scholarship Program/DOCTORADO NACIONAL /2019-21191692.

References

1. Christopher Bayliss, Angel A. Juan, Christine S.M. Currie, and Javier Panadero. A learnheuristic approach for the team orienteering problem with aerial drone motion constraints. *Applied Soft Computing*, 92:106280, 2020.
2. Felipe Cisternas-Caneo, Broderick Crawford, Ricardo Soto, Hanns de la Fuente-Mella, Diego Tapia, José Lemus-Romani, Mauricio Castillo, Marcelo Becerra-Rozas, Fernando Paredes, and Sanjay Misra. A data-driven dynamic discretization framework to solve combinatorial problems using continuous metaheuristics. In *Innovations in Bio-Inspired Computing and Applications*, pages 76–85, Cham, 2021. Springer International Publishing.
3. Broderick Crawford and Claudio León de la Barra. Los algoritmos ambidiestros. `https://www.mercuriovalpo.cl/impresa/2020/07/13/full/cuerpo-principal/15/`, 2020. Acceded 12-02-2021.
4. Broderick Crawford, Ricardo Soto, Gino Astorga, José García, Carlos Castro, and Fernando Paredes. Putting continuous metaheuristics to work in binary search spaces. *Complexity*, 2017, 2017.
5. Marco Dorigo and Luca Maria Gambardella. A study of some properties of ant-q. In *International Conference on Parallel Problem Solving from Nature*, pages 656–665. Springer, 1996.
6. Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.
7. Takayasu Fuchida, Kathy Thi Aung, and Atsushi Sakuragi. A study of q-learning considering negative rewards. *Artificial Life and Robotics*, 15(3):351–354, 2010.
8. Michael R Garey and David S Johnson. Computers and intractability: A guide to the theory of np-completeness, 1979.
9. Kashif Hussain, William Zhu, and Mohd Najib Mohd Salleh. Long-term memory harris' hawk optimization for high dimensional and optimal power flow problems. *IEEE Access*, 7:147596–147616, 2019.
10. Jose M Lanza-Gutierrez, NC Caballe, Broderick Crawford, Ricardo Soto, Juan A Gomez-Pulido, and Fernando Paredes. Exploring further advantages in an alternative formulation for the set covering problem. *Mathematical Problems in Engineering*, 2020, 2020.
11. Alessandro Lazaric, Marcello Restelli, and Andrea Bonarini. Reinforcement learning in continuous action spaces through sequential monte carlo methods. *Advances in neural information processing systems*, 20:833–840, 2007.

		QL1			QL2			QL3			QL4			QL5		
Inst.	Opt.	Best	Avg	RPD	Best	Avg	RPD	Best	Avg	RPD	Best	Avg	RPD	Best	Avg	RPD
41	429	**431.0**	439.19	**0.47**	435.0	442.72	1.4	439.0	444.06	2.33	438.0	444.32	2.1	438.0	444.13	2.1
42	512	**535.0**	549.52	**4.49**	537.0	553.71	4.88	541.0	554.89	5.66	542.0	555.45	5.86	537.0	552.39	4.88
43	516	532.0	545.48	3.1	534.0	552.03	3.49	535.0	552.19	3.68	535.0	550.52	3.68	**527.0**	547.0	**2.13**
44	494	**510.0**	527.48	**3.24**	514.0	530.44	4.05	516.0	530.43	4.45	512.0	531.9	3.64	511.0	530.68	3.44
45	512	532.0	547.45	3.91	537.0	553.17	4.88	**527.0**	549.48	**2.93**	532.0	551.58	3.91	531.0	550.16	3.71
46	560	573.0	585.74	2.32	573.0	588.68	2.32	577.0	588.68	3.04	576.0	591.77	2.86	**568.0**	588.84	**1.43**
47	430	**437.0**	447.26	**1.63**	441.0	449.77	2.56	440.0	449.84	2.33	439.0	449.81	2.09	439.0	451.06	2.09
48	492	502.0	511.55	2.03	509.0	516.39	3.46	**499.0**	513.87	**1.42**	503.0	514.13	2.24	507.0	514.77	3.05
49	641	**672.0**	690.26	**4.84**	683.0	697.48	6.55	685.0	697.42	6.86	686.0	696.55	7.02	676.0	697.52	5.46
410	514	**521.0**	532.0	**1.36**	**521.0**	533.88	**1.36**	529.0	537.26	2.92	526.0	535.58	2.33	522.0	534.52	1.56
51	253	**263.0**	269.13	**3.95**	264.0	272.75	4.35	264.0	272.32	4.35	264.0	273.16	4.35	265.0	273.29	4.74
52	302	326.0	332.55	7.95	327.0	335.58	8.28	**325.0**	334.61	**7.62**	328.0	335.48	8.61	327.0	335.71	8.28
53	226	231.0	234.94	2.21	**230.0**	235.62	**1.77**	231.0	235.81	2.21	232.0	236.29	2.65	**230.0**	235.77	**1.77**
54	242	**250.0**	253.93	**3.31**	**250.0**	254.6	**3.31**	**250.0**	255.39	**3.31**	252.0	255.39	4.13	251.0	254.84	3.72
55	211	216.0	219.85	2.37	218.0	221.46	3.32	217.0	220.87	2.84	**214.0**	221.39	**1.42**	217.0	220.97	2.84
56	213	**217.0**	228.23	**1.88**	221.0	231.26	3.76	225.0	231.23	5.63	223.0	231.9	4.69	220.0	230.9	3.29
57	293	305.0	312.97	4.1	304.0	316.4	3.75	**303.0**	317.03	**3.41**	306.0	318.32	4.44	312.0	317.48	6.48
58	288	**295.0**	299.37	**2.43**	296.0	301.32	2.78	296.0	300.77	2.78	**295.0**	300.9	**2.43**	**295.0**	301.1	**2.43**
59	279	285.0	291.73	2.15	**284.0**	293.42	**1.79**	289.0	293.1	3.58	288.0	294.1	3.23	286.0	293.23	2.51
510	265	**271.0**	278.19	**2.26**	274.0	281.35	3.4	277.0	281.74	4.53	273.0	281.9	3.02	274.0	281.84	3.4
61	138	**141.0**	145.77	**2.17**	144.0	148.16	4.35	144.0	148.42	4.35	146.0	148.29	5.8	146.0	148.65	5.8
62	146	**151.0**	156.26	**3.42**	152.0	159.06	4.11	154.0	158.77	5.48	152.0	158.58	4.11	153.0	158.06	4.79
63	145	**149.0**	151.19	**2.76**	**149.0**	151.29	**2.76**	**149.0**	151.65	**2.76**	**149.0**	151.74	**2.76**	**149.0**	151.74	**2.76**
64	131	**133.0**	135.35	**1.53**	**133.0**	136.03	**1.53**	**133.0**	135.74	**1.53**	134.0	136.29	2.29	134.0	136.32	2.29
65	161	**169.0**	179.87	**4.97**	173.0	183.26	7.45	173.0	183.19	7.45	173.0	182.84	7.45	175.0	182.97	8.7
a1	253	**262.0**	267.24	**3.56**	266.0	269.42	5.14	263.0	268.81	3.95	263.0	268.9	3.95	265.0	269.68	4.74
a2	252	**267.0**	271.65	**5.95**	**267.0**	273.8	**5.95**	269.0	274.16	6.75	269.0	274.1	6.75	268.0	274.23	6.35
a3	232	243.0	247.85	4.74	245.0	248.87	5.6	245.0	249.45	5.6	**242.0**	249.71	**4.31**	**242.0**	248.63	**4.31**
a4	234	**245.0**	250.81	**4.7**	**245.0**	252.61	**4.7**	249.0	252.77	6.41	**245.0**	252.13	**4.7**	**245.0**	252.68	**4.7**
a5	236	245.0	248.92	3.81	247.0	251.27	4.66	245.0	250.48	3.81	245.0	251.29	3.81	**244.0**	251.26	**3.39**
b1	69	**70.0**	71.81	**1.45**	71.0	72.68	2.9	72.0	72.9	4.35	72.0	72.87	4.35	72.0	73.03	4.35
b2	76	**76.0**	80.16	**0.0**	78.0	81.35	2.63	**76.0**	81.13	**0.0**	78.0	81.06	2.63	78.0	81.23	2.63
b3	80	**81.0**	82.85	**1.25**	82.0	83.87	2.5	82.0	83.77	2.5	82.0	83.58	2.5	**81.0**	83.55	**1.25**
b4	79	82.0	83.82	3.8	83.0	84.9	5.06	82.0	84.68	3.8	83.0	84.9	5.06	**81.0**	85.1	**2.53**
b5	72	**73.0**	74.55	**1.39**	**73.0**	75.03	**1.39**	74.0	74.9	2.78	74.0	75.23	2.78	**73.0**	74.84	**1.39**
c1	227	240.0	245.55	5.73	246.0	251.85	8.37	244.0	251.03	7.49	**239.0**	251.0	**5.29**	244.0	250.94	7.49
c2	219	234.0	240.19	6.85	237.0	242.89	8.22	238.0	242.61	8.68	**231.0**	242.65	**5.48**	233.0	242.19	6.39
c3	243	**255.0**	260.97	**4.94**	259.0	263.25	6.58	257.0	262.74	5.76	256.0	263.0	5.35	256.0	263.48	5.35
c4	219	232.0	236.03	5.94	**230.0**	236.1	**5.02**	233.0	236.94	6.39	232.0	237.13	5.94	232.0	236.97	5.94
c5	215	**225.0**	231.62	**4.65**	229.0	234.2	6.51	226.0	233.35	5.12	227.0	233.42	5.58	226.0	233.45	5.12
d1	60	**62.0**	64.42	**3.33**	64.0	65.97	6.67	64.0	66.0	6.67	64.0	65.81	6.67	63.0	66.06	5.0
d2	66	**67.0**	69.52	**1.52**	69.0	69.97	4.55	68.0	69.97	3.03	69.0	70.26	4.55	68.0	70.03	3.03
d3	72	**75.0**	78.0	**4.17**	76.0	78.86	5.56	**75.0**	78.74	**4.17**	**75.0**	78.58	**4.17**	76.0	78.77	5.56
d4	62	**62.0**	64.0	**0.0**	63.0	64.16	1.61	**62.0**	64.13	**0.0**	**62.0**	64.16	**0.0**	63.0	64.29	1.61
d5	61	**63.0**	65.55	**3.28**	64.0	66.35	4.92	**63.0**	66.13	**3.28**	64.0	66.06	4.92	**63.0**	66.16	**3.28**

Table 4: Results obtained by solving instances of OR-library

12. José Lemus-Romani, Broderick Crawford, Ricardo Soto, Gino Astorga, Sanjay Misra, Kathleen Crawford, Giancarla Foschino, Agustín Salas-Fernández, and Fernando Paredes. Ambidextrous socio-cultural algorithms. In *Computational Science and Its Applications - ICCSA 2020 - 20th International Conference, Cagliari, Italy, July 1-4, 2020, Proceedings, Part VI*, volume 12254 of *Lecture Notes in Computer Science*, pages 923–938. Springer, 2020.
13. Seyedali Mirjalili. Sca: a sine cosine algorithm for solving optimization problems. *Knowledge-based systems*, 96:120–133, 2016.
14. Bernardo Morales-Castañeda, Daniel Zaldivar, Erik Cuevas, Fernando Fausto, and Alma Rodríguez. A better balance in metaheuristic algorithms: Does it exist? *Swarm and Evolutionary Computation*, page 100671, 2020.
15. Alexander Nareyek. Choosing search heuristics by non-stationary reinforcement learning. In *Metaheuristics: Computer decision-making*, pages 523–544. Springer, 2003.
16. Xinyan Ou, Qing Chang, and Nilanjan Chakraborty. Simulation study on reward function of reinforcement learning in gantry work cell scheduling. *Journal of Manufacturing Systems*, 50:1 – 8, 2019.
17. Youngsoo Park, Peter Nielsen, and Ilkyeong Moon. Unmanned aerial vehicle set covering problem considering fixed-radius coverage constraint. *Computers Operations Research*, 119:104936, 2020.
18. Barbara M Smith. Impacs-a bus crew scheduling system using integer programming. *Mathematical Programming*, 42(1-3):181–187, 1988.
19. Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
20. Diego Tapia, Broderick Crawford, Ricardo Soto, Felipe Cisternas-Caneo, José Lemus-Romani, Mauricio Castillo, José García, Wenceslao Palma, Fernando Paredes, and Sanjay Misra. A q-learning hyperheuristic binarization framework to balance exploration and exploitation. In *International Conference on Applied Informatics*, pages 14–28. Springer, 2020.
21. Diego Tapia, Broderick Crawford, Ricardo Soto, Wenceslao Palma, José Lemus-Romani, Felipe Cisternas-Caneo, Mauricio Castillo, Marcelo Becerra-Rozas, and Sanjay Misra. Embedding q-learning in the selection of metaheuristic operators: The enhanced binary grey wolf optimizar case. In *Proceeding of 2021 IEEE International Conference on Automation/XXIV Congress of the Chilean Association of Automatic Control (ICA-ACCA), IEEE ICA/ACCA 2021*, page ARTICLE IN PRESS, 2021.
22. Sávio S.V. Vianna. The set covering problem applied to optimisation of gas detectors in chemical process plants. *Computers Chemical Engineering*, 121:388 – 395, 2019.
23. Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.
24. Kamal Z Zamli, Fakhrud Din, Bestoun S Ahmed, and Miroslav Bures. A hybrid q-learning sine-cosine-based strategy for addressing the combinatorial test suite minimization problem. *PloS one*, 13(5):e0195675, 2018.
25. Li Zhang, Brendan Shaffer, Tim Brown, and G. Scott Samuelsen. The optimization of dc fast charging deployment in california. *Applied Energy*, 157:111 – 122, 2015.

The horizontal linear complementarity problem and robustness of the related matrix classes*

Milan Hladík[1][0000−0002−7340−8491] and Miroslav Rada[2][0000−0002−1761−897X]

[1] Charles University, Faculty of Mathematics and Physics, Department of Applied Mathematics, Malostranské nám. 25, 11800, Prague, Czech Republic
hladik@kam.mff.cuni.cz
https://kam.mff.cuni.cz/~hladik

[2] University of Economics, Faculty of Informatics and Statistics, Department of Econometrics, Prague, W. Churchill's Sq. 4, 130 67 Prague, Czech Republic
miroslav.rada@vse.cz

Abstract. We consider the horizontal linear complementarity problem and we assume that the input data have the form of intervals, representing the range of possible values. For the classical linear complementarity problem, there are known various matrix classes that identify interesting properties of the problem (such as solvability, uniqueness, convexity, finite number of solutions or boundedness). Our aim is to characterize the robust version of these properties, that is, to check them for all possible realizations of interval data. We address successively the following matrix classes: nonnegative matrices, Z-matrices, semimonotone matrices, column sufficient matrices, principally nondegenerate matrices, R_0-matrices and R-matrices. The reduction of the horizontal linear complementarity problem to the classical one, however, brings complicated dependencies between interval parameters, resulting in some cases to higher computational complexity.

Keywords: Linear complementarity· Interval analysis· Special matrices· NP-hardness.

1 Introduction

The linear complementarity problem (LCP). The classical LCP problem is a feasibility problem

$$y = Mz + q, \quad y, z \geq 0, \tag{1a}$$

$$y^T z = 0, \tag{1b}$$

where $M \in \mathbb{R}^{n\times n}$ and $q \in \mathbb{R}^n$ are given and $y, z \in \mathbb{R}^n$ are variables. Condition (1a) is linear, but the (nonlinear) complementarity condition (1b) makes the problem NP-hard [2]. The LCP is called *feasible* if (1a) is feasible, and it is

* Supported by the Czech Science Foundation Grants P403-18-04735S (M. Hladík) and P403-20-17529S (M. Rada).

called *solvable* if (1a)–(1b) is feasible. The LCP appears in many optimization and operations research models such as quadratic programming, bimatrix games, or equilibria in specific economies. More one properties and algorithms for LCP see, e.g., the books [4,21].

The horizontal linear complementarity problem. The horizontal LCP [3] is a slight generalization of LCP, first formulated by Samelson et al. [25]. It reads

$$Ay = Bz + q, \quad y, z \geq 0, \tag{2a}$$

$$y^T z = 0, \tag{2b}$$

where $A, B \in \mathbb{R}^{n\times n}$ and $q \in \mathbb{R}^n$ are given and $y, z \in \mathbb{R}^n$ are variables. Clearly, provided A is nonsingular, we easily reduce the problem to LCP by multiplying A^{-1}

$$y = A^{-1}Bz + A^{-1}q, \quad y, z \geq 0, \tag{3a}$$

$$y^T z = 0, \tag{3b}$$

Otherwise, a reduction is possible only under certain conditions [7,28]. In our context, it is better to consider the horizontal form separately since the form $A^{-1}B$ brings complicated correlations into the matrix. Notice that for the horizontal and other extended forms of LCP, certain LCP-related matrix classes were generalized [26]. For the horizontal LCP, also special algorithms were developed [18,19,29].

Interval uncertainty. Properties of the solution sets of LCP and horizontal LCP relate with properties of the constraint matrices. In this paper, we study the situation when the matrix entries are not precisely known, but we have interval type uncertainty. A justifications of using intervals for modelling uncertainty is provided in [14,15] and many books, e.g., [20,22].

Formally, an interval matrix is a set

$$\boldsymbol{A} \coloneqq \{A \in \mathbb{R}^{m\times n};\ \underline{A} \leq A \leq \overline{A}\},$$

where $\underline{A}, \overline{A} \in \mathbb{R}^{m\times n}$, $\underline{A} \leq \overline{A}$, are given matrices and the inequality is understood entrywise. The corresponding midpoint and radius matrices are defined as

$$A_c \coloneqq \frac{1}{2}(\underline{A} + \overline{A}), \quad A_\Delta \coloneqq \frac{1}{2}(\overline{A} - \underline{A}).$$

The set of all interval $m\times n$ matrices is denoted by $\mathbb{IR}^{m\times n}$. Following the notation by Fiedler et al. [5], we introduce special matrices in $\boldsymbol{A}$. Given sign vectors $s \in \{\pm 1\}^m$ and $t \in \{\pm 1\}^n$, denote

$$A_{s,t} = A_c - D_s A_\Delta D_t \in \boldsymbol{A},$$

where D_s stands for the diagonal matrix with entries $s_1, \ldots, s_m$ and similarly for D_t. For more results and properties of interval computation, including interval arithmetic, we refer the readers, e.g., to books [20,22].

The LCP with interval uncertainties was investigated in [1,17], among others. They addressed the problem of computing an outer approximation of the solution set of all possible realizations of interval entries. Our goal is different – in this paper, we focus on the interval matrix properties that are related to the LCP.

Problem statement. Let $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{IR}^{n\times n}$ be given. We consider a class of the horizontal LCP problems with $A \in \boldsymbol{A}$ and $B \in \boldsymbol{B}$. Let $\mathcal{P}$ be a matrix property related to the (horizontal) LCP. We say that $\mathcal{P}$ holds *strongly* if it holds for each $A \in \boldsymbol{A}$ and $B \in \boldsymbol{B}$.

Our aim is to characterize strong versions of several fundamental matrix classes appearing in the context of the (horizontal) LCP. If property $\mathcal{P}$ holds strongly for an interval matrix $\boldsymbol{A}$, then we are sure that $\mathcal{P}$ is provably valid whatever are the true values of the uncertain entries. Therefore, the property holds in a robust sense for the (horizontal) LCP problem.

Notation. We use the shortage $[n] = \{1, \ldots, n\}$. Given a matrix $M \in \mathbb{R}^{n\times n}$ and index sets $I, J \subseteq [n]$, $M_{I,J}$ denotes the restriction of M to the rows indexed by I and the columns indexed by J; it is the empty matrix if I or J is empty. Similarly x_I denotes the restriction of a vector x to the entries indexed by I.

The identity matrix of size n is denoted by I_n, and the spectral radius of a matrix M by $\rho(M)$. The symbol D_s stands for the diagonal matrix with entries $s_1, \ldots, s_n$ and $e = (1, \ldots, 1)^T$ for the vector of ones. The relation $x \gneqq y$ between vectors x, y is defined as $x \geq y$ and $x \neq y$. Inequalities and the absolute value of matrices and vectors are understood entrywise.

2 Particular matrix classes

In the following sections, we consider important classes of matrices appearing in the context of the (horizontal) LCP. We characterize their strong counterparts when entries are interval valued. Other matrix properties were discussed, e.g., in [6,11,12,13,16] and in the context of LCP in Hladík [10].

Basically, due to reduction (3), we will tackle the matrix $A^{-1}B$ for $A \in \boldsymbol{A}$ and $B \in \boldsymbol{B}$. The expression $A^{-1}B$ imposes complicated dependencies between interval parameters, so that is why we have to deal with them carefully. A simple evaluation by interval arithmetic (and estimation of the inverse) leads to (possibly high) overestimation. For the sake of simplicity of exposition, we denote by $\boldsymbol{A}^{-1}\boldsymbol{B}$ the set

$$\{A^{-1}B;\ A \in \boldsymbol{A},\ B \in \boldsymbol{B}\}.$$

Throughout the paper we assume that $\boldsymbol{A}$ is strongly nonsingular. For characterization of nonsingularity and sufficient conditions see [23,24], for instance.

2.1 *S*-matrices

A matrix $M \in \mathbb{R}^{n\times n}$ is an *S-matrix* if there is $x > 0$ such that $Mx > 0$. The importance of this class is that the LCP is feasible for each $q \in \mathbb{R}^n$ if and only if M is an S-matrix.

Proposition 1. *We have that $\boldsymbol{A}^{-1}\boldsymbol{B}$ is strongly S-matrix if and only if the system*

$$A_{s,e}z = B_{-s,e}x,\ x > 0,\ z > 0. \tag{4}$$

is feasible for each $s \in \{\pm 1\}^n$.

Proof. We want to characterize feasibility of

$$(A^{-1}B)x > 0,\ x > 0$$

for each $A \in \boldsymbol{A}$ and $B \in \boldsymbol{B}$. Substitute $y \equiv Bx$ and $z \equiv A^{-1}y$. Then $Az = y = Bx$ and we reduced the problem to strong solvability of the interval system

$$\boldsymbol{A}z = \boldsymbol{B}x,\ x > 0,\ z > 0. \tag{5}$$

By [5,8], we obtain (4). □

The problem of checking strong S-matrix property is computationally intractable, which justifies the exponential formula 4.

Proposition 2. *Checking strong S-matrix property of $\boldsymbol{A}^{-1}\boldsymbol{B}$ is co-NP-hard even in the case when A is real.*

Proof. By [5], checking solvability of the system

$$|Mx| \le e,\ e^T|x| \ge 1 \tag{6}$$

is NP-hard even on a class of problems with M nonnegative positive definite. This is equivalent to weak solvability (i.e., solvability for at least one realization) of the interval system

$$|Mx| \le e,\ [-e,e]^T x \ge 1,$$

or to weak solvability of

$$Mx \le ey,\ -Mx \le ey,\ [-e,e]^T x \ge y,\ y > 0.$$

By Farkas' lemma, it is equivalent to the situation that the interval system

$$\begin{pmatrix} -M & M & [-e,e] \\ e^T & e^T & -1 \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \end{pmatrix},\quad u,v,w \ge 0 \tag{7}$$

is not strongly solvable. We claim that strong solvability of this system is equivalent to strong solvability of

$$\begin{pmatrix} 0 & -M & M & [-e,e] \\ 1 & e^T & e^T & -1 \end{pmatrix} \begin{pmatrix} z \\ u \\ v \\ w \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix},\quad z,u,v,w > 0. \tag{8}$$

If (8) has a solution (z, u, v, w), then $\frac{1}{z}(u, v, w)$ solves (7). Conversely, let (u, v, w) be a solution of (7). Situation $w = 0$ cannot happen since otherwise the second equation is violated. Thus we have $w > 0$. If $u, v > 0$, then we put $z := 1$ and (z, u, v, w) solves (8). Otherwise we put $u := u + e$, $v := v + e$ and $z := 2n + 1$ and (z, u, v, w) solves (8).

Eventually, we obtained the interval system in the form of (5), where

$$A = \begin{pmatrix} 0 & -M \\ 1 & e^T \end{pmatrix}, \quad \boldsymbol{B} = \begin{pmatrix} -M & [-e, e] \\ -e^T & 1 \end{pmatrix}.$$

Obviously, A is nonsingular. □

2.2 Nonnegative matrices

Nonnegative matrices are important in the LCP since they represent an efficiently recognizable subclass of copositive matrices.

Proposition 3. *We have that $\boldsymbol{A}^{-1}\boldsymbol{B}$ is strongly nonnegative if and only if the system*

$$A_{s,e}X = B_{-s,e}, \quad X \geq 0 \tag{9}$$

is feasible for each $s \in \{\pm 1\}^n$.

Proof. We need to characterize feasibility of

$$AX = B, \quad X \geq 0$$

for each $A \in \boldsymbol{A}$ and $B \in \boldsymbol{B}$. Thus we arrived at strong solvability of the interval matrix system. Fortunately, for strong solvability, the fact that the system is a matrix equation system makes no harm and we can simply call the characterization from [8], producing (9). □

It is an open problem if checking strong nonnegativity is intractable; we suspect it is.

2.3 Z-matrices

A matrix $M \in \mathbb{R}^{n \times n}$ is called a *Z-matrix* if $m_{ij} \leq 0$ for each $i \neq j$. Z-matrices emerge in the context of Lemke's complementary pivot algorithm, because it processes any LCP with a Z-matrix.

Recall that D_y denotes the diagonal matrix with entries $y_1, \ldots, y_n$, which in the following proposition play the role of variables.

Proposition 4. *We have that $\boldsymbol{A}^{-1}\boldsymbol{B}$ is strongly a Z-matrix if and only if the system*

$$A_{s,e}X - A_{-s,e}D_y = B_{s,e}, \quad X \leq 0, \; y \leq 0 \tag{10}$$

is feasible for each $s \in \{\pm 1\}^n$.

Proof. We need to characterize feasibility of

$$AX = B, \quad X_{ij} \le 0 \; i \ne j$$

for each $A \in \boldsymbol{A}$ and $B \in \boldsymbol{B}$. We express the diagonal of X as a difference of two nonpositive variables, which is an equivalent operation in view of [9]. Thus in matrix form we have $X \mapsto X - D_y$, where $X \le 0$ and $y \le 0$. By [8] strong feasibility of this system is equivalent to feasibility of (10) for each $s \in \{\pm 1\}^n$. □

2.4 Semimonotone matrices

A matrix $M \in \mathbb{R}^{n\times n}$ is *semimonotone* if for each $x \gneqq 0$ there is k such that $x_k > 0$ and $(Mx)_k \ge 0$. By [4], we can state two equivalent conditions of semimonotonicity. First, the LCP has a unique solution for each $q > 0$. Second, for each index set $\emptyset \ne I \subseteq [n]$ the system

$$M_{I,I}x < 0, \quad x \ge 0 \tag{11}$$

is infeasible. From the computational complexity perspective, checking whether M is semimonotone is a co-NP-hard problem [27].

Proposition 5. *We have that $\boldsymbol{A}^{-1}\boldsymbol{B}$ is strongly semimonotone if and only if for each index set $\emptyset \ne I \subseteq [n]$ the system*

$$\overline{A}_{[n],I}z_I + (A_{[n],J})_{-e,s}z_J \le \overline{B}_{[n],I}x, \tag{12a}$$

$$\underline{A}_{[n],I}z_I + (A_{[n],J})_{e,s}z_J \ge \underline{B}_{[n],I}x, \tag{12b}$$

$$D_s z_J \le 0, \; z_I < 0, \; x \ge 0 \tag{12c}$$

is infeasible for each $s \in \{\pm 1\}^{|J|}$, where $J = [n] \setminus I$.

Proof. Let $\emptyset \ne I \subseteq [n]$. We need to characterize infeasibility of

$$(A^{-1}B)_{I,I}x < 0, \quad x \ge 0. \tag{13}$$

Substitute $y \equiv B_{[n],I}x$ and $z \equiv A^{-1}y$. Then $Az = y = B_{[n],I}x$. Since $(A^{-1}B)_{I,I} = A^{-1}_{I,[n]}B_{[n],I}$, we can equivalently write (13) as follows

$$Az = B_{[n],I}x, \; z_I < 0, \; x \ge 0.$$

Since this system should be infeasible for each $A \in \boldsymbol{A}$ and $B \in \boldsymbol{B}$, we obtain by [5,8] the characterization (12). □

2.5 Principally nondegenerate matrices

A matrix $M \in \mathbb{R}^{n\times n}$ is *principally nondegenerate* if all its principal minors are nonzero. A principally nondegenerate matrix implies that the problem has finitely many solutions (including zero) for every $q \in \mathbb{R}^n$.

Proposition 6. *We have that $\boldsymbol{A}^{-1}\boldsymbol{B}$ is strongly principally nondegenerate if and only if for each index set $\emptyset \neq I \subseteq [n]$ and $s \in \{\pm 1\}^n$ the system*

$$(A_{[n],I})_{s,e} + (A_{[n],J})_{s,e}Z^1 - (A_{[n],J})_{-s,e}Z^2 = (B_{[n],I})_{s,-e}X^1 - (B_{[n],I})_{s,-e}X^2, \tag{14a}$$

$$Z^1, Z^2, X^1, X^2 \geq 0. \tag{14b}$$

is feasible, where $J = [n] \setminus I$.

Proof. Let $\emptyset \neq I \subseteq [n]$ and $k = |I|$. We need to characterize regularity of $(\boldsymbol{A}^{-1}\boldsymbol{B})_{I,I}$. For any particular instance, the system

$$A^{-1}_{I,[n]}B_{[n],I}X = I_k$$

should be feasible. Substitute $Y \equiv B_{[n],I}X$ and $Z \equiv A^{-1}_{J,[n]}Y$. Then $A_{[n],I} + A_{[n],J}Z = Y = B_{[n],I}X$. Thus we arrive at strong solvability of the interval matrix system

$$\boldsymbol{A}_{[n],I} + \boldsymbol{A}_{[n],J}Z = \boldsymbol{B}_{[n],I}X.$$

By [5,8], we obtain the characterization (14). □

2.6 Column sufficient matrices

A matrix $M \in \mathbb{R}^{n\times n}$ is *column sufficient* if for every $x \in \mathbb{R}^n$

$$[x_i(Mx)_i \leq 0 \ \forall i] \quad \Rightarrow \quad [x_i(Mx)_i = 0 \ \forall i].$$

Equivalently, by [4], for each pair of disjoint index sets $I, J \subseteq [n]$, $I \cup J \neq \emptyset$, the system

$$\begin{pmatrix} M_{I,I} & -M_{I,J} \\ -M_{J,I} & M_{J,J} \end{pmatrix} x \lneqq 0, \quad x > 0 \tag{15}$$

is infeasible. Notice that the above constraint matrix reduces to $M_{J,J}$ when $I = \emptyset$, and similarly it reduces to $A_{I,I}$ when $J = \emptyset$.

It is known [27] that checking column sufficiency is a co-NP-hard problem, which justifies necessity of inspecting all index sets I, J in (15). In the context of LCP, column sufficiency guarantees that for any $q \in \mathbb{R}^n$ the solution set of the LCP is a convex set (including possibly the empty set).

Proposition 7. *We have that $\boldsymbol{A}^{-1}\boldsymbol{B}$ is strongly column sufficient if and only if the system*

$$\overline{A}_{[n],I}z_I + \underline{A}_{[n],J}z_J \leq \overline{B}_{[n],I}x_I - \underline{B}_{[n],J}x_J, \tag{16a}$$

$$\underline{A}_{[n],I}z_I + \overline{A}_{[n],J}z_J \geq \underline{B}_{[n],I}x_I - \overline{B}_{[n],J}x_J, \tag{16b}$$

$$z_I \leq 0, z_J \geq 0, \ z \neq 0, \ x > 0. \tag{16c}$$

is infeasible for each admissible I, J.

Proof. Let admissible I, J be given. We want to characterize infeasibility of system

$$\begin{pmatrix} (A^{-1}B)_{I,I} & -(A^{-1}B)_{I,J} \\ -(A^{-1}B)_{J,I} & (A^{-1}B)_{J,J} \end{pmatrix} \begin{pmatrix} x_I \\ x_J \end{pmatrix} \lneqq 0, \quad x > 0.$$

Substitute

$$y \equiv B_{[n],I}x_I - B_{[n],J}x_J, \quad z \equiv A^{-1}y.$$

Then $Az = y = B_{[n],I}x_I - B_{[n],J}x_J$, so one can write the system as follows

$$Az = B_{[n],I}x_I - B_{[n],J}x_J, \quad z_I \le 0, z_J \ge 0, \ z \ne 0, \ x > 0.$$

By means of [5,8], infeasibility of this system for each $A \in \boldsymbol{A}$ and $B \in \boldsymbol{B}$ is characterized by (16). □

Notice that system (14) can easily be expressed as a system of linear inequalities. So checking its feasibility is a tractable problem by means of linear programming (for fixed I, J).

2.7 R_0-matrices

A matrix $M \in \mathbb{R}^{n\times n}$ is an *R_0-matrix* if the LCP with $q = 0$ has only the trivial solution $y = z = 0$. Equivalently, for each index set $\emptyset \ne I \subseteq [n]$, the system

$$A_{I,I}x = 0, \quad A_{J,I}x \ge 0, \quad x > 0 \tag{17}$$

is infeasible, where $J = [n] \setminus I$. The decision problem of a given matrix to be a R_0-matrix is a co-NP-hard [27] problem. If M is an R_0-matrix, then for any $q \in \mathbb{R}^n$ the LCP has a bounded solution set.

Proposition 8. *We have that $\boldsymbol{A}^{-1}\boldsymbol{B}$ is strongly R_0-matrix if and only if system*

$$\underline{A}_{[n],J}z_J \le \overline{B}_{[n],I}x_I, \tag{18a}$$
$$\overline{A}_{[n],J}z_J \ge \underline{B}_{[n],I}x_I, \tag{18b}$$
$$z_J \ge 0, \ x > 0 \tag{18c}$$

is infeasible for each admissible I, J.

Proof. Let admissible I, J be given. We want to characterize infeasibility of system

$$(A^{-1}B)_{I,I}x = 0, \quad (A^{-1}B)_{J,I}x \ge 0, \quad x > 0.$$

Substitute $y \equiv B_{[n],I}x_I$ and $z \equiv A^{-1}y$. Then $Az = y = B_{[n],I}x_I$ and the system reads

$$Az = B_{[n],I}x_I, \quad z_I = 0, \ z_J \ge 0, \ x > 0$$

or

$$A_{[n],J}z_J = B_{[n],I}x_I, \quad z_J \geq 0,\ x > 0.$$

This system is infeasible for each $A \in \boldsymbol{A}$ and $B \in \boldsymbol{B}$ if and only if system(18) is infeasible; see [5,8]. □

2.8 $\boldsymbol{R}$-matrices

A matrix $M \in \mathbb{R}^{n\times n}$ is an *R-matrix* if for each index set $\emptyset \neq I \subseteq [n]$, the system

$$M_{I,I}x + et = 0, \quad M_{J,I}x + et \geq 0, \quad x > 0,\ t \geq 0 \tag{19}$$

is infeasible w.r.t. variables $x \in R^{|I|}$ and $t \in \mathbb{R}$, where $J = [n] \setminus I$. In the context of the LCP, when M is an R-matrix, then for any $q \in \mathbb{R}^n$ the LCP has a solution.

Despite the (visual) similarity with R_0-matrix, the R-matrix property is much harder in the interval setting and for particular index sets I, J.

Proposition 9. *We have that $\boldsymbol{A}^{-1}\boldsymbol{B}$ is strongly R_0-matrix if and only if system*

$$-\overline{A}_{[n],I}et + (A_{[n],J})_{e,s}z_J \leq \overline{B}_{[n],I}x_I, \tag{20a}$$
$$-\underline{A}_{[n],I}et + (A_{[n],J})_{e,-s}z_J + et \geq \underline{B}_{[n],I}x_I, \tag{20b}$$
$$D_s z_J \geq 0,\ z_J + et \geq 0,\ x > 0,\ t \geq 0 \tag{20c}$$

is infeasible for each admissible I, J and $s \in \{\pm 1\}^{|J|}$.

Proof. Let admissible I, J be given. We want to characterize infeasibility of system

$$(A^{-1}B)_{I,I}x + et = 0, \quad (A^{-1}B)_{J,I}x + et \geq 0, \quad x > 0,\ t \geq 0.$$

Substitute $y \equiv B_{[n],I}x_I$ and $z \equiv A^{-1}y$. Then $Az = y = B_{[n],I}x_I$ and the system reads

$$Az = B_{[n],I}x_I, \quad z_I + et = 0,\ z_J + et \geq 0,\ x > 0$$

or

$$-A_{[n],I}et + A_{[n],J}z_J = B_{[n],I}x_I, \quad z_J + et \geq 0,\ x > 0,\ t \geq 0.$$

This system is infeasible for each $A \in \boldsymbol{A}$ and $B \in \boldsymbol{B}$ if and only if system(20) is infeasible for each $s \in \{\pm 1\}^{|J|}$; see [5,8]. □

3 Conclusion

We considered various matrix classes that appear in the context of the LCP and ensure that the problem has favourable properties (in view of its solvability and properties of the solution set). We fully characterized stability of these matrices on an interval domain and in the case the matrices originate from the horizontal LCP. Practically it brings characterization of robustness of these matrices because whatever are the realizations of the interval data, we are sure that the corresponding property is satisfied.

Several open problems emerged, too. For copositivity and P-matrix property, we presented no closed form characterization and we leave it for future research. Next, notice that many matrix properties are computationally hard to verify even in the real case, so the interval case cannot be easier. Therefore it would be interesting to investigate some polynomially recognizable cases or to come up with suitable sufficient conditions.

References

1. Alefeld, G., Schäfer, U.: Iterative methods for linear complementarity problems with interval data. Comput. **70**(3), 235–259 (Jun 2003)
2. Chung, S.J.: NP-completeness of the linear complementarity problem. J. Optim. Theory Appl. **60**(3), 393–399 (1989)
3. Cottle, R.W.: Linear complementarity since 1978. In: Giannessi, F., Maugeri, A. (eds.) Variational Analysis and Applications, NOIA, vol. 79, pp. 239–257. Springer, Boston (2005)
4. Cottle, R.W., Pang, J.S., Stone, R.E.: The Linear Complementarity Problem. SIAM, Philadelphia, PA, revised ed. of the 1992 original edn. (2009)
5. Fiedler, M., Nedoma, J., Ramík, J., Rohn, J., Zimmermann, K.: Linear Optimization Problems with Inexact Data. Springer, New York (2006)
6. Garloff, J., Adm, M., Titi, J.: A survey of classes of matrices possessing the interval property and related properties. Reliab. Comput. **22**, 1–10 (2016)
7. Gowda, M.: Reducing a monotone horizontal LCP to an LCP. Appl. Math. Lett. **8**(1), 97–100 (1995)
8. Hladík, M.: Weak and strong solvability of interval linear systems of equations and inequalities. Linear Algebra Appl. **438**(11), 4156–4165 (2013)
9. Hladík, M.: Transformations of interval linear systems of equations and inequalities. Linear Multilinear Algebra **65**(2), 211–223 (2017)
10. Hladík, M.: Stability of the linear complementarity problem properties under interval uncertainty (2019), `https://arxiv.org/abs/1908.07925`, preprint arXiv: 1908.07925
11. Hladík, M.: Tolerances, robustness and parametrization of matrix properties related to optimization problems. Optim. **68**(2-3), 667–690 (2019)
12. Hladík, M.: An overview of polynomially computable characteristics of special interval matrices. In: Kosheleva O et al (ed.) Beyond Traditional Probabilistic Data Processing Techniques: Interval, Fuzzy etc. Methods and Their Applications, Studies in Computational Intelligence, vol. 835, pp. 295–310. Springer, Cham (2020)

13. Horáček, J., Hladík, M., Černý, M.: Interval linear algebra and computational complexity. In: Bebiano, N. (ed.) Applied and Computational Matrix Analysis, Springer Proceedings in Mathematics & Statistics, vol. 192, pp. 37–66. Springer (2017)
14. Kreinovich, V.: Why intervals? A simple limit theorem that is similar to limit theorems from statistics. Reliab. Comput. **1**(1), 33–40 (1995)
15. Kreinovich, V.: Why intervals? Why fuzzy numbers? Towards a new justification. In: Mendel, J.M., Omori, T., Ya, X. (eds.) 2007 IEEE Symposium on Foundations of Computational Intelligence. pp. 113–119 (2007)
16. Kreinovich, V., Lakeyev, A., Rohn, J., Kahl, P.: Computational Complexity and Feasibility of Data Processing and Interval Computations. Kluwer, Dordrecht (1998)
17. Ma, H.q., Xu, J.p., Huang, N.j.: An iterative method for a system of linear complementarity problems with perturbations and interval data. Appl. Math. Comput. **215**(1), 175–184 (2009)
18. Mezzadri, F., Galligani, E.: Splitting methods for a class of horizontal linear complementarity problems. J. Optim. Theory Appl. **180**(2), 500–517 (2019)
19. Mezzadri, F., Galligani, E.: A modulus-based nonsmooth Newton's method for solving horizontal linear complementarity problems. Optim. Lett. (2020), in press, DOI: 10.1007/s11590-019-01515-9
20. Moore, R.E., Kearfott, R.B., Cloud, M.J.: Introduction to Interval Analysis. SIAM, Philadelphia, PA (2009)
21. Murty, K.G., Yu, F.T.: Linear Complementarity, Linear and Nonlinear Programming. Internet edn. (1997)
22. Neumaier, A.: Interval Methods for Systems of Equations. Cambridge University Press, Cambridge (1990)
23. Rex, G., Rohn, J.: Sufficient conditions for regularity and singularity of interval matrices. SIAM J. Matrix Anal. Appl. **20**(2), 437–445 (1998)
24. Rohn, J.: Forty necessary and sufficient conditions for regularity of interval matrices: A survey. Electron. J. Linear Algebra **18**, 500–512 (2009)
25. Samelson, H., Thrall, R.M., Wesler, O.: A partition theorem for euclidean n-spaces. Proc. Am. Math. Soc. **9**, 805–807 (1958)
26. Sznajder, R., Gowda, M.: Generalizations of P_0- and P-properties; extended vertical and horizontal linear complementarity problems. Linear Algebra Appl. **223-224**, 695–715 (1995)
27. Tseng, P.: Co-NP-completeness of some matrix classification problems. Math. Program. **88**(1), 183–192 (June 2000)
28. Tütüncü, R.H., Todd, M.J.: Reducing horizontal linear complementarity problems. Linear Algebra Appl. **223-224**, 717–729 (1995)
29. Zheng, H., Vong, S.: On convergence of the modulus-based matrix splitting iteration method for horizontal linear complementarity problems of H_+-matrices. Appl. Math. Comput. **369**, 124890:1–6 (2020)

Deep Reinforcement Learning for Dynamic Pricing of Perishable Products

Vibhati Burman, Rajesh Kumar Vashishtha, Rajan Kumar *, Sharadha Ramanan

TCS Research, Chennai
{vibhati.b,r.vashishtha,sharadha.ramanan}@tcs.com

Abstract. Dynamic pricing is a strategy for setting flexible prices for products based on existing market demand. In this paper, we address the problem of dynamic pricing of perishable products using DQN value function approximator. A model-free reinforcement learning approach is used to maximize revenue for a perishable item with fixed initial inventory and selling horizon. The demand is influenced by the price and freshness of the product. The conventional tabular Q-learning method involves storing the Q-values for each state-action pair in a lookup table. This approach is not suitable for control problems with large state spaces. Hence, we use function approximation approach to address the limitations of a tabular Q-learning method. Using DQN function approximator we generalize the unseen states from the seen states, which reduces the space requirements for storing value function for each state-action combination. We show that using DQN we can model the problem of pricing perishable products. Our results demonstrate that the DQN based dynamic pricing algorithm generates higher revenue when compared with conventional one-step price optimization and constant pricing strategy.

Keywords: Dynamic pricing · Deep reinforcement learning · Perishable items · Retail · Grocery · Fashion industry · Deep Q-network · Revenue management.

1 Introduction

Dynamic pricing, also referred as revenue management, is a strategy to adjust the selling prices of products at the right time for maximizing revenue under changing circumstances. These changing circumstances are the factors which affect the demand and supply. Examples include amount of inventory available, age of the product, weather and customer preferences.

Perishable items are those likely to spoil after a fixed time period. The scope of perishable item spans various industries including grocery, pharmaceuticals, fashion and airlines. The consumer spend on perishables is increasing and expected to further increase in the next few years. In particular, the demand for

* Rajan was an employee of TCS when this work was done.

perishable food items such as vegetables, fruits, milk and eggs, are continuing to increase as consumers are progressively becoming health conscious. In case of fashion items, the demand is often short-lived due to several factors including the impact of social media. The retailers are under great pressure to price perishables optimally to offload inventory as well as maximize overall revenue. Hence, there is a need to develop a dynamic pricing policy.

Feng et al. [8] developed an economic order quantity (EOQ) inventory model for perishable items and hypothesize that for perishable products the demand is dependent on its price, freshness and stock level. They have considered the following characteristics for perishable food items:

1. Demand for a perishable product is dependent on its price, age and inventory level.
2. The age of the product not only reduces the stocks but also decreases the demand rate.
3. Product can not be sold after its expiration date.

Some work has been done for dynamic pricing through reinforcement learning (RL), but there still exist some challenges. In this paper, we identify and address three important challenges. First, the existing work on dynamic pricing of perishable products using RL employ based tabular Q-learning approach. Pricing policy using tabular Q-learning approach cannot be generalized to previously unseen scenarios, i.e., we cannot estimate a pricing policy for a state unless it has been visited several times. Second, the exisiting literature on dynamic pricing through RL makes use of incremental method of learning a policy [22]. Incremental methods are not sample efficient and may lead to slower convergence of policy. Third, most of the existing work on dynamic pricing makes use of myopic approaches that try to optimize the immediate revenue. Harrison et al. [14] showed that myopic policies can lead to incorrect policies. Furthermore, Ravi Ganti et al. [12] have shown that far-sighted policies lead to increased profit in the long term.

In this paper, we have made the following important and original contributions in order to overcome the aforementioned challenges. First, we make use of a model-free DQN function approximator [26, 18]. By making use of a function approximator we address the problem of pricing for unseen states. Also, using a function approximator drastically reduces the space requirement for obtaining the pricing policy. To the best of our knowledge, this is the first application of DQN for dynamic pricing of perishables. Second, reinforcement learning provides an alternative approach for optimizing the revenue over the entire selling horizon of the item. This leads to better proftability in the long run.

The remainder of this paper is organized as follows. Section 2 provides a review of related work. Section 3 gives some background on reinforcement learning. Section 4 discusses MDP formulation. We discuss the methodology in Section 5. Experimental results and findings are explained in Section 6.

2 Related Work

In this section, we review literature relevant to our work. Dynamic pricing has been receiving a lot of attention due to increase in competition in the market and advancement of AI based modeling. Gallego et al. [11] formulated dynamic pricing model for perishables having stochastic demand with an arrival rate as a function of price over finite horizon. Bitran et al. [3] extended this work by considering demand as a Poisson process with an arrival rate as a function of general purchasing patterns. Feng et al. [10, 9] considered demand explicitly as a multivariate function of price, freshness and displayed stocks to obtained optimal price. Lu et al. [19] have maximized total profit to obtain the optimal joint dynamic pricing and replenishment policy for perishable items by applying pontryagins maximum principle. Duan et al. [7] proposed a dynamic pricing model for perishable food with quantity and quality deteriorating simultaneously. Their demand depends on the quality, the sales price and the reference price. They formulated an optimal control model to maximize the total profit and solved it by applying pontryagins maximum principle. Diaz et al. [1] studied the relation between dynamic price strategy and relevant factors such as price elasticity of demand, age-sensitivity of demand and age profile of initial inventory for perishables. They proposed a deterministic mathematical model that studied the influence of these factors on revenue and spoilage. Xiong et al. [27] studied the dynamic pricing problem of selling fixed stock of perishable items over a finite horizon, using fuzzy variables to model uncertain demand. They claim the effectiveness and robustness of their algorithm using a real world example. Robust optimization methods have been used to address dynamic pricing for perishable products [17, 21, 2]. These are one step optimization methods that result in myopic and static solutions, which focus only on maximizing immediate revenue rather than for a long term.

Dynamic pricing problem have also been addressed by using Reinforcement learning algorithms. Gosavi et al. [13] have used reinforcement learning as stochastic optimization for dynamically pricing the airline tickets. They formulated a semi-Markov Decision problem for their single leg problem over an infinite time horizon by involving some of the important factors affecting the pricing of tickets. Raju et al. [13] have developed dynamic pricing model for single seller and two seller maket scenario. Customer segmentation is an important aspect for their model. For single seller market, pricing decision is taken using Q-learning algorithm while for 2 seller market, actor-critic algorithm is used to decide the optimal price. Cheng [5] integrated real-time demand learning with look up table based Q-learning algorithm to optimaly price the identical products by a deadline. Rana et al. [22] used Q-learning and Q-leaning with eligibility traces (Q(λ)) to establish a pricing policy for products having fixed inventory and fixed time horizon. They defined state as remaining inventory, action as set of price points and reward as revenue. Rana et al. [23] established a pricing policy for inter-dependent perishable items or services. Inter-dependent products are the ones whose demand and prices are affected by one another. Q(λ) is used to obtain the optimal policy. The Markov Decision Process (MDP) formulation

is done as: set of all possible amount of inventory available for each item is considered as state, set of price points are actions and total revenue gathered is considered as reward. Chen et al. [4] used Q-learning algorithm for dynamic pricing of perishables in a competitive multi-agent retailer market. However, they pointed out that their pricing strategy was not always optimal in every market. There are also several papers that uses multi-agent reinforcement learning to learn the optimal pricing strategy [16, 15].

3 MDP formulation for dynamic pricing of perishables

We consider the dynamic pricing problem of a single perishable product, with a given initial inventory. The objective is to price the product dynamically so as to maximize the total expected revenue over a finite selling horizon.

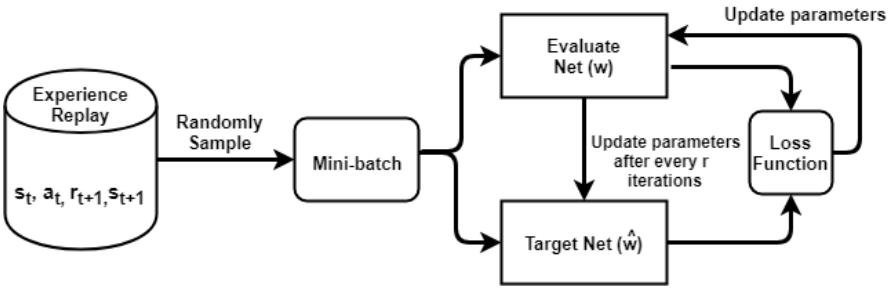


Fig. 1: DQN methodology

We formulate the dynamic pricing problem as a Markov Decision Process. Since we do not know the environment dynamics, we use a model-free reinforcement learning approach to solve the MDP. Specifically, we use DQN value function approximator to solve the MDP and develop an efficient pricing method.

A reinforcement learning task that satisfies the Markov property is called a Markov Decision Process, or MDP [24]. The task is a finite MDP if the state and action spaces are finite. A finite MDP is defined by set of states, set of actions and the one-step environment dynamics. In our MDP, the agent (decision maker) will choose a pricing action from the set A (action set) at each time step t. Since the environments response at time $t+1$ depends only on the state-action representation at time step t, the sequence of states s_t, with t= 1, 2, 3,...,m satisfies the Markov property.

We are dealing with a finite horizon problem that is treated as an episodic task in reinforcement learning. Each episode ends in a terminal state, which occurs at the end of the product lifetime. After termination of an episode, the state values are reset to the default initial value.

Selling horizon is the period of time during which the product is sold. The selling horizon is separated into m discrete decision times. Let t= 1, 2, 3,...,m denote the index of decision time. A pricing decision is made at the beginning of every decision time.

The elements of the MDP are described below:

1) **State**: The state is formulated as a vector of length two. The state at any time step t, for a product, is given by s_t=(inventory left, time since product launch). Here, the time since the product launch is considered as the age of the product. The pricing action is selected based on the current state. The current state is determined by the amount of inventory remaining and the age of the product. This state formulation contains all information about the past agent-environment interaction and thus, follows the Markov property.

2) **Action**: The agent transitions between different states by performing actions. Actions represents the choices the agent makes based on the current state. Here, the actions space includes all the discrete prices the product can be assigned. The action space A is given by $A = \{a_1, a_2, .., a_n\}$ for a product, where n is the number of discrete actions the agent can execute. Selecting a pricing action affects the future state of the environment and thus requires foresightedness.

3) **Reward**: The reward obtained at time $t + 1$ is determined by the state s_t and action a_t at time step t. In our case, the reward is given as the revenue generated by taking an action a_t at time step t. The reward, $r_{t+1} = a_t * demand_t$, is the product of price and demand. Demand is the units sold. We aim to maximize the expected return i.e., cummulative sum of rewards.

4 Methodology

Although classical reinforcement learning algorithms such as Q-learning and SARSA have been applied successfully for various applications [25, 20] in the past, these algorithms fail to scale up for real life problems involving large state spaces. A classical RL algorithm makes use of lookup table for storing the Q-values for every state-action combination it encounters. The size of this lookup table grows in proportion to the size of state space. Also, the information obtained from one state-action pair can not be propagated to other state-action pairs. So, rather than remembering the solutions, we are finding a general function to estimate the Q-value of a state-action pair. Neural networks, being a good function approximator, have been used here. Recently, a neural network based Deep Q-network or DQN technique proposed by Mnih et al. [26] was shown to successfully learn control policies from large state spaces. They applied DQN method to Atari games and demonstarted that it outperforms all previous approaches. We have used a DQN with similar structure as presented in [26] for dynamic pricing of perishables.

When reinforcement learning control algorithms are used with a non-linear function approximator, like a neural network, they are liable to instability. To improve the stability of the DQN, two neural networks of the same structure are used [26], evaluate network and target network. Target network is used

for generating the targets y_j in the Q-learning updates. The target network's parameters, $\hat{w}$, remain fixed for r steps after which they are updated to the latest values of the evaluate network parameters w. The evaluate network parameters, w, are updated at every time step.

Experience replay is another key feature of the DQN, introduced by Mnih et al. [26], that enhances its stability. The last N transitions of the form $(s_t, a_t, r_{t+1}, s_{t+1})$ are stored in memory. Here, s_t, a_t are the state, action at time step t, and r_{t+1}, s_{t+1} is the reward and state at time step $t+1$. From these N transitions, we randomly sample a mini-batch of D transitions. By sampling uniformly from a large memory, we can avoid the temporal correlations and make the data nearly i.i.d.

The methodology used for training our DQN based dynamic pricing algorithm is shown in Figure 1. We use an architecture where the input to the neural network is the state representation. The output layer consist of a separate neuron for each discrete action. Each output neuron represents the predicted state action value function $(Q(s, a))$. The Q values for all the actions for a given state are calculated in one forward pass of the neural network. A linear activation is used for the output layer and rectified linear unit, $(ReLU)$, is used as activation function for the hidden layers.

The loss function used for Q-learning update is shown in equation 1. Adam optimizer is used for updating the network weights.

$$L = \frac{1}{2n} \sum_{i=1}^{n} [\hat{Q}(s_t, a_t^i; \hat{w}) - Q(s_t, a_t^i; w)]^2 \tag{1}$$

where n is the total number of available control actions , Q is the action-value function of the evaluate network, $\hat{Q}$ is the action-value function of the target network.

Learning: The learning of DQN is outlined in Algorithm 1. The training of the DQN takes place offline. The algorithm starts by randomly initiaizing the parameters w of the evaluate network and the target network's parameters $\hat{w}$. We then initialize a zero filled replay memory D of size (N, (length of state vector*2)+2). The outer for loop controls the number of simulation episodes M. We reset the state to its initial value after reaching the end of an episode. Each episode lasts for a maximum of T time steps. Within each episode, we follow the following procedure. At each time step, an action a_t is selected at random with a probability ϵ. Otherwise, action is selected greedily. Next, the sale at a particular pricing action is given by the demand function. The reward r_{t+1} is the revenue generated by taking action a_t at state s_t. The inventory is now updated to obtain the next state s_{t+1}.The state transition tuple $(s_t, a_t, r_{t+1}, s_{t+1})$ is stored in the replay memory D. Then a mini batch is randomly sampled from this replay memory. The target vector is calculated as given in line 15 of Algorithm 1. The parameters w of evaluate neural network Q are updated by using the Adam optimizer. The parameters $\hat{w}$ of the target network remain fixed for r time steps after which they are updated to most recent value of evaluate network's parameters w.

Algorithm 1 DQN based pricing algorithm

Initialize replay memory D to size N
Initialize action-value function Q (evaluate network) with random weights w
Initialize target action-value function $\hat{Q}$ (target network) with random weights $\hat{w} = w$
for i= 1 to M **do**
  Reset state to initial value ($s_{t=1}$ = (initial inventory, 1))
  for t= 1 to T **do**
    With probability ε select a random action a_t
    Otherwise, select $a_t = argmax_a\ Q(s_t, a; w)$
    Execute action a_t
    sale = min (inventory, demand(a_t,t))
    reward: r_{t+1} = sale*a_t
    inventory=inventory-sale
    store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in D.
    randomly sample minibatch of transitions $(s_j, a_j, r_{j+1}, s_{j+1})$ from D

$$Set\ y_j = \begin{cases} r_{j+1}, \text{if } episode\ terminates\ at\ step\ j+1 \\ r_{j+1} + \gamma \max_{a'} \hat{Q}(s_{j+1}, a'; \hat{w}), \quad \text{otherwise} \end{cases}$$

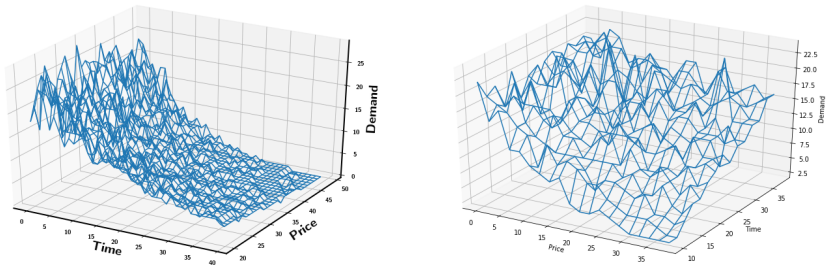
    Perform a gradient descent step on $[y_j - Q(s_j, a_j; w)]^2$ with respect to the evaluate net parameter w.
    every r steps, reset $\hat{w} = w$
  end for
end for

5 Experimental Results

In this section, numerical results are provided to evaluate the performance of our dynamic pricing method using DQN. We study the results of the DQN based pricing model by simulating the demand function.

Scenario 1: In practice, the demand for perishables is a function of various factors such as age of the product, its price, competitor price etc. Here, the demand function is considered to depend upon the price and freshness of the product. The demand of a perishable food item is, in general, observed to decrease as its age increases. Also, the price elasticity of perishable food items is mostly negative, i.e., as price increases the demand decreases. So, we have modelled demand as an exponentially decreasing function of time and price. Figure 2a shows the simulated demand function. We have introduced randomness in the demand function to account for dynamic market behaviour.

We consider a real-life instance of a grocery retailer who needs to sell a grocery product with a fixed initial inventory. The retailer has 580 units of a product with no replenishment and with shelf life of 40 days. According to [6] the prices of perishable food items are normally discounted by 20-50% during the last few days of the product's shelf-life. Therefore, we have selected the following



(a) Demand for scenario 1 (b) Demand for scenario 2

Fig. 2: Demand function.

discrete price points - 20, 25, 30, 35, 40. Also, here the prices can be changed daily.

We use DQN for learning the best pricing strategy for the simulated demand function. The time-steps are given by t= 1, 2, 3,...,40. The action set is the set of discrete prices, A = { 20, 25, 30, 35, 40 }. Since the state is defined by (inventory left, age of product), we have 40 x 580 states in the state-space.

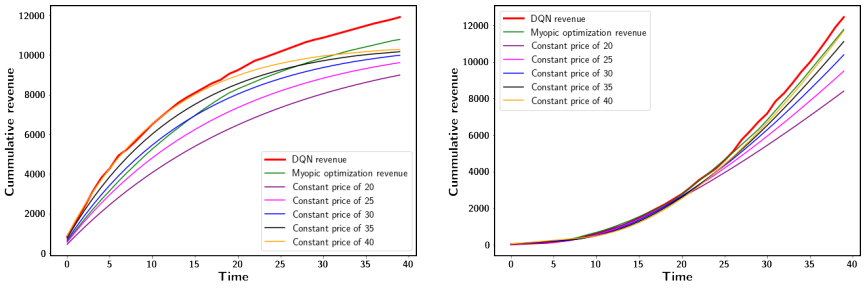
The DQN involves two similar networks-target network and evaluate network. Both these networks have the same architecture. The layout of the network and the hyper-parameters are listed in Table 1. We have trained the DQN for 200000 episodes with the simulated demand function. The DQN algorithm has been implemented using Python and Tensorflow on a MACOS Catalina system with 64-bit i5 processor @1.60 GHz and 8GB DDR3 RAM. The average execution time for training the DQN on 50000 episodes is 60 minutes and for training the DQN on 200000 episodes is 3 hours 50 minutes. GPU can be employed to significantly reduce this training time. After training the DQN, we evaluate the pricing policy given by it.

We compare the performance of the dynamic pricing algorithm using DQN with the performance of myopic (single-step) optimization algorithm. The single step price optimization model is formulated here as a non-linear function of sales and price elasticity that satisfies the price constraints and maximizes the revenue. We run this single-step optimization on the simulated demand function. The overall revenue is calculated as the sum of daily revenue. This is compared to the DQN results. Figure 3a shows the comparative plot of the cummulative revenue for RL algorithm and myopic optimization algorithm at different time steps. We can observe that the net revenue generated at the end of the product shelf-life is significantly higher for the RL based algorithm. We also observe that the cummulative revenue gap increases as the time increases. This shows the long term effectiveness of DQN in revenue maximization.

We also compare the performance of the DQN based pricing strategy with a fixed pricing strategy, which is a typical scenario in real world. In a fixed pricing strategy, the product price remains constant irrespective of the inventory left and

Table 1: Hyper-parameters of the DQN network

S.No.	Parameter name	value
1	Number of input layer neurons	2
2	Number of output layer neurons	5
3	Number of hidden layer	3
4	Number of hidden layer neurons	35,20,10
5	Optimizer	Adam
6	learning rate (α)	0.00001
7	Discount factor (γ)	.9
8	ϵ	0.5
9	Mini-batch size	64
10	Memory size	3000
11	r	200



(a) Cummulative revenue comparision for different pricing strategies, for scenario 1

(b) Cummulative revenue comparision for different pricing strategies, for scenario 2

Fig. 3: Cummulative revenue comparision.

the freshness of the product. Results for the net revenue generated at different conatant prices at the end of the shelf life, are tabulated in Table 2a. We observe that the RL based pricing policy generates the maximum revenue at the end of the product life cycle. A 7.8% increase in revenue is observed by using DQN based pricing policy, when compared with one-step price optimization.

Figure 4a exbhits the pricing strategy suggested by the trained DQN model. We observe that the DQN pricing strategy suggests different prices for different time periods. The pricing policy gradually decreases with time as the demand decreases. This seems rational as a perishable product near its expiry cannot be sold at the maximum selling price. Moreover, according to Chung and Li [6], 88% of consumers will check expiry dates before buying a perishable product. This also implies that the products cannot be sold at maximum price near their expiry.

Scenario 2: We have also experimented with the DQN based dynamic pricing algorithm for the case where the demand increases with time and

(a) Pricing policy suggested by DQN based dynamic pricing model, for scenario 1

(b) Pricing policy suggested by DQN based dynamic pricing model, for scenario 2

Fig. 4: Pricing policy suggested by DQN.

Table 2: Revenue comparison

(a) Revenue comparison, for scenario 1

Pricing strategy	**Total Revenue**
DQN	11635
1 step optimization	10792
constant price of 40	10283
constant price of 35	10137
constant price of 30	9940
constant price of 25	9552
constant price of 20	8901

(b) Revenue comparison, for scenario 2

Pricing strategy	**Total Revenue**
DQN	12445
1 step optimization	11771
constant price of 40	11700
constant price of 35	11112
constant price of 30	10384
constant price of 25	9449
constant price of 20	8400

decreases with price. The simulated demand function is shown in Figure 2b. This type of demand is frequently encountered in airline ticketing and hotel room booking. The number of seats available in a flight and the number of rooms available in a hotel can be considered analogous to inventory available for a perishable product.

Here, we consider a scenario where the initial inventory level is 650 and the distinct prices available are 20, 25, 30, 35, 40. Forty distinct time steps are considered and the pricing decision can be taken at each time step. The DQN architecture and hyper-parameter values are same as in scenario 1. The results for total revenue generated at the end of the selling horizon, for different pricing strategies are in Table. 2b.

In this scenario also, we observe that the DQN based pricing strategy generates the highest revenue when compared with single-step and constant pricing strategy. This can also be observed in Figure 3b . We observe a 5.7% increase in revenue by using DQN based pricing policy, when compared with one-

step price optimization. The pricing policy given by DQN is shown in Figure 4b. We see that as the demand increases with time, the prices also increase. This is typically observed in airline ticketing also, where the ticket prices increases as the date of journey approaches.

6 Conclusion

In the near future, consumer spend is only expected to increase exponentially. Social media impacts demand in continuous time and retailers need to respond with near real-time optimal pricing to persuade consumers to spend. Hence, dynamic pricing for perishables is a critical problem for retailers. In conclusion, this paper presents a deep reinforcement learning based approach to implement dynamic pricing for perishables. This approach is chiefly suitable for control problems with large state space. We have formulated the dynamic pricing problem as a Markov decision process and our results demonstrate that the DQN based dynamic pricing algorithm generates higher revenue when compared with constant pricing strategy and one-step price optimization. We are working to scale this approach to price multiple products for a real world use-case.

References

1. B Adenso-Díaz, S Lozano, and A Palacio. Effects of dynamic pricing of perishable products on revenue and waste. *Applied Mathematical Modelling*, 45:148–164, 2017.
2. Elodie Adida and Georgia Perakis. A robust optimization approach to dynamic pricing and inventory control with no backorders. *Mathematical Programming*, 107(1-2):97–129, 2006.
3. Gabriel R Bitran and Susana V Mondschein. Periodic pricing of seasonal products in retailing. *Management science*, 43(1):64–79, 1997.
4. Wenchong Chen, Hongwei Liu, and Dan Xu. Dynamic pricing strategies for perishable product in a competitive multi-agent retailers market. *Journal of Artificial Societies and Social Simulation*, 21(2), 2018.
5. Yan Cheng. Real time demand learning-based q-learning approach for dynamic pricing in e-retailing setting. In *2009 International Symposium on Information Engineering and Electronic Commerce*, pages 594–598. IEEE, 2009.
6. Jaekwon Chung and Dong Li. A simulation of the impacts of dynamic price management for perishable foods on retailer performance in the presence of need-driven purchasing consumers. *Journal of the Operational Research Society*, 65(8):1177–1188, 2014.
7. Yongrui Duan and Jingjing Liu. Optimal dynamic pricing for perishable foods with quality and quantity deteriorating simultaneously under reference price effects. *International Journal of Systems Science: Operations & Logistics*, pages 1–10, 2018.
8. Lin Feng, Ya-Lan Chan, and Leopoldo Eduardo Cárdenas-Barrón. Pricing and lot-sizing polices for perishable goods when the demand depends on selling price, displayed stocks, and expiration date. *International Journal of Production Economics*, 185:11–20, 2017.
9. Youyi Feng and Baichun Xiao. A continuous-time yield management model with multiple prices and reversible price changes. *Management science*, 46(5):644–657, 2000.

10. Youyi Feng and Baichun Xiao. Optimal policies of yield management with multiple predetermined prices. *Operations Research*, 48(2):332–343, 2000.
11. Guillermo Gallego and Garrett Van Ryzin. Optimal dynamic pricing of inventories with stochastic demand over finite horizons. *Management science*, 40(8):999–1020, 1994.
12. Ravi Ganti, Matyas Sustik, Quoc Tran, and Brian Seaman. Thompson sampling for dynamic pricing. *arXiv preprint arXiv:1802.03050*, 2018.
13. ABHUIT GOSAVII, Naveen Bandla, and Tapas K Das. A reinforcement learning approach to a single leg airline revenue management problem with multiple fare classes and overbooking. *IIE transactions*, 34(9):729–742, 2002.
14. J Michael Harrison, N Bora Keskin, and Assaf Zeevi. Bayesian dynamic pricing policies: Learning and earning under a binary prior distribution. *Management Science*, 58(3):570–586, 2012.
15. Ville Könönen. Dynamic pricing based on asymmetric multiagent reinforcement learning. *International journal of intelligent systems*, 21(1):73–98, 2006.
16. Erich Kutschinski, Thomas Uthmann, and Daniel Polani. Learning competitive pricing strategies by multi-agent reinforcement learning. *Journal of Economic Dynamics and Control*, 27(11-12):2207–2218, 2003.
17. Andrew EB Lim and J George Shanthikumar. Relative entropy, exponential utility, and robust dynamic pricing. *Operations Research*, 55(2):198–214, 2007.
18. Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.
19. Lihao Lu, Jianxiong Zhang, and Wansheng Tang. Optimal dynamic pricing and replenishment policy for perishable items with inventory-level-dependent demand. *International Journal of Systems Science*, 47(6):1480–1494, 2016.
20. Andrew Y Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger, and Eric Liang. Autonomous inverted helicopter flight via reinforcement learning. In *Experimental robotics IX*, pages 363–372. Springer, 2006.
21. Georgia Perakis and Anshul Sood. Competitive multi-period pricing for perishable products: A robust optimization approach. *Mathematical Programming*, 107(1-2):295–335, 2006.
22. Rupal Rana and Fernando S Oliveira. Real-time dynamic pricing in a non-stationary environment using model-free reinforcement learning. *Omega*, 47:116–126, 2014.
23. Rupal Rana and Fernando S Oliveira. Dynamic pricing policies for interdependent perishable products or services using reinforcement learning. *Expert Systems with Applications*, 42(1):426–436, 2015.
24. Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction.
25. Gerald Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.
26. Mnih Volodymyr, Kavukcuoglu Koray, Silver David, A Rusu Andrei, and Veness Joel. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.
27. Yu Xiong, Gendao Li, and Kiran Jude Fernandes. Dynamic pricing model and algorithm for perishable products with fuzzy demand. *Applied Stochastic Models in Business and Industry*, 26(6):758–774, 2010.

Cryptographic Primitives Optimization Based on the Concepts of the Residue Number System and Finite Ring Neural Network

Andrei Tchernykh[1,2,3,✉][0000-0001-5029-5212], Mikhail Babenko[4,2][0000-0001-7066-0061], Bernardo Pulido-Gaytan[1][0000-0002-7384-7670], Egor Shiryaev[4][0000-0002-2359-1291], Elena Golimblevskaia[4][0000-0001-7188-8711], Arutyun Avetisyan[2][0000-0002-0470-9944], Nguyen Viet Hung[5][0000-0002-9818-4455], and Jorge M. Cortés-Mendoza[3][0000-0001-7209-8324]

[1] CICESE Research Center, Ensenada, BC, Mexico
chernykh@cicese.mx, lpulido@cicese.edu.mx
[2] Ivannikov Institute for System Programming, Moscow, Russian Federation
arut@ispras.ru
[3] South Ural State University, Chelyabinsk, Russia
kortesmendosak@susu.ru
[4] North-Caucasus Federal University, Stavropol, Russian Federation
mgbabenko@ncfu.ru, ea_or@list.ru, elena.golimblevskaya@gmail.com
[5] Le Quy Don Technical University, Hanoi, Vietnam
hungnv@lqdtu.edu.vn

Abstract. Data encryption has become a vital mechanism for data protection. One of the main challenges and an important target for optimization is the encryption/decryption speed. In this paper, we propose techniques for speeding up the software performance of several important cryptographic primitives based on the Residue Number System (RNS) and Finite Ring Neural Network (FRNN). RNS&FRNN reduces the computational complexity of operations with arbitrary-length integers such as addition, subtraction, multiplication, division by constant, Euclid division, and sign detection. To validate practical significance, we compare LLVM library implementations with state-of-the-art, high-performance, portable C++ NTL library implementations. The experimental analysis shows the superiority of the proposed optimization approach compared to the available approaches. For the NIST FIPS 186-5 digital signature algorithm, the proposed solution is 85% faster, even though the sign detection has low efficiency.

Keywords: Residue Number System, Finite Ring Neural Network, Encryption, High-Performance, Cryptographic Primitives

1 Introduction

Security becomes commonplace in all modern computing areas and affects many fields, including casual people communication, Internet of Things (IoT), analytics, self-learning systems, cloud computing, etc. Advanced cryptographic algorithms provide key mechanisms for data confidentiality, integrity, authentication, non-repudiation, etc.

The cryptographic primitives are usually complex in terms of computational overhead and memory usage. They are designed based on mathematical theory, elliptic curves, Neural Networks (NNs), etc.

The high performance of cryptographic algorithms is important for numerous reasons. The principal one is the computational cost in terms of execution time. They can be executed by conventional computers, accelerated computing servers, and specialized hardware devices. In many cases, they are implemented as software components.

Many approaches are used to optimize encryption operations. Neuromorphic computing is concerned with emulating the neural structure and operation of the human brain. The main goals are to create a device that can extract better features, learn, recognize, classify, acquire new information, and even make a logical inference.

For instance, a single-chip prototype of the BrainScaleS 2, Intel Labs designed Loihi, and IBM's TrueNorth neuromorphic systems provide a proof-of-concept of a spiking neural network application to learn neurons and synapses [13, 14]. They include a hundred thousand neurons, each of which can communicate with thousands of others.

A Residue Number System (RNS) can achieve both fast computation and low power consumption. It is parallel, adaptable, and fault-tolerant, meaning it can produce results after components are failed [9, 10]. These properties allow for the successful development of cybersecurity systems [11-18].

RNS is a number system that represents integers by the remainders of division by several pairwise coprimes, called moduli. The arithmetic is called multi-modular arithmetic. It is widely used for computation with arbitrary length integers, for instance, in cryptography. It provides faster computation than with the usual numeral systems, even when converting between numeral systems is taken into account. By decomposing a large integer into a set of smaller integers, a large calculation is performed as a series of smaller calculations that can be performed independently and in parallel. The number of parallel elementary processes equals the number of RNS moduli.

In this paper, we propose a new optimization method RNS&FRNN of operations with arbitrary-length integers based on RNS and Finite Ring Neural Network (FRNN).

This paper is organized as follows. Section 2 describes the main concept of modular arithmetical operations. Section 3 introduces modular logical operations. Section 4 presents the scaling of RNS numbers by RNS base extension and introduces RNS&FRNN optimization method. Section 5 focuses on the experimental analysis. The conclusions and future work are discussed in the last Section 6.

2 Modular Arithmetical Operations

2.1 Addition, Subtraction, Multiplication, and Division

In the RNS, arithmetic operations are performed on each residue, according to the following general formula:

$$X \circ Y \xrightarrow{RNS} \left(|x_1 \circ y_1|_{p_1}, |x_2 \circ y_2|_{p_2}, \dots, |x_n \circ y_n|_{p_n}\right), \quad (1)$$

where $\{p_1, p_2, \dots, p_n\}$ is a moduli set of pairwise coprime numbers. "$\circ$" denotes the operation of addition, subtraction, or multiplication.

Integer numbers X and Y are defined in RNS as tuples $(x_1, x_2, \dots, x_n)$ and $(y_1, y_2, \dots, y_n)$, where x_i represents the remainder of the division of X by p_i, defined by $x_i = |X|_{p_i}$.

However, an additional restriction is imposed on the multiplication operation, which follows from the Chinese Remainder Theorem (CRT): $X \cdot Y < P$, where $P = \prod_{i=1}^{n} p_i$.

Integer division can be performed by various methods [2, 3, 4]. The most reliable algorithm is based on the scaling method. In this case, a dividend is an arbitrary number in the range $[0, P)$, and a divisor is any factor of $P = p_1 \cdot p_2 \cdot \dots \cdot p_n$.

This division is similar to dividing by numbers belonging to a certain limited set, which is faster than dividing by an arbitrary divisor (2).

$$X = \left\lfloor \frac{X}{p_1} \right\rfloor \cdot p_1 + x_1 \tag{2}$$

where X is the dividend, and p_1 is the divisor.

The dividend is represented by the residues $X \xrightarrow{RNS} (x_1, x_2, \dots, x_n)$, and the divisor is one of the moduli p_i. x_i is the residue of the division. In the first step of scaling, it is necessary to subtract the residue from the dividend (3):

$$X' \xrightarrow{RNS} (x_1', x_2', \dots, x_n') = \left(\left| x_1 - |x_i|_{p_1} \right|_{p_1}, \left| x_1 - |x_i|_{p_2} \right|_{p_2}, \dots, \left| x_n - |x_i|_{p_n} \right|_{p_n} \right). \tag{3}$$

In the second step, the division of X' by p_i is carried out directly by (4):

$$\left\lfloor \frac{X}{p_1} \right\rfloor \xrightarrow{RNS} \left(-, \left| x_1' \, |{p_1}^{-1}|_{p_2} \right|_{p_2}, \dots, \left| x_n' |{p_1}^{-1}|_{p_n} \right|_{p_n} \right) \tag{4}$$

where $|{p_1}^{-1}|_{p_i}$ is the multiplicative inversion of p_i.

At the end of the second stage, the residue x_i modulo p_i remains unknown, which can be found using the base extension (Section 4).

2.2 Euclidean division

Euclidean division is carried out using the approximate division method. The essence of the approximate method for calculating the positional characteristic to compare and restore the positional notation of the numbers in RNS. It is based on the relative values of the numbers to the full range determined by CRT.

We have:

$$X = \left| \sum_{i=1}^{n} \frac{P}{p_i} |P_i^{-1}|_{p_i} x_i \right|_P, \tag{5}$$

where $P = \prod_{i=1}^{n} p_i$, p_i is the RNS moduli, $|P_i^{-1}|_{p_i}$ is the multiplicative inversion of P_i relative to p_i, $P_i = \frac{P}{p_i}$.

If we divide the left and right sides of (5) by the constant P corresponding to the range of numbers, we obtain an approximate value

$$F(X) = \left|\frac{X}{P}\right|_1 = \left|\sum_{i=1}^{n} k_i x_i\right|_1, \quad (6)$$

where $k_i = \frac{\left|P_i^{-1}\right|_{p_i}}{p_i}$ and $|x|_1$ is fractional part real number x.

The result is obtained after summing and discarding the integer part of the number while maintaining the sum fractional part.

The fractional value $F(X) = \left|\frac{X}{P}\right|_1 \in [0, 1)$ contains both information about the value of the number and its sign [1]. If $\left|\frac{X}{P}\right|_1 \in [0, \frac{1}{2})$, then the number x is positive, and $F(X)$ is equal to the value of x divided by P. Otherwise, x is a negative number, and $1 - F(X)$ shows the relative value of the number x [5].

There are several methods of calculating $F(X)$ [6, 7, 8]. The method of integer division X/Y can be described by an iterative scheme, which is performed in two stages.

In the first stage, the search for the highest degree of 2^i is carried out when approximating the quotient with a binary series.

In the second stage, the approximation series is refined. To get a range larger than P, you can choose the value $P = P' \cdot p_{n+1}$, i.e., it is necessary to extend the RNS base by adding a redundant modulus. To avoid this base extension, a computationally complex operation, it is necessary to compare the current results of iteration i with previous values of iteration $i - 1$, and not dividends with intermediate divisors. This will satisfy the condition $0 < Y < P - 1$.

Known dividing algorithms determine the quotient based on the iteration $X' = X - Q_1 \cdot D$, where X and X' are the current and the next dividend respectively, D is the divisor, Q_1 is the quotient that is generated at each iteration from the full range of the RNS, and is not selected from a small set of constants.

In this method, the quotient is determined based on the iteration $r_i = X - B2^i$, where X is some divisible, B is the divisor, and 2^i is a member of the approximating series of the quotient. A comparison of the algorithms shows that the dividend in all iterations does not change, and the divisor is multiplied by a constant, which significantly reduces computational complexity.

The above method is easily modified in RNS using the approximate method of comparing modular numbers. In the iterative division process on a weighted number system, to search for the highest degree of a series of approximations of a quotient and to refine the approximating series, the dividend is compared with doubled divisors or with the sum of the members of the series.

The application of this idea for RNS can lead to an error in the division process. When the dynamic range is overflowed, the recovered number goes beyond the working range. For example, if the RNS moduli are $p_1 = 2, p_2 = 3, p_3 = 5,$ and $p_4 = 7$, then the range is $P = 2 \cdot 3 \cdot 5 \cdot 7 = 210$.

Suppose, during recovery, we got the number $X = 220$. In RNS $X = 220 = (0,1,0,3)$. The range P is exceeded by the number 10, which in the RNS is $(0,1,0,3)$. When using relative values, the number $X = 220$ is expressed as $X' = 10$, which is not true.

To overcome this difficulty, it is necessary to compare the current iteration values with the previous ones in the RNS. It allows to correct determining a larger or smaller

number. The overflow of the dynamic range in the RNS can be used to make the decision "more – less".

In the first iteration, the dividend is compared with the divisor, and at the other iterations, the doubled values of the divisors $q_i Y < q_{i+1} Y$ are compared. In each new iteration, the current value is compared with the previous one.

The number of iterations required depends on the divisible and divisor values. Successive application of this operation leads to the formation of a sequence of integers $Yq_1 < \ldots < Yq_n > Yq_{n+1}$.

Let the case $Yq_n > Yq_{n+1}$ be fixed at $n+1$ iterations, which corresponds to an overflow of the RNS range, i.e., $Yq_{n+1} > P$ and $X < Yq_{n+1}$. This completes the process of generating interpolation of the quotient by a binary series or by a set of constants in the RNS.

The process of approximating the quotient can be carried out by comparing only doubled neighboring approximate divisors. An important issue when implementing the function $F()$ is the accuracy of the coefficients.

It should also be noted that the number of characters in the fractional part should be twice as much as the number of characters in the RNS range. The modular numbers' division based on the approximate method of comparing numbers consists of the following steps (see Algorithm 1).

In this case, when the divisor has the minimum value and the dividend has the maximum, the threshold Δ_i is more than zero. It reduces the number of iterations when dividing a large divisible and a small divisor.

Algorithm 1. Euclidean division in RNS.

Input: $X \overset{RNS}{\longrightarrow} (x_1, x_2, \ldots, x_n)$, $Y \overset{RNS}{\longrightarrow} (y_1, y_2, \ldots, y_n)$, $F(X)$.
Output: $\omega = \left\lfloor \frac{X}{Y} \right\rfloor$, $\gamma = |X|_Y$.

Step 1. We calculate the approximate values of the divisible $F(X)$ and the divisor $F(Y)$ and compare them. If $F(X) < F(Y)$, then the division process ends and the quotient $\left\lfloor \frac{X}{Y} \right\rfloor = 0$. If $F(X) = F(Y)$, then the division process ends, and the quotient is equal to unity. If $F(X) > F(Y)$, then a higher degree 2^k is searched for by approximating the quotient with a binary code.

Step 2. We select the constant 2^k (the highest power of the series), multiply it by the divisor $F_1(X) = X2^k$ and introduce it into the comparison scheme. The constants $2^j \bmod p_i$, where $i = \overline{1,n}$, $1 \le j \le \log_2 P$ are previously stored in the memory.

Step 3. We find $\Delta_i = F(X) - F_1(Y)$. If in the sign digit Δ_i is "1", then the corresponding degree of the series is discarded, if it is "0", then in the adder of the quotient we add the value of a member of the series with this degree, that is 2^k.

Step 4. We find $F_1(Y)$, and check the term of the series with a degree 2^{k-1}.

Step 5. We find $\Delta_2 = \Delta_1 - F_1(Y)$ and perform the actions in accordance with paragraph 4.

Step 6. Similarly, we check all the remaining members of the series of the pre-zero degree. The resulting residue $\Delta_i = \Delta_{i-1} - F_{i-1}(Y) \approx 0$.

---

3 Modular Logical Operations

The operations of determining the sign of a number and comparing numbers can be performed using the approximate method discussed in Section 2.2.

The value obtained by (6) is a Positional Characteristic (PC). The determination of the sign detection is reduced to the PC, which consists of comparing the PC with the half value p_i, where $i = \overline{1,n}$. Thus, the sign of the number determines the following relation: if $\left|\frac{X}{P}\right|_1 < \frac{1}{p_i}$, then the number is positive; if $\left|\frac{X}{P}\right|_1 > \frac{1}{p_i}$, then the number is negative (Algorithm 2).

Algorithm 2. Sign detection in RNS.

Input:$X \xrightarrow{RNS} (x_1, x_2, \dots, x_n)$, $(p_1, p_2, \dots, p_n)$, P, p_j, $(P_1, P_2, \dots, P_n)$ j, for $i = \overline{1,n}$.
Output: $sign(X)$

1. **for** i **to** n **do**: $k_i = |P_i^{-1}|_{p_i}$
2. $F = 0$
3. **for** i **to** n **do**: $F \mathrel{+}= x_i \cdot k_i$
4. $F = F - (\boldsymbol{int})F$
5. **if** $F > 1/p_j$: **return** 1
6. **else**: **return** 0

The algorithm of sign detection in RNS is more complex in comparison with the algorithm of sign detection in the binary number system.

The numbers comparison can be viewed as calculating the PC of both numbers (X and Y) and comparing them:

if $\left|\frac{X}{P}\right|_1 - \left|\frac{Y}{P}\right|_1 = 0$, then $X = Y$; if $\left|\frac{X}{P}\right|_1 - \left|\frac{Y}{P}\right|_1 > 0$, then $X > Y$; if $\left|\frac{X}{P}\right|_1 - \left|\frac{Y}{P}\right|_1 < 0$, then $X < Y$. The implementation of the number comparison can provide an output $s = 0$ if $X = Y$, if $X > Y$ then $s = 1$, and $s = -1$ when $X < Y$.

4 Scaling RNS Numbers by Base Extension

Scaling an RNS number allows choosing a divisor that can simplify the division operation before dividing by p_i. The scaling depends on the coefficient K. The general formula for scaling the remainder is as follows:

$$x_i` = \left|\overline{x_{k_i}} \cdot K_i\right|_{p_i} \tag{7}$$

Thus, it is required to define two variables: x_k and K_i.

$$\overline{x_{k_i}} = |x_i - x_{n+1}|_{p_i}, \tag{8}$$

where $x_{n+1} = r_X$. r_X is a rank of a number, and in the RNS, it is reduced to the following calculations:

$$r_X = \frac{\sum_{i=1}^{n-1} x_i \cdot B_i}{P}, \tag{9}$$

where $B_i = |P_i^{-1}|_{p_i}$ is RNS modulo. Having defined $\overline{x_{k_i}}$, we need to calculate K_i, which is defined as $K_i = \left|K_i^{p_i-2}\right|_{p_i}$.

The base extension of the number in the RNS for neuromorphic computing can be effectively performed using the following procedure. The calculation of the new residue is based on the rank of the number, which can be defined as:

$$r_X = |\sum_{i=1}^{n} x_i B_i|_{p_n} \tag{10}$$

where $B_i = P_i \cdot |P_i^{-1}|_{p_i}$ is an orthogonal basis. Considering that, based on CRT and orthogonal basis, the number X in the base system $p_1, p_2, \dots, p_{n-1}$ can be written as

$$X = \sum_{i=1}^{n} a_i B_i - x_j P \tag{11}$$

Substituting (10) in (11), we obtain the following:

$$X = \left|\sum_{i=1}^{n-1} x_i |B_i|_{p_j} + x_j (p_j - |P|_{p_j})\right|_{p_j} \tag{12}$$

Based on the above, for the base extension, it is necessary to calculate the rank of the number x_j in the base system $p_1, p_2, \dots, p_{n-1}$ according to the expression (11) and find the remainder x_j by (12).

The proposed optimization method for the base extension is characterized by calculations for small modulo p_n. However, when compared with the CRT method, it simplifies the calculation with the large modulo P, and then the calculation with p_j. The residue of the number on the base extension is obtained by optimization method RNS&FRNN of RNS operations based on FRNN.

The constants of the expressions (11) and (12) can be calculated in advance. They determine the network structure. FRNN presented in Fig. 1 works as follows.

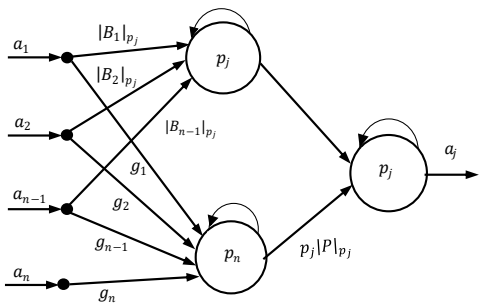


Fig. 1. FRNN architecture for the modular base extensions.

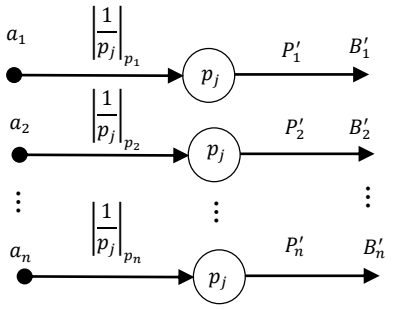


Fig. 2. The architecture of FRNN recalculation of the base extension.

The network input receives the modular values $x_1, \dots, x_n$. In the first stage, a modular neural network modulo p_n by weighted summation of the modular values of the number $x_1 \div x_n$ with coefficients $g_1 \div g_n$ calculates the rank of the number $r_A()$. Then the

modular network modulo p_j calculates the value $\sum_{i=1}^{n-1} a_i |B_i|_{p_{n+1}}$. In the second stage, $x_j = |X|_{p_j}$ is calculated using the computational model (11).

Each set of moduli of the modular code is characterized by an orthogonal basis, due to which, for the base extension, it is necessary to recalculate the basis $B_i', i = \overline{1, n+1}$. To recalculate them, the input data are: orthogonal basis $B_i, i = \overline{1, n}$, the moduli $p_1, p_2, \dots, p_n$ and the values of the extended modulo p_j. Since $P_i' = P'/p_i$ and P_i are coprime, we can calculate the orthogonal basis of the extended system as follows

$$B_i' \equiv \frac{P'}{p_i} \cdot |P_i'^{-1}|_{p_i} \tag{13}$$

To calculate it on a NN basis, it is necessary to calculate two constants: $\left|\frac{1}{p_j}\right|_{p_i}$ and $P_i' = \frac{P'}{p_i}$. Thus, the NN architecture can be presented as following (Fig. 2).

The proposed algorithm has lower computational complexity compared to the known methods. However, the method involves multiplying pre-calculated constants. These constants are usually known in advance.

5 Experimental Results

We perform experimental analysis on CPU 2.7 GHz Intel Core i5, RAM 8 GB 1867 MHz DDR3, macOS High Sierra version 10.13.6 operating system. We use NTL, a high-performance, portable C ++ library version 11.4.3, and LLVM's OpenMP runtime library version 10.0.0. RNS moduli are generated as a sequence of decreasing consecutive coprime numbers starting from $p_1 = 32{,}749$, …, $p_{285} = 29{,}789$, and $L = \lceil \log_2 P \rceil$. One million random values of X and Y are generated using RandomBnd() function, an NTL routine for generating pseudo-random numbers. Execution time T of arithmetic and logical operations are measured in microseconds (μs). The number of threads is four. The results are presented in Table 1.

First, we measure the relative performance of each operation independently. The speedup of RNS&FRNN is between 9,954 and 25,888 for the addition, 12,348 and 31,385 for the subtraction, 13,193.9 and 318,203 for multiplication, 15,353.5 and 140,290 for division by constant, and 17,815.5 and 40,359.7 for Euclid division, varying n and L. RNS sign detection performance is between 4.5 and 15 times lower.

Now, let us compare the performance of NIST FIPS 186-5 digital signature algorithm with two implementations. It is based on the operation of multiplying the point of an elliptic curve over $GF(q)$ by a scalar, the most time-consuming operation, where q is a prime number.

Different approaches for computing the elliptic scalar multiplication are introduced. Well-known Montgomery approach is based on the binary method, where scalar multiplication is defined to be the elliptic point resulting from adding value to itself several times. It performs addition and doubling in each iteration.

Table 1. Execution time of operations on NTL 11.4.3 (binary) and RNS&FRNN (RNS) (μs).

(a) Addition, Subtraction, and Multiplication

n	L	Addition			Subtraction			Multiplication		
		Binary	RNS	Binary/RNS	Binary	RNS	Binary/RNS	Binary	RNS	Binary/RNS
15	225	99,548	10	**9,954.8**	105,076	8	13,134.5	118,745	9	**13,193.9**
30	450	110,852	10	11,085.2	126,619	8	15,827.4	194,619	9	21,624.3
45	675	103,665	8	12,958.1	111,137	8	13,892.1	198,589	10	19,858.9
60	900	108,377	10	10,837.7	116,266	8	14,533.3	322,731	8	40,341.4
75	1,124	113,044	8	14,130.5	115,830	9	12,870	392,779	10	39,277.9
90	1,349	114,060	8	14,257.5	120,409	8	15,051.1	510,666	8	63,833.3
105	1,573	116,498	9	12,944.2	123,482	10	**12,348.2**	604,474	9	67,163.8
120	1,797	168,430	9	18,714.4	180,615	10	18,061.5	727,589	9	80,843.2
135	2,021	167,513	8	20,939.1	179,552	8	22,444	827,077	8	103,384.6
150	2,245	172,927	8	21,615.9	185,494	9	206,10.4	973,639	10	97,363.9
165	2,469	172,716	9	19,190.7	218,787	8	27,348.4	1,140,607	9	126,734.1
180	2,693	180,369	9	20,041.0	231,800	8	28,975	1,328,500	8	166,062.5
195	2,917	186,132	9	20,681.3	199,568	10	19,956.8	1,397,494	9	155,277.1
210	3,140	186,433	9	20,714.8	211,051	8	26,381.4	1,602,832	8	200,354.0
225	3,364	187,804	9	20,867.1	209,095	9	23,232.8	1,757,143	9	195,238.1
240	3,587	201,887	8	25,235.9	221,684	9	24,631.6	1,936,657	8	242,082.1
255	3,810	201,556	8	25,194.5	243,480	10	24,348	2,117,587	8	264,698.4
270	4,033	233,000	9	**25,888.9**	241,572	8	30,196.5	2,208,706	9	245,411.8
285	4,256	215,689	10	21,568.9	282,472	9	**31,385.8**	2,545,628	8	**318,203.5**

(b) Division by constant, Euclid division, and Sign detection

n	L	Division by constant			Euclid division			Sign detection		
		Binary	RNS	Binary/RNS	Binary	RNS	Binary/RNS	Binary	RNS	Binary/RNS
15	225	122,828	8	**15,353.5**	171,928	8	21,491.0	1	9	0.11
30	450	168,685	9	18,742.8	182,879	9	20,319.9	1	8	0.13
45	675	145,610	9	16,178.9	178,155	10	**17,815.5**	1	9	0.11
60	900	174,282	8	21,785.3	201,592	10	20,159.2	1	9	0.11
75	1,124	198,819	8	24,852.4	183,151	9	20,350.1	1	9	0.11
90	1,349	220,280	9	24,475.6	191,398	8	23,924.8	1	9	0.11
105	1,573	244,787	9	27,198.6	194,943	8	24,367.9	1	9	0.11
120	1,797	319,813	8	39,976.6	251,513	8	31,439.1	1	8	0.13
135	2,021	334,435	9	37,159.4	252,916	9	28,101.8	1	15	**0.07**
150	2,245	362,685	8	45,335.6	266,925	9	29,658.3	1	10	0.10
165	2,469	407,955	9	45,328.3	262,714	8	32,839.3	1	9	0.11
180	2,693	439,295	10	43,929.5	282,383	8	35,297.9	1	9	0.11
195	2,917	451,525	9	50,169.4	287,426	8	35,928.3	1	9	0.11
210	3,140	461,168	9	51,240.9	283,955	9	31,550.6	2	10	0.20
225	3,364	486,675	10	48,667.5	285,086	8	35,635.8	1	8	0.13
240	3,587	504,493	9	56,054.8	332,445	10	33,244.5	1	9	0.11
255	3,810	537,938	10	53,793.8	331,538	9	36,837.6	1	10	0.10
270	4,033	1,262,615	9	**140,290.6**	363,237	9	**40,359.7**	1	10	0.10
285	4,256	553,609	9	61,512.1	355,031	10	35,503.1	2	9	**0.22**

Let us evaluate the mathematical expectation of the number of additions and doubling.

Doubling can be expressed as:

$$\frac{1}{2^{\lceil \log_2 q \rceil}} \sum_i^{\lceil \log_2 q \rceil - 1} i \cdot 2^i = \frac{(\lceil \log_2 q \rceil - 2)2^{\lceil \log_2 q \rceil + 2}}{2^{\lceil \log_2 q \rceil}} \approx \lceil \log_2 q \rceil - 2 \qquad (14)$$

Addition can be expressed as:

$$\frac{1}{2^{\lceil \log_2 q \rceil}} \sum_{i=0}^{\lceil \log_2 q \rceil} i \cdot C_{\lceil \log_2 q \rceil}^{i} = \frac{\lceil \log_2 q \rceil}{2^{\lceil \log_2 q \rceil}} \cdot 2^{\lceil \log_2 q \rceil - 1} = \frac{\lceil \log_2 q \rceil}{2}, \quad (15)$$

where $C_b^a = \frac{b!}{(b-a)! \cdot a!}$.

Using the projective Jacobian coordinates for the case when $Z \neq 1$ and $a = -3$, it takes 16 multiplications to add points, and 8 multiplications to double a point.

Statistical analysis of the algirithm demonstrates that the mathematical expectation of number of modular multiplications is about

$$\frac{\lceil \log_2 q \rceil}{2} \cdot 16 + (\lceil \log_2 q \rceil - 2) \cdot 8 = 16\lceil \log_2 q \rceil - 16 \quad (16)$$

The execution time of the modular multiplication can be estimated as a sum of one multiplication and one addition; hence, $T_{Bin} = (16\lceil \log_2 q \rceil - 16)(M_{Bin} + A_{Bin})$, where M_{Bin} is the execution time of the multiplications and A_{Bin} is the execution time of the addition.

To assess the RNS implementation of the algorithm, first, we consider the RNS to binary T_C and binary to RNS T_E conversion times (Table 2).

Table 2. Time of Binary to RNS (T_C) and RNS to Binary (T_E) conversion (μs).

n	L	T_C	T_E
15	225	43,685,000	58,829,340
30	450	109,762,100	152,454,240
45	675	192,411,600	290,528,220
60	900	309,272,700	464,671,140
75	1,124	384,687,100	685,935,110
90	1,349	484,623,300	952,097,660
105	1,573	595,553,500	1,251,265,520
120	1,797	713,753,000	1,604,888,270
135	2,021	815,894,900	1,995,859,970
150	2,245	944,469,700	2,440,457,340
165	2,469	1,085,540,400	2,962,540,610
180	2,693	1,264,208,500	3,524,798,450
195	2,917	1,424,684,400	4,180,765,530
210	3,140	1,648,081,600	4,692,365,130
225	3,364	1,713,357,600	5,268,070,980
240	3,587	1,882,562,600	6,117,228,220
255	3,810	2,049,367,600	6,824,793,510
270	4,033	2,242,481,300	7,592,480,470
285	4,256	2,469,894,800	8,680,054,900

The modular multiplication of an elliptic curve point by a scalar in RNS requires one multiplication, one addition, $n(n-1)/4$ divisions by a constant, and one operation for determining the sign of a number, where n is the number of moduli.

The execution time of the RNS implementation can be estimated as

$$T_{RNS} = (16\lceil \log_2 q \rceil - 16)(M_{RNS} + A_{RNS} + \frac{n(n-1)}{4} DC_{RNS} + S_{RNS}) + 2(T_C + T_E),$$

where M_{RNS} is the execution time of multiplication of two numbers in RNS, A_{RNS} is the execution time of addition in RNS, DC_{RNS} is the execution time of the division by constant in RNS, S_{RNS} is the execution time of the sign detection, T_C is the time of binary to RNS conversion, and T_E is the time of RNS to binary conversion.

Thus, for $q = 2^{511} - 1$, $n = 75$, T_{Bin} and T_{RNS} are estimated, in the worst case, as $T_{Bin} = (16 \cdot 511 - 16) \cdot (392{,}779 + 113{,}044) = 4{,}127{,}515{,}680$, $T_{RNS} = (16 \cdot 511 - 16) \cdot (10 + 8 + 75 \cdot (75 - 1)/4 \cdot 8 + 9) + 2 \cdot (384{,}687{,}100 + 685{,}935{,}110) = 2{,}232{,}040{,}738$. Therefore, $T_{Bin}/T_{RNS} \approx 1.85$ times.

6 Conclusion

We propose an optimization of six encryption operations: addition, subtraction, multiplication, division by constant, Euclid division, and sign detection with integers of arbitrary length based on modular arithmetic and finite ring neural networks.

We show that they provide significant advantages in comparison with long arithmetic implemented in NTL. The higher benefits of RNS&FRNN are derived for the multiplication of large numbers. RNS shares them into smaller numbers that can be performed independently and in parallel without carries between them. We demonstrate that the proposed solution is 85% faster than NIST FIPS 186-5 digital signature algorithm, even, calculation of the sign detection operation is inefficient.

The structure of the non-positional operations of RNS, fault tolerance, and parallelism can be well suited for neuromorphic systems. However, the proposed approach does not provide an efficient implementation of logical operations, such as comparison, number sign determination, etc., which we will study in future work.

Acknowledgments

This work was partially supported by the Ministry of Education and Science of the Russian Federation (Project 075-15-2020-788).

References

1. Krasnobayev, V. A., Yanko, A. S., Koshman, S. A., “A Method for arithmetic comparison of data represented in a residue number system,” Cybern. Syst. Anal., vol. 52, no. 1, pp. 145–150, 2016.
2. Ruchkin, V., Romanchuk, V., Sulitsa, R., “Clustering, restorability and designing of embedded computer systems based on neuroprocessors,” in 2013 2nd Mediterranean Conference on Embedded Computing (MECO), 2013, pp. 58–61.
3. Vinogradov, I. M., Elements of number theory. Courier Dover Publications, 2016.
4. Yu, D.-J., Hu, J., Tang, Z.-M., Shen, H.-B., Yang, J., Yang, J.-Y., “Improving protein-ATP binding residues prediction by boosting SVMs with random under-sampling,” Neurocomputing, vol. 104, pp. 180–190, 2013.
5. Hu, J., Li, Y., Yan, W.-X., Yang, J.-Y., Shen, H.-B., Yu, D.-J., “KNN-based dynamic query-driven sample rescaling strategy for class imbalance learning,” Neurocomputing, vol. 191, pp. 363–373, 2016.

6. Babenko, M. et al., “Positional Characteristics for Efficient Number Comparison over the Homomorphic Encryption,” Program. Comput. Softw., vol. 45, no. 8, pp. 532–543, 2019.
7. Tchernykh, A. et al., “Scalable Data Storage Design for Non-Stationary IoT Environment with Adaptive Security and Reliability,” IEEE Internet of Things Journal, vol. 7, no. 10, pp. 10171-10188, 2020.
8. Burgess, N., “Scaling an RNS number using the core function,” in Proceedings 2003 16th IEEE Symposium on Computer Arithmetic, 2003, pp. 262–269.
9. Tchernykh, A., Miranda-López, V., Babenko, M., Armenta-Cano, F., Radchenko, G., Drozdov, A. Y., Avetisyan, A. “Performance Evaluation of Secret Sharing Schemes with Data Recovery in Secured and Reliable Heterogeneous Multi-Cloud Storage”. Cluster Computing. Vol. 22, Issue 4, pp 1173–1185, Springer, 2019
10. Miranda-López, V., Tchernykh, A., Babenko, M., Avetisyan, A., Toporkov, V., Drozdov. A.Y. “2Lbp-RRNS: Two-Levels RRNS with Backpropagation for Increased Reliability and Privacy-Preserving of Secure Multi-Clouds Data Storage”. IEEE Access pp. 1-16, 2020.
11. Babenko, M., Shiriaev, E., Tchernykh, A., Golimblevskaia, E. “Neural network method for base extension in residue number system”. ICCS-DE 2020- 2nd International Workshop on Information, Computation, and Control Systems for Distributed Environments, Irkutsk, Russia. July 6-7, 2020. Edited by: Igor Bychkov, Andrei Tchernykh, Alexander Feoktistov. CEUR-WS: Vol-2638, pp. 9-22, 2020
12. Babenko, M., Tchernykh, A., Golimblevskaia, E., Hung, N. V., Chaurasiya, V.K., “Computationally secure threshold secret sharing scheme with minimal redundancy”. ICCS-DE 2020- 2nd International Workshop on Information, Computation, and Control Systems for Distributed Environments, Irkutsk, Russia. July 6-7, 2020. Edited by: Igor Bychkov, Andrei Tchernykh, Alexander Feoktistov. CEUR-WS: Vol-2638, pp. 23-32
13. Davies, M., Srinivasa, N., Lin, T. H., Chinya, G., Cao, Y., Choday, S. H., ... & Wang, H. (2018). Loihi: A neuromorphic manycore processor with on-chip learning. IEEE Micro, 38(1), 82-99.
14. DeBole, M. V., Taba, B., Amir, A., Akopyan, F., Andreopoulos, A., Risk, W. P., ... & Modha, D. S. (2019). TrueNorth: Accelerating from zero to 64 million neurons in 10 years. Computer, 52(5), 20-29.
15. Babenko, M., Deryabin, M., Piestrak, S.J., Patronik, P., Chervyakov, N., Tchernykh, A., Avetisyan, A.: RNS Number Comparator Based on a Modified Diagonal Function. Electronics. 9, 1784 (2020). https://doi.org/10.3390/electronics9111784.
16. Miranda-Lopez, V., Tchernykh, A., Babenko, M., Kuchukov, V., Deryabin, M., Golimblevskaia, E., Shiryaev, E., Avetisyan, A., Rivera-Rodriguez, R., Radchenko, G., Talbi, E.-G.: Weighted Two-Levels Secret Sharing Scheme for Multi-Clouds Data Storage with Increased Reliability. In: 2019 International Conference on High Performance Computing & Simulation (HPCS). pp. 915–922. IEEE (2019). https://doi.org/10.1109/HPCS48598.2019.9188057.
17. Babenko, M., Deryabin, M., Tchernykh, A.: The Accuracy Estimation of the Interval-Positional Characteristic in Residue Number System. In: 2019 International Conference on Engineering and Telecommunication (EnT). pp. 1–5. IEEE (2019). https://doi.org/10.1109/EnT47717.2019.9030549.
18. Kucherov, N., Babenko, M., Tchernykh, A., Kuchukov, V., Vashchenko, I.: Increasing reliability and fault tolerance of a secure distributed cloud storage. In: The International Workshop on Information, Computation, and Control Systems for Distributed Environments. (2020). https://doi.org/10.47350/ICCS-DE.2020.16.

Formulation of a Layout-Agnostic Order Batching Problem*

Johan Oxenstierna[1,2][0000-0002-6608-9621], Louis Janse van Rensburg[3], Jacek Malec[1][0000-0002-2121-1937] and Volker Krueger[1][0000-0002-8836-8816]

[1] Department of Computer Science, Lund University, Lund, Sweden.
[2] Kairos Logic AB, Lund, Sweden.
[3] Thompson Institute, University of the Sunshine Coast, Queensland, Australia.

Abstract. To date, research on warehouse order-batching has been limited by reliance on rigid assumptions regarding rack layouts. Although efficient optimization algorithms have been provided for conventional warehouse layouts with Manhattan style blocks of racks, they are limited in that they fail to generalize to unconventional layouts. This paper builds on a generalized procedure for digitization of warehouses where racks and other obstacles are defined using two-dimensional polygons. We extend on this digitization procedure to introduce a layout agnostic minisum formulation for the Order Batching Problem (OBP), together with a sub-problem for the OBP for a single vehicle, the *single batch* OBP. An algorithm which optimizes the *single batch* OBP iteratively until an approximate solution to the OBP can be obtained, is discussed. The formulations will serve as the fundament for further work on layout-agnostic OBP optimization and generation of benchmark datasets. Experimental results for the digitization process involving various settings are presented.

Keywords: Order Batching Problem, vehicle routing, warehouse digitization.

1 Introduction

Order-picking is "the process of retrieving products from storage areas in response to a specific customer request" where "customer request" denotes a shipment order consisting of one or several products [1]. Order-picking is accountable for as much as 55% of all operating expenses in a warehouse and is considered an important process to optimize [2]. *Order-batching* is a common method with which to conduct order-picking. It means that each picker (vehicle) is set to pick a so-called *batch* of one or more orders [3]. As an optimization problem order-batching is known as the *Order Batching Problem* (OBP) [4] or the *Joint Order Batching and Picker Routing Problem* (JOBPRP) [5]. The *Picker Routing Problem* is a sub-problem of the OBP for one vehicle and is here treated as equivalent to the Traveling Salesman Problem (TSP) [6]. This paper

* This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

follows the convention that an "OBP" can include TSP optimization without having to include TSP optimization in the name of the problem (such as the JOBPRP) [4]. The *Picker Routing Problem* is henceforth referred to as TSP and the *Order Batching Problem*, which includes TSP optimization, as OBP. In the literature the OBP is usually formulated as a specific form of the more well-known Vehicle Routing Problem (VRP) [7], with two key amendments:

1. *Order-integrity*: In the OBP products in one order cannot be picked by more than one vehicle [8] whereas in the VRP this constraint is not used (there is no notion of a warehouse shipment "order" in the VRP) [7].
2. *Obstacle-layout:* We can observe two types of obstacle layouts (see Fig 1): In the *conventional* layout, racks are laid out in a Manhattan style blocks. In the *unconventional* layout, racks or other obstacles can be freely placed (see Fig 2. for examples). The *unconventional* layout includes the case when there are no racks or obstacles at all. All previous work on the OBP seems to require explicitly a conventional layout [5], [8]–[10], while the VRP does not have this requirement.

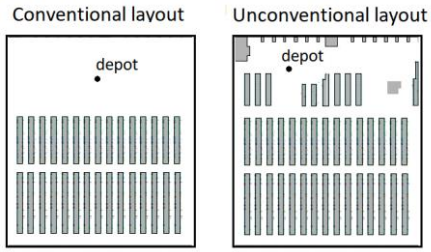


Fig. 1. Example of a *conventional* layout (left) with 30 *racks*, 16 *aisles* and 3 *cross-aisles*. Adding a single or a few irregular racks or other obstacles to the *conventional* layout renders it *unconventional*.

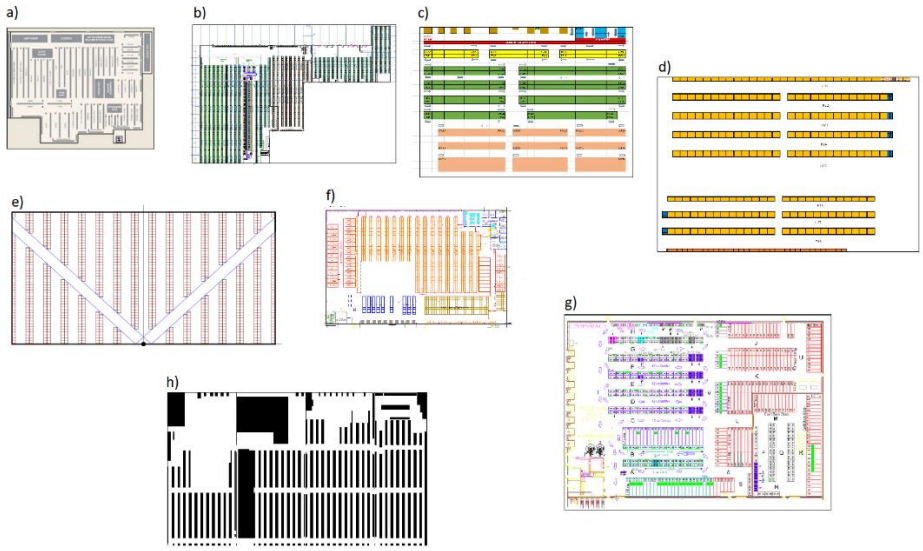


Fig. 2. Eight examples of unconventional warehouse layouts. a) and b) show cases where the layout has been built to fit within a non-rectangular outer wall. e) is the so called "fishbone" layout.

The aim of this paper is to formulate an OBP where orders and order-integrity are preserved, but where the layout is generalized towards any layout with or without polygonal obstacles. This is in line with a future research recommendation by Masae et al. [11]: "there is a strong need for developing […] algorithms for […] non-conventional warehouses". Below are some reasons for why this is important:

- It allows warehouses with unconventional layouts to formulate and optimize OBP's. This includes warehouses divided into zones where each zone has a conventional layout.
- It allows OBP optimization to be used as a tool with which to optimize warehouse layouts beyond conventional layouts.
- Problems in non-warehouse domains, such as agriculture, mining, road and aerial logistics to be explored as OBP's. The OBP is fundamentally similar to *batch processing* [12] where each process consists of constrained sub-processes (similar to *order-integrity*), and the Key Performance Indicator (KPI) depends on how well the sub-processes operate when they are combined. These types of broadened perspectives on the OBP can only be pursued if it is generalized beyond conventional layouts.

The paper continues with a literature review (Section 2), followed by the OBP formulation (Section 3). The formulation builds on a digitization process which generates the distances and shortest paths between all defined locations for a given warehouse [13]. The feasibility of the digitization process is examined in experiments involving various warehouse configurations (Section 4).

2 Literature Review

The OBP is a specific form of the Vehicle Routing Problem (VRP) [7] and a specific VRP-variant known as the *Steiner*-VRP [14]. A key feature of the *Steiner*-VRP is that multiple visits to same location (representing a vertex in a graph) are allowed [5], [8], [10], [14]. OBP's and VRP's are known to be NP-hard [15], [16]. OBP's have been formulated using integer programming (e.g. [14]) or set-partitioning (e.g. [4]), with a heavy reliance on heuristics for a *conventional* warehouse layout. The conventional layout is modeled such that obstacles (racks) are arranged with parallel "aisles" (between racks) and parallel "cross-aisles" (between sections of racks) [9], [14]. Using such restrictive definitions for aisles and cross-aisles makes it possible to formulate heuristics that reduce the solution space of an OBP. Briant et al. [9], for example, use cutting planes and various relaxation heuristics to formulate an OBP which they then propose optimality bounds for. They use a conventional layout with 8 aisles and 3 cross-aisles, which corresponds to the size of the warehouse shown in Fig. 1 d).

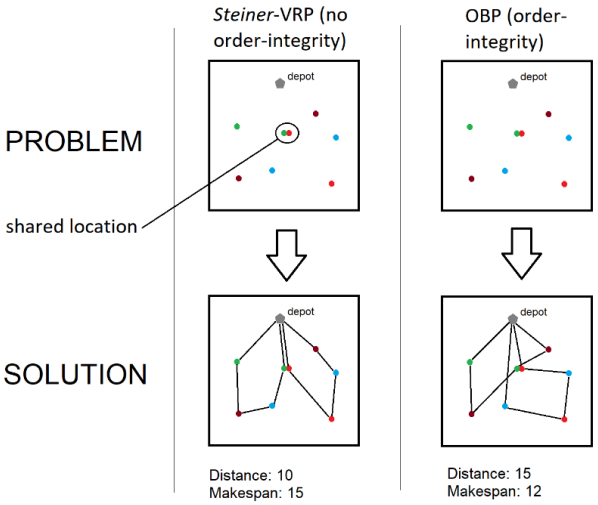


Fig. 3. A Steiner-VRP (left) plotted against the proposed layout-agnostic OBP in a setting without any obstacles. The dots denote products and the colors orders which the products belong to. The outlined green and red products in the middle share the same location. The difference between the Steiner-VRP and the OBP seen here is solely due to the *order-integrity* constraint. The vehicle distances may be longer in the OBP but the products which they are assigned to carry are more associated (by order color in this example). Order-integrity is used to e.g. reduce a later time-consuming sorting effort or to reduce pick-error i.e. the risk of the wrong product going into the wrong order.

The conventional layout appears in formulations as "number of aisles" [8], "the cross-distance between two consecutive aisles" [4], "number of vertices in the subaisle" [14] or "intra-aisle distance" [17]. They are used as required inputs for OBP optimization.

Some authors have called for formulations involving more layouts than the conventional layout [11], [18]–[21]. Without the conventional layout, however, it is a challenging task to effectively constrain an OBP solution space. This can for instance be exemplified in the scenario when there are no obstacles, and each order contains a single product. In that case the OBP is equivalent to a Steiner-VRP, and this problem has no yet proposed optimal solution [14]. Proposed OBP optimization algorithms for the conventional layout include dynamic programming [9], datamining [22], clustering [10] and meta-heuristics such as Tabu Search [23], Ant Colony Optimization [15] and Genetic Algorithms [24]. In the VRP research domain problem formulations are generally not concerned with obstacle layouts [25]. Instead the only requirement in a VRP is usually a cost matrix, providing the travel distance or time between all pairs of locations [7], [26]. In a VRP it is generally assumed that this cost matrix already exists, or that it is produced in a prior data collection process. In research on the OBP, on the other hand, plenty of attention is usually given to how to produce the cost matrix and how to define shortest paths or TSP's in an environment with obstacles. This can also be seen in some papers on VRP's that include obstacles (e.g. [27] and [28]). Concerning where vehicles begin and end their trips, most OBP papers assume that the origin and destination location is the same (usually this location is named *depot*). If this is not the case, the OBP is denoted *multi-depot* or a *Dial-A-Ride-Problem* (DARP) [21]. An example of this is when vehicles have one location where they drop off their picked orders, and where there are one or several locations where they can start their rides.

3 Problem Formulation

3.1 Preliminaries

The proposed OBP formulation is based on an undirected, symmetric and weighted graph. Without obstacles (racks or other) no graph is needed since distances between all pairs of locations in that case can be assumed to be Euclidean. Also, in the obstacle free case, the shortest path between any two locations can be assumed to be a single edge. With obstacles, however, shortest distances must be calculated based on the shortest paths that circumvent obstacles, and this is achieved here using the Floyd-Warshall graph algorithm [13], [29]. Concerning number of depots the below formulation assumes both an origin and a destination location for vehicles is formulated (but they can share the same coordinates).

First a set of locations is defined as $\mathcal{L} \subset \mathbb{R}^+ \times \mathbb{R}^+$. This set consists of different types of locations: $l_s \in \mathcal{L}$ is the starting (origin) location for all vehicles. $l_d \in \mathcal{L}$ is the destination location for all vehicles. $\mathcal{L}_\mathcal{P} \subset \mathcal{L}$ is the set of product locations. $\mathcal{L}_\mathcal{U} \subset \mathcal{L}$ is a union of sets of obstacles: $\mathcal{L}_\mathcal{U} = \cup_i u_i, i \in \mathbb{N}^+$ where each u_i is a polygonal obstacle with a set of corner locations $u_i = \{l_i^1, l_i^2, \dots, l_i^k\} \subseteq \mathcal{L}_\mathcal{U}, k \in \mathbb{N}^+$. All of the locations can thus be summarized as a union: $\mathcal{L} = \{l_s\} \cup \{l_d\} \cup \mathcal{L}_\mathcal{P} \cup \mathcal{L}_\mathcal{U}$. The products which are to be collected are defined as a set $\mathcal{P} = \{p_1, p_2, \dots, p_n\}, n \in \mathbb{N}^+$. Each product $p \in \mathcal{P}$ has a location $loc^p: \mathcal{P} \to \mathcal{L}_\mathcal{P}$, weight $w^p: \mathcal{P} \to \mathbb{R}^+$ and volume $vol^p: \mathcal{P} \to \mathbb{R}^+$. *The unassigned orders* which are to be batched are defined as a subset of all possible combinations of products $\mathcal{O} \subset 2^\mathcal{P}$. The locations of the products in an order $o \in \mathcal{O}$ are

defined as a function $loc^o: \mathcal{O} \to 2^{\mathcal{L}_\mathcal{P}}$. Order weight and volume quantities are defined as $w^o: \mathcal{O} \to \mathbb{R}^+$ and $vol^o: \mathcal{O} \to \mathbb{R}^+$. $w(o) = \sum_{p \in o} w(p), vol(o) = \sum_{p \in o} vol(p)$. *Vehicles* are defined as $\mathcal{M} = \{(w, vol, k, id) | w, vol, id \in \mathbb{R}^+, k \in \mathbb{N}^+\}$ where w denotes weight capacity, vol denotes volume capacity, k denotes the maximum number of orders the vehicle can carry and id a unique identifier of a vehicle. The capacities of a single vehicle $m \in \mathcal{M}$ are provided using functions $w^m: \mathcal{M} \to \mathbb{R}^+, vol^m: \mathcal{M} \to \mathbb{R}^+$ and $k^m: \mathcal{M} \to \mathbb{N}^+$.

The digital model of the warehouse is represented as a graph with a set of *vertices* $\mathcal{V} = \{v_1, v_2, \dots, v_n\}, n = |\mathcal{L}|$. $\mathcal{V}$ consists of different types of vertices denoted as follows: $v_s \in \mathcal{V}$ is a starting (origin) vertex for vehicles, $v_d \in \mathcal{V}$ is a destination vertex for vehicles, $\mathcal{V}_{\mathcal{L}_\mathcal{P}} \subset \mathcal{V}$ is a set of product location vertices and $\mathcal{V}_\mathcal{U} \subset \mathcal{V}$ is a set of obstacle corner vertices. The union of all vertices, $\mathcal{V} = \{v_s\} \cup \{v_d\} \cup \mathcal{V}_{\mathcal{L}_\mathcal{P}} \cup \mathcal{V}_\mathcal{U}$, are defined similarly to the locations apart from one important difference: There may be several products in one location and there is one vertex per *product location*, not one vertex per product (this is to limit the size of the graph). To get a set of locations from a corresponding set of vertices the function $loc^\mathcal{V}: \mathcal{V} \to \mathcal{L}$ is used. To get a set of vertices from a set of locations is similarly provided by the function $v^\mathcal{L}: \mathcal{L} \to \mathcal{V}$.

The set of possible batches is defined as $\mathcal{B} \subset 2^\mathcal{O}, b \in \mathcal{B}, b \in 2^\mathcal{O}, b \neq \emptyset$. The locations of the products in the batch can be obtained using function $loc^b: \mathcal{B} \to 2^{\mathcal{L}_\mathcal{P}}. loc(b) = \cup_{o \in b} loc(o)$. Similarly, the vertices in the batch are $v^b: \mathcal{B} \to 2^{\mathcal{V}_{\mathcal{L}_\mathcal{P}}}. v(b) = \cup_{o \in b} v(loc(o))$. Batch weight and volume quantities are defined as $w^b: \mathcal{B} \to \mathbb{R}^+$ and $vol^b: \mathcal{B} \to \mathbb{R}^+$. The number of orders in a batch is defined as $k^b: \mathcal{B} \to \mathbb{N}^+$ or $|b|$.

The set of edges E is defined such that each edge is an ordered pair $e \in E = \{(i, j), i, j \in \mathcal{V}, \mathrm{i} \neq j\}$ where i is an origin and j a destination vertex. E excludes any edge which passes through the hull of any polygon in $\mathcal{U}$ (for details on how this can be achieved see [13]). Edges between adjacent corners in any polygon $u \in \mathcal{U}$ are not excluded in E. The edges and vertices are then used to construct the *symmetric undirected weighted graph* $G = (\mathcal{V}, E)$.

A *shortest paths distance matrix* $D: \mathcal{V} \times \mathcal{V} \to \mathbb{R}^+$ provides the minimum sum of edge distances between any two vertices in $\mathcal{V}$ without crossing any hull in $\mathcal{L}_\mathcal{U}$. Each edge cost $d_{loc(i),loc(j)} \in D$ (henceforth d_{ij}) is between two vertices, $i, j \in \mathcal{V}, i \neq j$. If there exists an unobstructed path between $loc(i)$ and $loc(j)$ (which does not go through any obstacle hull) the distance is Euclidean $\|loc(i) - loc(j)\|$. If obstacles must be bypassed to go from $loc(i)$ to $loc(j)$, however, the distance is a sum of Euclidean distances following the shortest path between them (without crossing obstacles). The Floyd-Marshall graph algorithm is used to compute these shortest paths and distances exactly [13].

The set of vertices, including origin and destination vertex, that have to be visited to pick a batch is defined as $\mathcal{V}_b = \{v_s\} \cup v(b) \cup \{v_d\}, b \in \mathcal{B}$. A function can then be built which provides *the sequence of vertex visits in a batch TSP solution (tour)*:

$$T^b: \mathcal{V}_b \to \{v_i\}_{i=1}^n, n = |\mathcal{V}_b|, \tag{1}$$

$$T(b)_i = \begin{cases} v_s & i = 1 \\ v_k & 1 < i < n \\ v_d & i = n \end{cases} \quad (2)$$

where $v_k \in v(b)$ and i gives the sequence of visits. *The distance of a batch TSP solution (tour)* is similarly provided in a function:

$$D^b : T(b)_i \rightarrow \mathbb{R}^+, i \in \mathbb{N}^+, i \leq |T(b)|. \quad (3)$$

$$D(b) = \sum d_{T(b)_i T(b)_j}, i, j \in \mathbb{N}^+, j = i + 1, i < |T(b)| \quad (4)$$

Note D^b could be renamed D^{T^b} to clarify that the distance of a batch is computed over a certain *tour* to visit all the products in the batch. $\mathcal{V}, E, G$ and D are assumed to be produced in a digitization preprocessing step and the computational effort at this stage is assumed to not be included in subsequent OBP optimization. Out of $\mathcal{V}, E, G$ and D only D is needed as input for OBP optimization assuming vehicles are capable of finding the shortest path between any two locations on their own. $\mathcal{V}, E, G$ are also needed for directions on how to follow the shortest path, and if visualizations of edges are sought, both of which are arguably important in an industrial OBP optimization service. One example of a visualization of G and a small OBP optimization instance can be seen in Fig. 4 below:

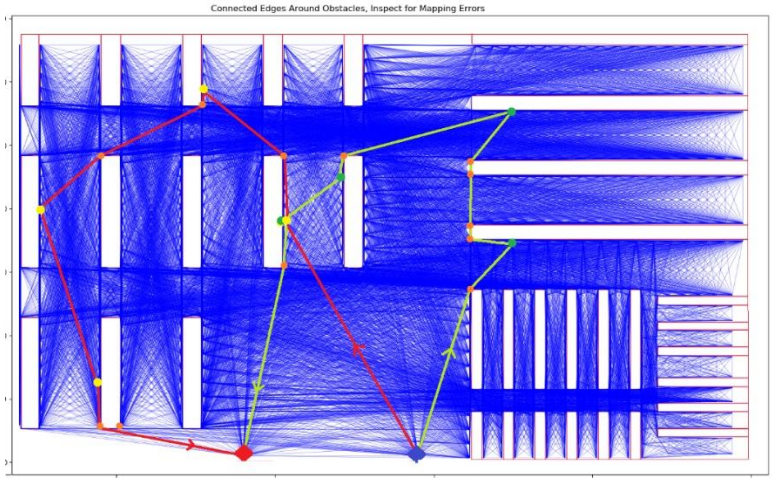


Fig. 4. Visualization of the digital graph (G) of a warehouse, and an example OBP with two orders, two vehicles and vehicle capacity of one order. Each blue line is an edge $e \in E$ that connects two vertices $(i, j), i, j \in \mathcal{V}$. The white hulls are racks (obstacles) laid out in an "unconventional" way and no edges pass through them. The orange vertices show a subset of $\mathcal{V}_U$ and the green and yellow vertices along the racks are the sought products in $\mathcal{V}_{\mathcal{L}_\mathcal{P}}$ (where color indicates which order it belongs to). Note one of the products is visited by both vehicles. At the bottom the origin and destination, v_S and v_d can be seen (blue and red respectively). The OBP solution is here shown as the red and lime edges following the shortest paths between v_S, the yellow or green vertices and v_d (the two tours are obtained using T^b above).

3.2 General OBP formulation

A set-partitioning formulation with an exponential number of binary variables is used to formulate the layout-agnostic general OBP. The binary decision variable x_{mb} is used to indicate whether batch $b \in \mathcal{B}$ is assigned to vehicle $m \in \mathcal{M}$ ($x_{mb} = 1$, if m is assigned to b, $x_{mb} = 0$ otherwise). The binary decision variable x_{mo} is used to indicate whether order $o \in \mathcal{O}$ is assigned to vehicle $m \in \mathcal{M}$ ($x_{mo} = 1$ if m is assigned $o, x_{mo} = 0$ otherwise). The binary decision variable x_{ml} is used to indicate whether vehicle m visits location $l \in \mathcal{L}_{\mathcal{P}}$ ($x_{ml} = 1$ if m visits $l, x_{ml} = 0$ otherwise).

$$min \sum_{b \in \mathcal{B}} D(b) x_{mb}, m \in \mathcal{M} \tag{5}$$

s.t.

$$\sum_{m \in \mathcal{M}} x_{mo} = 1, \forall o \in \mathcal{O} \tag{6}$$

$$\sum_{l \in loc(o)} x_{ml} \geq x_{mo}, \forall o \in \mathcal{O}, m \in \mathcal{M} \tag{7}$$

$$q(b) \leq q(m) x_{mb}, b \in \mathcal{B}, q \in \{w, vol, k\}, m \in \mathcal{M} \tag{8}$$

The optimization aim of the OBP (5) is to assign one batches to vehicles such that the sum of the distances of all batches is minimized. (6) ensures that each unassigned order is assigned to exactly one vehicle (*order-integrity*). (7) ensures that every product location in every order assigned to a vehicle is visited at least once. This inequality is what renders the OBP a general *Steiner*-VRP. (8) ensures capacity of vehicles is never exceeded.

3.3 Single batch OBP formulation

The general OBP formulation is problematic to work with due to the large number of possible combinations of vehicles and batches. Below is a proposal for a more tractable problem where the aim is to find a batch for an already selected vehicle. After vehicle *m* has been selected the aim is to assign as many orders as possible to it while keeping batch distance at a minimum:

$$\underset{b \in \mathcal{B}}{argmin}\, D(b) \tag{9}$$

$$\exists q\big(q(b) + q(o) \geq q(m)\big), \forall o \in \mathcal{O}, o \notin b, q \in \{w, vol, k\} \tag{10}$$

where $k(o)$ (i.e. the number of orders in an order) is 1. The aim in the *single batch* OBP (9) is to, for a given vehicle m, find a single batch b with the minimal batch distance. Constraints (6), (7), and (8) from the general OBP still apply (for the given vehicle). Constraint (10) is further added to ensure that the number of orders in the batch is as large as possible (for all unassigned orders there exists a weight, volume or number of orders quantity which will exceed vehicle capacity if the order is added to the batch). Without this maximization of number of orders an optimization algorithm would always create a batch with just a single order because this would produce the minimal batch distance. The single batch OBP formulation is a specific version of the so called *minimum cost maximal knapsack packing problem* (MCMKP) if distance is treated as "profit" and number of orders as knapsack "weight" (according to the definition by [30]). Note in the formulation here batch "weight" and "volume" are not included in the maximization since this would impose decision making over the importance of the different quantities (which one is most important to maximize while not exceeding vehicle capacity). The intention of the single batch OBP formulation is to provide the means with which to build an efficient *single batch* OBP optimization algorithm. This algorithm can then be used to produce one batch at a time within an algorithm which optimizes the general OBP, as proposed in Algorithm 1 below:

Algorithm 1. Iteration of single Batch OBP optimization.

1. total_OBP_cost $\leftarrow$ 0
2. **WHILE** $\mathcal{O}$**:**
3. $\quad m \leftarrow$ select_vehicle$(\mathcal{M})$
4. $\quad b \leftarrow$ single_batch$(\mathcal{O}, m, D)$
5. $\quad$ total_OBP_cost $+= D(b)$
6. $\quad \mathcal{O} \leftarrow$ remove_batched$(\mathcal{O}, b)$

Algorithm 1 runs with the assumption that there are always enough vehicles to choose from, and it creates single batches until there are no more unassigned orders left. The total cost is expressed in the TSP tour distances of the batches $D(b)$. After a batch has been created its orders are removed from $\mathcal{O}$.

4 Experimental results

This section evaluates the computational effort and memory requirement needed to generate the datastructures used by the formulation in Section 3. The only datastructure needed for OBP optimization is the distance matrix D, but graph G, including shortest paths between all locations are also included.

Name	Number of locations	Number of polygons	Memory allocated		CPU time (s)
			Disk (mb)	RAM (mb)	
c6953_B01	781	0	14.4	78	541
c7561_B02	288	16	2.7	65	268
c0543_B03	760	1	14.2	79	501
c9109_AYD	218	5	0.7	62	195
c3495_BER	579	69	61	83	620
c0054_JUL	752	42	12.2	81	763
c3401_DAD	1384	296	377	385	5616
c9543_ARA	4037	234	520	574	26028
c2456_CAG	6491	306	1219	1290	65232

CPU used: Intel Core i7-4710MQ 2.5 GZ 4 cores, 8GB of RAM

Computational time and memory requirement grows fast with number of locations in the digitization procedure. The largest instance included 6491 defined locations and required 18 hours of *CPU-time*. Please note the computation only has to be run once (and re-run if the obstacle layout is changed in the warehouse). Once the graph has been generated, distances and shortest paths can be queried quickly by pre-allocating them in Random Access Memory (RAM), which is why RAM usage is also a relevant parameter. "Number of locations", denoted as $|\mathcal{L}|$ in Section 3, and the number of products in each defined location, varies depending on precision sought in the digitization process. For example, the warehouse denoted c9543_ARA, holds around 40000 products, but there are only 4037 defined locations. Each location in that case represents the products within an area of around 3 m^2 on the horizontal axis and 5 shelf levels on the vertical axis, with a total of around 10 products represented by every defined location. Clearly, a faster digitization process would be achieved if more products were mapped to the same locations, but then the digital model would be less precise. The tradeoff between memory and CPU-time on the one hand, and digitization precision on the other, is an interesting topic left for future work.

5 Conclusion

This paper set out to formulate an Order Batching Problem (OBP) that does not depend on the way in which racks or other obstacles are laid out in the warehouse. A digitization procedure to generate necessary datastructures was first described. A minisum set-partitioning formulation with an exponential number of binary variables was introduced for the layout-agnostic OBP. A more tractable version of the OBP, the *single batch* OBP, was additionally formulated where the aim is to find a single batch for an already specified vehicle. Experiments evaluating CPU-times and memory footprints for generating necessary datastructures was presented. In ensuing work new layout agnostic OBP optimization algorithms and benchmark instances will be introduced.

References

[1] R. de Koster, T. Le-Duc, and K. J. Roodbergen, 'Design and control of warehouse order picking: A literature review', *European Journal of Operational Research*, vol. 182, no. 2, pp. 481–501, 2007, doi: https://doi.org/10.1016/j.ejor.2006.07.009.

[2] X. Jiang, Y. Zhou, Y. Zhang, L. Sun, and X. Hu, 'Order batching and sequencing problem under the pick-and-sort strategy in online supermarkets', *Procedia Computer Science*, vol. 126, pp. 1985–1993, 2018, doi: https://doi.org/10.1016/j.procs.2018.07.254.

[3] G. P. Sharp and D. R. Gibson, 'Order batching procedures', *58*, no. European Journal of Operational Research, 1992.

[4] N. Gademann and V. de S. Velde, 'Order batching to minimize total travel time in a parallel-aisle warehouse', *IIE Transactions*, vol. 37, no. 1, pp. 63–75, 2005, doi: 10.1080/07408170590516917.

[5] C. A. Valle and B. A. Beasley, 'Order batching using an approximation for the distance travelled by pickers', no. European Journal of Operational Research, 2019.

[6] H. Ratliff and A. Rosenthal, 'Order-Picking in a Rectangular Warehouse: A Solvable Case of the Traveling Salesman Problem', *Operations Research*, vol. 31, pp. 507–521, 1983.

[7] J.-F. Cordeau, G. Laporte, M. Savelsbergh, and D. Vigo, 'Vehicle Routing', in *Transportation, handbooks in operations research and management science*, vol. 14, 2007, pp. 195–224.

[8] Y. A. Bozer and J. W. Kile, 'Order batching in walk-and-pick order picking systems', *International Journal of Production Research*, vol. 46, no. 7, pp. 1887–1909, 2008, doi: 10.1080/00207540600920850.

[9] O. Briant, H. Cambazard, D. Cattaruzza, N. Catusse, A.-L. Ladier, and M. Ogier, 'An efficient and general approach for the joint order batching and picker routing problem', *European Journal of Operational Research*, vol. 285, no. 2, pp. 497–512, 2020, doi: https://doi.org/10.1016/j.ejor.2020.01.059.

[10] O. Kulak, Y. Sahin, and M. E. Taner, 'Joint order batching and picker routing in single and multiple-cross-aisle warehouses using cluster-based tabu search algorithms', *Flexible Services and Manufacturing Journal*, vol. 24, no. 1, pp. 52–80, Mar. 2012, doi: 10.1007/s10696-011-9101-8.

[11] M. Masae, C. H. Glock, and E. H. Grosse, 'Order picker routing in warehouses: A systematic literature review', *International Journal of Production Economics*, vol. 224, p. 107564, 2020, doi: https://doi.org/10.1016/j.ijpe.2019.107564.

[12] P.-Y. Chang, P. Damodaran *, and S. Melouk, 'Minimizing makespan on parallel batch processing machines', *International Journal of Production Research*, vol. 42, no. 19, pp. 4211–4220, 2004, doi: 10.1080/00207540410001711863.

[13] L. J. van Rensburg, 'Artificial intelligence for warehouse picking optimization - an NP-hard problem', M.Sc., Uppsala University, 2019.

[14] C. A. Valle, J. E. Beasley, and A. S. da Cunha, 'Optimally solving the joint order batching and picker routing problem', *European Journal of Operational Research*, vol. 262, no. 3, pp. 817–834, Nov. 2017, doi: 10.1016/j.ejor.2017.03.069.

[15] J. Li, R. Huang, and J. B. Dai, 'Joint optimisation of order batching and picker routing in the online retailer's warehouse in China', *International Journal of Production Research*, vol. 55, no. 2, pp. 447–461, 2017, doi: 10.1080/00207543.2016.1187313.

[16] H. Psaraftis, M. Wen, and C. Kontovas, ‘Dynamic Vehicle Routing Problems: Three Decades and Counting’, *Networks*, vol. 67, 2015, doi: 10.1002/net.21628.

[17] M. Bué, D. Cattaruzza, M. Ogier, and F. Semet, ‘A Two-Phase Approach for an Integrated Order Batching and Picker Routing Problem’, 2019, pp. 3–18.

[18] M. Bortolini, M. Faccio, E. Ferrari, M. Gamberi, and F. Pilati, ‘Design of diagonal cross-aisle warehouses with class-based storage assignment strategy’, *The International Journal of Advanced Manufacturing Technology*, vol. 100, no. 9, pp. 2521–2536, Feb. 2019, doi: 10.1007/s00170-018-2833-9.

[19] A. Fumi, L. Scarabotti, and M. Schiraldi, ‘The Effect of Slot-Code Optimization in Warehouse Order Picking’, *International Journal of Business and Management*, vol. 5, 2013, doi: 10.5772/56803.

[20] K. R. Gue and R. D. Meller, ‘Aisle configurations for unit-load warehouses’, *IIE Transactions*, vol. 41, no. 3, pp. 171–182, 2009, doi: 10.1080/07408170802112726.

[21] S. Henn, ‘Algorithms for on-line order batching in an order picking warehouse’, *Computers & Operations Research*, vol. 39, no. 11, pp. 2549–2563, 2012, doi: https://doi.org/10.1016/j.cor.2011.12.019.

[22] M.-C. Chen and H.-P. Wu, ‘An association-based clustering approach to order batching considering customer demand patterns’, *Omega*, vol. 33, no. 4, pp. 333–343, 2005, doi: https://doi.org/10.1016/j.omega.2004.05.003.

[23] S. Henn and G. Wäscher, ‘Tabu search heuristics for the order batching problem in manual order picking systems’, *European Journal of Operational Research*, vol. 222, no. 3, pp. 484–494, 2012, doi: https://doi.org/10.1016/j.ejor.2012.05.049.

[24] Ç. Cergibozan and A. Tasan, ‘Genetic algorithm based approaches to solve the order batching problem and a case study in a distribution center’, *Journal of Intelligent Manufacturing*, pp. 1–13, 2020, doi: 10.1007/s10845-020-01653-3.

[25] K. Braekers, K. Ramaekers, and I. V. Nieuwenhuyse, ‘The vehicle routing problem: State of the art classification and review’, *Computers & Industrial Engineering*, vol. 99, pp. 300–313, 2016, doi: https://doi.org/10.1016/j.cie.2015.12.007.

[26] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia, ‘A review of dynamic vehicle routing problems’, *European Journal of Operational Research*, vol. 225, no. 1, pp. 1–11, 2013, doi: https://doi.org/10.1016/j.ejor.2012.08.015.

[27] M. Mansouri, F. Lagriffoul, and F. Pecora, ‘Multi Vehicle Routing with Nonholonomic Constraints and Dense Dynamic Obstacles’, 2017, doi: 10.1109/IROS.2017.8206195.

[28] D. D. Bochtis and C. G. Sørensen, ‘The vehicle routing problem in field logistics part I’, *Biosystems Engineering*, vol. 104, no. 4, pp. 447–457, 2009, doi: https://doi.org/10.1016/j.biosystemseng.2009.09.003.

[29] R. D. Santis, R. Montanari, G. Vignali, and E. Bottani, ‘An adapted ant colony optimization algorithm for the minimization of the travel distance of pickers in manual warehouses’, *European Journal of Operational Research*, vol. 267, no. 1, pp. 120–137, 2018, doi: https://doi.org/10.1016/j.ejor.2017.11.017.

[30] F. Furini, I. Ljubić, and M. Sinnl, ‘An effective dynamic programming algorithm for the minimum-cost maximal knapsack packing problem’, *European Journal of Operational Research*, vol. 262, no. 2, pp. 438–448, 2017, doi: https://doi.org/10.1016/j.ejor.2017.03.061.

The Multi-Objective Team Set Orienteering Problem

H-E. Ben-Smida[1,2], Francisco Chicano[1], Saoussen Krichen [2]
[1]Universidad de Malaga, Spain
[2]University of Tunis, Tunisia

Abstract

In this paper we present a new problem called the Multi-Objective Team Set Orienteering Problem (MOTSOP). This problem can be used to model real-world problems like Tourist Trip Design Problems. MOTSOP is the generalization of the Team orienteering problem (TOP) [4] and the Set Orienteering Problem (SOP) [1]. Given a graph, we define clusters of nodes and associate a profit to each cluster. We also associate a visiting time to each node of the graph. We maintain the same logic as in SOP regarding the profit collection, where the profit is always linked to the cluster. That is, the profit is collected only if at least one node of a cluster is visited by one vehicle. This article presents opposing objectives which are minimizing the number of vehicles and maximizing profit. Our model seeks to find the best solution which can balance these two contradictory objectives. The difference between TSOP and SOP is that there is not only one single vehicle to collect the profit, instead the Team Set Orienteering Problem provides a fleet of vehicles to visit the nodes. We present a mixed integer linear programming formulation MILP for MOTSOP and we propose to solve the problem using new Genetic Iterated Neighborhood Algorithm (GINA). We show how the Multi-Objective Team Set Orienteering Problem can be used to solve the Tourist Trip Design Problem (TTDP).

Keywords— Set Orienteering Problem, Multi-Objective Optimization Problem, Team Set Orienteering Problem, Iterated local search, Genetic algorithm, Tourist Trip Design Problem

1 Introduction

According to the Tunisian federation of travel agencies FTAV, 98% of Tunsian travel agencies are threatened with bankruptcy and 28% of travel agency staff have been made redundant [2]. These figures show that the travel agency profession must be automated to reduce costs and ensure the services continuity. The main activity of travel agencies is to create a trip composed of a set or sequence of points-of-interests (POIs). This task is called Tourist trip design problem (TTDP) [5].

Typically, TOP [4] is a lightweight version of TTDP because it lacks the notion of Tourist different objectives and POIs categories. Adding new objectives and grouping POIs in clusters (categories), we can transform TTDP into MOTSOP. MOTSOP includes more realistic assumptions for the TTDP and, thus, is more beneficial on the scientific and operational side.

2 Multi-Objective Team Set Orienteering Problem (MOTSOP)

The extension of the SOP to multiple routes is defined as the team set orienteering problem (TSOP) with the objective of maximizing the collected profit across multiple routes. The extension of the TSOP to multiple objectives is defined as the Multi-Objective Team Set Orienteering Problem MOTSOP where the considered objectives are maximizing the profit obtained and minimizing the number of routes.

Each cluster has a profit that is obtained when one node in the cluster is visited. On the other hand, there is a maximum time budget for each route, what means that not all nodes can be visited in one route. Thus, maximizing profit usually implies increasing the number of routes, and the two objectives are in conflict.

MOTSOP is NP-hard because SOP reduces to MOTSOP and it has been proven in previous works that SOP is NP-hard. Therefore, no polynomial time algorithm is known to solve the MOTSOP to optimality.

2.1 Formulation

We have a directed graph $G(V, E)$ where V is the set of nodes and $E \subseteq V \times V$ is the set of edges. We label the nodes with non-negative integers and we distinguish the depot with number 0. The set $V/\{0\}$ is partitioned into pairwise disjoint sets called *clusters* and denoted with D_l where $l = 1, 2, \ldots, C$. Each cluster has a profit p_l associated. This profit is obtained if a node of the cluster is visited by one vehicle. A vehicle requires time t_{ij} to traverse edge $(i, j) \in E$. When a vehicle stops in node i, it stays there for s_i time, which is called the *visit time* of

node i. A route cannot last more than T_{max} time in total, including the visit time of the nodes. The bi-objective Team Set Orienteering Problem consists in maximizing the profit obtained and minimizing the number of routes. A mixed integer linear programming formulation follows.

$$\max \sum_{l=1}^{C} p_l z_l, \tag{1}$$

$$\min \sum_{(0,i)\in E} x_{0i}, \tag{2}$$

subject to:

$$\sum_{(j,i)\in E} x_{ji} = y_i \ \ \forall i \in V/\{0\}, \tag{3}$$

$$\sum_{(i,j)\in E} x_{ij} = y_i \ \ \forall i \in V/\{0\}, \tag{4}$$

$$z_l \leq \sum_{i\in D_l} y_i \ \ \forall 1 \leq l \leq C, \tag{5}$$

$$u_j - u_i \geq (t_{ij} + s_j)x_{ij} + (x_{ij} - 1)T_{max} \ \ \forall (i,j) \in E, \tag{6}$$

$$u_0 = 0, \tag{7}$$

$$u_j + c_{j0}x_{j0} \leq T_{max} \ \ \forall j \in V/\{0\}, \tag{8}$$

$$u_j \geq 0 \ \ \forall j \in V/\{0\}, \tag{9}$$

$$x_{ij} \in \{0,1\} \ \ \forall (i,j) \in E, \tag{10}$$

$$y_i \in \{0,1\} \ \ \forall i \in V/\{0\}, \tag{11}$$

where x_{ij} represents if edge (i,j) is traversed by any vehicle, y_i represents if node i is visited by any vehicle and z_l represents if any node of cluster D_l is visited by a vehicle. Equations (3) and (4) together ensure that at most one vehicle visits each node and leaves from it, providing also the correct value for y_i. Equation (5) ensures that a cluster is visited if at least one node in the cluster is visited. Equation (6) and (7) defines u_j as an upper bound of the leaving time at node j. Equation (8) ensures that no route will take more than T_{max} time. Finally, Equations (9) to (11) define the domain of the decision variables.

2.2 Adapt the TTDP into the MOTSOP

When tourists want to visit a city, they cannot visit every point of interest (POI), as they are constrained in time and budget. At the same time, travelers select POIs based on categories, since they does not have a clear idea about POIs. Most real-world touristic travels are multi-objective, since all travelers want to get the most out of their trip in fewer days. For this reason, we need to add the concept of categories and the multi-objective model to TSOP.

We can cluster POIs in some categories and assign a score to each category, where the score can represent the profit of each category. This profit is obtained only if at least one destination from this category is visited in one day at most. Another constraint is the visiting time, where at each point we consider the visiting time or the service time at each POI. In real life problem, the first goal is to plan the journey in minimum number of days and the second one is to enjoy and maximize leisure. There is a similarity between TTDP and MOTSOP if we model one day in the TTDP with one route in the MOTSOP and we relate maximizing leisure in TTDP to maximizing profit in MOTSOP.

Figure 1 could explains how to transform the TTDP into the MOTSOP. Imagine that a travel agent want to design a tailor-made trip. The goal is to minimize the number of night stays in a hotel to make the trip cheaper and at the same time increase the number of POIs categories.

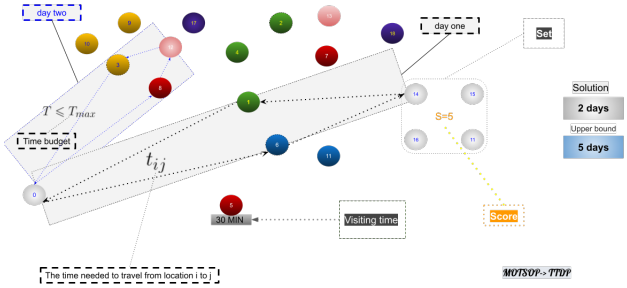


Figure 1: Example of solution containing two days trip. It illustrates how we adapt the TTDP into the MOTSOP.

3 Hybrid approach

In this work, we propose the Genetic Iterated Neighborhood Algorithm, GINA, which is a hybridization of a genetic algorithm and Iterated Neighborhood Algorithm (INA)[3] to jointly solve the MOTSOP. We incorporate INA into the GA to complement the genetic search by the advantages of INA features. The algorithm first generates a population of n feasible solution. Then, it applies binary tournament selection to select two solutions to apply a crossover operator. The solution resulting from the crossover is mutated and a local search is aplied, using the swap neighborhood [3]. Finally, an elitist replacement operator compares the new solution with the worst in the population and the best of these two is included in the population, removing the worst. One iteration of the algorithm is graphically illustrated in Figure 2.

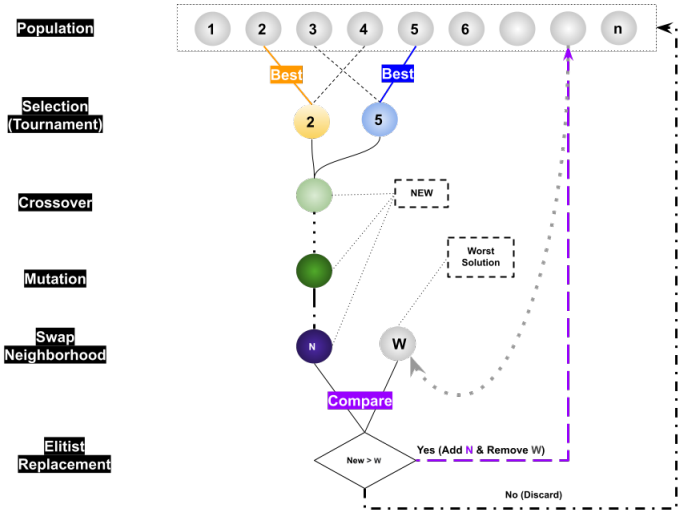


Figure 2: Algorithm Steps.

4 Conclusion

This abstract contains several contributions: the definition of a new problem with applications in tourism, a MILP formulation of the problem, and the proposal of a hybrid algorithm to solve it. In future work, we will introduce new elements to the problem to make it more realistic, like adding must-visit POIs, and time windows for each visit. These constraints can make the problem more real and we can apply it in the tourism industry to generate efficient app for travelers. We also plan to investigate the performance of several algorithms for solving the problem. In particular, we need to compare our proposed hybrid algorithm with ILP solvers using the proposed MILP formulation. Since no instances exist for this problem, we will take SOP instances and transform them to generate MOTSOP instances.

References

[1] Claudia Archetti, Francesco Carrabs, and Raffaele Cerulli. The set orienteering problem. *European Journal of Operational Research*, 267(1):264–272, 2018.

[2] Jabeur ben Attouch. Presque toutes les agences de voyages sont menacés description, 2020.

[3] Houssem E Ben-Smida, Francisco Chicano, and Saoussen Krichen. Iterated granular neighborhood algorithm for the taxi sharing problem. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*, pages 291–304. Springer, 2020.

[4] I-Ming Chao, Bruce L Golden, and Edward A Wasil. The team orienteering problem. *European journal of operational research*, 88(3):464–474, 1996.

[5] Warisa Wisittipanich and Chanagan Boonya. Multi-objective tourist trip design problem in chiang mai city. In *IOP Conference Series: Materials Science and Engineering*, volume 895, page 012014. IOP Publishing, 2020.

Mobile Clinics Routing in Response to Covid-19 Outbreak: An Intelligent Hyperheuristic Approach

, Kassem Danach [1, 4], Jomana Al-Haj Hassan[1], Abbas Tarhini[3], Shahin Gelareh[2]

[1]Department of Management Information System, Islamic University of Lebanon, Lebanon
[2]Département de Réseaux et Télécommunications, IUT de Bethune, Université Artois, Béthune, F-62400, France
[3]Lebanese American University, Lebanon, Beirut, Lebanon
[4] Laboratoire de Génie Informatique et d'Automatique de l'Artois (LGI2A), Université d'Artois, F-62400 Béthune, France

Abstract. On March 2020, the World Health Organization (WHO) announced the COIVD-19 as global pandemic that caused thousands of deaths and brought the world to a standstill with huge economic burden (World Health Organization, 2020). Health is an essential factor for sustaining a better life in a better world. Today, for different reasons, several districts in our countries would be deprived from the needed health support and thus, in such cases, we need to deliver health care to those regions. Despite of its considerable cost, mobile clinic remains one of the good solutions to deliver health care to critical areas in our countries. A recognized problem in this domain is minimizing the cost of mobile clinics route in a way that the number of served patients is maximized. This problem is known as the mobile clinics routing problem (MCRP). The purpose of this paper is to present a novel approach that, within the given limited resources, it minimizes the cost and the traveling distance of mobile clinics while maximizing the number of patients served as per priorities assigned according to the patients' medical status. This paper implements and tests an intelligent variable neighbourhood search algorithm for MCRP.

Keywords: pandemic management, Hyperheuristic, Machine Learning.

1 Introduction

Today, several reasons prevent governments in different countries from providing suitable health care to its citizens. Some of these reasons are related to pandemics, natural disasters, or even to terrorist acts that hit health care centers, which might take a long time for governments to recover. Other reasons could be the distributions of population in different geographic areas that lead to establishing health care centers in these areas become expensive and unfeasible projects. All this lead governments to

think about delivering health care to citizens instead of having them cross long distance to acquire it. Mobile clinics were one of the feasible solutions to be considered.

Mobile clinic is defined as “a typical clinic with all needed machines and equipment which can move from one place to another to provide better service. It is a way of having all the equipment built into a moving vehicle to be ready for usage at any time and to reach long distances where there are no fixed clinics” (Segen, 2012). Today, the need for mobile clinics is demanding to resolve the dilemma of an enormous number of people deprived from health care services. This fact is revealed in World Health Organization and the Word Bank’s (2015) report which reported that 400 million people do not have access to essential health services due to the increase in natural disasters, epidemics, food shortage, refugee aid and military conflicts. The basic problem reported is the lack of or destroyed infrastructure, which hinders the performance or the reconstruction of clinics. The report further “recommend that countries pursuing universal health coverage should aim to achieve a minimum of 80% population coverage of essential health services”, and that everyone everywhere should be protected from catastrophic and impoverishing health services. The demand for universal health coverage is further asserted by the World Health Organization (2020), which requires the “health policies and programs focus on providing quality health services for the poorest people, women and children, people living in rural areas and those from minority groups". Further, in its (2015) report on health care services, the Rural Health Services Review Committee of Alberta found that “some fixed clinics are closing in the rural area because of the declining population, economic recession, shortages of physicians and other health care professionals, a disproportionate number of elderly, poor, and underinsured residents, and high rates of chronic illness”. Participants in the report claimed that it is difficult for them to find out a professional doctor or a trusted health care center in rural areas. Among most of citizens in the rural areas believed that the government should provide clinics in the rural areas other than sending people to the other places and clinics.

At present, mobile clinics offer a feasible solution to the above dilemma. In addition to being equipped to respond to the COVID 19 outbreak, mobile clinics provide many programs and centers including the da Vinci® Surgical system, cancer services, diabetes, digestive health, emergency services, and many other health care services. In fact, such a variety of services offered by mobile clinics provides efficient health care, flexibility, accessibility to patients; still a major problem faces mobile clinics operators. The problem is to optimize the mobile clinics trip in a way that it minimizes the cost and the traveling distance of the mobile clinics routes while maximizing the number of patients served as per priorities assigned according to the patients’ medical status. Such a problem is very critical and hard since it incorporates a vehicle routing and scheduling problems. This makes the mobile clinic routing problem (MCRP) a nondeterministic polynomial time (NP)-complete problem. Such problems may be too computationally-intensive to find their exact solution. In such cases, heuristic techniques can be

effective. In fact, due to their random nature, heuristic techniques are not guaranteed to find an optimal solution for any problem, but they will find a good solution if one exists.

In this work, we propose a novel variable neighbourhood search (VNS) metaheuristic for solving MCRP. The selection of the VNS metaheuristic was motivated by Hansen and Mladenovic (2001) perception which states that "it aims at solving very rapidly, very large instances, and increasing precision and reducing solution time for combinatorial optimization problems; further VNS uses strategies for search diversification and intensification that have proved effective in a variety of optimization problems". This paper presents and tests the VNS algorithm with randomly generated input that simulates actual data in MCRP with instances of 200 patients and 12 mobile clinics. The test cases results clearly show a significant improvement in the cost of the objective function and the execution time.

The rest of this paper is organized as follows. The next section is devoted for the literature review. Section 3 presents the mathematical formulation. The Variable Neighbourhood Search metaheuristic is discussed in Section 4. Section 5 presents the empirical results. Finally, the conclusion is presented in Section 6.

2 Literature Review

Several researchers addressed the mobile clinics routing problem (MCRP). A number of researchers considered reducing the mobile clinics routes as the most important factor to optimize since the distance may sometimes hinder the usage of a medical service (Morrill, R. L. and Earickson, R., 1968; Shannon, G. et al., 1969; Muller, I., et al., 1998; Ruggiero, C.P. and Gloyd, S., 1995; Verter, V. and Lapierre, S.D., 2002). Literature also shows that many governments equip clinics with mobile healthcare facilities instead of building fixed healthcare centers (Ruggiero, C.P. et al.,1995; Fox-Rushby, J.A., 1995; Gilson, L., 1995; Dyer, J.J., 1996; Hodgson, M.J.,et al. 1998; Mackle, G.A. and Giles, M., 1995; Ruggiero, C.P. and Gloyd, S., 1995). Furthermore, other researchers studied the idea of building fixed healthcare centers that could be reached within a given walking distance instead, and thus aimed at finding the best location a healthcare center is suited, and how they should be staffed (Rahman, S. and Smith, D.K., 2000; Oppong, J.R., 1996; Mehrez, A. et al.,1996; Hindle, T. and Ngwube, C., 1990; Goldstein, S.M., et al., 2002; Galvao, R.D., et al., 2002; Doherty, J., et al. 1996; Chu, S. and Chu, L., 2000; Berghmans, L., et al., 1984; Hachicha et al. 2000). In fact, most of the previous works dealt with the problem as either a tour planning distance reduction problem or as a fixed healthcare allocation problem. In addition to that, only few articles addressed this problem in the past five years, at a time international healthcare organizations urge the need of further research in this area. All this motivated us to present our novel algorithm to solve this problem. To the best of our knowledge, none of the previously proposed solutions used VNS algorithm to solve the MCRP. Unlike other evolutionary methods, VNS uses strategies for diversification and intensification that

have proved effective in a variety of optimization problems (Hansen, P. and Mladenovic, N., 2001). In fact, our proposed solution was able to find very good results tested on large-sized problems with 200 patients and 12 mobile clinics within an acceptable running time.

3 Problem Statement

Disasters, such as pandemics, natural disasters, wars, etc., are intractable problems for humanity which lead to human loses that are not easy to recover. Routing of mobile clinics to cure victims in such environments is an important problem. The objective is to minimize the mobile clinic route time while taking into consideration other important parameters such as patients' priorities, road conditions, and clinics' capacities. In order to solve this problem we propose a mathematical model and a hyper-heuristic solution method that comprises several low-level heuristics guided by a reinforcement learning heuristic.

4 Mobile Clinic Routing Problem

In mobile clinic routing problem, mobile clinics are required to serve many patients living in a given spatial area. These patients are assigned priorities according to their medical status. Given a limited number of resources, a mobile clinic is required to serve the maximum possible number of patients, taking into consideration the patients priorities. In this problem, it is assumed that there exists an overall time limiting the whole process, and that each mobile clinic can handle one operation at a time, where each operation is processed within an uninterrupted period. It is worth noting that failing to find a feasible assignment of mobile clinic to patients would be very expensive in terms of wasting exceptional healthcare resources processed in unexpected time frames that would further lead to threatening patients' lives in some cases. The effect of such assignment is illustrated in the following example. This example shows two different permutations to assign one mobile clinic to eight patients taking into consideration the operation time, traveling time and patient priorities. The first permutation is illustrated in Figure 1. In this permutation, the mobile clinic is assigned to the corresponding eight patients consecutively 1, 2, 3, 4, 5, 6, 7, 8.

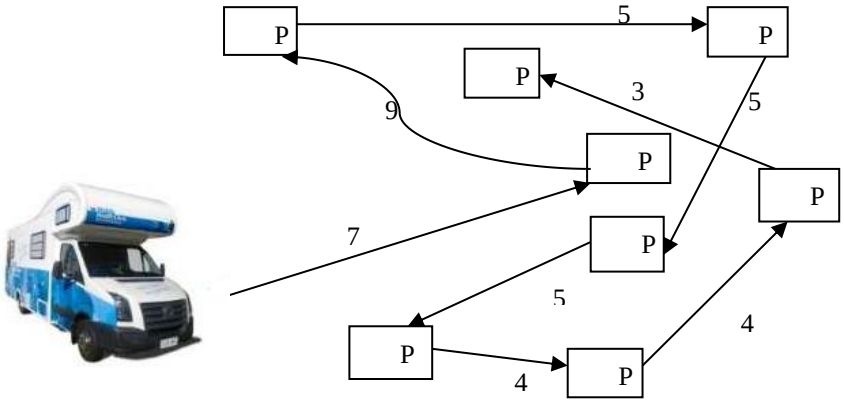


Fig. 1. Mobile clinic assignment – First permutation

Figure 1 clearly shows the path traversed by the mobile clinic to serve the eight patients. The numbers on the arrows represents the time taken to travel from one patient to another. For example, the time needed to reach patient 1 is 7, and to reach patient 2 from patient 1 we need 9 units of time. The cost of such a permutation would be calculated as follows (this formula will be verified in the following section): ∑(ti + tij) * PRi, = (5+7)*1+ (5+9)*2+ (7+6)*1+ (5+5)*1+ (8+5)*1+ (5+4)*1+ (5+4)*1+ (7+3)*1 = 104, where ti is the operation time at patient i, and tij is the time from patient i to patient j, and PRi is the priority of patient i.

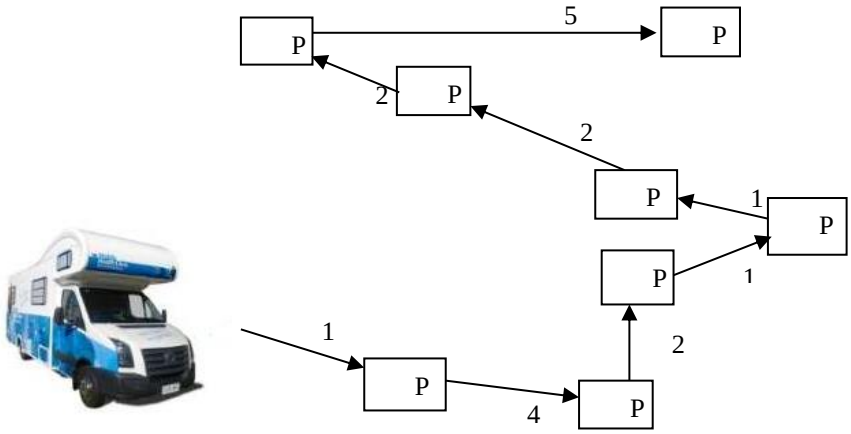


Fig. 2. Mobile clinic assignment – Second permutation

Obviously, the combination of the time and priority between patients has a tremendous effect on the total cost. Therefore, it is rational to assume that if we assigned the mobile clinic to patients with lower travelling time, we would get an optimized solution. To explain this concept further, consider Figure 2, which represents a different permutation from that given in Figure 1 where the mobile clinic is assigned to patients in a different sequence. The cost of such a permutation would be calculated as follows (this formula will be verified again in the following section): ∑(ti + tij) * PRi, = (8+1)*1+ (5+4)*1+ (5+2)*1+ (5+1)*1+ (5+1)*1+ (7+2)*2+ (5+2)*1+ (7+5)*1 = 72, where ti is the operation time at patient i, and tij is the time from patient i to patient j, and PRi is the priority of patient i. This clearly proves that reaching patients in different sequences may affect the total cost.

5 Mathematical Formulation

Assignment problems can be represented by graphs (Wood, 1968). Let G (VG, EG) be a graph in which vertex vi VG represents a patient, and |VG| = A is the total number of patients to be assigned to mobile clinics. Vertex weight PRi represents the priority that patient vi holds; further it also holds the time ti taken by each operation at

patient vi. Edge e □ EG joining two vertices vi and vj represents the existence of a flow between patient vi and patient vj; the weight of edge e, tij, represents the travel time between the two patients vi and vj.

$$\boldsymbol{Minimize}\ \sum_{i,j\neq i}^{A}(\boldsymbol{t_i} + \boldsymbol{t_{ji}}).\boldsymbol{PR_i} \qquad (1)$$

Given a fixed number of patients and mobile clinics distributed in the whole country. Distances between all mobile clinics and patients, in addition to distances between patients themselves is given as well. Each patient must be served exactly by one mobile clinic. Each mobile clinic can serve exactly one patient at a given time. Each mobile clinic takes an operation time at each patient.

6 Variable Neighborhood Search (VNS)

Variable neighborhood search (VNS), proposed by Mladenovi´c and Hansen (1997), is another kind of metaheuristics to solve combinatorial optimization problems. It explores neighborhoods found in one solution space, and moves from there to a new one randomly or intelligently if and only if an improvement was made. VNS includes two main phases: 1) the shaking phase and 2) the local search phase. The first one aims to diversify the search space in order to escape from local optima, while the objective of the second phase is to intensify the search around the current solution in order to improve it. VNS phases avoid local optimum and helps to explore and exploit the search space in order to achieve a near optimal solution. Each phase of the VNS applies more than one heuristic; further description on the heuristics selection method is found in the below subsections.

In fact VNS is used by several researchers to solve scheduling problems. In their work, Remde et al.(2007) proposed a VNS in order to solve workforce scheduling problem where it proved to be a powerful tool compared with the solution quality resulted by a genetic algorithm. Further, Hsiao (2012) propose a VSN based hyperheuristic method to solve more than one problem such as job shop scheduling, and bin packing problems.

In this work, the proposed VNS constructs a feasible solution as an initial step before moving to the shaking and the local search phases. In the shaking phase, the tabu search method is used. It selects a low level heuristic from the set of perturbation heuristic. In the local search phase, a random cycle of the existing predefined improvement heuristics is formed. It applies improvement heuristic while it is still improving the solution and switches to the next one in the cycle in case it stops improving the solution. This process keeps repeating until n consecutive non-improving solutions are reached. The proposed low level-heuristic are classified as constructive, improvement, and perturbation.

In our study, the constructive algorithm is applied only once at the beginning of the assignment problem in order to build the initial solution. In fact constructive heuristics build a solution from scratch based on several predefined rules. Thus, taking into consideration the overall (task length) time allocated to each mobile clinic, the proposed

constructive method of this study, CS1, schedules for each mobile clinic the corresponding adequate patients.

On the other hand, Improvement heuristics start from a complete solution and apply some moves in order to improve the objective function value. In our study, diff erent improvement heuristics are proposed:

1. Better-sequence-on-the-same-mobile-clinic: for a given mobile clinic, we reorder (one task per time) the sequence of allocated patients to be served by this mobile clinic and among the improving solution found, we move to the best found feasible solution in a greedy manner. The pseudocode for local search (LS1) is described in Algorithm 1 in Figure 4. We note that Q(Hs) represents patients assigned to mobile clinic s.
2. Inset/drop-between-two-diff erent-mobile-clinic: aiming at making a balanced utilization and fair distribution of tasks among mobile clinics, for two randomly chosen mobile clinics ri,rj($i \neq j$), we consider the following moves: 1) move one tasks from mobile clinic ri to mobile clinic rj ($i \neq j$) such that the diff erence between the objective functions of mobile clinic ri and mobile clinic rj, $\Delta = (OF(ri) - OF(rj))$, is minimized. 2) removing a patient from the list of patients being served by a given mobile clinic and insert it in a proper place within the sequence of patients being served by the mobile clinic with the least objective function.

Algorithm 1 LS1

```
    procedure LS1(S) .   INPUT: Clinics Queues
2:    for s = 1,...,m do                    ∀s = 1,...,m, (m = |H|)
          Queue temp;
4:    for i = 1,...,n do                    ∀i = 1,...,n, (n = |Q(Hs)|
          MinimumOF;
6:        CurrentPatient;
          if then MinimumOF > OF(Pi,Hs)
8:            MinimumOF = ← OF(Pi,Hs);
              CurrentPatient ← Pi;
10:       end if
          tabu ← Pi;
12:       Hs ← Pi;
      end for
14:   end for
      return H;
16: end procedure
```

Fig. 4. Local Search algorithm

Perturbation heuristics start with a complete solution and do some changes in order to inject some diversification. It consist of exploring the search space step in order to escape from a local optima. In this study, we implement two perturbation heuristics as follows:

1. *Re-Construct:* consists of destroy a part of the current solution and rebuild it using the proposed constructive heuristic.
2. *SW1:* consists of swapping two patients between two clinics if they can offer these patients their needed medical services.

3. *Crossover:* consists of swapping of 'genetic material' is made with the mobile clinic (part of the chromosome), while in the mutation operator is made with two mobile clinics, one in each chromosome.
4. *Remove/Insert:* removes a patient from a clinic queue and re-insert it to another one.

The pseudocode for *SW1* is described in Algorithm 2 in Figure 4.

Algorithm 2 SW1

```
1:  procedure SW1(S) .                    INPUT: Clinics Queues
2:      Random R1 ∈ 1,...,m                   (m = |H|)
3:      Random R2 ∈ 1,...,m−R1                (m = |H|)
4:      RandomPosition1 P1 ∈ 1,...,l          (l = |Q(HR1)|)
5:      RandomPosition1 P2 ∈ 1,...,k          (k = |Q(HR2)|)
6:      Swap(H_R1(P1), H_R2(P2))
7:      return S;
8:  end procedure
```

Fig. 4. Algorithm for swapping patients in perturbation heuristic

7 Experimental Results

In this section, the results of VNS algorithm are presented. This algorithm is tested on a set of 33 test cases; each of these test cases has a different number of patients and mobile clinics with different timing to reach neighbor patients. Since real data are not available to evaluate the performance of the proposed algorithms, we generated synthetic data. The test cases are illustrated in TABLE I. This table shows several parameters: the test case number, the number of patients, the number of used mobile clinics. In these 33 test cases we wanted to test how the performance is affected for different combinations of patients assigned to mobile clinics. Thus, in some test cases, (eg.TC1-TC3), we varied the number of patients and kept the number of mobile clinics fixed; in other test cases, the number of patients is fixed and the number mobile clinics is varied. Further, different test cases scaled up in size to test our algorithm on small, medium, and large sized-problems.

Table 1. Initial, Best solution and execution time results

TestCase #	Number of Patients	Number of MH
TC1	8	1
TC2	10	3
TC3	15	3
TC4	15	5
TC5	20	5
TC6	25	3
TC7	25	5

TestCase #	Number of Patients	Number of MH
TC18	50	10
TC19	75	7
TC20	75	9
TC21	75	10
TC22	100	7
TC23	100	9
TC24	100	10

TC8	25	7
TC9	30	3
TC10	30	5
TC11	30	7
TC12	35	3
TC13	35	5
TC14	35	7
TC15	50	5
TC16	50	7
TC17	50	9

TC25	100	12
TC26	150	7
TC27	150	9
TC28	150	10
TC29	150	12
TC30	200	7
TC31	200	9
TC32	200	10
TC33	200	12

8 Results and Discussion

Table 2 summarizes the best results of the VNS algorithms applied on the 33 test cases TC1-TC33. The VNS algorithm is implemented using Java language. Furthermore, the tests were carried out on a Pentium(R), dual-core CPU T4300- 2.10 GHz, and 2 GB RAM with Windows Vista SP1.

Table 2. Initial, Best solution and execution time results

Instance #	Number of Patients	Number of MH	Initial Objective Function (OF)	Best Objective Function	Execution Time
TC1	8	1	104	72	2.001
TC2	10	3	472	440	4.001
TC3	15	3	868	799	4.024
TC4	15	5	547	513	3.119
TC5	20	5	942	705	6.014
TC6	25	3	2375	2198	5.030
TC7	25	5	1475	1409	3.441
TC8	25	7	1093	1083	5.887
TC9	30	3	3413	3204	7.303
TC10	30	5	2089	1955	5.414
TC11	30	7	1586	1475	7.011
TC12	35	3	4340	4049	8.007
TC13	35	5	2665	2498	6.800
TC14	35	7	1980	1780	7.100
TC15	50	5	5772	5299	6.004
TC16	50	7	4116	3822	5.009
TC17	50	9	3312	3112	6.474
TC18	50	10	3002	2825	6.506
TC19	75	7	9556	8680	6.714
TC20	75	9	7502	6866	7.303
TC21	75	10	6752	6233	6.410
TC22	100	7	18557	17541	5.004
TC23	100	9	17373	16185	6.074

TC24	100	10	14245	13148	6.095
TC25	100	12	12216	11170	6.147
TC26	150	7	49165	45330	5.542
TC27	150	9	49056	46121	4.685
TC28	150	10	38394	35708	7.420
TC29	150	12	33388	30773	8.146
TC30	200	7	83511	76447	7.305
TC31	200	9	76780	68330	7.415
TC32	200	10	64016	57008	6.001
TC33	200	12	57662	52003	6.099

In Table 2, for each instance several parameters are shown such as the total time the initial objective function value, the best value of the objective function found by the VNS, and the execution time taken to run the proposed algorithm. The results in Table 2 are further illustrated in figure 7. Figure 7 shows the percentage of objective functions improvement using the VNS algorithm. It is worth noting the VNS algorithm was able to improve the solution in at least 5% in less than 10 seconds. As for the significant improvement in TC1 and TC5 and the minor improvement in TC8, it is mainly affected by the traversal time from one patient to another and the priority assigned to every patient. This is clarified in the example shown in Tables 3-a and Table 3-b.

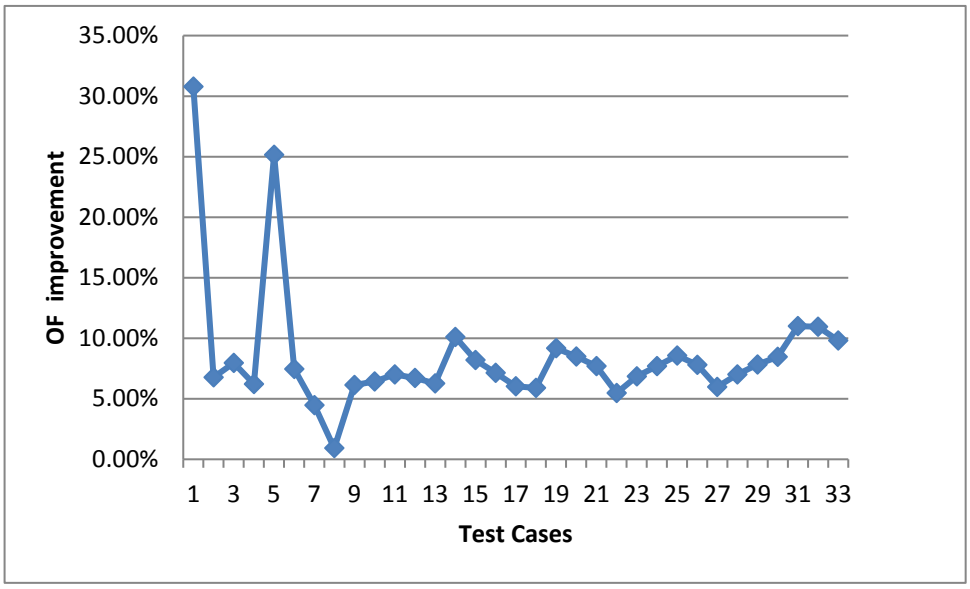


Fig. 7. Percentage of objective function improvemnet by VNS algorithm

Table 3 shows details related only to TC1. The parameters shown in the Table are the patient number, operation time taken at every patient location t_i, the traversal time from patient i to patient j t_{ij}, the priority assigned to every patient PR_i and the objective function OF.

Table 3. Priorities and traversal time between patients in TC1.

Patient#	t_i	t_{ij}	PR_i	OF
1	5	7	1	12
2	5	9	2	28
3	7	6	1	13
4	5	5	1	10
5	8	5	1	13
6	5	4	1	9
7	5	4	1	9
8	7	3	1	10
Objective Function				104

(a)

Patient#	t_i	t_{ij}	PR_i	OF
5	8	1	1	9
6	5	4	1	9
4	5	1	1	6
7	5	1	1	6
1	5	1	1	6
8	7	2	2	18
2	5	1	1	6
3	7	5	1	12
Best Objective Function				72

(b)

In Table 3-a, the mobile clinic path to serve all patients was sequential from 1 till 8 with almost all patients had low priorities with initial objective function equals to 104. The value of the objective function is improved to 72 as shown in Table 3-b by simply changing the order of reaching patients. It should be clear from this example that as patients don't have competing priorities and there exist a chance to find neighbor patients living close to each other, there is a chance of a significant improvement in the objective function.

9 Conclusion

In this paper, we presented our vision to solve the mobile clinic routing problem. The problem entails the assignment of a maximum number of patients with priorities to mobile clinics in such a way to reduce the total mobile clinics traversal time. We presented a variable neighborhood search (VNS) algorithm for the MCRP. VNS is a metaheuristic that we were the first to utilize to solve the assignment problem in mobile clinic routing problem. In fact the methods of the VNS have been enhanced to solve this problem. One contribution of this work is proposing and testing a unique algorithm (VNS) that has not been used before for solving the MCRP problem. The results indicate that our VNS algorithm was able to improve the solution in at least 5% in less than 10 seconds. This is due to the VNS nature that focus the Diversification and then intensification methods. A unique constructive method has been used to boost the quality of the initial population. Clearly, this process affected the final results as it aided in yielding the best known solution at earlier stages of the iterations giving an edge when compared to other related work and research that did not focus on generating such enhanced solutions. Additionally, the improvement and heuristic improvement methods have been also reinforced to give better results.

References

1. BERGHMANS, L., SCHOOVAERTS, P., TEGHEM, J., (1984). “Implementation of health facilities in a new city”, J. of the Operational Research Society 35, pp. 1047-1054
2. CHU, S., CHU, L., (2000). “Amodelingframeworkforcliniclocationandserviceallocation”, International Transactions in Operational Research 7, pp. 539-568
3. DOHERTY, J., RISPEL, L., WEBB, N., (1996). “Developing a Plan for Primary Health Care Facilities in Soweto, South Africa”, Part II: Applying Locational Criteria, in: Health Policy and Planning, vol. 11, no. 4, pp. 394-405, Oxford University Press, Oxford
4. DYER, J.J., (1996). “Comparative costs of mobile and fixed-clinic Primary Health Care services”, South African Medical Journal 86, pp. 528-530
5. FOORD, F., (1995). “Gambia: Evaluation of the mobile health care service in West Kang district”, World Health Statistics Quaterly 48, pp. 18-22
6. FOX-RUSHBY, J.A., (1995). “Gambia: cost and effectiveness of a mobile maternal health care service, West Kiang”, World Health Statistics Quaterly 48, pp. 23-27
7. GALVAO, R.D., ESPEJO, L.G.A., BOffEY, B., (2002). “A hierarchical model for the location of perinatal facilities in the municipality of Rio de Janeiro”, European J. of Operational Research 138, pp. 495-517
8. GILSON, L., (1995). “Keeping stocked in a problem health center. Management and health care reform in Sub-Saharian Africa”, Social Science and Medicine 40, pp. 695-710
9. GOLDSTEIN, S.M., WARD, P.T., LEONG, G.K., BUTLER, T.W., (2002). “The effect of location, strategy, and operations technology on clinic performance”, J. of Operations Management 20, pp. 63–75
10. HANSEN, P., MLADENOVIC, N. (2001). Variable Neighbourhood Search: Principles and Applications. European Journal of Operational Research, 130(1), pp.449-467.
11. HINDLE, T., NGWUBE, C., (1990). “The relative accessibility for health districts to regionally supplied specialist services”, Health Services Research 3, pp. 16-21
12. HODGSON, M.J., LAPORTE, G., SEMET, F., (1998). “A covering tour model for planning mobile health care facilities in Suhum district, Ghana”, J. of Regional Science 38, no. 4, pp. 621-639
13. MEHREZ, A., SINUANY-STERN, Z., ARAD-GEVA, T., BINYAMIN, S., (1996). “OntheImplementation of Quantitative Facility Location Models: The Case of a Clinic in a Rural Region”, J. of the Operational Research Society, no. 47, pp. 612-625
14. MORRILL, R. L., EARICKSON, R. (1968). “Clinic variation and patient travel distance”, Inquiry 5, pp. 26-34.
15. MACKLE, G.A., GILES, M., (1995). “The results of two years operation of the visiting specialist service to the Waikato Mobile Ear Clinic”, New Zealand Medical Journal 108, pp. 410-413
16. MULLER, I., SMITH, T., MELLOR, S., RARE, L., GENTON, B., (1998). “The effect of distance from home on attendance at a small rural health center in Papua New Guinea”, International Journal of Epidemiology 27, pp. 878-884
17. OPPONG, J.R., (1996). “Accomodating the rainy season in third world location-allocation applications”, Socio-Economic Planning Sciences 30, pp. 121-137
18. RAHMAN, S., SMITH, D.K., (2000). “Use of location-allocation models in health service development planning in developing nations”, European J. of Operational Research 123, 437-452
19. RUGGIERO, C.P., GLOYD, S., (1995). “Evaluation of vision services delivered by a mobile eye clinic in Costa Rica”, Optometry and Vision Science 72, pp. 241-248

20. RURAL HEALTH CARE REVIEW COMMITTEE REPORT. (2015) http://www.health.alberta.ca/newsroom/pub-annual-reports.html
21. SEGEN, J. (2012). Modern Medical Dictionalry. http://ww2.modern-medical-dictionaries.com/?folio=9POR7JU99
22. SHANNON, G. W., BASHSHUR, R.L., METZNER, CH. A. (1969), "The concept of distance as a factor in accessibility and utilization of health care", Medical Care Review 26, no. 2, pp. 143-161.
23. VERTER, V., LAPIERRE, S.D., (2002). "Location of Preventive Health Care Facilities", Annals of Operations Research 110, pp. 123-132
24. WOOD, D., (1968)."A system for computing university examination timetables," Computer Journal, vol. 11, pp. 41-47
25. WORLD BANK REPORT. (2015). http://www.worldbank.org/en/events/2015/06/12/trackinguhc
26. WORLD HEALTH ORGANIZATION (2015). http://www.who.int/healthinfo/universal_health_coverage/report/2015/en/
27. WORLD HEALTH ORGANIZATION. (2020, March 12). A report about health. Retrieved from https://www.euro.who.int/en/health-topics/health-emergencies/coronavirus-covid-19/news/news/2020/3/who-announces-covid-19-outbreak-a-pandemic

A genetic algorithm for the Three-dimensional Open Dimension Packing problem

C.-T.-T. Truong[1], L. Amodeo[1], and F. Yalaoui[1]

[1] LOSI, University of Technology of Troyes, Troyes, France
{cong_tan_trinh.truong, lionel.amodeo, farouk.yalaoui}@utt.fr

Abstract. The three-dimensional Open Dimension Packing problem (3D-ODPP) is a real-world driven optimization problem that aims at the minimization of package volume in right-size packaging systems. The problem can be found in many industrial scenarios, such as e-commerce secondary packaging. The objective of the 3D-ODPP is to find out the length, width, and height of the cardboard box that can be used to pack a given set of or products so that the volume of the box is minimal. Many literature researches have focused on exact methods to deal with the 3D-ODPP. Despite the fact that the exact methods are capable of finding the global solution, their applications are very limited in terms of problem size and computational time because the 3D-ODPP is NP-hard in the strong sense. In addition, constructive and meta-heuristic methods for solving the 3D-ODPP have not been discussed frequently in the literature and remain a gap in the state-of-the-art.
This paper proposes a genetic algorithm that deals with the 3D-ODPP. The genetic process is to find out the packing sequence and the orientation of products. To construct the solution, a new greedy-search product placement algorithm is developed. This placement algorithm is used to determine the position where each product is placed and to calculate the volume of the package. Literature instances are tested and the obtained solutions are compared with that given by existing exact methods. The experiments show that the proposed algorithm has the capacity of solving the 3D-ODPP in a reasonable time and gives competitive solutions compared with the benchmark methods, especially for problems with many products.

1 Introduction

The three-dimensional Open Dimension Packing problem (*3D-ODPP*), one of the Cutting and Packing problems according to the typology of [5], is a real-world driven optimization problem that aims at the minimization of package volume in right-size packaging systems. The problem can be found in many industrial scenarios, such as e-commerce secondary packaging. The objective of the *3D-ODPP* is to find out the length, width, and height of the cardboard box that can be used to pack a given set of small items (or products) so that the volume of the box is minimal.
Many literature researches, such as [2–4], have focused on exact methods to deal with the *3D-ODPP*. Despite the fact that the exact methods like that in [2–4] are capable of finding the global solution, their applications are very limited in terms of problem size and computational time because the *3D-ODPP* is NP-hard in the strong sense. In addition, constructive and meta-heuristic methods for solving the *3D-ODPP* have not been discussed frequently in the literature. Therefore, solving large-sized *3D-ODPP* is still a gap in the state-of-the-art.
This paper proposes a genetic algorithm called *"GA-ODP"* that deals with the *3D-ODPP*. The proposed method is inspired by the random-key biased genetic algorithm for solving 2D and 3D Bin Packing problem presented in [1]. The algorithm of [1] aims at minimizing the number of bins (container objects with fixed measurements) needed to pack a set of items. In order to construct a solution, three main decisions to be made are: the packing sequence indicating the order in which the items are packed; the orientation of each item; and the positions where the items are placed. [1] use a genetic algorithm to determine the orientation and the packing sequence of the items, then a constructive algorithm is applied to determine items' positions. The *GA-ODP* in this paper uses the same chromosome representation as that in [1]. However, the algorithm of [1] only deals with Bin Packing problems where bins' size is fixed, and the solutions are constructed based on the empty spaces rested inside the bins while for the *3D-ODPP*, all box measurements are variable, therefore, the encoding of item orientation, the placement strategy, and the fitness function of [1] cannot be used. This study proposes a new placement algorithm to construct the packing solutions.

The new placement algorithm is based on a greedy search algorithm that finds the local optimum for each item placement in a packing sequence. This placement algorithm allows constructing a solution for any packing order and item orientation given by the genetic algorithm. No chromosome repairing is needed in the evolutionary process.
Literature test instances with different problem sizes are tested and the obtained results are compared with that given by existing method [4]. The experiments show that the proposed algorithm has the capacity of solving the *3D-ODPP* and gives competitive solutions compared with state-of-the-art exact methods while the computational time is much shorter and therefore, the proposed method has the capacity of dealing with larger problems.

2 Problem formulation

The *3D-ODPP* addressed in this study is a problem arises in an e-commerce secondary packaging system where a set of cuboid-shaped items is to be packed into a single cardboard box before being shipped to client. Given n items of cuboid-shaped with their fixed length (p), width (q), and height (r). Every item can be rotated in six possible orthogonal orientations inside the box. Knowing that the capacity of the system is much higher than the items volume, which means any set of items can be packed by the packaging system. As all the measurements of the box are variable, the purpose is to determine the length (L_b), width (W_b), and height (H_b) of the minimal volume packing box. The mathematical model for solving the *3D-ODPP* is presented in [2]:
Parameters:

- n: number of items to be packed.
- p, q, r: the vectors indicating items' length, width and height, respectively.
- M: big number used in the model. $M = \left(\sum_{i=1}^{n} p_i\right)^3$

Variables:

- x_i, y_i, z_i ($i \in \{1...n\}$): Continuous variables indicating the coordinate of products.
- L_b, W_b, H_b: Continuous variables for length, width, height of the box, respectively.
- $o_{i,j}$ ($i \in \{1...n\}; j \in \{1...6\}$): Binary variables indicating weather the product i has orientation j. The orientations are defined as shown in Table 1.
- $a_{i,j}, b_{i,j}, c_{i,j}$ ($i, j \in \{1...n\}$): Binary variables indicating the "left-right", "front-behind", and "above-under" relative positions of products i and j. For example, if product i is on the left side of product j then $a_{2,3} = 1$, otherwise, $a_{2,3} = 0$. If there is at least one relative position between two product, then they are called "non-intersected".

Objective function:

$$Minimize \quad L_b \times W_b \times H_b \tag{1}$$

Subject to:

$$\sum_{j=1}^{6} o_{i,j} = 1 \quad \forall i \in \{1...n\} \tag{2}$$

$$a_{i,j} + a_{j,i} + b_{i,j} + b_{j,i} + c_{i,j} + c_{j,i} \geq 1 \quad \forall i, j \in \{1...n\}; i \neq j \tag{3}$$

$$x_i + p_i(o_{i,1} + o_{i,2}) + q_i(o_{i,3} + o_{i,4}) + r_i(o_{i,5} + o_{i,6}) \leq x_j + M(1 - a_{i,j}) \quad \forall i, j \in \{1...n\}; i \neq j \tag{4}$$

$$y_i + p_i(o_{i,3} + o_{i,5}) + q_i(o_{i,1} + o_{i,6}) + r_i(o_{i,2} + o_{i,4}) \leq y_j + M(1 - b_{i,j}) \quad \forall i, j \in \{1...n\}; i \neq j \tag{5}$$

$$z_i + p_i(o_{i,4} + o_{i,6}) + q_i(o_{i,2} + o_{i,5}) + r_i(o_{i,1} + o_{i,3}) \leq z_j + M(1 - c_{i,j}) \quad \forall i, j \in \{1...n\}; i \neq j \tag{6}$$

$$L_b \geq x_i + p_i(o_{i,1} + o_{i,2}) + q_i(o_{i,3} + o_{i,4}) + r_i(o_{i,5} + o_{i,6}) \quad \forall i \in \{1...n\} \tag{7}$$

$$W_b \geq y_i + p_i(o_{i,3} + o_{i,5}) + q_i(o_{i,1} + o_{i,6}) + r_i(o_{i,2} + o_{i,4}) \quad \forall i \in \{1...n\} \tag{8}$$

$$H_b \geq z_i + p_i(o_{i,4} + o_{i,6}) + q_i(o_{i,2} + o_{i,5}) + r_i(o_{i,1} + o_{i,3}) \quad \forall i \in \{1...n\} \tag{9}$$

$$\max_{i \in \{1...n\}} r_i \leq \phi \leq \sum_{i=1}^{n} p_i \qquad \forall \phi \in \{L_b, W_b, H_b\} \tag{10}$$

$$\sum_{i=1}^{n}(p_i \times q_i \times r_i) \leq L_b \times W_b \times H_b \leq \sum_{i=1}^{n} p_i \times \left(\max_{i\in\{1...n\}} q_i\right) \times \left(\max_{i\in\{1...n\}} r_i\right) \tag{11}$$

The constraint (2) assures that an item can only have at most one orientation. The constraints (3) to (6) define the relative positions of items. The constraints (7) to (9) make sure all products are entirely placed inside the box. Finally, the constraints (10) to (11) show the upper and lower bounds of box length, width, height, and volume.

[2–4] use the logarithm transformation and piecewise linearization technique to solve the model with a liner solver, e.g. CPLEX.

3 Genetic algorithm

As mentioned in section 1, the exact methods proposed by [2–4] are very limited by the problem size and require great computational power. Therefore, an heuristic approach would be necessary in many practical cases. This section describes how the genetic algorithm *GA-ODP* is applied to solve the *3D-ODPP*.

3.1 Solution encoding and decoding

In this genetic algorithm, an encoded solution (also known as a "chromosome") is an array made of $2n$ genes that contain the genetic information about items' packing sequence and orientation. The first n genes indicate the order in which the items are loaded into the box. Every genes of this part is a real number whose value is between 0 and 1. The second part of the chromosome includes n genes whose value is an entire number between 1 and 6 indicating the orientation of the items. The actual dimensions (l, w, h) of items among x, y, and z-axis corresponding to item orientations are as shown in Table 1.
Before constructing any solution, the chromosome must be decoded into the actual packing sequence and item orientation so that the constructing algorithm can turn them into item loading position and the box dimensions can be calculated. The decoded genes are represented as two following vectors:

- **The Vector of Loading Sequence *(VLS)*:** a vector of size $(1 \times n)$ that is made of n elements whose value is an entire number between 1 and n, without repeating, indicating the order in which the items are loaded into the box. The *VLS* is obtained by sorting the first n genes in the ascending order. In other words, the corresponding item of the gene with smaller value among the first n genes will be packed earlier.
- **The Vector of Item Orientation *(VIO)*:** a vector of size $(1 \times n)$ whose elements are entire numbers within $\{1, 2, \ldots, 6\}$. It indicates the exact orientation of n corresponding items. As mentioned in section 2, each item can have one out of the six possible orientations. This vector can be obtained by copying the second part of the encoded chromosome:
  $VIO_i = C_{n+i} \quad \forall i = 1...n.$

These vectors will be used as input arguments of the placement algorithm presented in section 3.2.

3.2 Solution construction

This section describes how to construct a solution from the decoded chromosomes and calculate the fitness of each solution. To construct a solution including items' orientation and loading position as well as the dimensions of the bounding box, the decoded chromosome of the solution will be used by a placement algorithm.

Placement algorithm: The placement algorithm is based on the greedy search algorithm. It places the items one after the other by the specific order defined by the *VLS*, until there is no item left. All the possible position, including x, y, and z-coordinates, where the items can be placed are precalculated and stored in three lists: Px, Py, and Pz, respectively. At the beginning, every list has only one element, which equals to 0. This means the first item will always be placed at the origin of the coordinate system. Every time a new item is placed, at most one coordinate will

be added into the each list. The new coordinates allow the following items can be placed next to the item that has just been placed. The placement algorithm will check all the combinations of x, y, and z-coordinates and find out the position for each item so that the increase of box volume caused by it's placement is minimal. Once all items are loaded, the dimensions of the bounding box will be calculated. This placement algorithm assures the feasibility of all solutions. The placement algorithm is shown in **Algorithm 1**.

Fitness function: The quality of a solution is considered as the volume of the bounding box given by the placement algorithm. As the box's length, width, and height are L_b, W_b, and H_b, the fitness function is:

$$f = L_b \times W_b \times H_b \tag{12}$$

3.3 Evolutionary process

A population is a set of individuals (chromosomes) created by a generator or by the evolutionary process. At the beginning, a set of n_p chromosomes are randomly generated to made up the initial population. The *GA-ODP* evolves the initial population through n_g generations to improve the solution and find out the best packing sequence as well as items' orientation for the given items. For each generation, the fitness value of every individual is computed, then the individuals are classified into two groups: the first group contains n_e elite individuals with highest fitness. The second group contains n_r "regular" individuals, which are the rest of the population. It is clear that $n_e + n_r = n_p$. To create the population of the next generation, new individuals are generated by the following operators:

- Copying: the chromosome of all the elite individuals are copied directly to the offspring's chromosome of the next generation without modification in the genes.
- Crossover: two individuals are selected from the population of the current generation. Their genes are mixed up by a specific operator to create the chromosome of a new offspring individual. Then the new individual is added to the population of the next generation.
- Mutation: Some individuals from the current population are selected to copy to the next generation but their genes will contain some random modifications created by the mutation operator.

Let n_c be the number of individuals created by the crossover operator, and n_m be the number of mutants, then, $n_r = n_c + n_m$. In other words, for any new generation, the population always has n_e individuals copied from the previous generation and n_r new individuals generated by the crossover and mutation operator. During both operators, there is a small possibility that some mutations appear on some genes and make the offspring's chromosome a little bit different before it is added to the population. Figure 3 illustrates the operations of generating a new population. The crossover and mutation operators are described as follows:

Copying: From the current population, the chromosome of all individuals with highest fitness will be copied to the offspring's chromosome of the population of the next generation. In other words, all elites of the current population will be copied to the next generation without modification.

Crossover: The *GA-ODP* uses a random-key crossover operator to generate new individuals. For each pair of parent individuals, a crossover vector of $2n$ random real numbers within $[0, 1]$ is generated. A given probability of crossover P_c is also used for this operator. For each number in the crossover vector, if the number is greater than P_c then the gene at the corresponding position of the first parent is copied to the offspring's chromosome. Otherwise, the gene from the second parent is copied to the offspring's chromosome. Figure 2 shows an example of crossover operator.

As mentioned in section 3.2, the placement algorithm guarantees that any chromosome can be decoded into a solution with no item intersection, therefore, no reparation process is needed for new individuals.

Mutation: The mutation operator is necessary to increase the diversity of the genetic pool and avoid premature convergence of the population. Let P_m be the probability of mutation, a mutation vector is created in the same way at the crossover vector above. If there is at least one element of the mutation vector is less than P_m, then there will be a mutation occurred on the gene at the corresponding position of the chromosome. Then the gene will be replaced by a new random gene whose value is between $[0, 1]$ if it is a packing sequence gene, and in $\{1, \ldots, 6\}$ if it is an item orientation gene.

Algorithm 1: Placement algorithm

Input: p, q, r, VLS, VIO
Output: L_b, W_b, H_b, x, y, z
Result: Items placement position and box dimensions
n = length of (p)
Calculate (l, w, h) corresponding to items' orientation VIO
Sort the items by order of VLS
$Px = Py = Pz = \{0\}$
Initialize the set of placed items: $B = \{\}$
for $i \in \{1...n\}$ **do**
  $V_{min} = M$
  for $x^* \in Px$ **do**
    for $y^* \in Py$ **do**
      for $z^* \in Pz$ **do**
        Calculate V^* when item i is placed at (x^*, y^*, z^*)
        if *($V^* \leq V_{min}$) & (No intersection between item i and any item in B)* **then**
          $V_{min} = V^*$
          $x_i = x^*$
          $y_i = y^*$
          $z_i = z^*$
  Add item i to B
  if $(x_i + l_i) \notin Px$ **then**
    Add $(x_i + l_i)$ to Px
  if $(y_i + w_i) \notin Py$ **then**
    Add $(y_i + w_i)$ to Py
  if $(z_i + h_i) \notin Pz$ **then**
    Add $(z_i + h_i)$ to Pz
Calculate L_b, W_b, and H_b

4 Computational experiments

To see the efficiency of the proposed algorithm, test instances derived from [4] are tested. The benchmark method is the mathematical model proposed in [4] resolved with CPLEX. The algorithms are programmed in C++ and the experiments are executed on an Intel Core i7-6820HQ CPU @2.70 GHz, Windows 7 PC with 32 GB of RAM.

For the *GA-ODP*, different combinations of parameters are tested to find out the optimal configuration for the genetic algorithm and reduce the computational time. n_g is selected from $\{50, 100, 200, 300\}$, $n_e \in \{0.1n_p, 0.2n_p, 0.3n_p\}$, $n_m \in \{0.1n_p, 0.2n_p, 0.3n_p\}$. As shown in [1], the population size n_p can significantly affect the calculation, and good results are obtained by indexing the population size into the problem size (or number of items). Therefore, in these experiments, the population size is also a function of problem size, the following values are tested to choose the best parameter: $n_p \in \{20n, 30n, 50n, 100n\}$. By testing all the combinations created by the parameters, the following configuration is chosen to be constant parameters for solving the test instances: $n_g = 200$, $n_p = 20n$, $n_e = 0.3n_p$, $n_m = 0.1n_p$, $P_c = 0.5$, and $P_m = 0.1$.

Table 2 shows the test results and the computational time for ten test instances derived from [4]. The row with header *"Solver"* shows the solutions given by the benchmark method, while the row

with header *"GA-ODP"* shows the experimental results given by the proposed method. It can be seen that the solutions given by the *GA-ODP* is competitive with that given by the exact method. There are six out of ten test instances where the difference of box volume of both method are the same. For the other instances, the difference is not enormous, which is from about 1.2% to 3.8%. In terms of computational time, for the small problem (1, 2, 7, and 8) where number of items is not greater than 5, the solver finds the solution faster than the *GA-ODP* (the Gap_t is negative). However, when there are more items, the solver needs much more time to find out the solution, and the computational time increases quickly among the number of items. In the other hand, the computational time of the *GA-ODP* does not increase in the same function as that of the solver. In that way, the *GA-ODP* can solve the problems with more items in a reasonable time.

The improvement of the *GA-ODP* over the benchmark method is calculated as follows:

$$GAP_V = (V_2 - V_1)/V_1 \tag{13}$$

$$GAP_t = (t_1 - t_2)/t_2 \tag{14}$$

Where V_1 and t_1 are box volume and computational time of the benchmark method [4], while V_2, t_2 are these values given by the proposed method.

The test results in Table 2 show that the proposed algorithm has the capacity of giving competitive solutions compared with exact method in the literature. In the other hand, the proposed algorithm can deal with larger problem while its computational time is not exploding among with the problem size.

The computational experiments also show that for most of the case, minimizing the adjusted box volume can lead to the minimal actual box volume. In terms of computational times, the proposed algorithm is not exploding while number of items increases.

5 Conclusion

This paper has proposed a new genetic algorithm to deal with the 3D-ODPP. The proposed algorithm has shown the capability of solving the 3D-ODPP in a reasonable computational time while the given solutions are competitive with those given by exact methods in the literature. The proposed method has significant advantage in terms of computational time when solving problems with more items to be packed. However, this work has not consider many practical constraints that often arise in real-world scenarios, such as item supporting, package balancing, weight distribution, etc. Therefore, solving the 3D-ODPP by a genetic algorithm with the consideration of practical constraints will be an interesting subject for future researches.

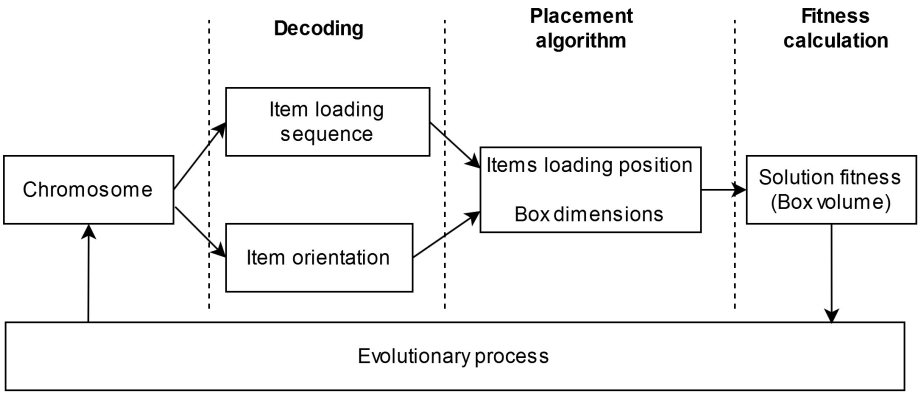


Fig. 1: Genetic algorithm

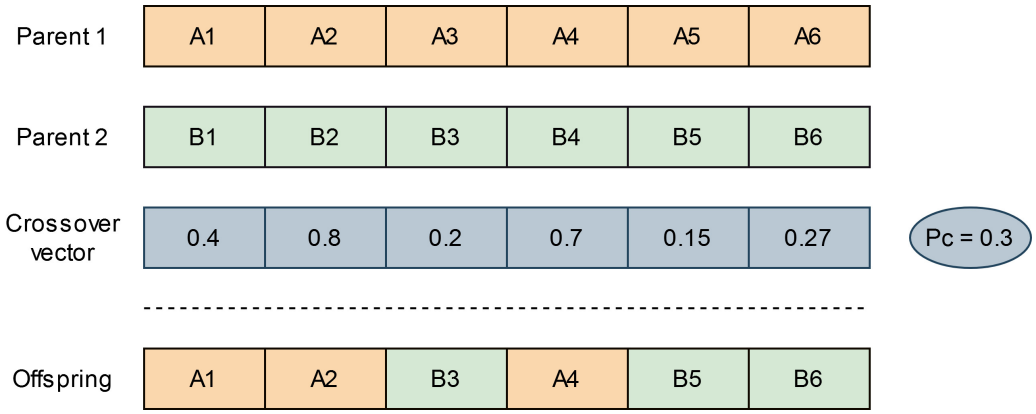


Fig. 2: Crossover operator

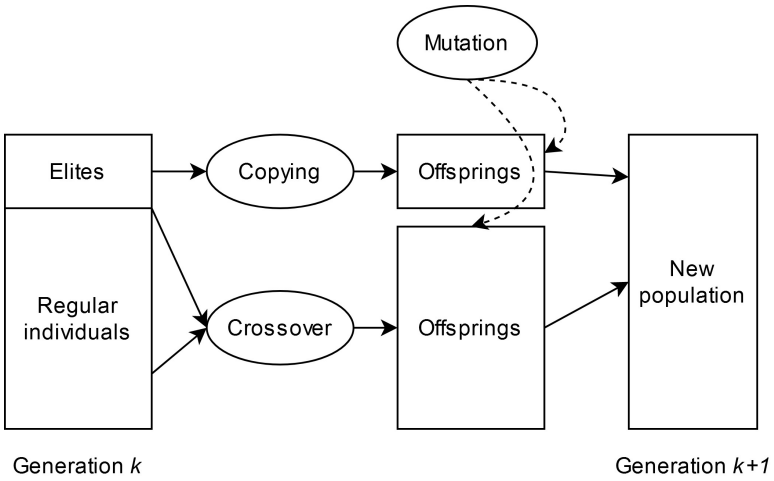


Fig. 3: Evolutionary process

Table 1: Item orientations.

Orientation	1	2	3	4	5	6
Among x-axis (l)	p	p	q	q	r	r
Among y-axis (w)	q	r	p	r	p	q
Among z-axis (h)	r	q	r	p	q	p

Table 2: Computational results

Problem		1	2	3	4	5	6	7	8	9	10
# items		4	5	6	7	8	9	4	5	6	7
Solver	L	28	30	35	43	9	10	127	102	92	101
	W	26	28	28	28	8	8	57	95	81	89
	H	6	6	6	6	5	6	30	30	50	51
	V	4368	5040	5880	7224	360	480	217170	290700	372600	458439
	t(s)	0.5	1.9	4.2	32.1	1.6	3.4	0.3	2.2	22.1	215.2
GA-ODP	L	28	30	31	25	9	10	127	102	90	106
	W	26	28	16	24	8	8	57	95	85	88
	H	6	6	12	12	5	6	30	30	50	50
	V	4368	5040	5952	7200	360	480	217170	290700	382500	466400
	t(s)	1.84	2.8	2.88	4.08	0.8	1.6	1.68	2.56	3.12	3.76
	Gap_V	0	0	0.012	0.023	0	0	0	0	0.027	0.038
	Gap_t	-0.73	-0.32	0.46	6.87	1.0	1.13	-0.82	-0.14	6.08	56.23

References

1. Gonalves, J.F., Resende, M.G.C.: A biaised random key genetic algorithm for 2D and 3D bin packing problem. Int. J. Production Economics 145 (2013), 500-510.
2. Truong, C.T.T.: A product arrangement optimization method to reduce packaging environmental impacts, in Conference Series: Earth and Environmental Science (2020).
3. Truong, C.T.T.: A Mathematical Model for Three-Dimensional Open Dimension Packing Problem with Product Stability Constraints. Optimization and Learning, Springer International Publishing, 2020, pp. 241-254.
4. Tsai, J.F., Wang, P.C., Lin, M.H.: A global optimization approach for solving three-dimensional open dimension rectangular packing problems, Optimization (2015), 64:12, 2601-2618
5. Wascher, Z., HauBner, B., Schumann, H.: A typology for Cutting and Packing problem. European Journal of Operational Research 183 (2007), 1109-1130.

Neighborhood Enumeration in Local Search Metaheuristics

Michiel Van Lancker[1 [0000−0002−2417−9928]], Greet Vanden Berghe[1 [0000−0002−0275−5568]], and Tony Wauters[1 [0000−0002−1014−6340]]

KU Leuven, Department of Computer Science, CODeS, Belgium
michiel.vanlancker@cs.kuleuven.be

Abstract. Neighborhood enumeration is a fundamental concept in the design of local search-based metaheuristics. It is often the only principle of intensification present in a metaheuristic and serves as the basis for various metaheuristics. Given its importance, it is surprising that academic reporting on enumeration strategies lacks the necessary information to enable reproducible algorithms. One aspect of neighborhood enumeration in particular has been under the radar of researchers: the order in which neighbors are enumerated. In this paper, we introduce a versatile formalism for neighborhoods which makes explicit enumeration order and we analyse the impact of enumeration order on the outcome of search procedures with a small set of benchmark problems.

Keywords: Enumeration Order · Local Search · Neighborhoods · Metaheuristics

1 Introduction

Metaheuristics have gained a somewhat ambiguous reputation over the years. On the one hand they are lauded for their useful characteristics in practical applications: metaheuristics are problem-independent, general optimization algorithms. They are not only capable of being reused over a wide variety of problems, but many are also anytime algorithms which maintain a valid solution throughout the entire search process. Furthermore, they can be implemented in a highly configurable fashion, enabling automated algorithm design and parameter tuning. This results in algorithm templates that can be instantiated and automatically tailored to solve specific problems or instances. On the other hand, metaheuristics research has not yet reached the scientific rigor found in other fields, with many researchers tending to focus on algorithmic efficiency – or worse, novelty – rather than algorithmic understanding. This has led to a large variety of algorithms which differ only slightly from one another or are identical except for the terminology used [6].

While big steps have been made – especially during the last two decades – to transform the field into a more academic one with rigorous scientific discipline built on formalized concepts, many publications continue to operate in the sphere of problem-solving rather than algorithmic understanding. This resulted

in many metaheuristics, but few insights. Nevertheless, efforts are underway to mature the discipline. Notable examples of this are (i) the endorsement by the Journal of Heuristics of the view that nature should no longer serve as an explicit inspiration for "novel" metaheuristics, (ii) the recognition of the need for white-box algorithm implementations, preferably described in a purely functional style [8], (iii) the call for rigorous evaluation and testing practices, and (iv) the active promotion of a view of what metaheuristics research ought to be [7].

In this paper we zoom in on one specific component of metaheuristics: the concept of local search neighborhoods. We argue that a gap exists between common theoretical neighborhood definitions and how they are implemented in practice. In other words: we argue that neighborhoods are not implemented according to the white-box principle, preventing algorithm reproducibility and standardized evaluation.

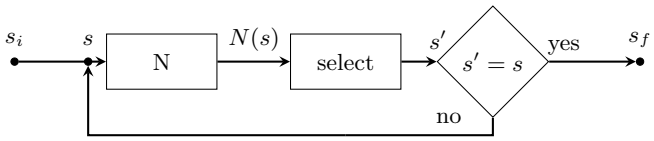


Fig. 1: Iterative improvement consists of repeatedly applying an improving operation to the solution.

Many optimization techniques can be considered instantiations of the iterative improvement-scheme (II-scheme), the distinction between which results from the interaction between their instantiating components. The II-scheme itself is straightforward: starting from an incumbent solution the search process consists of a series of iterations, where in each iteration a selection function selects an alternative solution of better quality than the incumbent solution. If a better solution is found, it replaces the incumbent solution. This process is repeated until no improving solution can be found.

In local search metaheuristics, a set of alternative solutions – called the neighborhood of the incumbent solution – is constructed by making a set of small modifications to the incumbent solution. Most, if not all, metaheuristics can be mapped to the II-scheme shown in Fig. 1. The difference between various metaheuristics yet again results from the differing interactions between their constituent components. Which components to consider and how to combine them is the responsibility of the (human) algorithm designer. Design choices which require some thought include how to generate a neighborhood of the incumbent solution, which solution to select from the neighborhood and how to compare solutions. Good neighborhood design is important when it comes to the efficiency of a local search (meta)heuristic. Choosing an appropriate selection criterion is equally important, as it strongly determines the behavior of the search and can have a dramatic impact on runtime.

Given the importance of these two design questions, it is fair to assume that reporting on metaheuristic algorithms should include complete information concerning which choices were made and, ideally, why. However, at present the opposite situation is the case: many publications concerning metaheuristics do not report neighborhood specifications to the level of detail required to facilitate reproducibility. Most obvious is the lack of information concerning how operators in a neighborhood are enumerated. This information is crucial if an order-dependent selection criterion is used and, indeed, virtually all deterministic selection criteria are order-dependent. A second, more subtle issue is the lack of information concerning which operators are a priori included in a neighborhood.

Our contributions in this paper are threefold. First, we introduce a formalism for the concept of a local search neighborhood, which makes explicit the enumeration order. Second, we analyze the effect of enumeration order on the outcome of a search procedure through a series of computational experiments. Third and finally, we provide several examples of the expressiveness of the proposed formalism.

The remainder of this paper is structured as follows. Beginning with the concept of iterative improvement, Section 2 introduces neighborhoods and selection methods and provides a brief overview of how neighborhood enumeration is commonly reported in metaheuristics research. In Section 3 we introduce a formalism for neighborhood enumeration. Section 4 then analyzes the effect of enumeration order on the outcome of a search procedure on a set of benchmark instances. Several examples demonstrating the flexibility of the formalism are given in Section 5. Section 6 then concludes the paper.

2 Iterative improvement, neighborhoods & selection

In this section we review the relationship between iterative improvement, neighborhoods and selection criteria. In doing so we identify a gap between the commonly used definitions for the aforementioned concepts and the components required to implement the II-scheme, resulting in an incomplete algorithm specification. The section ends with a brief analysis of how neighborhood enumeration is currently reported on in the academic literature.

To approach local search metaheuristics as instantiations of iterative improvement, strict definitions are required for the instantiating components. Consider the II-scheme shown in Fig. 1. It is clear that an instantiation of the scheme is determined by three factors, namely: a neighborhood generation function N, a neighbor selection function *select* and a condition to test whether or not the search has ended. Since we are only interested in improvement methods, the ending condition can be excluded from the analysis and thus the behavior of a deterministic II-procedure is dependent on only two functions: the neighborhood function $N : S \to \mathcal{P}(S)$ and the selection function $select : \mathcal{P}(S) \to S$. As is clear from its type, the neighborhood-function must map the incumbent solution to a set of alternative solutions, resulting in the common introductory definition of a neighborhood [2,9]:

Definition 1. *A neighborhood function is a mapping* $N : S \to \mathcal{P}(S)$ *which assigns to each solution* $s \in S$ *a set of solutions* $N(s) \subseteq S$*. The members of* $N(s)$ *are called neighbors of* s*.*

In the context of local search however, a different definition is sometimes used to more adequately capture the notion of operators and locality. A neighborhood is defined in terms of a relation – the local search operator – on S:

Definition 2. *The R-neighborhood* $N_R(s)$ *of solution* $s \in S$ *is the neighborhood defined by the relation* R *on* S*,* $N_R(s) = \{s' \in S : sRs'\}$*.*

The second component of the II-scheme is a selection function, which returns a single neighbor from the neighborhood it receives as input. We refrain from giving a general definition of selection criteria, but note that any selection criterion must be a function of type $select : \mathcal{P}(S) \to S$ and we shall examine how well two of the most popular selection criteria adhere to this definition.

The first criterion we will consider is the $argmin$ selection criterion (Eq. 1), which selects the best solution from the neighborhood. Next is the $firstmin$ selection criterion (Eq. 2), which selects the first improving solution from the neighborhood. More formal definitions of both criteria are as follows:

$$\underset{s' \in N(s)}{\operatorname{argmin}}\, c(s') := \{s' \mid \forall s'' \in N(s) : c(s') \leq c(s'')\} \tag{1}$$

$$\underset{s_i \in N(s)}{\operatorname{firstmin}}\, c(s_i) := \{s_i \in N_{\downarrow}(s) \mid \forall s_j \in N_{\downarrow}(s) : i \leq j\} \tag{2}$$

$$\text{where } N_{\downarrow}(s) := \{s' \in N(s) \mid c(s') \leq c(s)\}$$

Note that this definition of $argmin$ denotes a set of solutions instead of a single solution: if multiple solutions have the best objective value, all of these solutions will be returned. As such the definition specifies a function of type $\mathcal{P}(S) \to \mathcal{P}(S)$ and a modification, a tie-breaker, is needed to acquire the required type. Common tie-breakers are to select the first, the last or a random solution from the set of most improving solutions. Only the first two of these tie-breakers are deterministic and both of these are order-dependent.

For $firstmin$, the impact of order is obvious. To be able to return the first improving neighbor an order must be imposed on neighborhood N. In the worst case all solutions in the neighborhood are improving and thus each possible ordering of N will return a different solution. It follows that the neighborhood enumeration order must be known to achieve a full specification of a single iteration in the II-scheme. While the effect of enumeration order on the outcome of a single iteration is generally fairly limited, this is less so when considering the entire II-scheme. Since every iteration starts from the outcome of the previous iteration, the effect of an enumeration order compounds throughout the whole search.

Given the effect of enumeration order on the outcome of a search procedure, it is somewhat surprising that most publications do not contain any information

about it. Many publications only describe neighborhoods in terms of their local search operator. A notable exception is [5], which does not only mention the use of a random enumeration order, but also publishes the complete source code of its implementation.

Finally, let us examine some open-source implementations of metaheuristics and see how they implement neighborhood enumeration. The following two implementations serve as an example: the Java Metaheuristics Search Framework(JAMES) [4] and the suite of metaheuristic frameworks PARADISEO [3]. In JAMES it is possible for users to implement custom neighborhoods through a neighborhood- and operator-interface, but imposing orders on neighborhood sets through an interface is not possible and must be programmed from scratch by the user. When querying the full neighborhood, an eagerly constructed list of operators is returned. In PARADISEO, users can implement custom neighborhoods in a similar fashion, though here order *is* made explicit by means of an iterator-interface. Querying the full neighborhood returns a lazy iterator over the neighborhood. Furthermore, neighborhoods can be linked together into new neighborhoods.

Before continuing with the next section, we end this section with an example of issues arising when neighborhood definitions are incomplete. We will illustrate these issues by considering the $TwoOpt$-operator for the Traveling Salesperson Problem (TSP). Let $C = \{c_1, \ldots, c_n\}$ be a set of points on the Euclidean plane representing cities and let $d : C \times C \to \mathbb{N}$ be the distance between two cities. Then, the goal of the TSP is to find the shortest tour which visits each city once. Let permutation $\pi \in \Pi$ represent a tour through all cities in C and let $I_\pi = \{1, \ldots, n\}$ be the index set of π. Element $\pi_i \in \pi$, where $i \in I_\pi$, represents the i^{th} visited city in the tour. The objective value $c(\pi)$ is computed with (Eq. 3).

$$c(\pi) = \sum_{i \in I_\pi \setminus \{n\}} d(\pi_i, \pi_{i+1}) + d(\pi_n, \pi_1) \tag{3}$$

Applying the $TwoOpt$-operator to a solution for the TSP equals swapping two edges in the tour, or equivalently, inverting a subsequence of the solution representation π. The operator takes as input the current tour and two indices $i, j \in I_\pi$. To implement a function to generate the $TwoOpt$ neighborhood, a double for loop is typically used. A naive implementation would generate neighbors for all possible pairs $(i, j) \in I_\pi^2$. This is however redundant: $TwoOpt$ is a symmetric operator, thus a more efficient implementation would only generate neighbors for the pairs (i, j) for which $i < j$, as these are sufficient to cover the whole neighborhood. Aside from redundancy, which is unwanted but not problematic, if it is unclear which moves are included in the neighborhood and which are not, any order-dependent selection function can cause diverging search outcomes for two neighborhoods that "look" the same.

3 Neighborhood Enumeration

The previous section provided an introduction to how common definitions of neighborhoods, selection criteria and local optima are not sufficiently exact from an implementation perspective and how this in turn results in an incomplete algorithm specification. As suggested by the *TwoOpt*-example, there are two pieces of information missing from Definition 2: how many (i.e. which) solutions belong to a neighborhood and the order in which these solutions are visited. In this section we present an alternative definition of a neighborhood function, which makes the aforementioned information concrete. The purpose of the definition being introduced is to capture the structure of a local search neighborhood in such a way that the required implementation steps become clear.

Consider the neighborhood $N_M(s) \subseteq S$. For all $s_\phi \in N_M(s)$ we know that we can move from s to s_ϕ. Let $m_\phi : s \mapsto s_\phi$ be the function representing the move from s to s_i. There are $|N_M(s)|$ such functions, one for each $s_\phi \in N_M(s)$. Thus we can define the neighborhood as $N_M(s) = \{m_i(s)\}_{i \in \Phi}$, where Φ is an index over $N_M(s)$. Note that if we provide a constructor function $M : \Phi \to (S \to S)$, we can construct function $m_\phi : S \to S$ by evaluating $M(\phi)$. Given an iterator T over Φ, the first neighbor in the neighborhood can be generated as follows: take the first element ϕ from the iterator, call constructor M to construct move m_ϕ, and apply $m_\phi(s)$. To generate subsequent neighbors, take the next element from T and repeat the process until all elements from T have been consumed. The neighborhood can then be defined as:

Definition 3. *A neighborhood $N_M(s, T)$ is the set of solutions constructed by applying each function $m_\phi : S \to S$ for each $\phi \in T$ to s, where T is an iterator over (a subset of) Φ_M, the parameter space of operator $M : \Phi_M \to (S \to S)$. As T is ordered, a neighborhood enumeration is uniquely defined by the triple (s, M, T).*

This definition results in several extra design questions concerning the parameter space used in a neighborhood. While neighborhood design typically only considers the choice of operator, now two more design choices must be made: *which operator parameters should be included in a neighborhood* and *in what order they should be generated*. In the next two sections we take a more detailed look at what options are available regarding these choices.

3.1 Parameter spaces

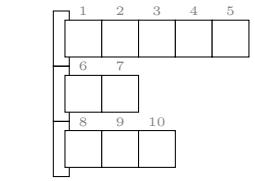
When considering operators, we make three observations: First, the parameter space Φ_M of operator M is dependent on the solution representation. Second, it is dependent on functional properties of its operator. Third, any subset of the parameter space can be used to generate a neighborhood.

Consider the TSP and three operators defined in Table 1. All three operators are quadratic and, since solution representation π is unconstrained, each operator can take any $(i, j) \in I_\pi^2$ as input, where I_π^2 is the Cartesian product of I_π.

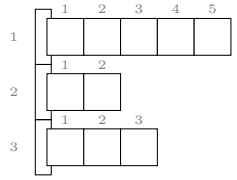
However, depending on the operator, we can eliminate some elements from I_π^2. For example, we know that the *Swap*- and *TwoOpt*-operators are symmetric operators and thus parameter combinations (i, j) and (j, i) will construct the same moves. Furthermore, for all three operators it is the case that no matter the state of the incumbent solution, parameter (i, i) will construct the identity move.

Operator	**Parameter space**	**Neighbor Relation**
Swap	$(i,j) \in I_\pi^2 : i < j$	$\pi'_i = \pi_j \wedge \pi'_j = \pi_i$
TwoOpt	$(i,j) \in I_\pi^2 : i < j$	$\forall k \in [0, j-i] : \pi'_{i+k} = \pi_{j-k}$
Shift	$(i,j) \in I_\pi^2 : i \neq j$	$\pi'_j = \pi_i$ $\pi'_k = \begin{cases} \forall k \in [i+1, j] : \pi_{k-1}, & \text{if } i < j \\ \forall k \in [j, i-1] : \pi_{k+1} & \text{otherwise} \end{cases}$

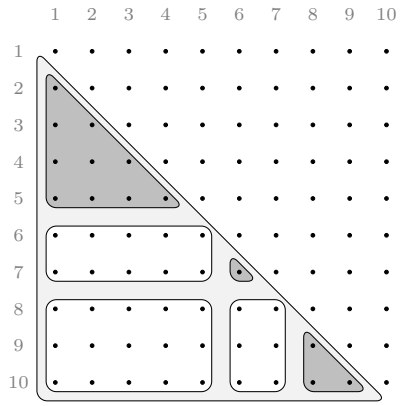
Table 1: Definitions of the Swap, 2opt and Shift operators and their respective parameter spaces.



(a) Single-level indexing



(b) Multi-level indexing



(c) Parameter space

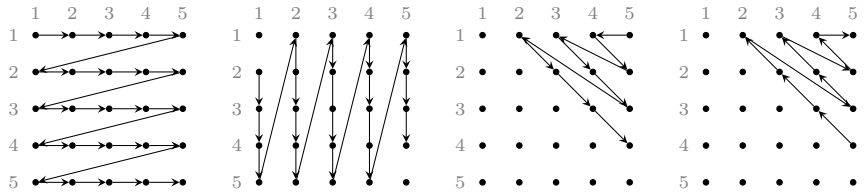
Fig. 2: The interpretation of operators and their respective parameter spaces is dependent on the indexing system used.

The importance of the chosen solution representation and index set becomes obvious when we consider more complex solution representations. Instead of permutation π, consider an ordered set of permutations ρ. To implement a neigh-

borhood for this structure, we require an index set to base our parameter space on. Looking at Fig. 2 it is clear that multiple options are available. We can use a single-level, linear index – like we did for permutation π – where every position in the representation is represented by a single integer: its position in the overall element order. Alternatively, a multi-level index can be used, where every position in the representation is represented by two integers: the position of the permutation in the set and the position within the permutation. Fig. 2c illustrates the correspondence between the parameter spaces of a symmetric operator using single-level and multi-level indexing. In light grey is the parameter space based on the single-level index. In dark grey are parameters corresponding to moves that operate inside a permutation of the set of permutations, using the multi-level index. Similarly, in white are the parameters corresponding to inter-permutation moves when using the multi-level index.

3.2 Enumeration order

The final step is to impose an order on the defined parameters. Given a set of parameters of size n, there are $n!$ ways to impose an order. However, some of these orders are more interesting than others. Of special interest are those that follow particular patterns, which can usually be efficiently implemented as an iterator which generates the parameter sequence lazily. Some of these patterned sequences can be interpreted as prioritizing certain moves: consider the $TwoOpt$ operator for the TSP and assume that we are using the $firstmin$ selection function. If $TwoOpt$ moves are enumerated according to the scheme $(1,2),(1,3),(1,4),\ldots$, the beginning position of the subsequence is considered more important than that of the end. Similarly scheme $(2,1),(3,1),(4,1),...$ deems the end position more important. Finally, scheme $(1,2),(2,3),(3,4),\ldots$ prioritizes moves corresponding to shorter subsequence inversions. Such semantic distinctions can help algorithm designers gain insights into the behavior and performance of their algorithms.



(a) I_π^2 (b) $I_\pi^2 : i \neq j$ (c) $I_\pi^2 : i < j$ (d) $I_\pi^2 : i < j$

Fig. 3: Various iterators over I_π^2.

Four iterators for quadratic operators are shown in Fig. 3 which differ in terms of their parameters included, order and direction. Fig. 3a illustrates an

iterator over the full parameter space – the Cartesian product I_π^2 – ordered along the rows. Fig. 3b is ordered along the columns and eliminates parameters $(i, i) \in I_\pi^2$. Figures 3c and 3d are both ordered along the diagonals and eliminate parameters $(i, j) \in I_\pi^2$ for which $i \geq j$, but they differ in the direction they take.

4 Experimental Evaluation

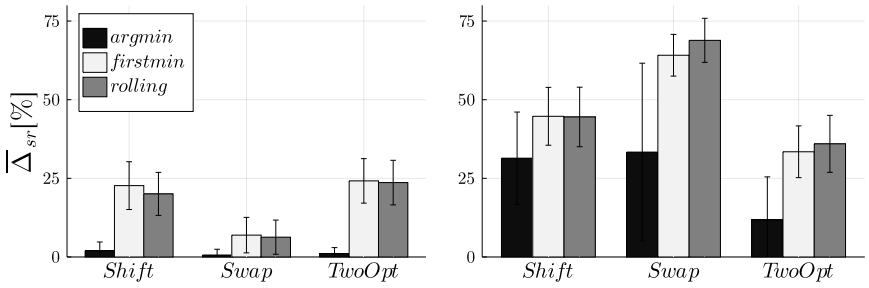
To evaluate the influence of enumeration order on search procedures we consider a search procedure to be a program of type $solve : S \to S$. This program takes an initial solution s_i and returns a local optimum as final solution s_f. We refer to the change induced on s_i by $solve$ as $\Delta_s = |C| - |e_c| - 1$, where $|C|$ is the number of cities and $|e_c|$ is the number of edges s_i and s_f have in common. In a similar fashion, we refer to the difference between the objective value of s_i and s_f as $\Delta_v = c(s_f) - c(s_i)$ and its runtime as Δ_t.

Constructive	**Select**	**Operator**	***Order***	***Direction***
random	argmin	Swap	Column	Forward
greedy	firstmin	TwoOpt	Row	Reverse
	rolling	Shift	Diagonal	

Table 2: The set of algorithm design parameters considered when experimentally evaluating enumeration order.

To study the impact of enumeration order on the search we compare Δ_s, Δ_v and Δ_t for *solve* procedures instantiated with different design parameters. Table 2 lists these design parameters. As the first three columns do no influence enumeration order, they can be considered design parameters resulting in different "contexts" in which the effect of enumeration order is evaluated. These parameters serve to broaden the scope of our analysis. All of the included design parameters have been defined in earlier sections of this paper, except for the selection function *rolling*. This selection function is an adapted version of $firstmin$. Whereas $firstmin$ begins from scratch in the next iteration after selecting the first improving neighbor $s_i = m_i(s)$, *rolling* will continue enumerating from its current position. The last two columns determine enumeration order. Three different iterators are used as parameter **Order**, each of which can be used in two **Directions**, resulting in six enumeration order. Every configuration is tested on 42 TSP instances from TSPLIB. All algorithms and experiments are implemented in the Julia programming language for technical computing [1] and run in a single-core-per-run configuration on an Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz machine with 16 cores. A complete description of the experimental setup and data is available online[1].

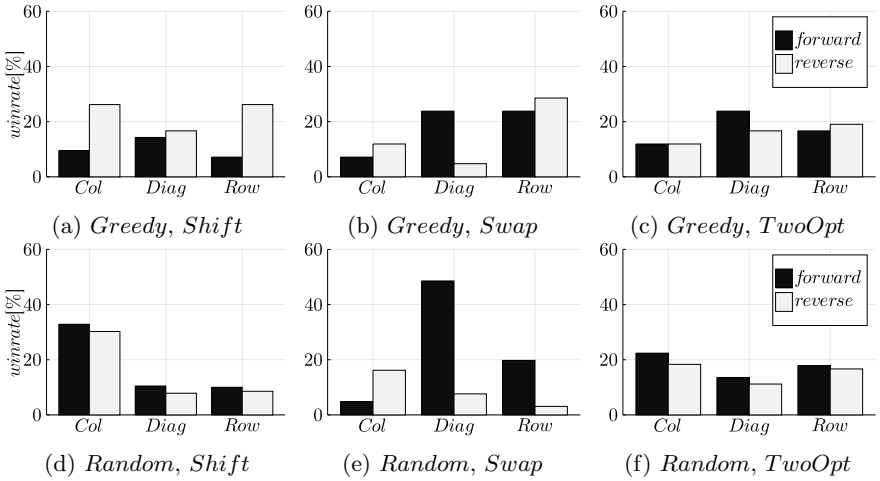
[1] `github.com/Michiel-VL/Neighborhood_Enumeration_Data`



(a) Greedy (b) Random

Fig. 4: The relative difference between the final solutions should be zero if order had no influence.

First, we examine the effect of enumeration order on the solution state. If no such effect were to exist, then the final solutions of the six runs for a given context and instance should be identical, independent of the enumeration order parameters. To measure if there is an effect of enumeration order on the solution state, we compute the mean relative pairwise distance $\overline{\Delta}_{sr}$ between the set of final solutions of a given context and instance. Fig. 4 is given for each of the 18 contexts. It is clear that the enumeration order does have an influence on the search outcome. Even for *argmin* selection, which is just barely order-dependent, $\overline{\Delta}_{sr}$ is fairly large, suggesting that the effect compounds quickly over the iterations of a search procedure.



(a) *Greedy*, *Shift* (b) *Greedy*, *Swap* (c) *Greedy*, *TwoOpt*

(d) *Random*, *Shift* (e) *Random*, *Swap* (f) *Random*, *TwoOpt*

Fig. 5: The numbers of wins for different orders and operators.

Fig. 5 shows the relative number of wins per enumeration order for different constructive heuristics and local search operators. While enumeration order does seem to affect the winrate, the results are inconclusive as to which order should be preferred for a given operator or constructive heuristic.

5 Discussion

Modeling a local search neighborhood as the combination of an operator with its own parameter space and an iterator over this parameter space has several advantages. First, it renders explicit the enumeration order to explore the neighborhood, which we have shown has an impact on the search outcome. Furthermore it is modular, as operator, set of parameters and order are completely separable implementation-wise. This not only enables easy reuse of code but it is also expressive, offering a range of neighborhood structures at virtually no cost.

It is also possible to encode structural properties of the problem in the neighborhood. As shown in Section 3, parameter spaces based on structured index sets can be used to distinguish between different parts of a solution representation. By opening up a neighborhood's structure through its parameter space, it is possible to use a wide variety of known algorithms to construct parameter spaces and reuse these over various neighborhoods.

Given a set of neighborhood definitions, new neighborhoods can be constructed in an algorithmic manner. Using function composition, operators can be composed into new operators and through the Cartesian product and disjoint union, various enumeration structures are available. Furthermore, given that in many programming languages iterators are denoted by a data structure that is composable in various ways – like filtering, linking or zipping – the definition as a whole is very expressive and enables concise descriptions of algorithms like Variable Neighborhood Descent and concepts such as path relinking or higher-order neighborhoods.

Note that defining a neighborhood as a triple $(s, M, T(\Phi_M))$ replaces the nested for-loops in many neighborhood implementations with a single foreach-loop. This triple separates three different neighborhood-design concerns that are typically entangled in code: local-search operators, neighborhood size and enumeration order. This enables algorithm designers not only to reuse operator, parameter space and enumeration order implementations for multiple neighborhoods, but it also leads to a more descriptive way of handling neighborhoods, enabling swift development and automated algorithm configuration.

6 Conclusion

In this paper we introduced a novel definition for neighborhoods aimed at formalizing their implementation. Defining local search neighborhoods in terms of a parametrized local search operator and an iterator over the parameter space of the operator leads to an expressive, composable definition which can be readily used during implementation. The iterator makes explicit two algorithm design

considerations that are typically overlooked: in what order should neighbors be generated and which neighbors should be included in a neighborhood. Furthermore, by basing the operator parameter spaces on the indexing mechanism of a solution representation, significant parts of neighborhood design can be automatically derived from a solution representation. Finally, many enumeration orders can be efficiently implemented as a lazy sequence and therefore neighborhoods can be generated lazily.

While we only considered unconstrained problem representations, it would be interesting to look at constrained problems to examine how particular types of constraints affect the use of the definition, as complex constraints could prevent efficient iterator implementations. Though interesting, this primarily concerns implementation efficiency rather than formalization and thus lay outside the scope of this paper.

Acknowledgements

Research supported by Data-driven logistics (FWO-S007318N). Editorial consultation provided by Luke Connolly, KU Leuven.

References

1. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B.: Julia: A fresh approach to numerical computing. SIAM Review **59**(1), 65–98 (2017). https://doi.org/10.1137/141000671
2. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM computing surveys (CSUR) **35**(3), 268–308 (2003)
3. Cahon, S., Melab, N., Talbi, E.G.: Paradiseo: A framework for the reusable design of parallel and distributed metaheuristics. Journal of heuristics **10**(3), 357–380 (2004)
4. De Beukelaer, H., Davenport, G.F., De Meyer, G., Fack, V.: James: An object-oriented java framework for discrete optimization using local search metaheuristics. Software: Practice and Experience **47**(6), 921–938 (2017). https://doi.org/https://doi.org/10.1002/spe.2459, `https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2459`
5. Mecler, J., Subramanian, A., Vidal, T.: A simple and effective hybrid genetic search for the job sequencing and tool switching problem. Computers & Operations Research p. 105153 (2020). https://doi.org/https://doi.org/10.1016/j.cor.2020.105153, `http://www.sciencedirect.com/science/article/pii/S0305054820302707`
6. Sörensen, K.: Metaheuristics—the metaphor exposed. International Transactions in Operational Research **22**(1), 3–18 (2015)
7. Swan, J., Adraensen, S., Brownlee, A.E., Johnson, C.G., Kheiri, A., Krawiec, F., Merelo, J., Minku, L.L., Özcan, E., Pappa, G.L., et al.: Towards metaheuristics" in the large". arXiv preprint arXiv:2011.09821 (2020)
8. Swan, J., Adriaensen, S., Bishr, M., Burke, E.K., Clark, J.A., De Causmaecker, P., Durillo, J., Hammond, K., Hart, E., Johnson, C.G., et al.: A research agenda for metaheuristic standardization. In: Proceedings of the XI metaheuristics international conference. pp. 1–3 (2015)
9. Talbi, E.G.: Metaheuristics: from design to implementation, vol. 74. John Wiley & Sons (2009)

A hybrid FLP-AHP approach for Optimal Product Mix in Pulp and Paper Industry

Meenu Singh[1] and Millie Pant[2]
[1, 2] Department of Applied Science & Engineering
Indian Institute of Technology (IIT) Roorkee
Roorkee, India
msingh1@as.iitr.ac.in, pant.milli@as.iitr.ac.in

Abstract. Pulp and Paper Industries (PPI) manufactures a wide range of papers based on three different GSM (Grams/ sq. meter). i.e., lower GSM, middle GSM and higher GSM. In order to maximize the profit, the PPI must efficiently utilize its available resources thereby producing optimal units of three different GSMs. Such problems lie under the category of product mix problems and forms an important part of production planning for every paper mill. In the present study, this problem is represented as a Fuzzy Linear Programming (FLP) model, to include the inherent vagueness and uncertainties. The solutions obtained through FLP are further refined with the help of AHP (Analytical Hierarchical Process) to determine the most profitable solution. Results indicate that ranking results obtained by integrating AHP into FLP may help in providing a better guidance to the Decision Maker (DM) for determining an optimal product mix.

Keywords: Indian Pulp and Paper Industry (IPPI), product mix optimization, fuzzy linear programming (FLP), multi-criteria decision making (MCDM).

1 Introduction

The Pulp and Paper Industry (PPI) plays an important role in Indian economy due to several reasons [1]. Different types of paper produced by a paper mill can be broadly classified into Cultural and Industrial papers [2]. The major production of mills deals with the cultural paper involving all types of writing and printing papers with three different levels of GSM (Grams/ sq. meter). i.e., lower GSM, medium GSM, and higher GSM. The wrapping, packing, photographic and other functional papers are called industrial papers. The various stages of pulping and papermaking process are presented in Fig. 1.

Production planning is an important decision making for any production industry including PPI, where the main objective is to maintain a tradeoff between production and consumption.

The focus of the present study is to suggest an optimal production plan for Indian Pulp and Paper Industries (IPPI) producing a variety of papers. The objective here is to maximize the profit by suggesting an optimal product mix on the basis of different levels of GSM. The problem is formulated as a Fuzzy Linear Programming (FLP) model due to the inherent uncertainties in the model parameters. Further, sensitivity

analysis of the proposed model is done by examining the effect of alpha-cuts on profit. Finally, Analytical Hierarchical Process (AHP) is introduced into the model for evaluating the feasible solutions as alternatives.

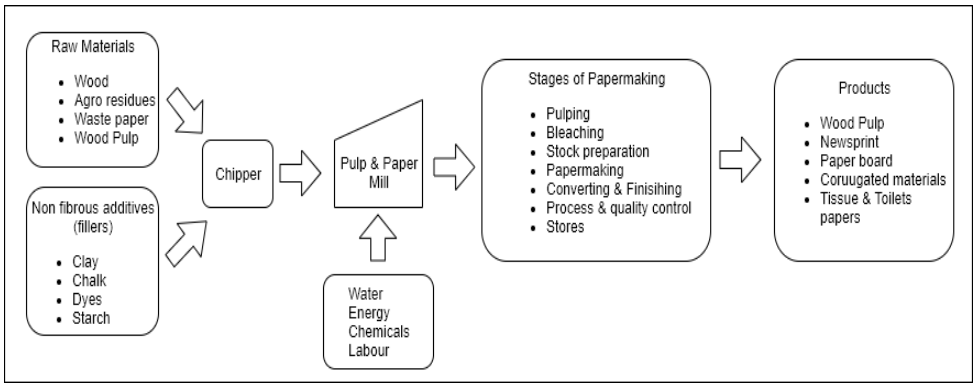


Fig. 1. Process flow diagram of PPI.

Rest of the paper is organized as follows. In section 2, literature review on FLP for production planning is given. In section 3, brief description of the methodology is provided. Section 4 and 5, presents a hypothetical but realistic case study illustrating the applicability of the proposed approach through a mathematical model. Sections 6 provide the results and discussion of the model and the decision making process respectively. Finally, the paper concludes with section 7, summarizing the present study and providing future research directions.

2 LITERATURE REVIEW

Linear Programming (LP) has gained its reputation as one of the best decision making tools for maximizing the goal achievements or minimizing the costs while satisfying all the constraints and restrictions. Literature is full of instances advocating the effectiveness of LP -in different areas [3] and [4].

Introduction of fuzzy logic in LP was suggested long back in 1970's after the concept of fuzzy theory was established by Zadeh in 1965 [5]. In [6-8], the authors suggested that fuzzy set theory can be integrated with other mathematical programming approaches like non-linear programming, quadratic programing, dynamic programming and goal programming for a more realistic representation of the problem. The fuzzy theory is used to optimize the solutions for which constraints have fuzzy coefficients, fuzzy inequalities or fuzzy variables.

Several instance are available in literature where the researchers have successfully implemented FLP by substituting the crisp coefficient with fuzzy numbers in an LP problem [3, 4, 9, 10]. Researchers have also shown that most of real life problems with intensive decision making like product mix, manpower allocation, flow shop scheduling, transportation, production planning [9]–[13] can be dealt efficiently through an

FLP approach. However, there are work in the area of production planning activities [14-16] but the authors were not able to find any relevant literature relating to production planning in a fuzzy environment for an IPPI.

The proposed methodology integrates AHP, a well-known Multi-Criteria Decision Making (MCDM) technique with FLP to select the most profitable solution from the set of solutions achieved in the optimization process.

3 METHODOLOGY

An integrated FLP-AHP model is proposed for determining the optimal product mix for an Indian Pulp and Paper Industry (IPPI). It is a two phase methodology, as illustrated in Fig. 2.

In Phase I, fuzzification of the problem is done, while in Phase 2, AHP is invoked to select the best possible alternative, out of the solutions obtained in Phase I. These phases are further divided into a number of steps as described below and as illustrated in Figure 2.

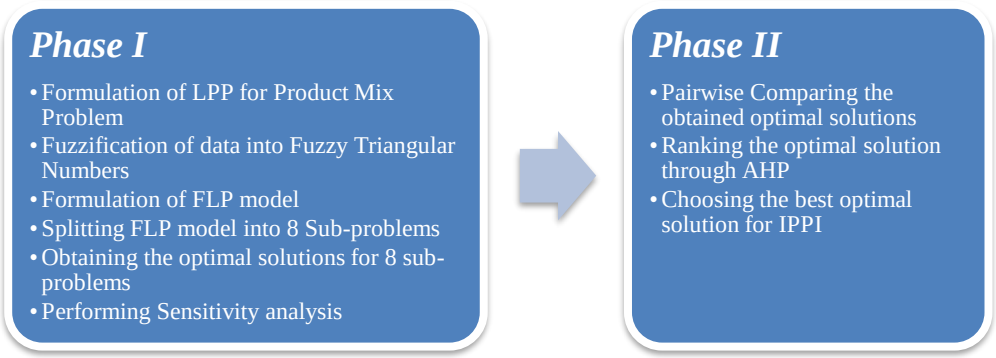


Fig. 2. Schematic presentation of FLP-AHP model

Phase I
Step 1. Formulation of LPP: develop the LPP model with the help of the available crisp data.
Step 2. Fuzzification of data: the available crisp data is changed into fuzzy triangular numbers (FTN).
Step 3. Formulation of FLP model: using the data generated in Step 2, the LPP developed in Step 1 is converted into an FLP model.
Step 4. Splitting: the model developed in Step 3, is split into a number of sub problems.
Step 5. Solution: Obtain the solution through different values of aspirants.

Phase II.

Step 1. Decision Making through AHP: the solutions obtained for different sub problems are treated as alternatives and the best solution is ranked through AHP.

4 CASE STUDY

4.1 Background

A hypothetical but realistic PPI in India is considered, that produces three types of writing/ printing paper based on three different levels of GSM: lower GSM (58 to 64 GSM), medium GSM (68 to 80 GSM) and higher GSM (90 to 120 GSM) papers. Each kind of GSM papers are produced by a combination of raw material fibers such as eucalyptus and poplar with fillers such as china clay, GCC, PCC etc. The range of percentage composition of the raw material fibers with the fillers are presented in Table 1.

Table 1. The % of raw material used in each grade of paper.

Raw material	Lower GSM	Medium GSM	Higher GSM
Pulp	90%	86%	85%
Filler	10%	14%	15%

The data and details for the case study were constructed following a series of discussions with the experts from management and R&D department of an Indian paper mill.

During the papermaking process, the raw material goes through different processes including pulping, bleaching, stock preparation, papermaking, converting and finishing. During each process, the yield of pulp is lost to some extent due to the processing of a particular grade of paper while the remaining amount undergoes the next process.

Table 2. Processing of paper with net losses at each department.

Departments	Lower GSM			Medium GSM			Higher GSM		
	Input	***Losses***	***Output***	***Input***	***Losses***	***Output***	***Input***	***Losses***	***Output***
Input	1.000	-	-	1.000	-	-	1.000	-	-
Pulping Process	1.000	0.01	0.990	1.000	0.02	0.983	1.000	0.02	0.981
Bleaching Process	0.990	0.09	0.901	0.983	0.14	0.845	0.981	0.19	0.795
Stock Preparation Process	0.901	-	0.901	0.845	-	0.845	0.795	-	0.795
Papermaking Process	0.901	0.05	0.856	0.845	0.07	0.786	0.795	0.08	0.734
Converting & Finishing Process	0.856	0.07	0.796	0.786	0.07	0.738	0.735	0.08	0.676
Output	-	-	0.796	-	-	0.738	-	-	0.676

The expected average losses from one ton of input for each grade of paper; average capacity of plant at each unit and the maximum output of the three kinds of papers produced in a day is summarized in Table 2. A blank entry represents “no loss” implying that the same amount is carried on to the next process.

The profit of each GSM for a ton is evaluated after estimating the selling price and other expenses of each GSM as per the current market trend and as per the opinion of experts. The profit for the three GSM with the amount is given in Table 3.

Table 3. Details of plant capacity.

Papermaking process	Tons/ day	Max production of each grade	Tons/ day	Profit of each grade of paper produced	Rs./ Ton
Pulping process	230	Lower GSM	34	Lower GSM	5600
Bleaching process	120	Medium GSM	158	Medium GSM	5000
Stock preparation Process	220	Higher GSM	34	Higher GSM	5100
Converting & Finishing Process	220				

5 Mathematical Model

The above-mentioned data for the papermaking processes and objective function was analyzed to obtain estimates for LP problem model parameters. The decision variables of the model are notated as x_1, x_1 and x_3 representing the amount of: lower GSM, medium GSM and higher GSM to be produced. The objective function Z is to maximize the profit per ton for the three kinds of papers.

The LPP model of the problem can be formulated as below, by using the given data in Table 1-3:

$$
\begin{aligned}
&Max\ Z = 5600x_1 + 5000x_2 + 4200x_3 \\
&s.t. \\
&0.90x_1 + 0.86x_2 + 0.85x_3 \leq 230 && \text{(Raw Material Constraint)} \\
&0.12x_1 + 0.16x_2 + 0.17x_3 \leq 22.7 && \text{(Filler Constraint)} \\
&x_1 + x_2 + x_3 \leq 230 && \text{(Pulping process constraint)} \\
&0.9x_1 + 0.983x_2 + 0.988x_3 \leq 120 && \text{(Bleaching process constraint)} \\
&0.901x_1 + 0.891x_2 + 0.869x_3 \leq 220 && \text{(Stock preparation process constraint)} \\
&0.869x_1 + 0.860x_2 + 0.835x_3 \leq 220 && \text{(Converting \&Finishing Process)} \\
&0.15x_1 + 0.7x_2 + 0.14x_3 \leq 24 && \text{(Time constraint)} \\
&x_1 \leq 34 && \text{(Input constraint for Lower GSM)} \\
&x_2 \leq 158 && \text{(Input constraint for Medium GSM)} \\
&x_3 \leq 34 && \text{(Input constraint for Higher GSM)}
\end{aligned}
\tag{1}
$$

However, there are several factors that affect the preciseness of data like: non-availability of raw materials or fillers, improper management of resources, inadequate power supply, and underutilization of capacity etc.

Presence of uncertainty in data justifies the application of fuzzy set theory for modeling such problems [11-13]. In this study, FLP is implemented to maximize the value of Z

obtained through LP by using fuzzy triangular numbers. Fuzzy numbers generated for each coefficient are given in Table 4.

Table 4. Fuzzy numbers of each coefficient.

	Lower GSM			**Medium GSM**			**Higher GSM**		
Process	***Lower***	***Crisp***	***Higher***	***Lower***	***Crisp***	***Higher***	***Lower***	***Crisp***	***Higher***
Pulp	0.88	0.90	0.90	0.84	0.86	0.86	0.83	0.85	0.85
Filler	0.10	0.12	0.12	0.14	0.16	0.16	0.15	0.17	0.17
Bleaching process	0.989	0.990	0.990	0.983	0.985	0.985	0.980	0.981	0.981
Stock preparation Process	0.890	0.901	0.901	0.837	0.845	0.845	0.790	0.795	0.795
Converting & Finishing Process	0.846	0.856	0.856	0.783	0.786	0.786	0.731	0.735	0.735
Time	0.14	0.15	0.15	0.696	0.7	0.7	0.14	0.15	0.15
Max production	34	34	35	158	158	161	34	34	35

5.1 Formulation of FLP

STEP 1. The crisp data mentioned in the Table 1-3 is utilized to calculate the fuzzy numbers for each coefficient as presented in Table 4.

STEP 2. Using the fuzzy data in Table 4, the Fuzzy Linear Programming model for the case company is formulated as (2):

$$
\begin{aligned}
&Max\ Z = (5600,5600,6500)x_1 + (5000,5000,6200)x_2 + (4200,4200,5800)x_3 \\
&s.t. \\
&(0.88,0.90,0.90)x_1 + (0.84,0.86,0.86)x_2 + (0.83,0.85,0.85)x_3 \leq (230,280,280) \\
&(0.10,0.12,0.12)x_1 + (0.14,0.16,0.16)x_2 + (0.15,0.17,0.17)x_3 \leq (22.7,34.1,34.1) \\
&(0.989,0.990,0.990)x_1 + (0.983,0.985,0.985)x_2 + (0.980,0.981,0.981)x_3 \leq (120,120,120) \\
&(0.890,0.901,0.901)x_1 + (0.837,0.845,0.845)x_2 + (0.790,0.795,0.795)x_3 \leq (220,220,220) \\
&(0.846,0.856,0.856)x_1 + (0.783,0786,0.786)x_2 + (0.731,0.735,0.735)x_3 \leq (220,220,220) \\
&(0.14,0.15,0.15)x_1 + (0.696,0.7,0.7)x_2 + (0.14,0.15,0.15)x_3 \leq (24,24,24) \\
&x_1, x_3 \leq (34,34,35) \\
&x_2 \leq (158,158,161)
\end{aligned}
\tag{2}
$$

STEP 3. To obtain the better optimization result from the fuzzy numbers, the FLP has been split into eight sub-problems of linear crisp programming, as seen in Table 5, by taking in account, the total number of possible combinations of objective values with all the constraints values.

STEP 4. The optimal solutions of each sub-problem (3) to (9) are obtained through Lingo 18.0 software and are presented in Table 6.

Table 5. The sub-problems of FLP.

Sub-problem 1

$$
\begin{aligned}
&Max = 5600x_1 + 5000x_2 + 4200x_3 \\
&s.t. \\
&0.90x_1 + 0.86x_2 + 0.85x_3 \le 280 \\
&0.10x_1 + 0.14x_2 + 0.15x_3 \le 34.1 \\
&0.990x_1 + 0.983x_2 + 0.981x_3 \le 120 \\
&0.901x_1 + 0.845x_2 + 0.795x_3 \le 220 \\
&0.856x_1 + 0.786x_2 + 0.735x_3 \le 220 \\
&0.15x_1 + 0.7x_2 + 0.14x_3 \le 24 \\
&x_1, x_3 \le 34 \\
&x_2 \le 158
\end{aligned}
\tag{3}
$$

Sub-problem 2

$$
\begin{aligned}
&Max = 6500x_1 + 6200x_2 + 5800x_3 \\
&s.t. \\
&0.90x_1 + 0.86x_2 + 0.85x_3 \le 280 \\
&0.10x_1 + 0.14x_2 + 0.15x_3 \le 34.1 \\
&0.990x_1 + 0.983x_2 + 0.981x_3 \le 120 \\
&0.901x_1 + 0.845x_2 + 0.795x_3 \le 220 \\
&0.869x_1 + 0.786x_2 + 0.735x_3 \le 220 \\
&0.15x_1 + 0.7x_2 + 0.14x_3 \le 24 \\
&x_1, x_3 \le 34 \\
&x_2 \le 158
\end{aligned}
\tag{4}
$$

Sub-problem 3

$$
\begin{aligned}
&Max = 5600x_1 + 5000x_2 + 4200x_3 \\
&s.t. \\
&0.90x_1 + 0.86x_2 + 0.85x_3 \le 230 \\
&0.10x_1 + 0.14x_2 + 0.15x_3 \le 22.7 \\
&0.990x_1 + 0.983x_2 + 0.981x_3 \le 120 \\
&0.901x_1 + 0.845x_2 + 0.795x_3 \le 220 \\
&0.856x_1 + 0.786x_2 + 0.735x_3 \le 220 \\
&0.15x_1 + 0.7x_2 + 0.14x_3 \le 24 \\
&x_1, x_3 \le 35 \\
&x_2 \le 161
\end{aligned}
\tag{5}
$$

Sub-problem 4

$$
\begin{aligned}
&Max = 6500x_1 + 6200x_2 + 5800x_3 \\
&s.t. \\
&0.90x_1 + 0.86x_2 + 0.85x_3 \le 230 \\
&0.10x_1 + 0.14x_2 + 0.15x_3 \le 22.7 \\
&0.990x_1 + 0.983x_2 + 0.981x_3 \le 120 \\
&0.901x_1 + 0.845x_2 + 0.795x_3 \le 220 \\
&0.869x_1 + 0.786x_2 + 0.735x_3 \le 220 \\
&0.15x_1 + 0.7x_2 + 0.14x_3 \le 24 \\
&x_1, x_3 \le 35 \\
&x_2 \le 161
\end{aligned}
\tag{6}
$$

Sub-problem 5

$$
\begin{aligned}
&Max = 5600x_1 + 5000x_2 + 4200x_3 \\
&s.t. \\
&0.88x_1 + 0.84x_2 + 0.83x_3 \le 280 \\
&0.12x_1 + 0.16x_2 + 0.17x_3 \le 34.1 \\
&0.989x_1 + 0.983x_2 + 0.980x_3 \le 120 \\
&0.890x_1 + 0.837x_2 + 0.790x_3 \le 220 \\
&0.846x_1 + 0.783x_2 + 0.731x_3 \le 220 \\
&0.15x_1 + 0.696x_2 + 0.15x_3 \le 24 \\
&x_1, x_3 \le 34 \\
&x_2 \le 158
\end{aligned}
\tag{7}
$$

Sub-problem 6

$$
\begin{aligned}
&Max = 6500x_1 + 6200x_2 + 5800x_3 \\
&s.t. \\
&0.88x_1 + 0.84x_2 + 0.83x_3 \le 280 \\
&0.12x_1 + 0.16x_2 + 0.17x_3 \le 34.1 \\
&0.989x_1 + 0.983x_2 + 0.980x_3 \le 120 \\
&0.890x_1 + 0.837x_2 + 0.790x_3 \le 220 \\
&0.846x_1 + 0.783x_2 + 0.731x_3 \le 220 \\
&0.15x_1 + 0.696x_2 + 0.15x_3 \le 24 \\
&x_1, x_3 \le 34 \\
&x_2 \le 158
\end{aligned}
\tag{8}
$$

Sub-problem 7

$$
\begin{aligned}
&Max = 5600x_1 + 5000x_2 + 4200x_3 \\
&s.t. \\
&0.88x_1 + 0.84x_2 + 0.83x_3 \le 230 \\
&0.12x_1 + 0.16x_2 + 0.17x_3 \le 22.7 \\
&0.989x_1 + 0.983x_2 + 0.980x_3 \le 120 \\
&0.890x_1 + 0.837x_2 + 0.790x_3 \le 220 \\
&0.846x_1 + 0.783x_2 + 0.731x_3 \le 220 \\
&0.15x_1 + 0.696x_2 + 0.15x_3 \le 24 \\
&x_1, x_3 \le 35 \\
&x_2 \le 161
\end{aligned}
\tag{9}
$$

Sub-problem 8

$$
\begin{aligned}
&Max = 6500x_1 + 6200x_2 + 5800x_3 \\
&s.t. \\
&0.88x_1 + 0.84x_2 + 0.83x_3 \le 230 \\
&0.12x_1 + 0.16x_2 + 0.17x_3 \le 22.7 \\
&0.989x_1 + 0.983x_2 + 0.980x_3 \le 120 \\
&0.890x_1 + 0.837x_2 + 0.790x_3 \le 220 \\
&0.846x_1 + 0.783x_2 + 0.731x_3 \le 220 \\
&0.15x_1 + 0.696x_2 + 0.15x_3 \le 24 \\
&x_1, x_3 \le 35 \\
&x_2 \le 161
\end{aligned}
\tag{10}
$$

Table 6. Optimal solutions of eight sub-problems.

	x_1	x_2	x_3	**Z**	
X_{11}	34	19.71	34	431771.4	*Lower bound*
X_{12}	34	19.71	34	540428.6	
X_{13}	35	19.29	35	439428.6	
X_{14}	35	19.29	35	550071.4	
X_{15}	34	20.80	34	437223.0	
X_{16}	34	19.71	34	540428.6	
X_{17}	35	19.29	35	439428.6	
X_{18}	35	19.28	35	550071.4	*Upper bound*

STEP 5. The crisp fuzzy linear model for maximizing the value of aspiration (λ) is written as below:

$$Max\ Z = \lambda$$
$$s.t.$$
$$118300.0\lambda - 6500x_1 - 6200x_2 - 5800x_3 + 550071.4 \leq 0$$
$$(0.90 - 0.02\lambda)x_1 + (0.86 - 0.02\lambda)x_2 + (0.85 - 0.02\lambda)x_3 - 50\lambda - 280 \leq 0$$
$$(0.12 - 0.02\lambda)x_1 + (0.16 - 0.02\lambda)x_2 + (0.17 - 0.02\lambda)x_3 - 11.4\lambda - 34.1 \leq 0$$
$$(0.990 - 0.001\lambda)x_1 + (0.985 - 0.002\lambda)x_2 + (0.981 - 0.001\lambda)x_3 - 120 \leq 0$$
$$(0.901 - 0.011\lambda)x_1 + (0.845 - 0.008\lambda)x_2 + (0.795 - 0.005\lambda)x_3 - 220 \leq 0$$
$$(0.856 - 0.010\lambda)x_1 + (0.786 - 0.003\lambda)x_2 + (0.735 - 0.004\lambda)x_3 - 220 \leq 0$$
$$(0.15 - 0.01\lambda)x_1 + (0.7 - 0.004\lambda)x_2 + (0.15 - 0.001\lambda)x_3 - 24 \leq 0$$
$$x_1 - \lambda - 34 \leq 0$$
$$x_2 - (3.0\lambda) - 158 \leq 0$$
$$x_3 - \lambda - 34 \leq 0$$
$$0 \leq \lambda \leq 0$$
$$x_1, x_2, x_3 \geq 0 \qquad (10)$$

STEP 6. The solutions in Table 7 are obtained by varying the value of λ in Lingo 18.0 software. It is observed that the maximum profits of the case company varies depending on the value of λ and variables of product mix. Also, only 3 solutions are found to be feasible (highlighted in grey) while 4 solutions are infeasible.

Table 7. Senstivity analysis of value of aspirations.

λ	x_1	x_2	x_3	Z	**Solution Type**
0	34.5	20.603	34.5	₹ 5,52,091.39	*Feasible*
0.3	34.62	26.150	34.62	₹ 4,70,027.35	*Feasible*
0.5	34.7	29.848	34.7	₹ 4,89,300.75	*Non-Feasible*
0.7	34.78	33.546	34.78	₹ 5,08,574.15	*Non-Feasible*
0.9	34.86	37.244	34.86	₹ 5,27,847.55	*Non-Feasible*
1	34.9	39.093	34.9	₹ 5,37,484.25	*Non-Feasible*
0.3309	34.63237	26.722	34.63237	₹ 4,73,006.38	*Feasible*

5.2 AHP

In order to select the best optimal solution from the Table 6, Analytical Hierarchy Process (AHP) is employed [17] as discussed in the next section.

The hierarchy structure in Fig. 3 presents the three feasible solutions (S1, S2, S3) for selection of the best solution. The priority conditions and alternatives were decided in the interview of three decision-makers (DMs), such as (DM 1, DM 2, DM 3). In order to prevent biases in their judgement, the DMs advocated for the equal. The same weights were therefore assigned to them as (1/3, 1/3, 1/3). Implementation of AHP consisted of the following steps:

1. Different comparison matrices were formed for the alternatives with respect to each criterion individually for DMs.
2. The final comparison matrix is aggregated as geometric mean of individual DMs matrix.

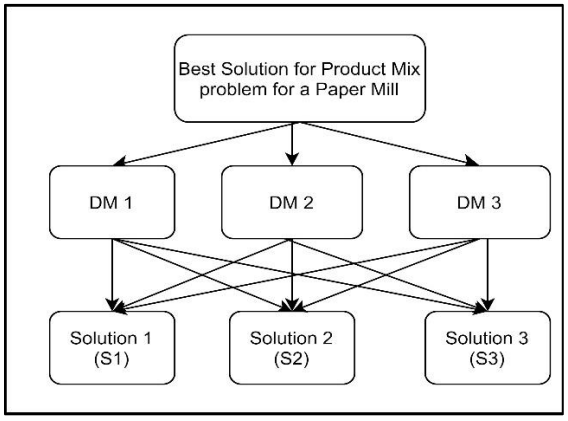


Fig. 3. A hierarchy structure of the problem.

6 Results and Discussions

6.1 Sensitivity Analysis: Membership of aspiration values contributed towards managing the profit of the mill is discussed in this section. The trends of profits and the trend of decision variables against the value of $\boldsymbol{\lambda}$ are presented in Fig. 4(a) & Fig. 4(b). The following observations were made:

1. The profit of product mix decreased in tandem with the decrease in the value of aspiration level. The maximum and minimum profits were obtained for λ =1 and λ =0.3 respectively. This is shown graphically in Fig 4(a).

2. Decision variables x_1 and x_2 have the same values and the gap between them with x_2 is smaller as the value of λ increases. At λ = 0.9 or 1, the production amount of x_2 is greater than the x_1 & x_3. This trend can be visualized through Fig. 4(b).

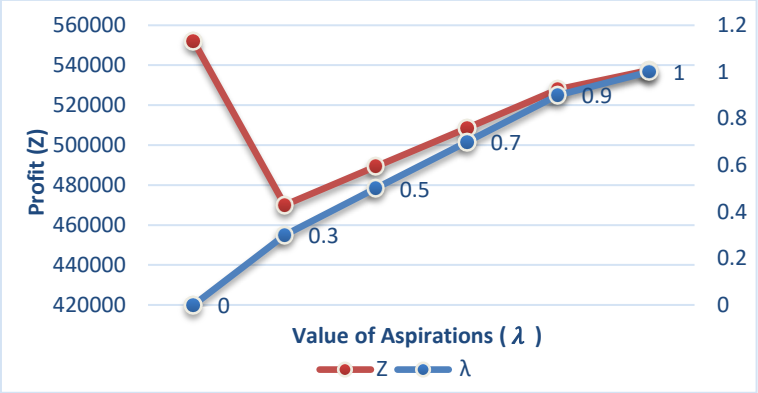


Fig. 4 (a). Trends of Profit against the value of aspirations.

3. Through the trends mentioned in (a) and (b), it can be said that the profit (in Rupees) in a closed interval of [470027.35, 552091.39] can be obtained if the lower GSM, medium GSM and higher GSM were produced (in Tons) in the interval of [34.5, 34.62], [20.6, 26.15] and [34.5, 34.62], **respectively**.

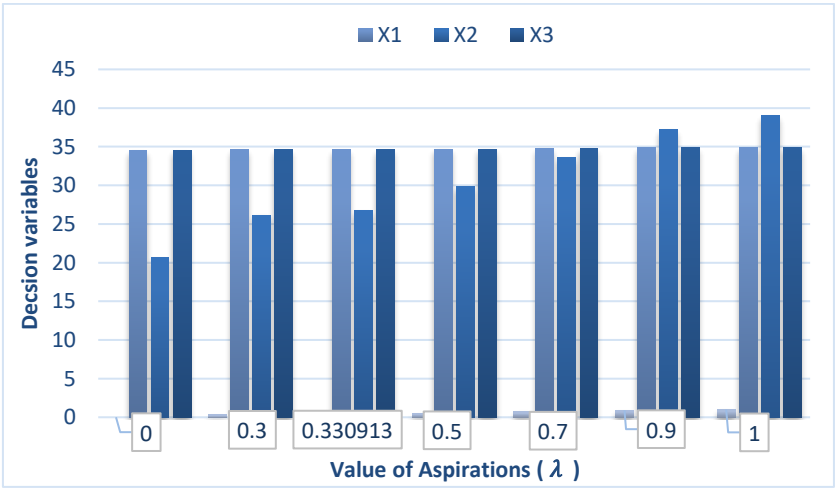


Fig.4 (b). Decision variables vs. aspiration level.

4. The optimal feasible solutions were obtained by λ value in the range of [0, 0.330**].**

5. The crisp profit earned through LPP was Rs 434200.0, which improved up to 21.35%. i.e., Rs 117891.40 by the use of FLP.

6.2 Decision making through AHP

1. After determining the local priorities and checking the consistency at 10%. The global final priority is evaluated and is depicted in Fig. 5.
2. It can be clearly seen that the S1 is the most profitable solution among all the three optimum solutions, due to its overall weight (0.599). It's the best combination for the product mix of the case company with the production of 89.60 tons/ day in which 34.5 tons/ day is for lower GSM, 20.60 tons/ day for medium GSM and 34.5 for higher GSM.

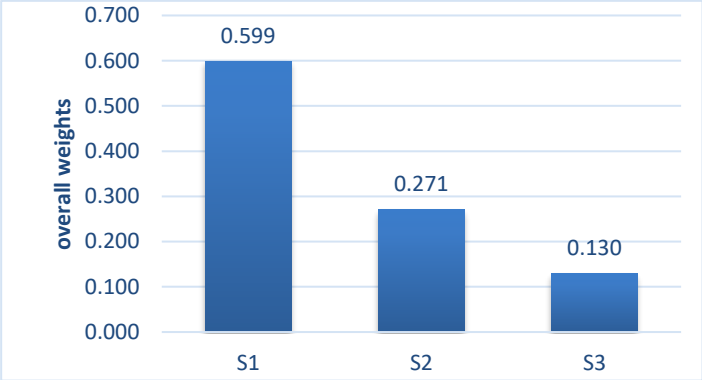


Fig. 5. AHP overall weights for the three optimal solutions.

7 Conclusion

The product mix optimization for maximizing the profit has attracted a lot of attention in recent years. This paper presented an integrated FLP-AHP approach for determining the optimal product mix for an IPPI. The research tends to be a valuable method for improving production planning for three high grades of writing/ printing paper with minimal resources. Some conclusions that can be drawn from the research:

- Implementation of fuzzy theory helped in enhancing the profit by 21.35%, which is a significant improvement for a real life situation.
- Integrating AHP into FLP will help managers make better decisions by assisting in identifying the most profitable solution among the available solutions.
- In this study, the authors have considered an example of an IPPI. However, due to the generic nature of FLP-AHP, it can be applied to other manufacturing sectors as well to determine product mix.

Further study can be undertaken by including more variables and constraints that take place during production planning activities, constructing a multi-objective model or considering other fuzzy membership functions like trapezoidal, logistics, S-curve etc. Sustainability factors may also be included to make the model more realistic.

References

1. Challenges and Opportunities for the Pulp and Paper Industry | SciTech Connect, http://scitechconnect.elsevier.com/challenges-and-opportunities-for-the-pulp-and-paper/, last accessed: 2020/11/27.
2. Jain, R.K.: Compendium of Census survey of Indian Paper Industry. 1st edn. Central Pulp and Paper Institute, Saharanpur (2015).
3. Spitter, J. M., Hurkens, C. A., De Kok, A. G., Lenstra, J. K., & Negenman, E. G.: Linear programming models with planned lead times for supply chain operations planning. *European Journal of operational research*, *163*(3), 706-720 (2005).
4. Li, Z., & Li, Z.: Linear programming-based scenario reduction using transportation distance. *Computers & Chemical Engineering*, *88*, 50-58.
5. Zadeh, L. A.: Fuzzy Sets, Information and Control, 8: 338-353 (1965).
6. Tanaka, H., & Asai, K.: Fuzzy linear programming problems with fuzzy numbers. *Fuzzy sets and systems*, *13*(1), 1-10 (1984).
7. Rommelfanger, H.: Fuzzy linear programming and applications. *European journal of operational research*, *92*(3), 512-527 (1996).
8. Bector, C. R., & Chandra, S.: *Fuzzy mathematical programming and fuzzy matrix games*, vol. 169, Springer, Berlin (2005).
9. Buckley, J. J.: Possibilistic linear programming with triangular fuzzy numbers. *Fuzzy sets and Systems*, *26*(1), 135-138 (1988).
10. Zimmermann, H. J.: Fuzzy programming and linear programming with several objective functions. *Fuzzy sets and systems*, *1*(1), 45-55 (1978).
11. Pendharkar, P. C.: A fuzzy linear programming model for production planning in coal mines. *Computers & operations research*, *24*(12), 1141-1149 (1997).
12. Wu, Y. K.: On the manpower allocation within matrix organization: a fuzzy linear programming approach. *European Journal of Operational Research*, *183*(1), 384-393 (2007).
13. Abdullah, L., & Abidin, N. H.: A fuzzy linear programming in optimizing meat production. *International Journal of Engineering and Technology*, *6*(1), 436-444 (2014).
14. Fang, C. C., Lai, M. H., & Huang, Y. S., Production planning of new and remanufacturing products in hybrid production systems. *Computers & Industrial Engineering*, *108*, 88-99 (2017).
15. Saniuk, S., & Saniuk, A., Decision support system for rapid production order planning in production network. In *International Conference on Intelligent Systems in Production Engineering and Maintenance* (pp. 217-226). Springer, Cham (2017).
16. Rodríguez, G. G., Gonzalez-Cava, J. M., & Pérez, J. A. M., An intelligent decision support system for production planning based on machine learning. *Journal of Intelligent Manufacturing*, 1-17 (2019).
17. Saaty, T. L.: The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation: McGraw-Hill. *Inc. New York, NY* (1980).

Robustness of Adversarial Images against Filters

Raluca Chitic[1 [0000−0003−1113−2343]], Nathan Deridder[1 [0000−0003−1215−1239]], Franck Leprévost[1 [0000−0001−8808−2730]], and Nicolas Bernard[2 [0000−0003−3857−9591]]

[1] University of Luxembourg
House of Numbers, 6, avenue de la Fonte, L-4364 Esch-sur-Alzette
G-D of Luxembourg
Raluca.Chitic@uni.lu & Nathan.Deridder.001@student.uni.lu & Franck.Leprevost@uni.lu
[2] La Fraze
1288, chemin de la Fraze, F-88380 Arches, France
Nicolas.Bernard@lafraze.com

Abstract. This article addresses the robustness issue of adversarial images against filters. Given an image $\mathcal{A}$, that a convolutional neural network and a human both classify as belonging to a category $c_{\mathcal{A}}$, one considers an adversarial image $\mathcal{D}$ that the neural network classifies in a category $c_t \neq c_{\mathcal{A}}$, although a human would not notice any difference between $\mathcal{D}$ and $\mathcal{A}$. Would the application of a filter F (such as the Gaussian blur filter) to $\mathcal{D}$ still lead to an adversarial image $F(\mathcal{D})$ that fools the neural network? To address this issue, we perform a study on VGG-16 trained on CIFAR-10, with adversarial images obtained thanks to an evolutionary algorithm run on a specific image $\mathcal{A}$ taken in one category of CIFAR-10. Exposed to 4 individual filters, we show that the outputted filtered adversarial images essentially do remain adversarial in some sense. We also show that combining filters may render our EA attack less effective. We therefore design a new evolutionary algorithm, whose aim is to create adversarial images that do pass the filter test, do fool VGG-16 and do remain close enough to $\mathcal{A}$ that a human would not notice any difference. We show that this is indeed the case by running this new algorithm on the same image $\mathcal{A}$.

1 Introduction

During the last decade, Neural Networks, and particularly Convolutional Neural Networks (CNNs), have established themselves as the leading way to recognise objects in images. From there, they can be applied to automated image classification, image segmentation, video feed monitoring, etc. However, they are not absolutely foolproof. *Trompe-l'œil* can fool a human into seeing something that is not really there. In the same way, a CNN can be wrong from time to time, misclassifying an object in a picture as something else. *Adversarial* images are specially crafted to this purpose.

Significant work has been performed on adversarial attacks which are designed to fool CNNs trained for object recognition. Among the different types of successful adversarial attacks are those based on Evolutionary Algorithms (EAs). Although EA-based attacks produce adversarial images that are misclassified by CNNs, these images often contain noise-like artefacts. This can pose an issue for the similarity between the original, unmodified and the adversarial images, which is a requirement for adversarial attacks [2]. Moreover, noise-removing filters are a staple of image processing. It raises questions regarding the robustness of noisy adversarial images: Would it be enough to add a filter in front of a CNN to protect it against such existing attacks? Or from the attacker's point of view, given an adversarial image fooling a CNN, is it robust? Does the filtered adversarial image remain adversarial? If it does not, then is it possible to modify an EA-based attack to fool the combination Filter + CNN? This article addresses these questions in a specific context, and in elaborated scenarios.

The considered CNN, briefly described in section 2, is VGG-16 [8, 12] trained on the CIFAR-10 [9] dataset to classify images according to 10 categories. The adversarial images are obtained by the evolutionary algorithm $\mathrm{EA}_{L_2}^{\mathrm{target}}$ introduced in [5, 4] for the *target scenario*. In a nutshell, this scenario considers two different categories c_t and $c_{\mathcal{A}}$, and an image $\mathcal{A}$ classified by a trained CNN in $c_{\mathcal{A}}$. An EA then aims at evolving $\mathcal{A}$ into an adversarial image $\mathcal{D}$, that the CNN classifies as belonging to c_t, while remaining close to $\mathcal{A}$ for a human eye. Section 3 summarizes the main features of $\mathrm{EA}_{L_2}^{\mathrm{target}}$ in the context of the *target scenario* instantiated on VGG-16 trained on CIFAR-10. Section 4 first explicits the implementation aspects and the parameters of $\mathrm{EA}_{L_2}^{\mathrm{target}}$. Then, one example is detailed. This section displays both the ancestor image in the category *dog*, and the adversarial descendent image in each of the 9 distinct remaining target categories of CIFAR-10, obtained by explicitly running $\mathrm{EA}_{L_2}^{\mathrm{target}}$ on the chosen ancestor with these parameters. To address the questions at the origin of this paper, a series of filters are compared in section 5, and then applied in section 6 to the images of section 4.

A first outcome is that the adversarial images created by $\mathrm{EA}_{L_2}^{\mathrm{target}}$ and the ancestor essentially have the same pattern once exposed to individual filters. Depending on the filter, filtered adversarial images remain adversarial, either for the *target* scenario or for the *untargeted* scenario, for which one only requires the adversarial image to be classified in a different category than the original one. However, using composition of different filters render the $\mathrm{EA}_{L_2}^{\mathrm{target}}$ attack less effective, not only for the *target* but also for the *untargeted* scenario. This outcome leads to the construction in section 7 of the variant $\mathrm{EA}_{L_2}^{\mathrm{target},F}$ of the EA, whose fitness function natively includes the robustness against the filter F for the *target* scenario. Section 7 shows the adversarial images obtained by running $\mathrm{EA}_{L_2}^{\mathrm{target},F}$ for a specific composition of filters on the same *dog* ancestor image, and the behavior of these new adversarial images towards filters. Section 8 summarizes the conclusions of this case study, some characteristics of the new

black-box, targeted, non-parametric creation process $\mathrm{EA}_{L_2}^{\mathrm{target},F}$ of adversarial images robust against filters, and provides a series of research directions.

This article, formalizing some aspects of the bachelor student project of the second author, is an additional contribution to the research program announced in [2].

2 VGG-16 trained on CIFAR-10

Although applicable to any CNN trained at image classification on some dataset, we instantiate our approach on a concrete case: VGG-16 trained on CIFAR-10. On the one hand, the dataset CIFAR-10 [9] encompasses $50,000$ training images, and $10,000$ test images of size $32 \times 32 \times 3$, meaning that each image has a width and height of 32 pixels, each pixel having a color resulting from the 3 RGB values. The images are sorted according to $\ell = 10$ categories (see Table 1).

Table 1. CIFAR-10.– For $1 \leq i \leq 10$, the 2^{nd} row specifies the category c_i of CIFAR-10. In our experiment, we shall use the picture $n^\circ 16$ in the *dog* category from the test set of CIFAR-10 as ancestor.

i	1	2	3	4	5	6	7	8	9	10
c_i	plane	car	bird	cat	deer	dog	frog	horse	ship	truck

On the other hand, an input image $\mathcal{I}$ given to VGG-16 [12] is processed through 16 layers to produce a classification output vector $\mathbf{o}_{\mathcal{I}}$ of size $\ell = 10$ in the case considered, namely $\mathbf{o}_{\mathcal{I}} = (\mathbf{o}_{\mathcal{I}}[1], \cdots, \mathbf{o}_{\mathcal{I}}[10])$, where $0 \leq \mathbf{o}_{\mathcal{I}}[i] \leq 1$, and $\sum_{i=1}^{10} \mathbf{o}_{\mathcal{I}}[i] = 1$. Each value $\mathbf{o}_{\mathcal{I}}[i]$ measures the probability that the image $\mathcal{I}$ belongs to the category c_i. As a consequence, an image $\mathcal{I}$ is classified as belonging to the category c_k if $k = \arg\max_{1 \leq i \leq 10}(\mathbf{o}_{\mathcal{I}}[i])$.

3 Target and untargeted scenarios, and design of $\mathbf{EA}_{\boldsymbol{L_2}}^{\mathbf{target}}$

The *target* scenario consists in first choosing two different categories $c_t \neq c_{\mathcal{A}}$ among the 10 categories of CIFAR-10. Then one is given an ancestor image $\mathcal{A}$ labelled by VGG-16 as belonging to $c_{\mathcal{A}}$. Finally one constructs a new image $\mathcal{D}$, classified by VGG-16 as belonging to c_t, although $\mathcal{D}$ remains so close to $\mathcal{A}$ that a human would likely classify $\mathcal{D}$ as belonging to $c_{\mathcal{A}}$ or even be unable to distinguish $\mathcal{D}$ from $\mathcal{A}$. The classification threshold value is set at 0.95, meaning that such a $\mathcal{D}$ has achieved its purpose if $\mathbf{o}_{\mathcal{D}}[t] \geq 0.95$. We shall also encounter in section 6 the slightly different *untargeted* scenario. An adversarial image $\mathcal{D}$ is still required to be similar to $\mathcal{A}$ for a human eye, while VGG-16 classifies $\mathcal{D}$ as

belonging to a category $c \neq c_{\mathcal{A}}$, in the sense that the label value of c outputted by VGG-16 for $\mathcal{D}$ is the largest among all label values, and is strictly larger than the label value of $c_{\mathcal{A}}$. In particular, an image adversarial for the *target* scenario is also adversarial for the *untargeted* scenario, but the inverse may not be true.

Keeping these notations, let us summarize the strategy adopted in [5, 4] to construct an evolutionary algorithm $\mathrm{EA}_{L_2}^{\mathrm{target}}$, that creates such adversarial images fooling VGG-16 trained on CIFAR-10 for the target scenario. The main components of our EA are as follows.

Population initialization. The initial population is set to 160 copies of the ancestor image $\mathcal{A}$.

Evaluation. This operation is performed on each individual image *ind* of a given generation g_p *via* the fitness function $fit_{L_2}(ind, g_p)$ that takes into account a dual goal made of both the evolution of *ind* towards the target category c_t, and its proximity with the ancestor $\mathcal{A}$, measured thanks to the L_2-norm:

$$fit_{L_2}(ind, g_p) = A(g_p, ind)\mathbf{o}_{ind}[c_t] - B(g_p, ind)L_2(ind, \mathcal{A}) \geq 0, \tag{1}$$

where the quantities $A(g_p, ind), B(g_p, ind) \geq 0$ weight and balance the dual goal (see section 4 for their values). The L_2-norm is used to calculate the difference between the pixel values of the ancestor and of the considered image *ind*:

$$L_2(ind, \mathcal{A}) = \sum_{p_j} |ind[p_j] - \mathcal{A}[p_j]|^2, \tag{2}$$

where p_j is the pixel in the j^{th} position, and $0 \leq ind[p_j], \mathcal{A}[p_j] \leq 255$ are the corresponding pixel values of the images *ind* and $\mathcal{A}$.

Evolution. Once the fitness function of each individual in the population is computed (starting with the first generation made of the initial population), the on-going generation is split into 3 groups. The "elite" consists of the 10 best individuals in the population. The "didn't make it", consisting of the lower scored half of the population, is discarded. The "middle class" consists of the remaining individuals. The "elite" is kept unchanged. Each of the 80 individuals of the "didn't make it" group is replaced by an individual resulting of the mutation of elements from the "elite" and the "middle-class". All "middle-class" individuals are mutated. The performed mutations are those described in [4] (they remain similar to some extent to those of [1]). Cross-overs (see [4]) are applied to all individuals except those of the "elite". Pixel values are modified in a range ± 3 in the version used here of $\mathrm{EA}_{L_2}^{\mathrm{target}}$. These operations lead to the 160 individuals composing the new generation subject to the next round of evaluation.

This loop is performed as many times as necessary to create the adversarial image $\mathcal{D}$ as the result of $\mathrm{EA}_{L_2}^{\mathrm{target}}$ run on $\mathcal{A}$ for the target category c_t. Hence $\mathcal{D} = \mathrm{EA}_{L_2}^{\mathrm{target}}(\mathcal{A}, c_t)$ satisfies $\mathbf{o}_{\mathcal{D}}[t] \geq 0.95$.

4 Obtention of the adversarial images: Running $\mathbf{EA}_{\boldsymbol{L_2}}^{\mathbf{target}}$

Concretely, for any generation g_p, one sets $B(g_p, ind) = 10^{-\log_{10}(L_2(ind,\mathcal{A}))}$. The value of $A(g_p, ind)$ depends on $\boldsymbol{o}_{ind}[c_t]$ (note that $\log_{10} \boldsymbol{o}_{ind}[c_t] \leq 0$).

$$A(g_p, ind) = \begin{cases} 10^{-3+\log_{10} \boldsymbol{o}_{ind}[c_t]} & \text{if } \boldsymbol{o}_{ind}[c_t] < 10^{-3} \\ 10^{-2+\log_{10} \boldsymbol{o}_{ind}[c_t]} & \text{if } 10^{-3} \leq \boldsymbol{o}_{ind}[c_t] < 10^{-2} \\ 10^{-1+\log_{10} \boldsymbol{o}_{ind}[c_t]} & \text{if } 10^{-2} \leq \boldsymbol{o}_{ind}[c_t] \end{cases} \tag{3}$$

$\mathrm{EA}_{L_2}^{\mathrm{target}}$ was implemented in Python 3.7 with the NumPy [11] library. Keras [6] was used to load and run the VGG-16 [12] model. Our experiments were run on a computer with an Nvidia RTX 3080 GPU and an Amd Ryzen 7 5800X CPU.

Figure 1 shows the ancestor image $\mathcal{A}$ taken in the category *dog* (image $n°16$ in the category c_6 of the test set of CIFAR-10, used throughout this article). Besides $\mathcal{A}$, Figure 1 also presents the 9 evolved adversarial images $\mathcal{D}_i = \mathrm{EA}_{L_2}^{\mathrm{target}}(\mathcal{A}, c_i)$, with $i \neq 6$, classified by VGG-16 as belonging to the category c_i with the notations of Table 1. By slightly changing many pixels instead of heavily changing a few pixels, this approach, that enhances the indistinguishability between the adversarial image and the ancestor image, differs substantially from [13], where one single pixel is changed, but this modification is noticeable for a human without difficulty.

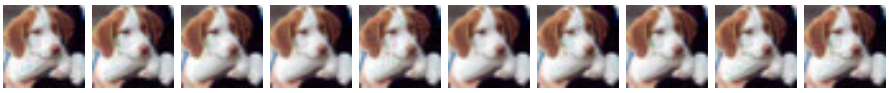


Fig. 1. Comparison of the ancestor $\mathcal{A}$ (chosen as the image $n°16$ in the *dog* category c_6) in the 6^{th} position with the adversarial images $\mathcal{D}_i = \mathrm{EA}_{L_2}^{\mathrm{target}}(\mathcal{A}, c_i)$ in the i^{th} position ($i \neq 6$). VGG-16 classifies $\mathcal{A}$ in the *dog* category with probability 0.9996387, and classifies $\mathcal{D}_i$ in the target category c_i with probability ≥ 0.95.

Table 2 specifies the number of generations and the execution time required by $\mathrm{EA}_{L_2}^{\mathrm{target}}$ to create the adversarial images $\mathcal{D}_i$ of Figure 1. The images pictured in Figure 1 are tested in section 6. More precisely, filters performed on these images create new images that are given as input to VGG-16 for classification. The choice of these filters is described in the next section.

5 Selection of filter

In image processing, a filter or a *Kernel* [14] is essentially given by a square $f \times f$ matrix for an odd integer f. Filtering an image $\mathcal{I}$, say of size $n \times n$, is an operation performed pixel for pixel as follows. For each pixel p of $\mathcal{I}$, one puts

Table 2. For $1 \leq i \leq 10$, $i \neq 6$, the 2^{nd} row specifies the number of generations required by $\text{EA}_{L_2}^{\text{target}}$ to create the adversarial image $\mathcal{D}_i$ pictured in Figure 1. The 3^{rd} row shows the total execution time, measured in seconds, while the 4^{th} row represents the average number of generations per second.

i	1	2	3	4	5	7	8	9	10
# of generations	815	960	494	127	1011	376	286	970	526
total time (in seconds)	46.22	69.14	27.02	8.27	53.93	20.67	16.2	52.36	28.21
# of generations/second	17.63	13.88	18.28	15.36	18.74	18.19	17.65	18.53	18.65

in matrix form a $f \times f$ area of the image centered on p. The coefficients of the resulting $f \times f$ matrix $\mathcal{I}_p$ are the RGB values of the corresponding pixels in the considered area. The convolution operation of the kernel matrix and of $\mathcal{I}_p$ leads to a $f \times f$ matrix $F * \mathcal{I}_p$. The values of the pixel p of the filtered image $F(\mathcal{I})$ is the sum of the coefficients of $F * \mathcal{I}_p$. Pixels at a distance $< f$ of an edge of $\mathcal{I}$ require a special treatment to ensure that the size of the filtered image $F(\mathcal{I})$ is also of size $n \times n$ (otherwise, its size would be reduced to $(n-f+1) \times (n-f+1)$).

Although one could consider a large list of filters, we focus in this article on the following four [10, chapters 7 and 8], that have a significant impact on images. In our computations performed on images of size 32×32, we shall take $f = 1$ for F_1 and $f = 3$ for F_2, F_3, F_4, and used the OpenCV implementation library [3].

The *inverse filter* F_1 replaces all colors by their complementary colors. This operation is performed pixel for pixel by subtracting the RGB value $(255, 255, 255)$ of white by the RGB value of that pixel.

The *Gaussian blur filter* F_2 uses a Gaussian distribution to calculate the Kernel, $G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$, where x is the distance from the origin on the x-axis, y is the distance from the origin on the y-axis and σ is the standard deviation of the Gaussian distribution. By design, the process gives more priority to the pixels in the center, and blurs around it with a lesser impact as one moves away from the center.

The *median filter* F_3 is used to reduce noice and artefacts in a picture. Though under some conditions it can reduce noise while preserving the edges, this does not really occur for small images like those considered here. In general, one selects a pixel, and one computes the median of all the surrounding pixels.

The *unsharp mask filter* F_4 enhances the sharpness and contrast of images. The unsharp masked image is obtained by blurring a copy of the image using a Gaussian blur, which is then weighted and subtracted from the original image.

Any such filter F, or any combination of filters $F_{i_1}, F_{i_2}, \cdots, F_{i_k}$ operating successively (in that order) on an image $\mathcal{I}$, creates a filtered image $F(\mathcal{I})$ or $F_{i_k} \circ$

$\cdots \circ F_{i_2} \circ F_{i_1}(\mathcal{I})$. In section 6 we make use of the previous four filters taken individually, and of the combination $F_3 \circ F_4$.

6 Filtering the ancestor and the adversarial images

The ancestor image $\mathcal{A}$, and the adversarial images $\mathcal{D}_i = \mathrm{EA}_{L_2}^{\mathrm{target}}(\mathcal{A}, c_i)$ $(i \neq 6)$ represented in Figure 1 are tested against the filters of section 5. Figure 2 shows the outcome of this process. From left to right, the 10 pictures on the k^{th} row represent $F(\mathcal{A})$ in the 6^{th} position and $F(\mathcal{D}_i)$ in the i^{th} position for $1 \leq i \neq 6 \leq 10$, with $F = F_k$ for $1 \leq k \leq 4$, and $F = F_3 \circ F_4$ for the 5^{th} row. The reason for the choice of $F_3 \circ F_4$ is that F_4 is used to amplify and highlight detail, while F_3 is used to remove noise from an image without removing detail. Therefore, a combination of these filters could remove the noise created by the EA while maintaining a high level of detail.

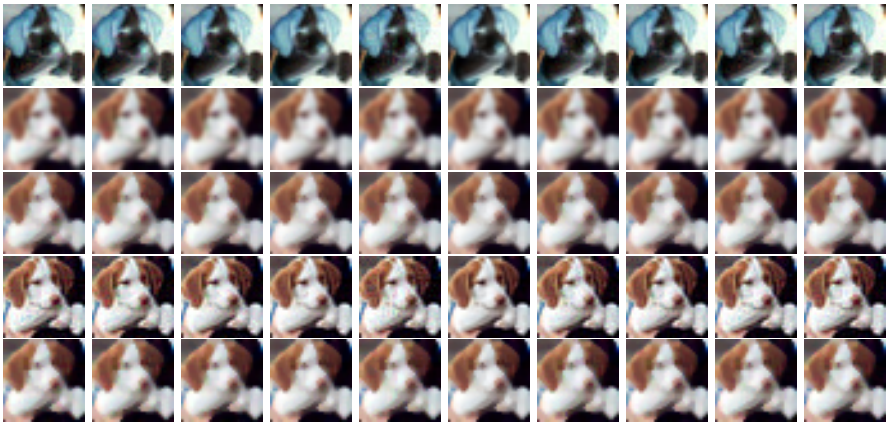


Fig. 2. Comparison of the impact of filters on the ancestor $\mathcal{A}$ and on the adversarial images $\mathcal{D}_i$. The k^{th} row represents $F(\mathcal{A})$ (in 6^{th} position) and $F(\mathcal{D}_i)$ (in i^{th} position, $i \neq 6$), where $F = F_k$ for $1 \leq k \leq 4$, and $F = F_3 \circ F_4$ for $k = 5$.

These filtered images are given to VGG-16 for classification. Table 3 shows the outcome with filters $F = F_k$ for $1 \leq k \leq 3$, while Table 4 is produced with F_4, and Table 5 with $F_3 \circ F_4$. Each Table has (groups of) rows showing the probability of the filtered images for the c_6 category, the target class c_i, the maximum probability and its corresponding class outputted by VGG-16, respectively. In all tables, we set $\mathcal{D}_6 = \mathcal{A}$ to ease the notations.

The inverse, Gaussian blur and median filters (F_1, F_2 and F_3, Table 3) produce images that are adversarial against VGG-16 for the *untargeted* scenario. Indeed,

Table 3. Label values in the category c_6, target class c_i, maximum probability and its corresponding class given by VGG-16 for the filtered ancestor $F(\mathcal{A})$ and the filtered adversarial images $F(\mathcal{D}_i)$ ($i \neq 6$) for $F = F_1$ (1^{st} group of 5 rows), F_2 (2^{nd} group of 5 rows) and F_3 (3^{rd} group of 5 rows).

i	1	2	3	4	5	6	7	8	9	10
$\mathbf{o}_{F_1(\mathcal{D}_i)}[6]$	8e-04	1e-02	8e-03	8e-04	5e-03	1e-03	2e-02	1e-02	3e-03	8e-04
$\mathbf{o}_{F_1(\mathcal{D}_i)}[i]$	5e-04	8e-04	8e-05	0.99	6e-04	1e-03	2e-02	4e-05	1e-02	0.91
$\max(\mathbf{o}_{F_1(\mathcal{D}_i)})$	0.98	0.72	0.98	0.99	0.68	0.99	0.95	0.97	0.82	0.91
$c_{\arg\max(\mathbf{o}_{F_1(\mathcal{D}_i)})}$	truck	frog	cat	cat	frog	cat	cat	cat	truck	truck
i	1	2	3	4	5	6	7	8	9	10
$\mathbf{o}_{F_2(\mathcal{D}_i)}[6]$	1e-04	1e-02	3e-03	1e-04	6e-04	3e-04	1e-03	1e-03	1e-04	2e-04
$\mathbf{o}_{F_2(\mathcal{D}_i)}[i]$	1e-05	2e-06	8e-05	0.99	2e-05	3e-04	1e-05	2e-05	3e-06	2e-06
$\max(\mathbf{o}_{F_2(\mathcal{D}_i)})$	0.99	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
$c_{\arg\max(\mathbf{o}_{F_2(\mathcal{D}_i)})}$	cat	cat	cat	cat	cat	cat	cat	cat	cat	cat
i	1	2	3	4	5	6	7	8	9	10
$\mathbf{o}_{F_3(\mathcal{D}_i)}[6]$	1e-02	0.84	0.22	6e-03	1e-02	0.26	0.14	0.12	2e-02	0.11
$\mathbf{o}_{F_3(\mathcal{D}_i)}[i]$	1e-05	3e-06	1e-03	0.99	2e-05	0.26	5e-05	6e-05	4e-06	5e-06
$\max(\mathbf{o}_{F_3(\mathcal{D}_i)})$	0.98	0.84	0.77	0.99	0.98	0.73	0.86	0.88	0.97	0.89
$c_{\arg\max(\mathbf{o}_{F_3(\mathcal{D}_i)})}$	cat	dog	cat	cat	cat	cat	cat	cat	cat	cat

Table 4. Label values in the category c_6, target class c_i, maximum probability and its corresponding class given by VGG-16 for the filtered ancestor $F_4(\mathcal{A})$ and the filtered adversarial images $F_4(\mathcal{D}_i)$ ($i \neq 6$).

i	1	2	3	4	5	6	7	8	9	10
$\mathbf{o}_{F_4(\mathcal{D}_i)}[6]$	1e-03	8e-05	1e-02	0.15	3e-04	0.99	3e-04	1e-02	1e-03	5e-05
$\mathbf{o}_{F_4(\mathcal{D}_i)}[i]$	0.94	0.99	0.97	0.84	0.99	0.99	0.99	0.98	0.99	0.99
$\max(\mathbf{o}_{F_4(\mathcal{D}_i)})$	0.94	0.99	0.97	0.84	0.99	0.99	0.99	0.98	0.99	0.99
$c_{\arg\max(\mathbf{o}_{F_4(\mathcal{D}_i)})}$	plane	car	bird	cat	deer	dog	frog	horse	ship	truck

Table 5. Label values in the category c_6, target class c_i, maximum probability and its corresponding class given by VGG-16 for the filtered ancestor $F_3 \circ F_4(\mathcal{A})$ and the filtered adversarial images $F_3 \circ F_4(\mathcal{D}_i)$ ($i \neq 6$).

i	1	2	3	4	5	6	7	8	9	10
$\mathbf{o}_{F_3\circ F_4(\mathcal{D}_i)}[6]$	0.17	0.99	0.75	0.11	8e-02	0.91	0.51	0.32	0.22	0.58
$\mathbf{o}_{F_3\circ F_4(\mathcal{D}_i)}[i]$	3e-05	1e-06	3e-03	0.88	3e-05	0.91	1e-04	8e-05	8e-06	7e-06
$\max(\mathbf{o}_{F_3\circ F_4(\mathcal{D}_i)})$	0.82	0.99	0.75	0.88	0.91	0.91	0.51	0.67	0.77	0.58
$c_{\arg\max(\mathbf{o}_{F_3\circ F_4(\mathcal{D}_i)})}$	cat	dog	dog	cat	cat	dog	dog	cat	cat	dog

the c_6 probabilities of the $F(\mathcal{D}_i)$ are very low, and there is a category $c \neq c_6$ of probability strictly larger. Only one adversarial image reverts to the *dog* class after being filtered (filter F_3, class 2). The noticeable predominance of the *cat*

class as the predicted category of the filtered images is likely due to the similarity in features of the CIFAR-10 *dog* and *cat* images that VGG-16 was trained on. Another reason for the inefficacy of filters F_1, F_2 and F_3 to protect VGG-16 for the *untargeted* scenario is that the c_6 probability of $F(\mathcal{A})$ is drastically reduced from the initial 0.9996387 of $\mathcal{A}$, which is undesired. Although adversarial for the *untargeted* scenario, these filtered images can not be considered adversarial for the *target* scenario. Still, almost all $F(\mathcal{D}_i)$ are classified in the same category *cat* as $F(\mathcal{A})$, hence follow $F(\mathcal{A})$'s pattern.

Filters F_4 and $F_3 \circ F_4$ do not significantly reduce the c_6 probability of $F(\mathcal{A})$ (Tables 4 and 5). The $F_4(\mathcal{D}_i)$s are classified in c_i with high confidence, and are definitively adversarial for the *untargeted* scenario. Moreover, they are all adversarial for the *target* scenario, while $F_4(\mathcal{A})$ simultaneously remains classified as *dog*. In that sense, our $\text{EA}_{L_2}^{\text{target}}$ attack is robust against the unsharp mask filter for the *target* and a fortiori the *untargeted* scenario.

The final filter $F_3 \circ F_4$ has a particular impact. First, $F_3 \circ F_4(\mathcal{A})$ remains classified as *dog*. Second, the c_6 probabilities of the $F_3 \circ F_4(\mathcal{D}_i)$s (Table 5) are either the largest or the second largest. Out of the 9 adversarial images, 4 reverted to the *dog* class after filtering, while the other 5 were classified as *cat*, with *dog* being the second most likely category. Hence the $F_3 \circ F_4$ combination of filters, which brings back a significant proportion of filtered images to the ancestor category, may render our EA-based attack less effective, not only for the *target*, but also for the *untargeted* scenario.

7 The variant $\text{EA}_{L_2}^{\text{target},F}$

Results of the previous section lead to the conception of $\text{EA}_{L_2}^{\text{target},F}$. This variant of $\text{EA}_{L_2}^{\text{target}}$ natively takes into account the goal to create adversarial images that remain adversarial for the *target* scenario once filtered, in addition to remaining close to the ancestor and being classified as belonging to a target category. The main modification is clearly in the fitness function :

$$fit_{L_2}^{F}(ind, g_p) = A(g_p, ind)(\mathbf{o}_{ind}[c_t] + \mathbf{o}_{F(ind)}[c_t]) - B(g_p, ind)L_2(ind, \mathcal{A}) \quad (4)$$

where the last component measures the probability that the individual filtered with F is classified as the target category. Since $F_3 \circ F_4$ is the only filter which can revert a significant proportion of the adversarial images to c_6, it makes sense to explore $\text{EA}_{L_2}^{\text{target},F_3 \circ F_4}$. For the sake of consistency, the range of pixel value modification is set to ± 3 as well for $\text{EA}_{L_2}^{\text{target},F_3 \circ F_4}$.

Running $\text{EA}_{L_2}^{\text{target},F_3 \circ F_4}$ for the target scenario on the same dog ancestor image as during the previous experiments leads to the $\mathcal{D}_i^{F_3 \circ F_4}$ adversarial images pictured in Figure 3.

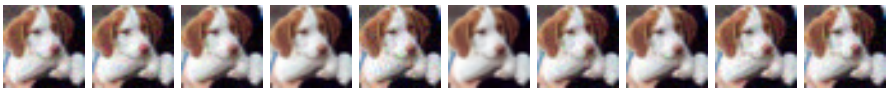


Fig. 3. Comparison of the ancestor $\mathcal{A}$ in the 6^{th} position with the adversarial images $\mathcal{D}_i^{F_3 \circ F_4} = EA_{L_2}^{\text{target}, F_3 \circ F_4}(\mathcal{A}, c_i)$ in the i^{th} position ($i \neq 6$, from left to right). VGG-16 classifies $\mathcal{D}_i^{F_3 \circ F_4}$ as belonging to the target category c_i with probability ≥ 0.95.

The ancestor image $\mathcal{A}$, and the adversarial images $\mathcal{D}_i^{F_3 \circ F_4} = \text{EA}_{L_2}^{\text{target}, F_3 \circ F_4}(\mathcal{A})$ ($i \neq 6$) pictured in Figure 3 are tested against the filters of section 5. Figure 4 shows the outcome of this process. More precisely, from left to right, the picture in the i^{th} position on the k^{th} row represent $F(\mathcal{D}_i^{F_3 \circ F_4})$ for $i \neq 6$ and $F(\mathcal{A})$ for $i = 6$, filtered with $F = F_k$ for $1 \leq k \leq 4$, and $F = F_3 \circ F_4$ for $k = 5$.

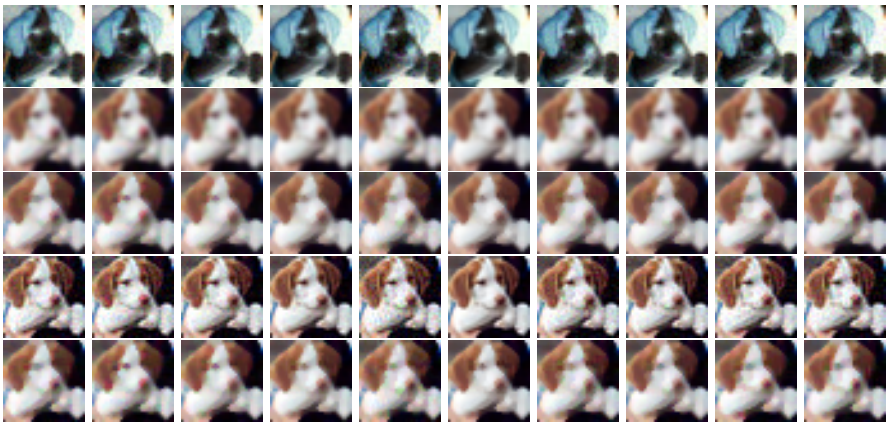


Fig. 4. Comparison of the impact of filters on the ancestor $\mathcal{A}$ and on the adversarial images $\mathcal{D}_i^{F_3 \circ F_4}$. The k^{th} row represents $F(\mathcal{A})$ (in 6^{th} position) and $F(\mathcal{D}_i^{F_3 \circ F_4})$ (in i^{th} position, $i \neq 6$), where $F = F_k$ for $1 \leq k \leq 4$, and $F = F_3 \circ F_4$ for $k = 5$.

These filtered images are given to VGG-16 for classification. Table 6 is produced with the filters F_1, F_2, F_3, while Table 7 is produced with F_4, and Table 8 with $F_3 \circ F_4$. Each Table has (groups of) rows showing the probability of the filtered images for the c_6 category, the target class c_i, the maximum probability and its corresponding class outputted by VGG-16, respectively. In all tables, we set $\mathcal{D}_6^{F_3 \circ F_4} = \mathcal{A}$ to ease the notations.

The first outcome of Tables 6, 7 and 8 is that the $\text{EA}_{L_2}^{\text{target}, F_3 \circ F_4}$ attack is robust against all individual filters considered for the *untargeted* scenario. This result is not surprising, since the F_k ($1 \leq k \leq 4$) filters were already not a good de-

Table 6. Label values in the category c_6, target class c_i, maximum probability $\max \mathbf{o} = \max\left(\mathbf{o}_{F\left(\mathcal{D}_i^{F_3\circ F_4}\right)}\right)$ and its corresponding class $c_{\max}$, where $c_{\max} = \arg\max\left(\mathbf{o}_{F\left(\mathcal{D}_i^{F_3\circ F_4}\right)}\right)$ given by VGG-16 for the filtered ancestor $F(\mathcal{A})$ and the filtered adversarial images $F(\mathcal{D}_i^{F_3\circ F_4})$ ($i \neq 6$) for $F = F_1$ (1st group of 5 rows), F_2 (2nd group) and F_3 (3rd group).

i	1	2	3	4	5	6	7	8	9	10
$\mathbf{o}_{F_1\left(\mathcal{D}_i^{F_3\circ F_4}\right)}[6]$	3e-03	2e-03	8e-04	1e-03	1e-03	1e-03	7e-03	0.11	1e-03	1e-04
$\mathbf{o}_{F_1\left(\mathcal{D}_i^{F_3\circ F_4}\right)}[i]$	2e-03	1e-04	7e-05	0.99	6e-04	1e-03	0.22	5e-05	1e-02	0.99
max $\mathbf{o}$	0.58	0.66	0.97	0.99	0.85	0.99	0.73	0.86	0.75	0.99
$c_{\max}$	frog	cat	cat	cat	frog	cat	cat	cat	frog	truck
i	1	2	3	4	5	6	7	8	9	10
$\mathbf{o}_{F_2\left(\mathcal{D}_i^{F_3\circ F_4}\right)}[6]$	2e-04	3e-04	6e-03	1e-04	1e-03	3e-04	6e-03	1e-03	3e-04	1e-03
$\mathbf{o}_{F_2\left(\mathcal{D}_i^{F_3\circ F_4}\right)}[i]$	3e-05	3e-06	2e-04	0.99	4e-05	3e-04	2e-05	5e-05	4e-06	3e-06
max $\mathbf{o}$	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
$c_{\max}$	cat	cat	cat	cat	cat	cat	cat	cat	cat	cat
i	1	2	3	4	5	6	7	8	9	10
$\mathbf{o}_{F_3\left(\mathcal{D}_i^{F_3\circ F_4}\right)}[6]$	9e-04	4e-02	1e-03	3e-04	4e-02	0.26	4e-02	4e-03	1e-02	0.10
$\mathbf{o}_{F_3\left(\mathcal{D}_i^{F_3\circ F_4}\right)}[i]$	0.84	0.17	0.99	0.99	0.88	0.26	0.84	0.98	1e-02	8e-02
max $\mathbf{o}$	0.84	0.72	0.99	0.99	0.88	0.73	0.84	0.98	0.97	0.80
$c_{\max}$	plane	cat	bird	cat	deer	cat	frog	horse	cat	cat

Table 7. Label values in the category c_6, target class c_i, maximum probability $\max \mathbf{o} = \max\left(\mathbf{o}_{F_4\left(\mathcal{D}_i^{F_3\circ F_4}\right)}\right)$ and its corresponding class $c_{\max}$, where $c_{\max} = \arg\max\left(\mathbf{o}_{F_4\left(\mathcal{D}_i^{F_3\circ F_4}\right)}\right)$ given by VGG-16 for the filtered ancestor $F_4(\mathcal{A})$ and the filtered adversarial $F_4(\mathcal{D}_i^{F_3\circ F_4})$ ($i \neq 6$).

i	1	2	3	4	5	6	7	8	9	10
$\mathbf{o}_{F_4\left(\mathcal{D}_i^{F_3\circ F_4}\right)}[6]$	4e-04	6e-05	4e-03	6e-02	4e-04	0.99	3e-04	5e-03	1e-04	1e-04
$\mathbf{o}_{F_4\left(\mathcal{D}_i^{F_3\circ F_4}\right)}[i]$	0.96	0.99	0.99	0.93	0.97	0.99	0.99	0.99	0.99	0.98
max $\mathbf{o}$	0.96	0.99	0.99	0.93	0.97	0.99	0.99	0.99	0.99	0.98
$c_{\max}$	plane	car	bird	cat	deer	dog	frog	horse	ship	truck

fense for the CNN when using adversarial images created with $\mathrm{EA}_{L_2}^{\mathrm{target}}$. Hence, although the $\mathcal{D}_i^{F_3\circ F_4}$ adversarial images were only designed to circumvent filter $F_3 \circ F_4$, they are also robust against filters F_1 to F_4.

Table 8. Label values in the category c_6, target class c_i, maximum probability $\max\ \mathbf{o} = \max\left(\mathbf{o}_{F_3\circ F_4\left(\mathcal{D}_i^{F_3\circ F_4}\right)}\right)$ and its corresponding class $c_{\max}$, where $c_{\max} = \arg\max\left(\mathbf{o}_{F_3\circ F_4\left(\mathcal{D}_i^{F_3\circ F_4}\right)}\right)$ given by VGG-16 for the filtered ancestor $F_3 \circ F_4(\mathcal{A})$ and the filtered adversarial $F_3 \circ F_4(\mathcal{D}_i^{F_3\circ F_4})$ $(i \neq 6)$.

i	1	2	3	4	5	6	7	8	9	10
$\mathbf{o}_{F_3\circ F_4(\mathcal{D}_i^{F_3\circ F_4})}[6]$	6e-05	2e-04	1e-04	2e-04	1e-04	0.91	1e-04	1e-04	5e-04	8e-05
$\mathbf{o}_{F_3\circ F_4(\mathcal{D}_i^{F_3\circ F_4})}[i]$	0.99	0.99	0.99	0.99	0.99	0.91	0.99	0.99	0.99	0.99
$\max\ \mathbf{o}$	0.99	0.99	0.99	0.99	0.99	0.91	0.99	0.99	0.99	0.99
$c_{\max}$	plane	car	bird	cat	deer	dog	frog	horse	ship	truck

None of the $\mathrm{EA}_{L_2}^{\mathrm{target},F_3\circ F_4}$ adversarial images filtered with F_1 to F_3 were classified as *dog*. Nonetheless, the predicted categories vary between filters. While the Gaussian filter predicts *cat* for all images, the inverse and median filters also predict the target class for some images. By simply observing the images of Figure 4, one can see that the Gaussian filter produces the blurriest images, hence reducing not only the object details, but also the noise added by the EA. This might explain why the predictions corresponding to the Gaussian-filtered images do not contain much information related to the target class.

As is the case with $\mathrm{EA}_{L_2}^{\mathrm{target}}$, the adversarial images produced by $\mathrm{EA}_{L_2}^{\mathrm{target},F_3\circ F_4}$, once filtered with F_4 are all classified as the target class, hence being adversarial for the *target* scenario. This is probably due to the fact that, while the unsharp mask increases the contrast of the *dog* object, it also intensifies the noise added by the EA, which directs the image towards the adversarial class. The noisy aspect of the filtered images can be seen in the 4^{th} row of Figure 4.

Finally, Table 8, corresponding to $F_3 \circ F_4$, shows a clear improvement compared to Table 5, as $\mathrm{EA}_{L_2}^{\mathrm{target},F_3\circ F_4}$ produces images that are no longer vulnerable to this filter for the *target* scenario.

Although $\mathrm{EA}_{L_2}^{\mathrm{target},F_3\circ F_4}$ provides an increase in robustness compared to $\mathrm{EA}_{L_2}^{\mathrm{target}}$, it is also interesting to compare the time efficiency of the two algorithms. Table 9 gives the number of generations and the amount of time required by $\mathrm{EA}_{L_2}^{\mathrm{target},F_3\circ F_4}$ to create the adversarial images $\mathcal{D}_i^{F_3\circ F_4}$.

Comparing Tables 2 and 9 shows that the higher robustness of $\mathrm{EA}_{L_2}^{\mathrm{target},F_3\circ F_4}$ requires both a longer execution time per generation and more generations. Firstly, $\mathrm{EA}_{L_2}^{\mathrm{target},F_3\circ F_4}$ must satisfy not two, but three conditions. More generations are needed to have not only the plain adversarial images with a target class probability higher than 0.95, but also its filtered version. Secondly, the drop in the average number of generations computed per second is due to the additional filtering step in $\mathrm{EA}_{L_2}^{\mathrm{target},F_3\circ F_4}$.

Table 9. For $1 \leq i \leq 10$, $i \neq 6$, the 2^{nd} row specifies the number of generations required by $\text{EA}_{L_2}^{\text{target},F_3\circ F_4}$ to create the adversarial images $\mathcal{D}_i^{F_3\circ F_4}$. The 3^{rd} row gives the total execution time, while the 4^{th} row gives the average number of generations per second.

i	1	2	3	4	5	7	8	9	10
# of generations	1495	1164	717	194	1812	820	541	1262	1330
total time (in seconds)	159.2	136.04	80.23	22.85	199.08	82.98	53.26	133.33	139.01
# of generations/second	9.39	8.56	8.94	8.49	9.1	9.88	10.16	9.47	9.57

8 Conclusion

This ongoing work addresses the issue of the robustness against filters of adversarial images fooling CNNs. By considering VGG-16 trained at image classification on CIFAR-10, adversarial images $\mathcal{D}_i$ created by the $\text{EA}_{L_2}^{\text{target}}$ evolutionary algorithm performed on the *dog* instantiation of the "target scenario" with ancestor image $\mathcal{A}$, and specific filters F_1, F_2, F_3, F_4, we first prove that the $F(\mathcal{D}_i)$ images, while no longer adversarial for the *target* scenario for $F = F_1, F_2, F_3$, are not only adversarial for the *untargeted* scenario, but foremost follow $F(\mathcal{A})$'s pattern for these three individual filters. We also prove that the $F_4(\mathcal{D}_i)$ images remain adversarial for the *target* scenario, and a fortiori for the *untargeted* scenario, while $F_4(\mathcal{A})$ is classified in the same category as $\mathcal{A}$. Hence, by essentially following the ancestor's behavior towards these four individual filters, the adversarial images $\mathcal{D}_i$ acquire an additional similarity with $\mathcal{A}$. These results confort their adversarial profile, and enhance the robustness and quality of the $\text{EA}_{L_2}^{\text{target}}$ attack.

We secondly show that the $F_3 \circ F_4$ combination of filters brings back a significant proportion of filtered images in the ancestor category, while $F_3 \circ F_4(\mathcal{A})$ is classified in the same category as $\mathcal{A}$. Since this may render $\text{EA}_{L_2}^{\text{target}}$ less effective, not only for the *target*, but also for the *untargeted* scenario, a third outcome of this work is the construction of the variant $\text{EA}_{L_2}^{\text{target},F}$ of the evolutionary algorithm, that natively takes into account the robustness of adversarial images against a generic filter F. We instantiate this EA on $F = F_3 \circ F_4$. The produced images $\mathcal{D}_i^{F_3\circ F_4}$ are adversarial against $F_3 \circ F_4$ for the *target* scenario, but also essentially against F_4 and F_3. They are to a large extent adversarial against F_1 and F_2 for the *untargeted* scenario as well. The performance of this new EA is compared to that of $\text{EA}_{L_2}^{\text{target}}$, showing that the provided robustness comes at the cost of more, as well as longer generations.

These preliminary results lead to a series of future work. We intend to extend our methodology to all images of Figure 10 of [4], beyond the sole *dog* series of the present article, potentially with more than one ancestor and more than one descendant in each given category. We also plan to assess the efficiency of the creation of adversarial images with $\text{EA}_{L_2}^{\text{target},F}$ depending on which ancestor im-

age is provided as input: either the original $\mathcal{A}$ or the processed adversarial image $\mathrm{EA}_{L_2}^{\mathrm{target}}(\mathcal{A}, c_i)$. An important direction would be to consider larger images, such as those of ImageNet [7], since the small 32×32 images of this study are naturally *grainy*. Finally, one could consider to address these issues with a different choice of the measure of proximity between images, for instance with SSIM [15] instead of L_2, and with different scenarios, for instance the *flat scenario* of [4].

References

1. Bernard, N., Leprévost, F.: Evolutionary algorithms for convolutional neural network visualisation. In: High Performance Computing – 5th Latin American Conference, 2018 (Bucaramanga, Colombia, Sep 23-28, 2018). Communications in Computer and Information Science, vol. 979, pp. 18–32. Springer, Heidelberg (2018)
2. Bernard, N., Leprévost, F.: How evolutionary algorithms and information hiding deceive machines and humans for image recognition: A research program. In: Proceedings of the OLA'2019 International Conference on Optimization and Learning (Bangkok, Thailand, Jan 29-31, 2019). pp. 12–15. Springer, Heidelberg (2019)
3. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000), `https://www.drdobbs.com/open-source/the-opencv-library/184404319`
4. Chitic, R., Bernard, N., Leprévost, F.: Evolutionary algorithms deceive humans and machines at image classification: An extended proof of concept on two scenarios. Journal of Information and telecommunication (2020), `http://dx.doi.org/10.1080/24751839.2020.1829388`
5. Chitic, R., Bernard, N., Leprévost, F.: A proof of concept to deceive humans and machines at image classification with evolutionary algorithms. In: Intelligent Information and Database Systems, 12th Asian Conference, ACIIDS 2020 (Phuket, Thailand, March 23-26, 2020). pp. 467–480. Springer, Heidelberg (2020)
6. Chollet, F.: Keras. GitHub code repository (2015-2018), `https://github.com/fchollet/keras`
7. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: The imagenet image database (2009), `http://image-net.org`
8. Geifman, Y.: cifar-vgg (2018), `https://github.com/geifmany/cifar-vgg`
9. Krizhevsky, A., et al.: The CIFAR datasets (2009), `https://www.cs.toronto.edu/~kriz/cifar.html`
10. Lim, J.S.: Two-Dimensional Signal and Image Processing. Prentice Hall (1989)
11. Oliphant, T.E.: A guide to NumPy. Trelgol Publishing USA (2006)
12. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR **abs/1409.1556** (2014), `http://arxiv.org/abs/1409.1556`
13. Su, J., Vargas, D.V., Sakurai, K.: One pixel attack for fooling deep neural networks. CoRR **abs/1710.08864** (2017)
14. Sung Kim, R.C.: Applications of Convolution in Image Processing with MATLAB. University of Washington (2013), `https://www.semanticscholar.org/paper/Applications-of-Convolution-in-Image-Processing-Casper/391f4dc0567f671b0718f80834fdc1e83a9fd54b`
15. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing **13**(4), 600–612. (2004)

Simultaneous Lot Sizing and Scheduling Problem in Tire Industry : Eligibility Constraint Sensitivity Analysis

C. Koch[1,2], T. Arbaoui[1], Y. Ouazene[1], F. Yalaoui[1], H. De Brunier[2], N. Jaunet[2]

[1] Industrial Systems Optimization Laboratory, University of Technology of Troyes, 10004 Troyes, France
cyril.koch@utt.fr, taha.arbaoui@utt.fr, yassine.ouazene@utt.fr, farouk.yalaoui@utt.fr
[2] Manufacture Française des Pneumatiques Michelin

1 Introduction

Lot sizing problems have been studied widely in the literature over the last decades. The expected output of lot sizing is to give a complete picture over a planning horizon of how many parts to produce at each period and how many pieces to carry in inventory. It takes its origin in the well-known Economic Order Quantity (EOQ) model [1] under the assumption of single item, constant demand and infinite planning horizon. Since then, numerous researchers have built more realistic models to tackle real world problems. A literature review of this problem have been done and the problem positioning is highlighted in Figure 1.

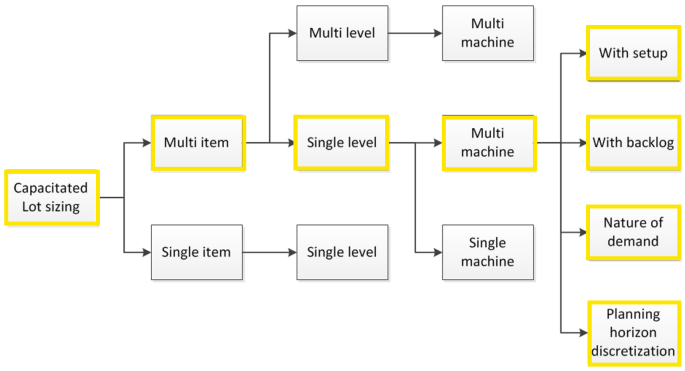


Fig. 1. Problem Positioning

2 Tire Manufacturing Process

The whole production process can be divided in 5 major sub-processes (See Fig.3) [11].

First, a banbury mixer creates a homogeneous rubber material based on natural rubber, carbon black, resins and other chemicals. All these components are mixed in a banbury mixer to obtain the homogeneous rubber material which is the basic raw material to build a tire, in the form of thin layers. In addition to the rubber compounds, two more raw materials are necessary to build a tire : textile reinforcements and steel wires, ensuring its rigidity and geometry. The second sub-process is the production of semi-finite products. Two different technologies are used. The first one is the extruding and calendaring process. Different types of rubber material from the first sub-process are warmed up, mixed and then pushed through dies of the appropriate shape. The second one is the profiling and cutting process. Textiles are cut at the right dimensions thanks to fabric bias cutter to provide crown plies and fabric used in the casing plies (among other pieces of fabric). Then the task of building the tire begins. This third sub-process is called the assembly sub-process. The tire building machine also needs a resource called a drum (a rotative cylinder) on which the different parts of the tire are assembled. The end result is called a “green” or uncured

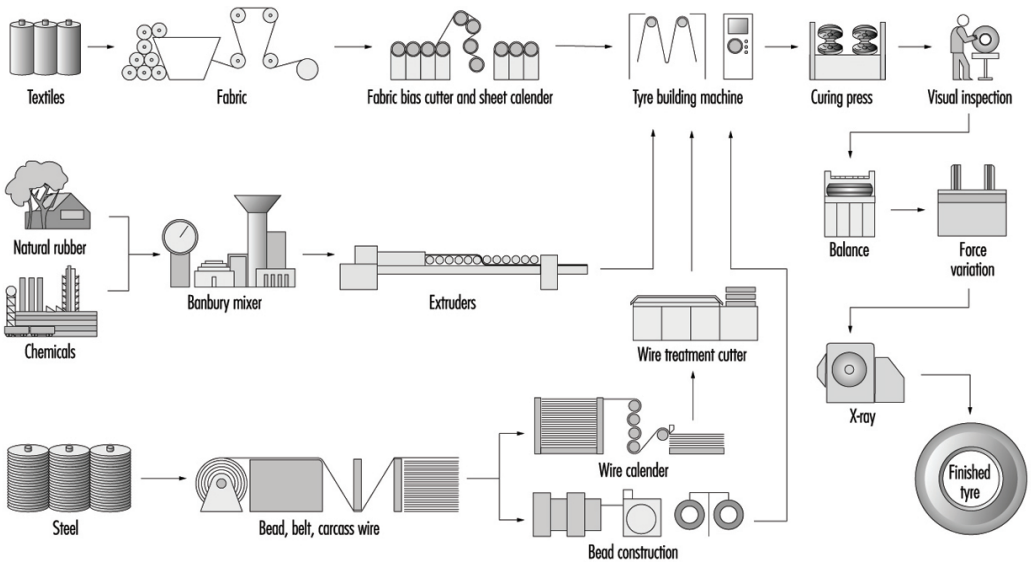


Fig. 2. Tire Manufacturing Process

tire. Fourth is the curing and vulcanizing process. A tire-specific mold is placed into the curing machine, which is also called a heater or curing press, and the green tire is put into that mold. The increase of temperature causes the sulfur contained in the rubber compound to bound with the rubber molecules. This is what we call vulcanization. The rubber is then transformed from a plastic to an elastic state. When ejected from the mold and after cooling the tire has taken on his final shape and properties. Finally inspection and finishing operations remains before the tire is stored in the warehouse.

3 Problem description and Contribution

We focus on the curing sub-process production planning problem. It is the most important stage as it has been identified as the bottleneck by the company, requires consequent setup times and is highly restricted by tire - heater eligibility matrix. The plant management face a complex production planning problem with a wide portfolio of tires to be produced on unrelated parallel machines with numerous eligibility constraints. Furthermore market trends tend to dilute demand signal on more and more different references of tire. Thus the portfolio is getting wider and wider to match customer expectations and makes even more difficult the planning problem of the company. The production is based on a make-to-stock inventory policy, so that the inventory level stays between a minimum and a maximum level calculated to prevent shortage and keep Working Capital Requirement to a minimum.

During the curing process the green tire is put into a mold that provides a specific pattern for the tire. Each mold is tire-specific: it can be used for exactly one type of tire. For some tire references several molds are available, though for most tires there is only one mold. Every mold can be placed in several heaters, respecting the tire-heater eligibility matrix. Still each heater can contain at most one mold at a time. The curing time depends on the tire produced and the heater used. The heaters capacity therefore links together different tire references that compete for the same resource - available time of a given heater where the molds can be placed in. Except for the first and the last period of the production campaign, tires are produced in a continuous run and production is always done at full capacity. This type of production is often referred as “all-or-nothing” production. Also, only one type of tire can be cured in a heater within one period. Thus our problem is classified as a small-bucket lot sizing problem.

In [2] a MIP formulation of the presented problem is proposed. The contribution of this study is to explore the sensitivity of one particular input parameter which is very critical and erratic

according to the company : the tire-heater eligibility constraint. We use the same solving method as in [2] and generate different scenarios for the tire-heater eligibility matrix with the company supply chain manager's help. This study allows the company to use our modelling as a decision support system and evaluate different scenarios for future projects.

4 Conclusion

The originality of the problem studied in this paper is the application of a simultaneous lot-sizing and scheduling problem in tire industry with specific constraints. This work is a continuation of the previous work presented in [2]. It allows the company to make a step further in his journey to industry 4.0 and an agile and digital manufacturing system.

References

1. F. W. Harris : How many parts to make at once. Factory, The magazine of management, vol.10 (1913)
2. C. Koch, T. Arbaoui, Y. Ouazene, F. Yalaoui, A. De Wulf, N .Jaunet : A dedicated lot sizing problem in tire industry. IFAC World Congress 2020, Germany (2020)
3. L. Lasdon, RC. Terjung : An efficient algorithm for multi-item scheduling. Operations research, vol. 19 , (1971)
4. Z. Degraeve, L. Schrage : A tire production scheduling system for Bridgestone/Firestone Off-The-Road. Operations Research, vol 45, (1997)
5. R. Jans, Z. Degraeve : An industrial extension of the discrete lot-sizing and scheduling problem. IIE transactions, vol.36, (2004)
6. S. de Araujo, M. Arenales, A. Clark : Joint rolling-horizon scheduling of materials processing and lot-sizing with sequence-dependent setups. Journal of Heuristics, vol. 13, (2007)
7. N. Absi, S. Kedad-Sidhoum : MIP-based heuristics for multi-item capacitated lot-sizing problem with setup times and shortage costs. RAIRO-Operations Research, vol.41, (2007)
8. S. Helber, F. Sahling : A fix-and-optimize approach for the multi-level capacitated lot sizing problem. International Journal of Production Economics, vol 123, (2010).
9. C. Toledo, M. da Silva Arantes, M. Hossomi, P. França, K. Akartunalı: A relax-and-fix with fix-and-optimize heuristic applied to multi-level lot-sizing problems. Journal of Heuristics, vol. 21, (2015)
10. M. Ghirardi, A. Amerio : Matheuristics for the lot sizing problem with back-ordering, setup carry-overs, and non-identical machines. Computers & Industrial Engineering, vol. 127, (2019)
11. A. Boudha, S. Kumar, Srimannarayana, F.Koyan: Tyre Manufacturing Processes Seminar Report. Indian Institute of Technology, Bombay (2011)

Iterated Local Search with Neighbourhood Reduction for the Pickups and Deliveries Problem Arising in Retail Industry*

Hanyu Gu, Yefei Zhang, and Yakov Zinder

School of Mathematical and Physical Sciences, University of Technology Sydney
15 Broadway Ultimo NSW 2007, Australia
Hanyu.Gu@uts.edu.au, Ye.f.zhang@student.uts.edu.au, yakov.zinder@uts.edu.au

Abstract. The paper studies a vehicle routing problem with simultaneous pickups and deliveries that arises in the retail sector, which considers a heterogeneous fleet of vehicles, time windows of the demands, practical restrictions on the drivers and a roster specifying the order of vehicle loading at the depot. The high competition in this industry requires that a viable optimisation approach must achieve a good balance of solution time, quality and robustness. In this paper, a novel iterated local search algorithm is proposed which dynamically reduces the neighbourhood so that only the most promising moves are considered. The results of computational experiments on real-world data demonstrate the high efficiency of the presented optimisation procedure in terms of computation time, stability of the optimisation procedure and solution quality.

Keywords: Vehicle routing problem · Iterated local search · Neighbourhood reduction

1 Introduction

This paper considers a vehicle routing problem with simultaneous pickups and deliveries (VRPSPD) which arises in the retail sector. The features of this problem include: a heterogeneous fleet of vehicles, time window for pickups and deliveries, open routes, restriction on shift length and loading roster at the depot. In spite of the practical importance of these features, few applications in the literature considered all of them simultaneously [9], [6]. Furthermore, the objective of the considered problem is to maximise the number of allocations which is practically essential, but is rarely considered in the literature [9].

Since VRPSPD is NP-hard in the strong sense [3], the majority of the publications in this topic present various heuristics and metaheuristics [9], [6]. In practice, a scheduler expects to produce a good solution within a short time limit, typically no more than one minute. In contrast, most research in the literature focuses more on solution quality. In this paper, an iterated local search [7] based optimisation algorithm is presented to achieve a satisfactory balance between solution quality and computation time.

The iterated local search algorithm (ILS) has been widely used to solve combinatorial optimisation problem [7]. It iteratively generates a sequence of local optimums. At each iteration a local search is performed on a problem-specific neighbourhood structure. A perturbation mechanism is employed to avoid local optimum and expand the search space. By allowing infeasible solutions in

* Supported by an Australian Government Research Training Program Scholarship

the designed neighborhood structure [12], [5], ILS has been demonstrated to be much faster than the state-of-the-art for solving the Workforce Scheduling and Routing Problem, which is, from a practical application viewpoint, similar to the studied problem in this paper.

The most time-consuming component in ILS is the evaluation of potential moves in the local search procedure due to the large size of the neighbourhood of the current solution. It is also critical to select proper moves to increase the probability of converging to the global optimum. This paper presents a mechanism to reduce the neighbourhood dynamically, which makes the move evaluation faster, and at the same time direct search in the most promising part of the neighbourhood.

Contributions of this paper include

- development of a MIP model for a VRPSPD problem with many features from the retail sector
- introduction of neighbourhood reduction to speedup the ILS algorithm
- computational studies on real-world data

The remainder of the paper is organised as follows. Section 2 presents the problem formulation. Section 3 describes the proposed iterated local search. Section 4 presents the results for the computational experiments. Section 5 concludes the paper.

2 Problem statement

The considered vehicle routing problem can be stated as follows. Let $G = \{L, A\}$ be a directed graph, where the set of vertices $L = \{0\} \cup C$ and $C = \{1, 2, ..., l\}$, the set of arcs $A = A_D \cup A_C$ and $A_D = \{(0, i) | i \in C\}$, $A_C = \{(i, j) | i \neq j, \forall i, j \in C\}$. Vertex 0 represents the depot and the remaining vertices represent the l customers. Each arc $(i, j) \in A$ has an associated travel time $t_{i,j}$.

The delivery to customer $i \in C$ is characterised by its weight w_i^d and volume v_i^d. The pickup from customer $j \in C$ is characterised by its weight w_i^p and volume v_i^p. For customer $i \in C$, the associated time window $[a_i, b_i]$ indicates the earliest and latest time when the driver can start the corresponding services, and let $p_i > 0$ be the service time required for the driver to complete the service.

Let T be the set of all vehicles. Each vehicle $i \in T$ is differed by its weight capacity W_i and volume capacity V_i. All vehicles $i \in T$ depart from the same single depot and are not required to return to depot after serving all allocated customers. The driver in each vehicle $i \in T$ finishes the shift after serving the last allocated customers. Due to the loading capacity of the depot, each vehicle $i \in T$ arrives at the depot at the specified starting time r_i with loading time δ_i. Furthermore, there exists an upper bound S_i on the shift time of the driver in vehicle $i \in T$, which is the length of time interval between the time when driver starts loading at the depot and the time when driver finishes the service of the last allocated customers.

Each customer $i \in C$ can be served only once, but not all vehicles are capable to serve certain customers. In this paper, two types of vehicles are considered, i.e., the one-man vehicle $T' \subset T$ and the two-men vehicle $T'' \subset T$. The customers are also classified as either one-man customer $C' \subset C$, or two-men customer $C'' \subset C$. The one-man customer can be served by all vehicles, while two-men customer can only be served by two-men vehicles.

The objective is to maximise the total number of allocated customer services while respecting all the constraints on drivers, vehicles and the depot.

Let x_{jk}^i be a binary variable indicating if customer j is the immediate predecessor of customer k in the route of vehicle i; η_j^i be a binary variable indicating if customer j is allocated to vehicle i; γ_j^i

be a binary variable indicating if customer j is the first customer to visit after vehicle i departing from the depot; θ_j^i be a binary variable indicating if customer j is the last customer in the route of vehicle i. Denote the time when driver in vehicle i starts serving customer k by s_k^i; the weight of the vehicle when leaving customer j by y_j ; the volume of the vehicle when leaving customer j by z_j. The considered problem can be formulated as follows:

$$J = \max \sum_{i \in T} \sum_{j \in C} \eta_j^i \tag{1}$$

subject to

$$\sum_{i \in T} \eta_j^i \leq 1, \quad \forall j \in C \tag{2}$$

$$\sum_{j \in C} \gamma_j^i \leq 1, \quad \forall i \in T \tag{3}$$

$$\gamma_j^i + \sum_{k \in C} x_{k,j}^i = \eta_j^i, \quad \forall i \in T, j \in C \tag{4}$$

$$\theta_j^i + \sum_{k \in C} x_{j,k}^i = \eta_j^i, \quad \forall i \in T, j \in C \tag{5}$$

$$a_j \leq s_j^i \leq b_j, \quad \forall j \in C, i \in T \tag{6}$$

$$(r_i + \delta_i + t_{0,k})\gamma_k^i \leq s_k^i, \quad \forall i \in T, k \in C \tag{7}$$

$$s_j^i + (p_j + t_{j,k})x_{j,k}^i + (a_k - b_j)(1 - x_{j,k}^i) \leq s_k^i, \quad \forall i \in T, \forall (j,k) \in A_C \tag{8}$$

$$p_j + s_j^i - r_i - (p_j + b_j - r_i)(1 - \theta_j^i) \leq S_i, \quad \forall j \in C, i \in T \tag{9}$$

$$\sum_{k \in C} w_k^d \eta_k^i \leq W_i, \quad \forall i \in T \tag{10}$$

$$y_k \leq W_i + (\max_{e \in T} W_e - W_i)(1 - \eta_k^i), \quad \forall i \in T, k \in C \tag{11}$$

$$\sum_{j \in C} w_j^d \eta_j^i - w_k^d + w_k^p - (\max_{e \in T} W_e - w_k^d + w_k^p)(1 - \gamma_k^i) \leq y_k, \quad \forall i \in T, k \in C \tag{12}$$

$$y_j - w_k^d + w_k^p - (\max_{e \in T} W_e - w_k^d + w_k^p)(1 - x_{j,k}^i) \leq y_k, \quad \forall i \in T, \forall (j,k) \in A_C \tag{13}$$

$$\sum_{k \in C} v_k^d \eta_k^i \leq V_i, \quad \forall i \in T \tag{14}$$

$$z_k \leq V_i + (\max_{e \in T} V_e - V_i)(1 - \eta_k^i), \quad \forall i \in T, k \in C \tag{15}$$

$$\sum_{j \in C} v_j^d \eta_j^i - v_k^d + v_k^p - (\max_{e \in T} V_e - v_k^d + v_k^p)(1 - \gamma_k^i) \leq z_k, \quad \forall i \in T, k \in C \tag{16}$$

$$z_j - v_k^d + v_k^p - (\max_{e \in T} V_e - v_k^d + v_k^p)(1 - x_{j,k}^i) \leq z_k, \quad \forall i \in T, \forall (j,k) \in A_C \tag{17}$$

$$\sum_{i \in T'} \sum_{k \in C''} \eta_k^i = 0 \tag{18}$$

$$x_{j,k}^i \in \{0,1\}, \quad \forall \{j,k\} \in A_C, i \in T \tag{19}$$

$$\eta_j^i \in \{0,1\}, \quad \forall i \in T, j \in C \tag{20}$$
$$\theta_j^i \in \{0,1\}, \quad i \in T, j \in C \tag{21}$$
$$y_j \geq 0, \quad \forall j \in C \tag{22}$$
$$z_j \geq 0, \quad \forall j \in C \tag{23}$$

where (7) and (3) guarantee that a vehicle either stays at the depot or visits exactly one customer; (4) and (5) ensure that each customer must have an immediate successor from the same route except for the last customer; together with (2) ensure that a customer is visited by at most one vehicle; the arrival time, loading time at the depot, travelling time between vertices, the time window are taken into account by (7), (8) and (6)-(8) respectively; the shift length, weight capacity, volume capacity are enforced by (9), (10)-(13), and (14)-(17) respectively; (6) and (8) eliminate subtours by virtue of $p_i > 0$.

3 ILS with neighbourhood reduction

A critical component of ILS is the design of proper neighbourhood structures. It has been demonstrated by many publications that permitting infeasible solutions in local search together with the use of an augmented objective function can significantly boost the performance of the meta-heuristics in the field of vehicle routing problem [1, 2, 8, 12]. The neighbourhood structures considered in this paper are defined by the commonly used edge exchange operators, which allows the violation of the time window, shift length, weight and volume capacity constraints. However, the algorithm presented in this paper reduces the size of the neighbourhood by only allowing moves that lead to more allocations than the best known feasible solution. To be specific, let $J(s)$ be the number of allocated customers in a solution s which can be infeasible; $H(s, O)$ be the neighbourhood of a solution s induced by an edge exchange operator O permitting infeasible solution. The corresponding reduced neighbourhood is defined as

$$\widehat{H}(s, O) = \{s' \in H(s, O) | J(s') > J(s^*)\}$$

where s^* be the best known feasible solution. In the studied problem, it is permitted to have customers not allocated. Therefore, feasible solutions can be efficiently generated using simple heuristics (see Section 3.1 for more details). It should be noted that the reduced neighbourhood is dynamic since s^* can be updated in the iterative process of ILS. Since ILS can quickly find good solutions, the size of the reduced neighbourhood becomes significantly smaller after just a few iterations, which leads to faster convergence of the algorithm. Also, the solution process can be more stable because only solutions with more allocations are considered in the local search process.

The paper considers two edge exchange operators

- inter-route swap O_1: exchanges a sequence of up to two consecutive customers in a route with a sequence of up to two consecutive customers in another route; exchanges a sequence of up to two consecutive customers in a route with at most one unallocated customer;
- intra-route swap O_2: extract at most two consecutive customers from a route and insert it into a different position of the same route; reverse the order of a sequence of consecutive customers in the route.

It should be noted that O_2 cannot increase the number of allocated customers. Therefore, it is used mainly for repair infeasibility in the local search procedure.

In the local search procedure, the solution in the reduced neighbourhood is evaluated based on the augmented objective function

$$f(s) = J(s) - \alpha \times TW(s) - \beta \times WD(s) - \sigma \times Weight(s) - \psi \times Volume(s) \tag{24}$$

where $TW(s)$, $WD(s)$, $Weight(s)$, $Volume(s)$ are the total violation for constraints on time window, working duration, weight, volume corresponding to s and $\alpha, \beta, \sigma, \psi$ are non-negative weights for $TW(s)$, $WD(s)$, $Weight(s)$, $Volume(s)$. Furthermore, $TW(s)$, $WD(s)$, $Weight(s)$, $Volume(s)$ are computed by the technique used in [8, 10, 11].

The details of the local search procedure based on the reduced neighbourhood (NRS) are given in Algorithm 1.

Algorithm 1 NRS(s)

1: **while** TRUE **do**
2: **if** $\widehat{H}(s, O_1) == \emptyset$ **then** return s^* **end if**
3: $s' = s$
4: $s = argmax_x\{f(x)|x \in \widehat{H}(s, O_1)\}$
5: **if** $f(s') < f(s)$ **then**
6: $s' = s$
7: **if** s is feasible **then** $s^* = s$ **end if**
8: **else**
9: Break
10: **end if**
11: **end while**
12: $s = s'$
13: **if** $\widehat{H}(s, O_2) \neq \emptyset$ **then** $s = argmax_x\{f(x)|x \in \widehat{H}(s, O_2)\}$ **end if**
14: **if** $f(s') < f(s)$ **then**
15: **if** s if feasible **then** $s^* = s$ **end if**
16: **else**
17: $s = s'$
18: **end if**
19: return s

In this pseudocode, the input solution s is permitted to be infeasible. The edge exchange operator O_1 is applied until a local optimum is found under the reduced neighbourhood $\widehat{H}(s, O_1)$. Since the size of the reduced neighbourhood is related to the number of allocations in the current global optimum s^*, s^* is updated whenever a new global optimum is found (line 7 and 15). It should be noted that $\widehat{H}(s, O_1)$ is empty only if the current s^* has all customers allocated, which is also the global optimum. Following the strategy in [12], local search based on the edge exchange operator O_2 is performed for at most one iteration after the local optimum under O_1 is found (line 13). In line 13, $\widehat{H}(s, O_2)$ is empty only if s is a feasible solution. The output of NRS is either the input solution, or a solution with more allocated customers and higher augmented objective function value.

Algorithm 2 ILS with neighbourhood reduction (ILS-NR)

1: $s' =$ INITIAL()
2: $s^* = s'$
3: $t = J(s^*)$
4: $h = 1$
5: **while** s^* has unallocated customers **and** $h \leq M$ **do**
6: $\alpha = \beta = \sigma = \psi = 1$
7: $e = 1$
8: **repeat**
9: $\bar{s} = s'$
10: $s' =$NRS(s')
11: **if** $f(\bar{s}) \neq f(s')$ **then** Update $\alpha, \beta, \sigma, \psi$ **end if**
12: $e++$
13: **until** $f(\bar{s}) == f(s')$ **or** s^* has unallocated customers **or** $e > E$
14: **if** $J(s^*) > t$ **then**
15: $t = J(s^*)$
16: $h = 1$
17: **end if**
18: $s' =$ PERTURB(h)
19: $h++$
20: **end while**
21: **return** s^*

The ILS with neighbourhood reduction (ILS-NR) is now presented in Algorithm 2. It begins with the INITIAL procedure which generates a feasible solution for the problem (line 1). The details of INITIAL is given in Section 3.1. This solution is also the current best known solution s^* (line 2). It should be noted that the current best known s^* is a global variable and may be updated inside the NRS and PERTURB procedure.

After the call of the INITIAL procedure, the WHILE loop (line 5 - 20) is executed if the current best known solution s^* has at least one unallocated customer. The WHILE loop terminates if the current best known solution allocates all customers, or counter h exceeds M which is a parameter. Each iteration of the WHILE loop (line 5 - 20) attempts to find a solution with more allocations than the current best known solution s^* applying the local search procedure (line 8 - 13).

Each iteration of the local search (line 8 - 13) is an applications of NRS which aims to find a solution with a better value of the augmented objective function (24). The penalties for violation of corresponding constraints are updated to force the convergence to feasible solutions. Following [1, 2, 12], at the beginning of each iteration of the local search (line 8 - line 13), the initial value for weights $\alpha, \beta, \sigma, \psi$ in the augmented objective function (24) are set to one (line 6). If NRS returns an improving solution, a weight is multiplied by $1 + \Delta$ if the corresponding constraint has a positive violation; otherwise the weight is divided by $1 + \Delta$. Δ is a parameter that controls the strength of the adjustment. This weight updating mechanism is effective in producing feasible solutions, which explains why O_2 is only applied for one iteration in NRS (line 10). Local search terminates if either the NRS procedure fails to obtain a solution with better value of the augmented objective function (24), the current best known solution s^* allocates all customers or the count e exceeds E which is a parameter.

3.1 INITIAL procedure

The INITIAL procedure is a sweep heuristic [4] that constructs a feasible solution for the problem. First a list of customers is constructed based on the geographic coordinates of the customers. Then the customers are inserted to a route one by one until no customer can be inserted, in which case a new route is constructed. Since one-man vehicles can only serve one-man customers, whereas two-men vehicles can serve all-types of customers, the procedure constructs the routes for one-man vehicles first, then followed by the routes for two-men vehicles. When inserting a customer into the route, the procedure chooses the insertion position that respects all the constraints and gives the smallest increase in travel time. The procedure terminates until either no customers can be inserted into the vehicle's route, or all customers have been allocated.

3.2 PERTURB procedure

The PERTURB procedure expands the search space by randomly perturbing the current best solution s^*. An unallocated customer is randomly chosen, and then inserted into a position among the routes which gives the largest value of (24) when $\alpha = \beta = \sigma = \psi = 1$. Then, two randomly selected sequences of consecutive customers are swapped between two randomly selected routes. This random swap will be performed multiple times which depends on the counter h in the pseudocode for the ILS-NR (Algorithm 2). To be specific, the number of swaps starts from one and increases by one each time when counter h in Algorithm 2 increase. The current best solution s^* may also be updated in this process.

4 Computational study

This section presents the results of computational experiments aimed at the evaluation of the performance of ILS-NR. A total of 60 instances were provided by a transportation company working in the retail industry. Each instance represents the real-world situation on a particular day. The travel time from the location of the depot to each customer, and the travel time between the location of each customer are specified by a symmetric matrix. The time when driver arrives at the depot is specified by a roster and each driver can work for a maximum of 10 hours. ILS-NR is implemented in c++, and compiled with g++ O3. The following settings are used throughout the experiments [12]:

- The maximum permissible number of consecutive unsuccessful attempts to improve the current best known solution (the parameter M in Algorithm 2) is computed as $|C| + \lambda|T|)$, where C is the set of all customers, T is the set of all vehicles, $\lambda = 10$.
- The maximum number of exchange operations in the perturbation is five.
- The parameter Δ for adjusting the weights (Section 3) is 0.5.

In addition, the maximum permissible iterations for local search (the paramter E in Algorithm 2) is 100. All computational experiments are conducted on a computer with Intel Xeon CPU E5-2697 v3 2.60GHz and 8GB RAM.

We first compare the performance of ILS-NR with CPLEX which solves the IP model in Section 2. Furthermore, we test the performance of CPLEX when the best solution from ILS-NR is used as a warm start. Both CPLEX and CPLEX with warm start have a time limit of 6 hours and memory limit of 7.5GB RAM. Version 12.10 of CPLEX is used for all the tests. In Table 1, the groups

Table 1: CPLEX vs ILS-NR vs CPLEX warm start

			CPLEX			**ILS-NR**				**CPLEX warm start**			
Instances	$\|C\|$	$\|T\|$	Obj	Gap(%)	Time(s)	Avg.	Max.	#Max	Time(s)	Input	Obj	Gap(%)	Time(s)
M-2017-07-23	30	3	27	10.88	9112	**28.00**	**28**	**30**	**0.13**	**28**	**28**	**0.00**	**3711**
M-2017-07-24	26	2	21	9.52	14257	21.93	22	28	0.10	22	22	9.09	21600
M-2017-07-25	14	2	**14**	**0.00**	**1**	**14.00**	**14**	**30**	**0.00**	**14**	**14**	**0.00**	**0**
M-2017-10-08	28	2	24	12.50	10288	24.63	26	1	0.17	26	26	3.85	21600
M-2017-10-09	22	2	21	4.76	21600	21.00	21	30	0.03	21	21	4.76	21600
M-2017-10-10	22	2	17	11.76	21600	17.00	17	30	0.07	17	17	11.76	11304
M-2017-10-16	34	2	26	19.99	21600	26.10	27	3	0.30	27	27	15.55	21600
M-2017-10-17	24	2	21	9.52	21600	21.30	22	10	0.10	22	22	4.55	21600
M-2017-10-21	34	2	24	29.17	21600	26.87	28	1	0.33	28	28	12.75	21600
M-2017-10-24	17	2	**17**	**0.00**	3	**16.90**	**17**	**27**	**0.00**	**17**	**17**	**0.00**	**0**
M-2017-10-30	37	2	27	29.63	21600	28.90	30	1	0.40	30	30	16.85	21600
M-2017-12-22	72	7	66	9.09	21600	69.43	70	13	3.47	70	70	2.86	21600
M-2017-12-23	70	5	59	18.64	21600	65.80	67	3	3.47	67	67	4.48	21600
M-2017-12-24	70	5	50	40.00	21600	57.47	59	1	3.20	59	59	18.64	21600
M-2017-12-25	70	5	52	25.00	21600	57.50	59	1	3.40	59	59	10.17	21600
R-2017-07-23	47	5	**47**	**0.00**	463	**47.00**	**47**	**30**	**0.00**	**47**	**47**	**0.00**	**1**
R-2017-07-24	65	3	48	14.58	21600	52.13	53	5	2.60	53	53	3.77	21600
R-2017-07-25	43	4	**42**	**0.00**	19472	**42.00**	**42**	**30**	**0.50**	**42**	**42**	**0.00**	**0**
R-2017-10-08	88	6	80	8.71	21600	**85.60**	**86**	**18**	**7.40**	**86**	**86**	**0.00**	**18582**
R-2017-10-09	63	4	54	5.56	21600	55.27	56	8	2.37	56	56	1.79	21600
R-2017-10-10	44	5	**44**	**0.00**	593	**44.00**	**44**	**30**	**0.00**	**44**	**44**	**0.00**	**0**
R-2017-10-16	72	5	64	9.37	21600	68.67	69	20	3.50	69	69	1.77	21600
R-2017-10-17	37	4	34	8.82	5084	35.93	36	28	0.37	36	36	2.78	21600
R-2017-10-21	60	5	55	5.45	21600	**58.00**	**58**	**30**	**1.80**	**58**	**58**	**0.00**	**1**
R-2017-10-24	53	6	**53**	**0.00**	**790**	**53.00**	**53**	**30**	**0.00**	**53**	**53**	**0.00**	**1**
R-2017-10-30	71	7	69	2.90	21600	**70.67**	**71**	**20**	**1.43**	**71**	**71**	**0.00**	**1**
R-2017-12-12	52	4	49	6.12	21600	**51.43**	**52**	**18**	**0.67**	**52**	**52**	**0.00**	**1**
R-2017-12-19	52	4	46	10.87	21600	**50.47**	**51**	**14**	**1.20**	**51**	**51**	**0.00**	**0**
R-2017-12-22	62	4	53	15.09	21600	57.03	58	3	2.47	58	58	5.17	21600
R-2017-12-23	70	5	63	9.52	21600	67.73	68	22	3.23	68	68	1.47	21600
R-2017-12-24	70	5	56	10.71	21600	**60.70**	**62**	**1**	**3.27**	**62**	**62**	**0.00**	**2**
R-2017-12-25	70	5	65	7.69	21600	**69.77**	**70**	**23**	**0.93**	**70**	**70**	**0.00**	**1**
T-2017-07-23	64	5	63	1.59	21600	**64.00**	**64**	**30**	**0.00**	**64**	**64**	**0.00**	**1**
T-2017-07-24	70	5	67	2.99	21600	**69.00**	**69**	**30**	**2.63**	**69**	**69**	**0.00**	**1**
T-2017-07-25	57	4	55	3.64	21600	**56.77**	**57**	**23**	**0.57**	**57**	**57**	**0.00**	**0**
T-2017-10-08	65	8	**65**	**0.00**	**3834**	**65.00**	**65**	**30**	**0.00**	**65**	**65**	**0.00**	**1**
T-2017-10-09	43	7	**43**	**0.00**	**31**	**43.00**	**43**	**30**	**0.00**	**43**	**43**	**0.00**	**1**
T-2017-10-10	46	5	**46**	**0.00**	**675**	**46.00**	**46**	**30**	**0.00**	**46**	**46**	**0.00**	**0**
T-2017-10-16	63	7	**63**	**0.00**	**6631**	**63.00**	**63**	**30**	**0.00**	**63**	**63**	**0.00**	**2**
T-2017-10-17	56	4	49	12.24	21600	52.53	53	16	1.43	53	53	3.77	13380
T-2017-10-21	76	4	58	8.62	21600	61.93	62	28	4.23	62	62	1.61	21600
T-2017-10-24	62	4	52	10.05	21600	55.33	56	10	2.30	56	56	1.79	21600
T-2017-10-30	36	5	**36**	**0.00**	**13**	**36.00**	**36**	**30**	**0.00**	**36**	**36**	**0.00**	**0**
T-2017-12-12	63	7	**63**	**0.00**	**1345**	**63.00**	**63**	**30**	**0.00**	**63**	**63**	**0.00**	**2**
T-2017-12-19	54	5	**54**	**0.00**	**923**	**54.00**	**54**	**30**	**0.00**	**54**	**54**	**0.00**	**1**
T-2017-12-22	91	7	75	18.67	21600	**88.73**	**89**	**22**	**7.47**	**89**	**89**	**0.00**	**7**
T-2017-12-23	70	5	63	11.11	21600	**69.93**	**70**	**28**	**0.67**	**70**	**70**	**0.00**	**1**
T-2017-12-24	70	5	63	9.52	21600	67.10	68	3	3.77	68	68	1.47	21600
T-2017-12-25	70	5	64	9.37	21600	68.53	69	16	3.23	69	69	1.45	21600
T-2017-12-26	70	5	65	4.62	21600	**67.97**	**68**	**29**	**3.07**	**68**	**68**	**0.00**	**1**
A-2017-10-16	100	4	53	30.19	21600	61.73	63	2	4.90	63	63	9.52	21600
A-2017-12-22	100	7	76	22.37	21600	82.70	84	2	9.20	84	84	10.72	21600
B-2017-10-08	100	6	72	15.16	21600	79.60	80	18	8.70	80	80	3.65	21600
B-2017-10-16	100	5	71	15.49	21600	78.93	80	6	8.03	80	80	2.50	21600
B-2017-10-30	100	7	81	13.58	21600	86.93	88	5	9.40	88	88	4.55	21600
B-2017-12-22	100	4	58	41.79	21600	67.97	70	1	7.90	70	70	17.49	21600
C-2017-07-24	100	5	86	9.30	21600	92.83	93	25	8.77	93	93	1.65	21600
C-2017-10-16	100	7	96	2.08	21600	**97.97**	**98**	**29**	**6.97**	**98**	**98**	**0.00**	**5**
C-2017-10-21	100	4	67	20.47	21600	75.40	76	12	9.67	76	76	6.06	21600
C-2017-12-22	100	7	89	11.24	21600	**97.87**	**99**	**1**	**10.20**	**99**	**99**	**0.00**	**5**

Table 2: ILS vs ILS-NR

			ILS			**ILS-NR**					
Instances	$\|C\|$	$\|T\|$	Avg.	Max	Time(s)	Avg.	%	Max.	%	Time(s)	%
M-2017-07-23	30	3	28.00	28	0.17	28.00	0.00	28	0.00	**0.13**	**20.00**
M-2017-07-24	26	2	21.57	22	0.13	**21.93**	**-1.70**	22	0.00	**0.10**	**25.00**
M-2017-07-25	14	2	14.00	14	0.00	14.00	0.00	14	0.00	0.00	0.00
M-2017-10-08	28	2	24.13	25	0.17	**24.63**	**-2.07**	**26**	**-4.00**	0.17	0.00
M-2017-10-09	22	2	21.00	21	0.07	21.00	0.00	21	0.00	**0.03**	**50.00**
M-2017-10-10	22	2	16.97	17	0.03	**17.00**	**-0.20**	17	0.00	0.07	-100.00
M-2017-10-16	34	2	25.83	26	0.33	**26.10**	**-1.03**	**27**	**-3.85**	**0.30**	**10.00**
M-2017-10-17	24	2	21.03	22	0.10	**21.30**	**-1.27**	22	0.00	0.10	0.00
M-2017-10-21	34	2	26.23	28	0.27	**26.87**	**-2.41**	28	0.00	0.33	-25.00
M-2017-10-24	17	2	17.00	17	0.03	16.90	0.59	17	0.00	**0.00**	**100.00**
M-2017-10-30	37	2	28.47	29	0.40	**28.90**	**-1.52**	**30**	**-3.45**	0.40	0.00
M-2017-12-22	72	7	69.30	70	4.37	**69.43**	**-0.19**	70	0.00	**3.47**	**20.61**
M-2017-12-23	70	5	65.70	67	4.33	**65.80**	**-0.15**	67	0.00	**3.47**	**20.00**
M-2017-12-24	70	5	57.47	59	3.87	57.47	0.00	59	0.00	**3.20**	**17.24**
M-2017-12-25	70	5	57.37	58	4.00	**57.50**	**-0.23**	**59**	**-1.72**	**3.40**	**15.00**
R-2017-07-23	47	5	47.00	47	0.00	47.00	0.00	47	0.00	0.00	0.00
R-2017-07-24	65	3	51.80	52	2.93	**52.13**	**-0.64**	**53**	**-1.92**	**2.60**	**11.36**
R-2017-07-25	43	4	42.00	42	0.67	42.00	0.00	42	0.00	**0.50**	**25.00**
R-2017-10-08	88	6	85.50	86	16.10	**85.60**	**-0.12**	86	0.00	**7.40**	**54.04**
R-2017-10-09	63	4	55.07	56	2.90	**55.27**	**-0.36**	56	0.00	**2.37**	**18.39**
R-2017-10-10	44	5	44.00	44	0.00	44.00	0.00	44	0.00	0.00	0.00
R-2017-10-16	72	5	68.90	70	4.73	68.67	0.34	69	1.43	**3.50**	**26.06**
R-2017-10-17	37	4	36.00	36	0.43	35.93	0.19	36	0.00	**0.37**	**15.38**
R-2017-10-21	60	5	58.00	58	2.30	58.00	0.00	58	0.00	**1.80**	**21.74**
R-2017-10-24	53	6	53.00	53	0.00	53.00	0.00	53	0.00	0.00	0.00
R-2017-10-30	71	7	70.77	71	1.33	70.67	0.14	71	0.00	1.43	-7.50
R-2017-12-12	52	4	51.10	52	1.10	**51.43**	**-0.65**	52	0.00	**0.67**	**39.39**
R-2017-12-19	52	4	50.50	51	1.47	50.47	0.07	51	0.00	**1.20**	**18.18**
R-2017-12-22	62	4	56.67	58	9.67	**57.03**	**-0.65**	58	0.00	**2.47**	**74.48**
R-2017-12-23	70	5	67.77	68	4.07	67.73	0.05	68	0.00	**3.23**	**20.49**
R-2017-12-24	70	5	60.73	61	3.73	60.70	0.05	**62**	**-1.64**	**3.27**	**12.50**
R-2017-12-25	70	5	69.83	70	0.87	69.77	0.10	70	0.00	0.93	-7.69
T-2017-07-23	64	5	64.00	64	0.00	64.00	0.00	64	0.00	0.00	0.00
T-2017-07-24	70	5	69.00	69	3.20	69.00	0.00	69	0.00	**2.63**	**17.71**
T-2017-07-25	57	4	56.47	57	0.93	**56.77**	**-0.53**	57	0.00	**0.57**	**39.29**
T-2017-10-08	65	8	65.00	65	0.00	65.00	0.00	65	0.00	0.00	0.00
T-2017-10-09	43	7	43.00	43	0.00	43.00	0.00	43	0.00	0.00	0.00
T-2017-10-10	46	5	46.00	46	0.00	46.00	0.00	46	0.00	0.00	0.00
T-2017-10-16	63	7	63.00	63	0.00	63.00	0.00	63	0.00	0.00	0.00
T-2017-10-17	56	4	52.47	53	1.63	**52.53**	**-0.13**	53	0.00	**1.43**	**12.24**
T-2017-10-21	76	4	61.30	62	4.37	**61.93**	**-1.03**	62	0.00	**4.23**	**3.05**
T-2017-10-24	62	4	55.03	56	2.60	**55.33**	**-0.55**	56	0.00	**2.30**	**11.54**
T-2017-10-30	36	5	36.00	36	0.00	36.00	0.00	36	0.00	0.00	0.00
T-2017-12-12	63	7	63.00	63	0.00	63.00	0.00	63	0.00	0.00	0.00
T-2017-12-19	54	5	54.00	54	0.00	54.00	0.00	54	0.00	0.00	0.00
T-2017-12-22	91	7	88.97	89	15.00	88.73	0.26	89	0.00	**7.47**	**50.22**
T-2017-12-23	70	5	69.50	70	2.17	**69.93**	**-0.62**	70	0.00	**0.67**	**69.23**
T-2017-12-24	70	5	66.90	68	4.03	**67.10**	**-0.30**	68	0.00	**3.77**	**6.61**
T-2017-12-25	70	5	68.17	69	3.97	**68.53**	**-0.54**	69	0.00	**3.23**	**18.49**
T-2017-12-26	70	5	67.87	68	12.83	**67.97**	**-0.15**	68	0.00	**3.07**	**76.10**
A-2017-10-16	100	4	61.23	62	5.97	**61.73**	**-0.82**	**63**	**-1.61**	**4.90**	**17.88**
A-2017-12-22	100	7	82.63	84	10.90	**82.70**	**-0.08**	84	0.00	**9.20**	**15.60**
B-2017-10-08	100	6	79.33	81	16.80	**79.60**	**-0.34**	80	1.23	**8.70**	**48.21**
B-2017-10-16	100	5	78.17	80	9.20	**78.93**	**-0.98**	80	0.00	**8.03**	**12.68**
B-2017-10-30	100	7	86.47	88	17.07	**86.93**	**-0.54**	88	0.00	**9.40**	**44.92**
B-2017-12-22	100	4	67.20	68	8.90	**67.97**	**-1.14**	**70**	**-2.94**	**7.90**	**11.24**
C-2017-07-24	100	5	92.50	93	12.00	**92.83**	**-0.36**	93	0.00	**8.77**	**26.94**
C-2017-10-16	100	7	98.00	98	10.33	97.97	0.03	98	0.00	**6.97**	**32.58**
C-2017-10-21	100	4	75.20	76	16.60	**75.40**	**-0.27**	76	0.00	**9.67**	**41.77**
C-2017-12-22	100	7	98.20	99	13.73	97.87	0.34	99	0.00	**10.20**	**25.73**
Average			56.19	56.70	4.05	56.33	-0.32	56.82	-0.31	2.67	17.61

titled “CPLEX” and “CPLEX warm start” contain results obtained by CPLEX and CPLEX with warm start. In these groups, the objective value, optimality gap, computational time are displayed in columns titled “Obj”, “Gap(%)” and “Time(s)”. The column titled “Input” in group “CPLEX warm start” displays the objective value of the warm start solution. ILS-NR is run 30 times on each instance with the average objective value (“Avg.”), best objective value (“Max.”), number of runs the best objective value is obtained (“#Max”) and computation time (“Time(s)”) being reported under the group “ILS-NR”.

According to Table 1, CPLEX can prove optimality for 13 instances. With warm start, CPLEX can prove optimality for another 16 instances with significantly reduced CPU time. Among these 29 instances proved optimality by CPLEX, ILS-NS can find optimal solutions with high frequency (#Max) within 10.2 seconds. For 45 out of 60 instances, the average objective values produced by ILS-NR are better than the objective values produced by CPLEX which has a time limit of 6 hours.

To demonstrate the effectiveness of the neighbourhood reduction, Table 2 presents the computational results for ILS-NR and ILS without neighbourhood reduction. The performance of ILS-NR was measured against ILS by the percentage difference

$$\frac{X_{ILS} - X_{ILS-NR}}{X_{ILS}} \times 100 \tag{25}$$

where X can either be the average objective value (column “Avg.”), best found objective value (“Max”) or CPU time (“Time”); X_{ILS-NR} is the value obtained by ILS-NR and X_{ILS} is the value obtained by ILS. Therefore, a negative percentage difference indicates that ILS-NR is better with respect to the average objective value and best found objective value, while a positive percentage difference indicates that ILS-NR is better with respect to CPU time. For readers’ convenience, the superior results produced by ILS-NR are shown in bold.

In Table 2, ILS-NR is faster than ILS on 41 out of 60 instances with an average difference of 17.61%, which clearly demonstrates the improvement on computation time due to neighbourhood reduction. In terms of stability, the average objective value produced by the ILS-NR outperforms the average objective value produced by ILS on 49 instances.

5 Conclusion

This paper considers a practical vehicle routing problem with simultaneous pickups and deliveries which arises in the retail sector. A novel neighbourhood reduction technique is introduced to enhance the performance of the state-of-the-art iterated local search algorithm. Computational experiments carried out on a set of real-world instances demonstrate the superior performance of the proposed algorithm in terms of computational time, solution quality and stability. The advantage of the proposed algorithm is more conspicuous for time-critical applications given the longest computation time among the test instances is just 10.2 seconds.

Bibliography

[1] Cordeau, J.F., Gendreau, M., Laporte, G.: A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks: An International Journal **30**(2), 105–119 (1997)

[2] Cordeau, J.F., Laporte, G., Mercier, A.: A unified tabu search heuristic for vehicle routing problems with time windows. Journal of the Operational research society **52**(8), 928–936 (2001)

[3] Garey, M.R., Johnson, D.S.: Computers and intractability, vol. 174. freeman San Francisco (1979)

[4] Gillett, B.E., Miller, L.R.: A heuristic algorithm for the vehicle-dispatch problem. Operations research **22**(2), 340–349 (1974)

[5] Gu, H., Zhang, Y., Zinder, Y.: Lagrangian relaxation in iterated local search for the workforce scheduling and routing problem. In: International Symposium on Experimental Algorithms. pp. 527–540. Springer (2019)

[6] Koç, Ç., Laporte, G., Tükenmez, İ.: A review on vehicle routing with simultaneous pickup and delivery. Computers & Operations Research p. 104987 (2020)

[7] Lourenço, H.R., Martin, O.C., Stützle, T.: Iterated local search: Framework and applications. In: Handbook of metaheuristics, pp. 129–168. Springer (2019)

[8] Nagata, Y., Bräysy, O., Dullaert, W.: A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. Computers & operations research **37**(4), 724–737 (2010)

[9] Parragh, S.N., Doerner, K.F., Hartl, R.F.: A survey on pickup and delivery problems. Journal für Betriebswirtschaft **58**(2), 81–117 (2008)

[10] Vidal, T., Crainic, T.G., Gendreau, M., Prins, C.: A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. Computers & operations research **40**(1), 475–489 (2013)

[11] Vidal, T., Crainic, T.G., Gendreau, M., Prins, C.: A unified solution framework for multi-attribute vehicle routing problems. European Journal of Operational Research **234**(3), 658–673 (2014)

[12] Xie, F., Potts, C.N., Bektaş, T.: Iterated local search for workforce scheduling and routing problems. Journal of Heuristics **23**(6), 471–500 (2017)

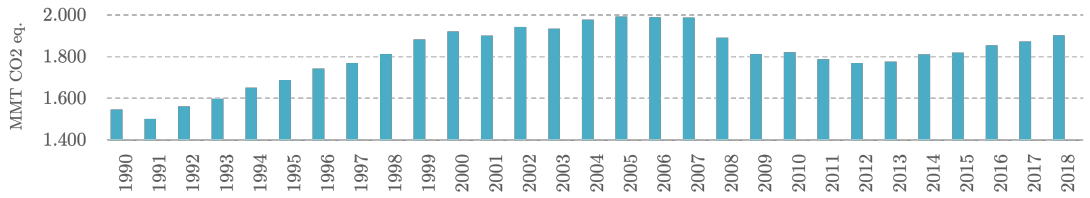
Multi-Objective Optimization Model for Sustainable Urban Transportation Ensuring the Quality of Service

D. Peña[1], P. Ruiz[1], A. Tchernykh[2] and B. Dorronsoro[1]

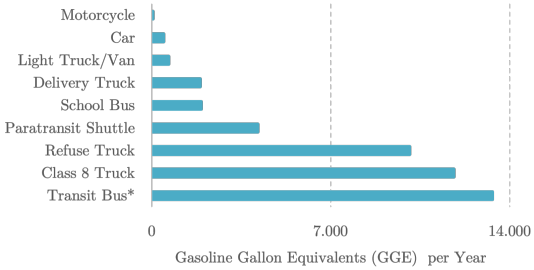
[1] Escuela Superior de Ingeniería, Universidad de Cádiz, Av. Universidad de Cádiz, 10, 11519, Campus Universitario de Puerto Real, Cádiz, Spain
{david.morales, patricia.ruiz, bernabe.dorronsoro}@uca.es
[2] CICESE Research Center, Carretera Ensenada-Tijuana 3918, 22860, Ensenada, Baja California, México
chernykh@cicese.mx

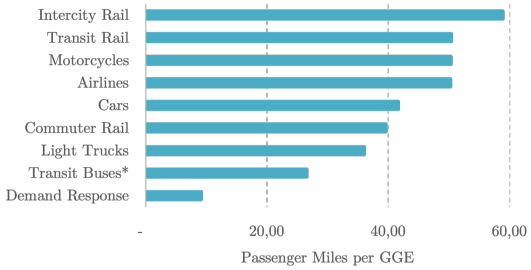
1 Introduction

Global CO_2 emissions reached a record of 33.5 $GtCO_2$ in 2018, driven by the population's robust growth and different economic activities. In combination with the transportation sector, power generation accounted for more than two-thirds of total emissions; these two sectors are responsible for almost all global growth since 2010 [5]. For example, in the U.S. the greenhouse gas (GHG) emissions from the transportation sector increased by 23.3% from 1990 to 2018 (Fig. 1(a)) and represent over 28% of the total emissions (transportation sector supposes 24% of worldwide emissions) [4]. The Intergovernmental Panel on Climate Change (IPCC) report concluded that GHG emissions must be reduced by 50% to 85% before 2050 [6], thereby avoiding many of the worst impacts of climate change. Besides, nowadays, 54% of the world's population lives in urban areas instead of 30% in 1950. The projection of the United Nations for 2050 claims that the number of city dwellers will increase to 68% [8], which implies an increase in demand for all sectors that interact with people straightly. This situation will lead to many significant challenges, such as overpopulation that encourage the request for a limited supply of essential resources like h2O, combustible material, electric energy, or services, including health care, security, education, and transportation. Reducing GHG emissions from transportation will likely require a broad range of strategies, as increasing vehicles' efficiency, lowering the carbon content of fuels, and optimizing routes. Public transportation can contribute to the solution. Fig. 1(b) shows the average yearly fuel use per vehicle for the principal land transportation sector vehicles, measured in Gasoline Gallon Equivalents (GGEs). GGEs represents the amount of fuel (e.g., diesel or electricity) it takes to equal the energy held in a gasoline gallon. The vehicle's mean distance traveled per year and their fuel economy are the main factors that affect the annual fuel use on average (Fig. 1(c)). Buses for public transport are relatively inefficient because of the high number of trips per day and the stop-and-go drive cycles, spending more fuel on average than any other kind of vehicle. Regarding fuel economy (FE), transit buses are not quite efficient due to their passenger load rates being less than their capacity. Vehicles like UBER or taxis (demand response) are the least efficient because they use combustible to pick up the passengers in reply to users' calls or requests at mobile applications [4]. Public transport vehicles like transit buses that travel with low ridership have been shown to produce more per-passenger-km emissions than private automobiles [2]. Transit bus scheduling has conflicting objectives; for example, the governments strive to reduce the negative environmental impact while users want better service, more vehicles, and lower fares. Those objectives are tackled in this paper. Our multi-objective formulation is based on the importance of minimizing the number of buses transiting with low ridership, which are wasting fuel and increasing air pollution. Simultaneously, from the passengers' perspective, reducing the waiting times and moving in safe and comfortable vehicles is desirable. This study focuses on the timetable optimization that assigns different types of buses (i.e., extra capacity, energy consumption, weight, etc.) to cover a defined route, assessing its effects on GHG emissions when the number of transit buses is reduced, as well as the consequences in terms of degradation of quality of service (QoS). Additionally, this study attempts to highlight the benefits of determining a timetable combining different types of vehicles inspired by the notion suggested by Potter [7] to enhance vehicle utilization preventing an overloaded scenario. We focused on tactical planning and operating emissions of public transport (without considering the total life cycle GHG emissions, e.g., infrastructure, maintenance, and operation), applying a heuristic approach based on Multi-Objective Evolutionary Algorithms (MOEA) to estimate a solution to the Multiple Vehicle-Types Timetabling Problem (MVTTP).



(a) Greenhouse Gas (GHG) Emissions by Transportation Sector





(b)Yearly average of Fuel Use by Vehicle Type (c) Fuel Economy by Travel Mode

Fig. 1. (a) GHG emissions in million metric tons carbon dioxide equivalent (MMT CO2 eq.). (b) The average annual fuel used to represent the quantity of fuel with the same amount of energy contained in a gallon of gasoline. (c) *Transit buses are not efficient at their current ridership rates or load factor (i.e., number of passengers on board), where, on average, a given bus is less than 25% full. [4] [2]

2 The multi-objective Multiple Vehicle-Types Timetabling Problem

In our work, we split the timetabling phase, proposed by Ceder [1] along with three other fundamental phases for a transit planning problem (network route design, timetable development, vehicle and crew scheduling) into two tasks: frequency decision and timetable design. Accordingly, we assume that the route map and its bus stops are established. Moreover, information about route conditions like route elevation or average speed is available. The passenger demand is provided for each time-period in every stop (e.g., the number of passengers in the route).
MVTTP models a complex problem where a set of transit buses of different types is assigned to cover a predefined route. The multi-objective problem (MOP) is to find an appropriate distribution of multiple bus-types to minimize two essential objectives simultaneously: I) reduce GHG emissions related to the fuel consumption of the vehicle used for a trip, and II) minimize the unsatisfied user demand. Minimizing the function of the GHG emissions, associated with the negative environmental impact, can also contribute to reducing the operational costs (e.g., fuel, driver salary, or repairs) and enhance the traffic flow due to the reduction of needless bus circulation and unoccupied vehicles wasting fuel and polluting. QoS, represented by the unsatisfied user demand function, should guarantee a good experience in terms of safety, availability, and comfort. The latter is usually associated with the load factor and the waiting time (i.e., delay in boarding the vehicle).

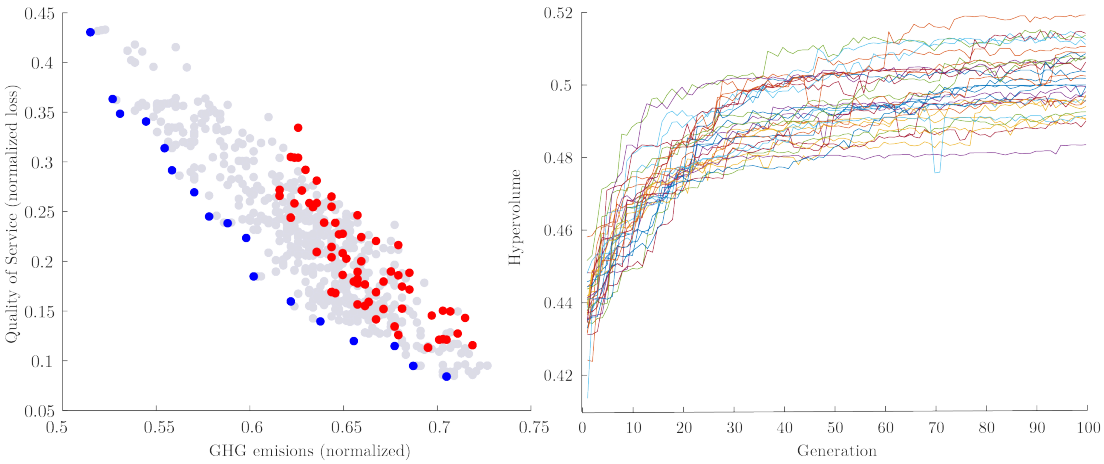
2.1 Encoding and solution representation

Evolutionary Algorithms (EAs) are efficient search methods belonging to the family of heuristics. Inspired by the evolutionary process of living beings, they use the principle of natural selection to solve problems. EAs are capable of performing robust and multi-directional searches in complex spaces working with a population of potential solutions, where each individual is the coding of a possible solution to the problem (chromosome). One virtue of such algorithm is that they require little information about the problem and generally obtain relevant approximate solutions in reasonable execution times. In this work, we are looking for the "best set of timetables" that simultaneously minimize the conflicting objective functions described before, using a heterogeneous fleet to cover every trip in a specific period where each possible timetable is an individual (solution). Solutions are encoded as arrays of integers (genotype), representing the bus type assigned to cover different

trips of the route (phenotype). We propose the utilization of zeros to mark the time-periods finish and the start of the next one. The order of departures is specified in the sequence.

3 Results and discussion

For these preliminary results, we used a well-known MOEA proposed by Deb et al. called NSGA-II [3], employing three crossover and two mutation operators with different probabilities (10% to 100% in steps of 10), we define the algorithm parameters based on 600 different combinations. The final settings as defined as follows: population size of 100 individuals (solutions), 100 generations per run, 10% of probability of mutation, and a crossover probability of 100%. The results obtained from the experiments carried out in this work indicate that the MOEA used provides a set of solutions that represent diverse timetables for a possible real-world scenario, with a static ridership. Those timetables assign different kinds of transit buses to cover the daily demand on the route, which can reduce the environmental impact while maintaining an acceptable QoS. The decision-maker could adopt the solution set to choose the best one according to governmental restrictions for example. Furthermore, we develop an extensive experimentation, showing a stable hypervolume behaviour and acceptable performance in terms of diversity and approximation to the *Pareto front* (Fig. 2).



(a) *Pareto front* approximation behaviour (b) Hypervolume performance

Fig. 2. (a) The evolution process applying the crossover and mutation methods on the initial population (red dots) attaining the *Pareto front* approximation for the last generation (blue dots), (b) Hypervolume performance for 30 independent runs of NSGA-II.

Acknowledgements

This work was supported by the Spanish Ministerio de Ciencia, Innovación y Universidades and the ERDF for the support provided under contract RTI2018-100754-B-I00 (iSUN project).

References

1. Ceder, A. Public Transit Planning and Operation. Public Transit Planning and Operation (2016).
2. Davis, S., Williams, S., & Boundy, R. Transportation Energy Data Book (35th ed.). U.S. Department of Energy (2016). http://cta.ornl.gov/data/tedb35/Edition35_Full_Doc.pdf
3. Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6, 182–197 (2002).
4. Environmental Protection Agency. Greenhouse Gas Inventory Data Explorer (2020).
5. IEA: CO2 Emissions from Fuel Combustion: Overview 2020. IEA, France (2020)
6. IPCC, Global Warming of 1.5°C.An IPCC Special Report on the impacts of global warming of 1.5°C above pre-industrial levels and related global greenhouse gas emission pathways, in the context of strengthening the global response to the threat of climate change, sustainable development, and efforts to eradicate poverty (2018)
7. Potter, S. Transport Energy and Emissions: Urban Public Transport. in Handbook of Transport and the Environment (Handbooks in Transport) 247–262 (2003).
8. United Nations, D. World urbanization prospects 2018: highlights (ST/ESA/SER. A/421) (2019)

Solving QAP with Auto-parameterization in Parallel Hybrid Metaheuristics

Jonathan Duque[1], Danny Múnera[1], Daniel Diaz[2], and Salvador Abreu[3]

[1] Facultad de Ingeniería, Universidad de Antioquia, Medellín Colombia
jonathan.duque@udea.edu.co danny.munera@udea.edu.co
[2] CRI, Université Paris 1, Paris France
daniel.diaz@univ-paris1.fr
[3] NOVA-LINCS, Universidade Évora, Évora Portugal
spa@uevora.pt

Abstract. The Quadratic Assignment Problem (QAP) is one of the most challenging combinatorial optimization problems with many real-life applications. Currently, the best solvers are based on hybrid and parallel metaheuristics, which are actually highly complex and parametric methods. Finding the best set of tuning parameters for such methods is a tedious and error-prone task. In this paper, we propose a strategy for auto-parameterization of QAP solvers. We show evidence that auto-parameterization can further improve the quality of computed solutions. Our auto- parameterization scheme relieves the user from having to find the right parameters while providing a high quality solution.

1 Introduction

The Quadratic Assignment Problem (QAP) is a hard combinatorial optimization problem with many real-life applications such as scheduling, facility location, electronic chipset layout, production, process communications, turbine runner balancing, data center network topology, among many others [14, 5]. QAP has been shown to be NP-Hard and finding effective algorithms to solve it is an active research topic in recent years.
Medium size problems can be solved using exact methods (e.g., size ≤ 30), which can find an optimal solution or prove that a problem has no solution [1]. Exact methods consider the entire search space: either explicitly by exhaustive search or implicitly, by pruning some portions of the search space that have been detected as irrelevant for the search.
To tackle harder problems, one must resort to incomplete methods which provide good, albeit potentially sub-optimal solutions in a reasonable time. Such is the case for metaheuristics, which are high-level procedures that make choices to efficiently explore part of the search space, so as to make problems tractable. Metaheuristics usually have several parameters to adjust their behavior depending on the problem to solve [9]. Examples of metaheuristics include genetic algorithms, tabu search, local search and simulating annealing.
Metaheuristics operate on two main working principles: intensification and diversification. The former refers to the method's ability to explore more deeply a promising region of the search space, while the latter refers to the exploration of different regions of the search space. By design, some metaheuristics methods are better at intensifying the search while others are so at diversifying it. However, the behavior of most metaheuristics can be controlled via a set of parameters. A fine tuning of these parameters is therefore crucial to achieve an effective trade-off between intensification and diversification, and hence good performance in solving a given problem. Unfortunately, selecting the best set of parameters is a tedious and error-prone task. This process is even harder because the best parameters values vary with the problem structure and even just for different instances of the same problem, as stated by the Non-Free-Lunch theorem [48].
Each metaheuristic has its own strengths and weaknesses, which may vary according to the problem or even to the instance being solved. The trend is thus to design hybrid metaheuristics, which combine diverse methods in order to benefit from the individual advantages of each one [7]. However, this increases the number of parameters (parameters of individual metaheuristics and new parameters to control the hybridization). The design and implementation of a hybrid metaheuristic is a complex process; tuning the resulting parameters, to reach the best performance, is also very challenging.
Despite the good results obtained with the use of hybrid metaheuristics, it is still necessary to reduce the processing times needed for the hardest instances [42]. One of the most plausible

options entails parallelism [17]. In parallel metaheuristics one can have multiple instances of the same (or different) metaheuristics running in parallel, either independently or cooperatively through concurrent process communications [12, 43]. Not only does parallelism help to decrease processing time, but it can also be a means to easily implement hybridization.
In previous work we proposed a Cooperative Parallel Local Search solver, called CPLS [37, 35]. CPLS embeds various simple local search metaheuristics and then relies on cooperative parallelization to concurrently execute several metaheuristic instances, which cooperate during the search process. We later extended CPLS, by proposing PHYSH (Parallel HYbridization of simple Heuristics) [32, 31]. PHYSH supports the combination of population-based and single-solution metaheuristics by proposing a strategy for efficiently exchanging information between metaheuristics of a different nature.
CPLS and PHYSH facilitate the creation of parallel hybrid methods, but they also require the fine tuning of a larger number of parameters, since more metaheuristics (of different types) are involved. Moreover, the configuration of the parallel interaction itself (communication between the methods) involves yet another set of parameters which need to be adjusted. Tuning this increasing number of parameters makes it even more difficult to find the appropriate setting for the algorithm to behave optimally.
Automating the task of finding good parameters is thus desirable and has attracted significant attention from researchers. We may identify two kinds of strategies for automatic tuning: *parameter tuning* and *parameter control* [24]. In parameter tuning (off-line tuning) the set of parameters are defined before applying the algorithm to a specific problem (static definition of parameters). Several strategies for automatic parameter tuning of metaheuristics have been proposed [25, 26]. In contrast, parameter control strategies (online-tuning) adapts the values of the controlled parameters during the algorithm execution (dynamic adaptation of parameters). The idea is to find the best parameters setting during the solving process, using some mechanism to alter the parameter values according to the algorithm performance.
Parameter tuning can be seen as a pre-process pass which is executed before the solving in order to determine the adequate values for parameters. This does not affect the implementation of the solver. On the other hand, *parameter control* has to be implemented in the kernel of the solver. The former may appear easier but when the number of parameters become large it is hard to use in practice. Indeed, it usually requires many runs to identify the best parameter settings, making this a time-consuming process. These methods are often limited by the number of parameters and the computational power available. In that case, parameter control strategies emerge as a viable solution to deal with the high complexity of current solvers (hybrid and/or parallel).
In this paper we propose a parallel hybrid method with a parameter control strategy for solving the QAP, called DPA-QAP. DPA-QAP embeds multiple metaheuristic methods in a parallel hybrid execution and self-adapts the parameters of the metaheuristics using an iterative process, adaptation is performed based on performance measures. We carried out an experimental evaluation which shows that the auto-parametrization strategy outperforms a simpler version of DPA-QAP with no auto-parametrization, i.e., a parallel hybrid method with static parametrization. We perform the evaluation using the classical QAPLIB instances and also a particular set of very hard QAP instances.
In the remaining of this paper we present the related work on Section 2. Section 3 presents the general structure of DPA-QAP and Section 4 introduces the auto-parametrization strategy. Section 5 contains the experimental evaluation performed which validates our strategy. A short conclusion ends the paper.

2 Related work

The Quadratic Assignment Problem (QAP) was first proposed by Koopmans and Beckmann in 1957 [30] as a model for a facilities location problem. This problem consists in assigning a set of n facilities to a set of n locations, while minimizing the cost associated with the *flows* of items among facilities and the *distance* between them. Let F be the flow matrix, where f_{ij} is the flow between facilities i and j, let D be the distance matrix, where d_{kl}, is the distance between the locations k and l. The goal is to find an optimal assignment of facilities to locations at minimum total cost, which is defined as the sum of all products between flows and distances [28]. Equation 1 presents a formulation of QAP, where $\phi(i)$ is the location which facility i is assigned to and $d_{\phi(i)\phi(j)}$ is the distance between locations $\phi(i)$ and $\phi(j)$.

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} * d_{\phi(i)\phi(j)} \quad (1)$$

Metaheuristic methods have been successfully applied for solving QAP. From the early 90s, several metaheuristic methods have emerged as a suitable option to solve this problem, e.g. Simulated Annealing [15], Iterated Local Search [44], Tabu Search [45], Genetic Algorithms [47], among several others. These methods perform well on a wide range of QAP instances, however, some hard instances still require very long runs to achieve quality solutions. Moreover, no method was able to get good performance on an extensive set of instances. The aforementioned problems spurred the emergence of new techniques based on hybridization and parallelization.

The idea on hybrid metaheuristics is to combine different methods to integrate their strengths and counter their weaknesses. For instance, one of the fundamental methods of hybrid metaheuristics is the memetic algorithm (MA) [34]. MA is an effective approach which combines an evolutionary algorithm with a local search procedure. Hybrid metaheuristics are intricate procedures, tricky to design, implement, debug and tune, therefore, it is unsurprising that hardly any of them only combine more that a couple of methods.

Parallelizing metaheuristics grants access to using powerful computational platforms with the aim of speeding up the search process [16]. A straightforward implementation of parallel metaheuristics is the *Independent multi-walks* approach which speeds up the search process by performing concurrent executions of multiple metaheuristic instances, therefore augmenting the probability to get quickly a good solution [13]. Another kind of parallel metaheuristics allows the concurrent instances to cooperate by exchanging information during the search process, aiming to improve the efficiency of the solver [46, 33]. We identify these methods as *Cooperative multi-walk* approaches. Cooperative parallelization not only improves the processing times but also open a plethora of possibilities to create new hybrid algorithms.

We proposed a way to create hybridization through cooperative parallelization in our CPLS framework [37, 35]. CPLS allows the user to code the individual metaheuristics, and the framework manages parallelism and communications. In CPLS, different local search (single-solution) metaheuristics concurrently interact by exchanging relevant information about the search. This interaction provides a cooperative way to intensify the search. This framework has been successfully used to solve hard variants of Stable Matching Problems [38] and hard instances of QAP [36, 35]. Since CPLS does not support population-based methods, we proposed an extension of the framework called PHYSH [32, 31], which provides an efficient strategy to promote cooperation between population-based and single-solution methods. Both CPLS and PHYSH have proved able to efficiently solve several hard combinatorial optimization problems, including QAP.

Parallel hybrid metaheuristics often have many parameters which modify the algorithm behaviour. Setting these parameters has a heavy influence on the performance of the method, however, finding the optimal values for these parameters is usually a hard task [25]. Using hybridization and parallelism makes this task even more difficult for mainly two reasons: First, hybrid metaheuristics inherit the parameters of each "low level" metaheuristic, so one needs to find the setting of more parameters, since a parameter configuration for one algorithm usually is not suitable for another. Second, cooperative parallel strategies also require parameters to define their behaviour, e.g., for determining how frequently metaheuristics should interact or how each metaheuristic has to use the received information,...

Tuning metaheuristic parameters (known as *offline-tuning*) has been carried out in different ways, in earlier times the tuning process was done by hand, another popular approach was to take parameters values from similar algorithms reported in the literature. More recently, the use of specialized tools for automatic parameter tuning has become prevalent, these techniques use advanced methodologies and tools from a theory of experiment design to machine learning approaches, among others [23]. Several methods have been proposed for parameter tuning, for instance SPO [4], CALIBRA [2], REVAC [39], F-Race [6], ParamILS [25], SMAC [26], HORA [3]. However, these methods have limitations when tuning a large number of parameters or when they require significant computational resources to perform the test runs [24].

Parameter control (*online-tuning*) emerges as a reasonable option. Some strategies have been proposed for specific metaheuristics such as [41] for swarm intelligence and [27] for evolutionary algorithms. Also, some specific strategies has been proposed for the QAP, such as [20] which proposes a strategy for self-control parameters on a Tabu Search method.

Hyper-heuristics present another way to face the problem of metaheuristic parameter control. These form a novel research approach in which a high level strategy selects or generates the best metaheuristics with their respective parameters and acceptance criteria. This approach is used with the aim of having more general methods, not designed for a single problem or for a few instances of a problem [11]. To the best of our knowledge, only one hyperheuristic method solves the QAP and uses parallelism in its design [18]: the authors propose a parameter control method using a genetic algorithm (GA) acting as a high-level strategy in the hyperheuristic approach. The GA, generation by generation, performs the adaptation of parameters through cross-over and mutation operators, ending up with the parameters at their best adjustment for each method.
We achieve a form of hyperheuristic using cooperative parallelism. The key idea is to use the parallel computational power to not only create a hybrid metaheuristic but also to automatically control the parameters of the metaheuristic involved in the parallel hybrid method.

3 DPA-QAP method

This section presents the general structure of DPA-QAP, a Dynamic Parameter Adaptation method for solving the Quadratic Assignment Problem. DPA-QAP is build on the top of a parallel hybrid metaheuristic solver, similar to the one presented in [35]. Figure 1 presents the two main components of DPA-QAP, the *Worker* nodes and a *Master* node. Worker nodes run a set of metaheuristics, in parallel, carrying out the search process. We design each worker node to run in a separate thread, ideally bound to its own dedicated core, each thread runs a specific metaheuristic instance.

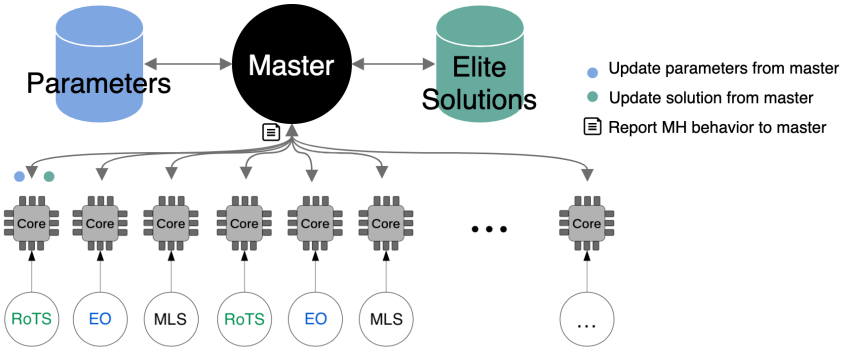


Fig. 1: DPA-QAP top-level view.

Each worker reports periodically its current candidate solution and some contextual information (e.g., solution cost, performance metrics, etc.) to the master node, which stores best intermediate solutions into an elite pool. When the master receives a solution from a worker node, it merges it into the elite pool. If the incoming solution is already present, it gets mutated by performing two random swaps. This mechanism promotes some diversity for the candidate solutions in the pool. When the elite pool is full, the master node sends solutions to worker nodes, ensuring the receiver implements a different metaheuristic from the one that inserted that solution into the population. This process constitutes a flexible interaction feature which eases the hybridization of metaheuristics promoting cross-fertilization among different types. The size of this population is equal to the number of worker nodes, since the population must have a solution for each method.
On the top of this cooperative parallel search, DPA-QAP implements a dynamic adaptation strategy which is tasked to automatically adjust the parameters of the metaheuristics during their execution, looking for the best setting and trying to ensure a balance between intensification and diversification in the search.

3.1 Metaheuristics used in the DPA-QAP method

We select three different metaheuristic methods for the worker nodes: Robust Tabu Search (RoTS), Extremal Optimization (EO), and Multistart Local Search (MLS). We select these

metaheuristics because they are commonly used in combinatorial optimization problems particularly for the QAP. We now present a brief description of each of these methods.

Robust Tabu Search The name Tabu Search (TS) refers to the use of an adaptive memory and special problem-solving strategies, called intelligent strategies, in order to get a better local search method [21]. The idea is to memorize within a structure the elements that for the LS will be forbidden to use (*tabu*) and thus avoid getting trapped in local optima. TS looks for the best solution within the neighborhood but does not visit the solutions of previous neighbors if they have been visited before or have been marked as prohibited locations [18]. Robust Tabu Search is an adaptation of TS to the QAP and has been one of the best performing methods for this problem [45].

Extremal Optimization Extremal Optimization (EO) is a metaheuristic inspired by self-organizing processes as frequently found in nature: for EO this is *self-organized criticality* (SOC). The version proposed by [8] has only one adjustable parameter: τ, and uses of a Probability Distribution Function (PDF). EO proceeds like this: it inspects the all candidate configurations (all assignments in the neighborhood). Each one is assigned a fitness value, by means of the goal function. The configurations are then ranked from worst to best. After that is turn for the PDF, which introduces uncertainty in the search process. EO resorts to the PDF to choose a solution from organized configurations. The role of the τ parameter is to provide different search strategies from pure random walk ($\tau = 0$) to deterministic (greedy) search ($\tau \Rightarrow \infty$). In previous work, we extended the basic EO metaheuristic to support not only a power-law PDF, but also an exponential and a gamma-law PDFs [36].

Multistart Local Search A simple Local Search (LS) is one of the oldest and most frequently used metaheuristics. LS starts from an initial solution and repeatedly improves it within a defined neighborhood. Neighbor solutions can be generated by applying minor changes to the initial solution. LS ends when no improved solutions are found in the neighborhood of the current solution, achieving a local optima [49]. Multistart Local Search (MLS) is a modification of LS that iteratively performs multiple different searches, executing each local search from a different starting point. When MLS reaches a local optimum, it tries to escape by restarting the search from scratch or performing some random moves in the current solution.

Metaheuristics Parameters Table 1 presents the parameters considered for each metaheuristic, together with the range of variation for each parameter. These ranges are picked from the best performances, as reported in the literature. For RoTS we use the parameters reported in [45], for EO we select the parameters reported in [36] and for MLS, the only parameter used is the restart process, then no range is needed.

Table 1: Metaheuristic's parameter ranges (note that n stands for QAP instance's size).

Metaheuristic	Parameter name	Range
RoTS	Tabu duration factor	$[4n$ - $20n]$
	Aspiration factor	$[n^2$ - $10n^2]$
EO	PDF	Power - Exponential - Gamma
	τ	[0,1]
MLS	Start type	Restart from scratch Random swaps

4 Automatic Parameter Adaption in DPA-QAP

The DPA-QAP method operates within an iterative process. At the beginning, worker nodes are initialized with random parameters. DPA-QAP dynamically adapts the best setting

of parameters in every worker (which is executing a metaheuristic instance). Parameter tuning depends on the performance in the solving process for an individual worker at each iteration. Each worker periodically reports relevant information to the master. With this information, the master node evaluates the worker's performance and tweaks its parameters, trying to strike a balance between intensification and diversification in the search. Figure 2 depicts the flow diagram of this process. Gray boxes represent the functionality executed by worker nodes, in parallel. White boxes specify the iterative adaptation process by the master. The master waits while the worker nodes perform the search. When it receives a metaheuristic report, it develops a *performance evaluation* for each worker and executes the *parameters' adaptation* procedure. The master then sends a new, evolved, set of parameters and a new configuration back to the worker nodes. Worker nodes resume the search with the settings they received: parameters and restarting from a new initial solution (from the master's elite pool). DPA-QAP repeats this process until an established number of iterations is accomplished or when the solution target is reached.

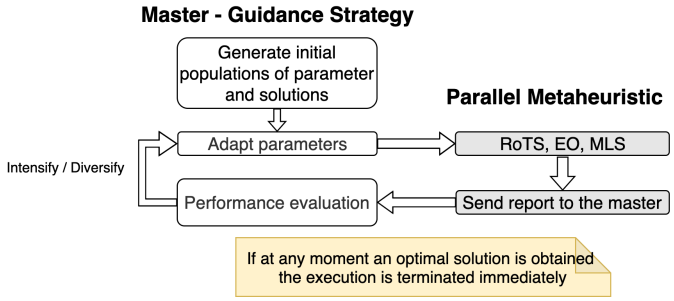


Fig. 2: DPA-QAP flow diagram.

4.1 Metaheuristics Performance Metrics

At each iteration of the parameters' adaptation process, each metaheuristic runs for a given time, `iteration_time`. When the `iteration_time` is running out, worker nodes report to the master node the initial solution and the best found solution in the interval with theirs associated costs. In order to assess the performance of a worker using a specific set of parameters, the master computes the distance between the initial and final solution (pair-wise difference) and the percentage gain for that iteration. The percentage gain is defined as:

$$\text{gain} = \frac{cost_{initial} - cost_{final}}{cost_{initial}}$$

Evaluating the performance of the metaheuristics is a critical process, and selecting the right set of metrics affects the overall performance of the parameters' adaptation process. In this work we consider two classical metrics, the percentage gain in the cost of the objective function and the distance between solutions. The gain acts as a direct indicator of the metaheuristic's performance, meanwhile the distance is assessing how diverse the search is. Other metrics can be also considered, for instance, the time spend on local optima, the number of iterations without improvements, among many others.

4.2 Performance Evaluation

The parameters' adaptation process evaluates the workers' performance by processing the percentage gain and the similarity between the initial and final solution (see Figure 3).
Through experimentation we verify that the gain is usually bigger at initial stages of the search than at the final stage. For this reason, DPA-QAP changes the value of the diversification gain limit during the search process, inspired by how the temperature decreases in simulated annealing [29]. Figure 4 shows how the diversification gain limit decreases in DPA-QAP during the search process. Using this dynamic limit, DPA-QAP diversifies the search more easily at the beginning than at the end of the search process.

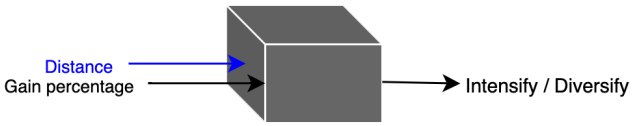


Fig. 3: Performance evaluation in DPA-QAP.

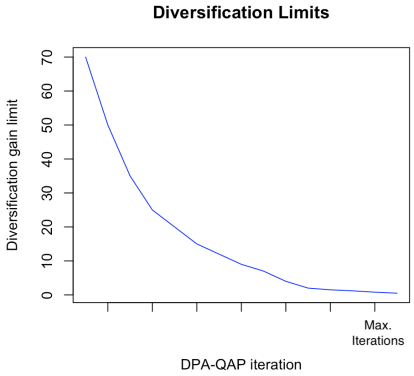


Fig. 4: DPA-QAP gain diversification limits.

The similarity criterion is computed comparing the distance between the initial and final solutions. If this distance is lower than one-third of the QAP size (i.e., 66% of the variables are equal), we consider both solutions as "very similar".
Considering these two criteria, we defined the following rules to determine which action must be taken for adapting the worker's parameters:

- If the gain obtained by the method and its pair-wise difference is lower than the corresponding limits, the component adapts the metaheuristic parameters to *diversify* the search.
- If the gain is higher than the corresponding diversification gain limit or the pair-wise difference is higher than the distance solution limit, the component adapts the parameters to *intensify* the search.

Both the dynamic diversification limits and the distance solution limit are hyper-parameters of the auto-parametrization strategy. We plan to test different limits in future work.

4.3 Adapting the Parameters

The evaluation of the worker's performance outputs a mandate which can be, intensify or diversify. This output is used as input for the parameters adaptation process. For each possible case we define a behavior depending of the metaheuristic type.

Extremal Optimization In EO the parameter τ is in the range 0 to 1 and, depending on its value and the PDF, this may lead the metaheuristic to intensify or diversify the search, by adding or subtracting a delta value belonging to the range (see Figure 5). The parameters are then adjusted by adding to their values using deltas, so the master performs a search process that looks for the best parameters setting for a given metaheuristic.

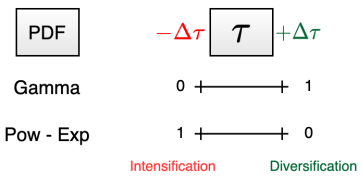


Fig. 5: Parameters adaptation in EO.

Robust Tabu Search We define the parameter adaptation process for Robust Tabu search as follows: if the parameter adaptation component returns *diversify*, a delta of $n/2$ is added to the tabu duration and a delta of $n^2/2$ is added to the aspiration parameters. If the parameter adaptation component returns *intensify*, the tabu duration is subtracted by $n/2$ and the aspiration is decreased by $n/2$. For intensification, the delta for the adaptation of the aspiration parameter is different to diversification. This is done intending to slow down the intensification process, avoiding to stagnates on a local optimum.

Multi-start Local Search For the case of MLS, if there is any gain in cost, the type of restart is retained. If there is no gain, the algorithm changes to the other option.

5 Experimental Evaluation

In this section we present an experimental evaluation of our proposed method, DPA-QAP, comparing its performance against an independent parallel hybrid metaheuristic method. We consider three sets of very hard benchmarks: the 20 hardest instances of QAPLIB [10] and two sets of even harder instances: Drezner's [19] `dre`XX and Palubeckis's [40] `Inst`XX instances. Each instance is executed 30 times stopping as soon as the Best Known Solution (BKS) is found or when a time limit of 5 minutes is hit, in case the BKS is not reached. All experiments have been carried out on a quad-AMD Opteron 6380 system, totaling 64 cores running at $2.5GHz$ and 128 GB of RAM.
At present, DPA-QAP is systematically configured with 30 worker nodes: 10 running RoTS, 10 running EO and 10 running MLS. Each worker node randomly initializes each parameter of its metaheuristic. To do so, a value is randomly picked from the admissible values according to Table 1. These parameters are then periodically adapted as explained in the previous section. In this experiment, parameter control is triggered every 15 or 20 seconds, depending on the size of the problem. Each metaheuristic can thus adapt its parameters up to 20 times during the 5 minutes global execution cap. We plan to study the impact of varying this interval and determine if it is also possible and useful to dynamically adapt it.
We compare DPA-QAP to a *base solver* (called BASE-QAP) which is statically parametrized (from an implementation point of view, this solver is actually derived from DPA-QAP by disabling the parameter control mechanisms). Other than that, BASE-QAP is identical to DPA-QAP: it also creates 30 metaheuristic instances (10 of each type of metaheuristic); each metaheuristic instance also randomly initializes its parameters, which instead remain fixed during the execution. Our goal is to compare this pre-process parameterization (parameters fixed) with self-parameterization. Usually the parameter tuning pre-process is a time-consuming task, the idea is to avoid this offline tuning step by having an online method able to adapt its parameters meanwhile the problem solution is carry out.
Both methods are similarly implemented in Java 11 using the `ForkJoinPool` and `Atomic`*`Type`* classes to handle the parallelism in a shared memory model [4]. In all cases we made sure that each worker node is actually mapped by the JVM onto a different physical core, at runtime.

5.1 Evaluation on QAPLIB

We evaluated the performance of the DPA-QAP on QAPLIB, a well-known collection of 134 QAP problems of various sizes and difficulties [10]. The instances are named as `name`XX where `name` corresponds to the first letters of the author and XX is the size of the problem. For each instance, QAPLIB also includes the Best Known Solution (BKS), which is sometimes the optimum. Many QAPLIB instances are easy for a parallel solver, we therefore selected the 20 hardest ones (removing all systematically solved instances). We ran both DPA-QAP and BASE-QAP under the same conditions (30 repetitions, time limit of 5 minutes).
Table 2 presents the results. For each solver, the table lists the number of times the BKS is reached across the 30 executions (**#BKS**), the Average Percentage Deviation (**APD**), which is the average of the 30 relative deviation percentages computed as follows: $100 \times \frac{Avg-BKS}{BKS}$, where Avg is the average of the 30 found costs, and finally the average execution time (**Time**). Execution times are given in seconds (as a decimal number). This time is the elapsed (wall clock) time, and includes the time to install all solver instances, solve the problem, communications and the time to detect and propagate the termination. To compare the performance of both solvers, we first compare the number of BKS found, then (in case of tie), the APDs and finally the execution times. For each benchmark, the best-performing solver row is highlighted and the discriminant field is enhanced in bold font.

[4] source and instances are at https://github.com/JonathanDuque/QAPMetaheuristic/tree/DPA-QAP

Table 2: Evaluation of dynamic adaptation on 20 hardest instances of QAPLIB.

		DPA-QAP			BASE-QAP		
	BKS	#BKS	APD	Time	#BKS	APD	Time
sko72	66256	**28**	0.010	130.9	24	0.012	161.2
sko81	90998	**20**	0.012	209.6	10	0.011	242.3
sko90	115534	**9**	0.022	262.2	8	0.016	274.9
sko100a	152002	**12**	0.027	245.0	4	0.029	279.3
sko100b	153890	**20**	0.012	223.1	14	0.014	242.9
sko100c	147862	**27**	0.010	268.7	20	0.010	287.2
sko100d	149576	6	0.024	287.7	**9**	0.021	285.9
sko100e	149150	**20**	0.012	266.1	16	0.015	271.2
sko100f	149036	8	0.018	267.8	**9**	0.017	265.9
tai40a	3139370	**4**	0.082	272.7	3	0.085	290.6
tai50a	4938796	0	**0.386**	300.0	0	0.401	300.0
tai60a	7205962	0	**0.479**	300.0	0	0.519	300.0
tai80a	13499184	0	**0.689**	300.0	0	0.780	300.0
tai100a	21044752	0	**0.647**	300.0	0	0.685	300.0
tai80b	818415043	**14**	0.031	282.1	13	0.028	254.5
tai100b	185996137	5	0.084	282.9	**10**	0.077	285.3
tai150b	498896643	0	0.654	300.0	0	**0.601**	300.0
tai256c	44759294	0	0.183	300.0	0	**0.179**	300.0
tho150	8133398	0	0.095	300.0	0	**0.086**	300.0
wil100	273038	**19**	0.011	292.0	13	0.013	293.0
Summary		**192**	**0.174**	**269.5**	153	0.180	276.7

DPA-QAP outperforms BASE-QAP on 14 out of 20 of the hardest QAPLIB instances, while the reverse only occurs for 6 instances. 7 instances can never been solved by any solver. Clearly, a time limit of 5 minutes is too short for those hard problems: we plan to experiment with larger time limits. The "summary" row shows that DPA-QAP obtains a better *#BKS* than BASE-QAP (192 vs. 153, a 25% increase). The average APD is also better (0.174 vs. 0.180). It is worth noticing that solutions of better quality are obtained in a slightly shorter average execution time (269.5 *sec* vs. 276.7 *sec*).
Notice that BASE-QAP is indeed an efficient solver for this benchmark, it implements a parallel hybridization strategy and its parameters, despite being randomly initialized, are selected within a range taken from state-of-the-art solvers which report competitive results. Still, DPA-QAP managed to outperform BASE-QAP in most instances.

5.2 Evaluation on harder instances

We also evaluated DPA-QAP on two more sets of instances, artificially crafted to be very difficult for metaheuristics: the `dre`*XX* instances proposed by Drezner at al. [19] and the `Inst`*XX* instances by Palubeckis [40]. These problems are generated with a known optimum. For this test we used the same machine and configuration as for QAPLIB (30 cores and a time limit of 5 min with 30 repetitions).
Table 3 presents the results for Drezner's instances. We have omitted small instances which are systematically solved by both solvers in less than 15 seconds. We start with `dre42` which is solved by both solvers at each replication; even on this case DPA-QAP is much faster than BASE-QAP: 33 *sec* vs. 61 *sec*. In all instances, DPA-QAP outperforms BASE-QAP. Regarding the 3 largest instances, we note that a time limit of 5 minutes is clearly too short and we need to experiment with a higher timeout. As a whole, DPA-QAP reaches more BKS (60 vs. 38) and, when the optimum is not reached, solutions provided by DPA-QAP are of much better quality than BASE-QAP as shown by the APDs (23.558 vs. 32.408), and it does so in a shorter period of time.
Table 4 presents the results for Palubeckis' instances. As in the previous case, we did not include small instances which are systematically solved by both solvers in less than 15 seconds. Here again, DPA-QAP performs better than BASE-QAP on all instances of the benchmark. As for Drezner's instances, the time limit of 5 minutes appears too short to solve large instances. However, DPA-QAP does find more BKS (83 vs. 60) and dynamic parameter

Table 3: Evaluation on Drezner instances.

		DPA-QAP			BASE-QAP		
	OPT	#BKS	APD	Time	#BKS	APD	Time
dre42	764	30	0.000	**33.5**	30	0.000	61.4
dre56	1086	**21**	14.140	212.6	8	21.020	259.3
dre72	1452	**9**	27.391	264.7	0	34.944	300.0
dre90	1838	0	**22.052**	300.0	0	28.031	300.0
dre110	2264	0	**36.072**	300.0	0	52.111	300.0
dre132	2744	0	**41.694**	300.0	0	58.343	300.0
Summary		**60**	**23.558**	**235.1**	38	32.408	253.5

adaptation makes it possible to improve the quality of solutions wrt. BASE-QAP as shown by the APDs (0.147 vs. 0.157).

Table 4: Evaluation on Palubeckis' instances.

		DPA-QAP			BASE-QAP		
	OPT	#BKS	APD	Time	#BKS	APD	Time
Inst40	837900	**29**	0.150	107.8	26	0.173	150.6
Inst50	1840356	**23**	0.096	199.4	18	0.124	237.8
Inst60	2967464	**20**	0.165	188.3	11	0.147	248.5
Inst70	5815290	**9**	0.121	266.6	3	0.156	293.3
Inst80	6597966	2	**0.184**	291.8	2	0.193	291.8
Inst100	15008994	0	**0.179**	300.0	0	0.184	300.0
Inst150	58352664	0	**0.140**	300.0	0	0.142	300.0
Inst200	75405684	0	**0.137**	300.0	0	0.138	300.0
Summary		**83**	**0.147**	**244.2**	60	0.157	265.2

6 Conclusions and future work

We have proposed a dynamic parameter adaptation scheme for parallel and hybrid solvers based on metaheuristics to solve the QAP. The basic principle of this approach is to have a master node which periodically collects the progress of each metaheuristic. This node has a global view of the overall search progress, therefore it can provide each metaheuristic with new parameter values in order to increase its effectiveness. We proposed DPA-QAP: an implementation of this architecture in Java, embedding three well-known metaheuristics: Robust Tabu Search, Extremal Optimization and Multistart Local Search. The first experiments performed on very difficult instances of QAP validate our approach by significantly improving solution quality.

We plan to extend this work in several directions. First, we will experiment on machines with more cores and with time limits greater than the 5 minutes cap which was allowed in this work. We will also try to determine the best settings for parameter reporting and adjustment: in this experiment we used a constant interval which needs to be refined. Another line of potential experiments consists in including efficient metaheuristics, such as Ant Colony Optimization [22]; or embedding population-based metaheuristics, such as genetic algorithms. Finally, we plan to address larger instances of the QAP as well as other difficult problems. As an outcome, we aim to design and propose a general framework for self-adaptation able to address a wide variety of combinatorial search and optimization problems.

Acknowledgement

This research was supported by the CODI (*Comité para el desarrollo de la Investigación*) project PRV19-1-01 funded by the University of Antioquia in Medellín,Colombia.

References

1. M. Abdel-Basset, G. Manogaran, H. Rashad, and A. N. H. Zaied. A comprehensive review of quadratic assignment problem: variants, hybrids and applications. *Journal of Ambient Intelligence and Humanized Computing*, 0(0):1–24, 2018.
2. B. Adenso-Díaz and M. Laguna. Fine-tuning of algorithms using fractional experimental designs and local search. *Operations Research*, 54(1):99–114, 2006.
3. E. B. Barbosa and E. L. Senne. A heuristic for optimization of metaheuristics by means of statistical methods. *ICORES 2017 - Proceedings of the 6th International Conference on Operations Research and Enterprise Systems*, 2017-January(Icores):203–210, 2017.
4. T. Bartz-Beielstein, C. Lasarczyk, and M. Preuss. Sequential Parameter Optimization. In *2005 IEEE Congress on Evolutionary Computation*, volume 1, pages 773–780. IEEE, 2005.
5. R. K. Bhati and A. Rasool. Quadratic Assignment Problem and its Relevance to the Real World: A Survey. *International Journal of Computer Applications*, 96(9):42–47, 2014.
6. M. Birattari and J. Kacprzyk. *Tuning metaheuristics: a machine learning perspective*, volume 197. Springer, 2009.
7. C. Blum, J. Puchinger, G. R. Raidl, and A. Roli. Hybrid metaheuristics in combinatorial optimization: A Survey. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7505 LNCS(6):1–10, 2011.
8. S. Boettcher and A. Percus. Nature's way of optimizing. *Artificial Intelligence*, 119(1):275–286, 2000.
9. I. Boussaïd, J. Lepagnot, and P. Siarry. A survey on optimization metaheuristics. *Information Sciences*, 237:82–117, 2013.
10. R. E. Burkard, S. Karisch, and F. Rendl. QAPLIB - a Quadratic Assignment Problem Library. *European Journal of Operational Research*, 55(1):115–119, 1991.
11. E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward. A Classification of Hyper-Heuristic Approaches: Revisited. In *International Series in Operations Research and Management Science*, chapter 14, pages 453–477. Springer Berlin Heidelberg, 2019.
12. Y. Caniou, P. Codognet, F. Richoux, D. Diaz, and S. Abreu. Large-scale parallelism for constraint-based local search: the costas array case study. *Constraints*, 20(1):1–27, 2014.
13. P. Codognet, D. Munera, D. Diaz, and S. Abreu. Parallel local search, 2018.
14. C. W. Commander. A survey of the quadratic assignment problem, with applications. *Morehead Electronic Journal of Applicable Mathematics*, 4:MATH–2005–01, 2005.
15. D. T. Connolly. An improved annealing scheme for the QAP. *European Journal of Operational Research*, 46(1):93–100, 1990.
16. C. Cotta, E.-G. Talbi, and E. Alba. Parallel Hybrid Metaheuristics. In *Parallel Metaheuristics*, pages 347–370. John Wiley & Sons, Inc., Hoboken, NJ, USA, sep 2005.
17. T. G. Crainic and M. Toulouse. Parallel Meta-heuristics. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 497–541. Springer US, 2010.
18. T. Dokeroglu and A. Cosar. A novel multistart hyper-heuristic algorithm on the grid for the quadratic assignment problem. *Engineering Applications of Artificial Intelligence*, 52:10–25, 2016.
19. Z. Drezner. The Extended Concentric Tabu for the Quadratic Assignment Problem. *European Journal of Operational Research*, 160(2):416–422, 2005.
20. N. Fescioglu-Unver and M. M. Kokar. Self Controlling Tabu Search algorithm for the Quadratic Assignment Problem. *Computers & Industrial Engineering*, 60(2):310–319, 2011.
21. F. Glover. Tabu search—part II. *ORSA Journal on Computing*, 2:4–32, 1990.
22. Y. Hani, L. Amodeo, F. Yalaoui, and H. Chen. Ant colony optimization for solving an industrial layout problem. *European Journal of Operational Research*, 183(2):633–642, 2007.
23. H. Hoos. Automated Algorithm Configuration and Parameter Tuning. In Y. Hamadi, E. Monfroy, and F. Saubion, editors, *Autonomous Search*, pages 37–71. Springer Berlin Heidelberg, Berlin, 2012.
24. C. Huang, Y. Li, and X. Yao. A Survey of Automatic Parameter Tuning Methods for Metaheuristics. *IEEE Transactions on Evolutionary Computation*, 24(2):201–216, 2020.

25. F. Hutter, H. Hoos, K. Leyton-Brown, and T. Stützle. ParamILS: an Automatic Algorithm Configuration Framework. *Journal of Artificial Intelligence Research*, 36:267–306, 2009.
26. F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6683 LNCS:507–523, 2011.
27. G. Karafotias, M. Hoogendoorn, and Á. E. Eiben. Parameter control in evolutionary algorithms: Trends and challenges. *IEEE Transactions on Evolutionary Computation*, 19(2):167–187, 2014.
28. L. Kaufman and F. Broeckx. An Algorithm for the Quadratic Assignment Problem Using Benders' Decomposition. *European Journal of Operational Research*, 2:207–211, feb 1978.
29. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220 4598:671–80, 1983.
30. T. C. Koopmans and M. Beckmann. Assignment Problems and the Location of Economic Activities. *Econometrica*, 25(1):53–76, 1957.
31. J. Lopez, D. Munera, D. Diaz, and S. Abreu. On integrating population-based metaheuristics with cooperative parallelism. In *Proceedings - 2018 IEEE 32nd International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2018*, 2018.
32. J. López, D. Múnera, D. Diaz, and S. Abreu. Weaving of Metaheuristics with Cooperative Parallelism. In A. Auger, C. M. Fonseca, N. Lourenço, P. Machado, L. Paquete, and D. Whitley, editors, *Parallel Problem Solving from Nature – PPSN XV*, volume 11101 LNCS, pages 436–448, Cham, 2018. Springer International Publishing.
33. L. Loukil, M. Mehdi, N. Melab, E. G. Talbi, and P. Bouvry. A parallel hybrid genetic algorithm-simulated annealing for solving Q3AP on computational grid. *IPDPS 2009 - Proceedings of the 2009 IEEE International Parallel and Distributed Processing Symposium*, (June 2014), 2009.
34. P. Moscato, C. Cotta, and A. Mendes. Memetic Algorithms. In *Handbook of Heuristics*, volume 1-2, pages 53–85. Springer Berlin / Heidelberg, 2004.
35. D. Munera, D. Diaz, and S. Abreu. Hybridization as Cooperative Parallelism for the Quadratic Assignment Problem. In *10th International Workshop, HM 2016*, volume 9668 of *Lecture Notes in Computer Science*, pages 47–61, Plymouth, UK, 2016. Springer International Publishing.
36. D. Munera, D. Diaz, and S. Abreu. Solving the Quadratic Assignment Problem with Cooperative Parallel Extremal Optimization. In *The 16th European Conference on Evolutionary Computation in Combinatorial Optimisation*, Porto, 2016.
37. D. Munera, D. Diaz, S. Abreu, and P. Codognet. A Parametric Framework for Cooperative Parallel Local Search. In *European Conference on Evolutionary Computation in Combinatorial Optimization (EvoCOP)*, volume 8600, Granada, Spain, 2014.
38. D. Munera, D. Diaz, S. Abreu, F. Rossi, V. Saraswat, and P. Codognet. Solving Hard Stable Matching Problems via Local Search and Cooperative Parallelization. In *AAAI*, Austin, TX, USA, 2015.
39. V. Nannen and A. E. Eiben. Relevance estimation and value calibration of evolutionary algorithm parameters. *IJCAI International Joint Conference on Artificial Intelligence*, pages 975–980, 2007.
40. G. Palubeckis. An Algorithm for Construction of Test Cases for the Quadratic Assignment Problem. *Informatica, Lith. Acad. Sci.*, 11(3):281–296, 2000.
41. R. S. Parpinelli, G. F. Plichoski, R. S. D. Silva, and P. H. Narloch. A review of techniques for online control of parameters in swarm intelligence and evolutionary computation algorithms. *International Journal of Bio-Inspired Computation*, 13(1):1–20, 2019.
42. M. Saifullah Hussin. *Stochastic Local Search Algorithms for Single and Bi-objective Quadratic Assignment Problems*. PhD thesis, Université de Bruxelles, 2016.
43. A. Silva, L. C. Coelho, and M. Darvish. Quadratic assignment problem variants: A survey and an effective parallel memetic iterated tabu search. *European Journal of Operational Research*, (xxxx), 2020.
44. T. Stützle. Iterated local search for the quadratic assignment problem. *European Journal of Operational Research*, 174(3):1519–1539, 2006.
45. E. Taillard. Robust taboo search for the quadratic assignment problem. *Parallel Computing*, 17(4-5):443–455, 1991.
46. E.-G. Talbi and V. Bachelet. COSEARCH: A parallel cooperative metaheuristic. *Journal of Mathematical Modelling and Algorithms*, 5(1):5–22, 2006.

47. D. M. Tate and A. E. Smith. A genetic approach to the quadratic assignment problem. *Computers & Operations Research*, 22(1):73–83, 1995.
48. D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.
49. M. Yagiura and T. Ibaraki. Local Search, 2002.

Combination of Machine Learning and Optimization: A Survey of Recent Methods for Large Complex Problems

O. Bahri and L. Amodeo

Industrial Systems Optimization Laboratory,
University of Technology of Troyes, 10004 Troyes, France
oumayma.bahri@utt.fr lionel.amodeo@utt.fr

1 Introduction

Industrials and scientists are often faced decision-making problems that are too difficult for humans to solve. The complexity stems from the same source: data collection and integration as in the case of large volume of health-care data from multiple sources in the hospital environment (medical records, patient surveys and comments, administrative databases, etc.); or for example real-time re-planning in the event of anomaly or incident detection in the transportation & logistic domain.

In order to address such real-life problems, we observe today more and more research works which deal first with the integrative aspect (throughout the data collection and analysis until the problem modeling) using techniques derived from Machine Learning (ML); then which cover the paradigm of Optimization (OPT) to find the best possible solution(s).

Over many years, ML & OPT fields have provided great theoretical insights, offered many successful methods and found practical applications in all areas of science. In fact, ML analysts prefer simpler algorithms that work in reasonable computational time for specific classes of problems. Thus, ML research advances and develops rapidly, which has made a lot of works applied in various popular fields such as image recognition, recommender systems and anomaly detection, natural language processing, etc. In turn, OPT researchers often address more complex (or NP-hard) problems by deriving the core resolution process and using specific optimizers to solve them. OPT methods and approaches have attracted much attention in almost all decision-making fields.

While the two domains having more evolved independently from each other, their classical approaches have become increasingly inadequate with the recent data explosion. Then, to deal with the exponential increase of data volume and complexity, business and research communities have paid a lot of attention to the synergy and interplay between ML & OPT. Indeed, the two scientific disciplines are deeply interwoven: On one hand, OPT lies at the heart of ML in the sense that most ML problems, once formulated, can be solved as OPT problems. On the other hand, OPT concepts and methods equip ML researchers with tools for training large families of models. Besides, modern OPT algorithms are using ML theories and techniques to improve their efficiency.

Recently, numerous research works have been focused on the developments of new techniques and tools inspired at the same time by the ML principles and OPT concepts. In this context, we propose to give some classification and summary of the most popular studies, which can offer guidance and inspiration to contribute in both domains. Thus, we first suggest to distinguish two major types of research directions:

1. OPT for ML: The first question is how the combination with OPT can help ML researchers to rapidly develop new tools for more complex families of learning models? The second question is how to use OPT methods to solve specific ML problems? For instance, how to yield optimal estimations or predictions based on large or dynamic collected data?

2. ML for OPT: The question here is how to improve the OPT algorithmic structure, process and strategies by means of ML techniques? An interesting challenge involves performance measurements problems namely, how to use ML models to efficiently report and assess the results quality of a particular OPT algorithm?

In the following paragraphs, we seek to examine a brief state-of-the-art of existing approaches along the two axes.

2 OPT for ML

This section presents the recent advances in ML community, especially the studies that use OPT principles or methods to exploit novel ML branches. A wide range of works aimed at extending well-known OPT methods to create novel mixed learning models and paradigms [3]. For instance, [10] discussed the extensive use of mathematical programming methods in ML. In [4], convex optimization methods are applied for nonlinear kernel approximation or classification problems. In [6], constraint-based OPT methods are developed for incorporating domain knowledge into graphical learning models. In [12], parametric or hybrid OPT methods are used to find the optimal solution for simulation-based problems.

Moreover, many researchers have discussed the role of some analytical methods of optimization in very popular fields like neural network (NN), reinforcement learning, meta learning, etc. They have shown that the development of optimization algorithms in specific ML fields can be inspiring to perform more informative learning [8]. For example, learning the parameters of complex NNs is one of the most well studied problems in the field of ML. [1, 7] have proposed an adaptive gradient-based methods for online NN learning.

Unfortunately,the gradient descent scheme can result in poor learning training and performance in the case of NNs that have multiple hidden layers (i.e. deep networks). Thus, recent works have focused on the combination of new layer-wise training methods with stochastic gradient-based algorithms (i.e. which are first-order optimization approach) [5]. The results of such a combination lead to new general high-order optimizers which can efficiently learn deep models without any need for pre-training [9, 11]. Some problems are emerging when applying these adaptive optimizers. For example, the learning rate can be oscillating in the later training stage, which may lead to non-converging problem. Thus, further OPT algorithms based on variance reduction were proposed to improve the convergence rate [13]. Other popular methods that have a significative influence on various ML fields are: optimization-based meta-learning methods, Adam optimization for image super resolution, trust-region optimization for deep reinforcement learning, etc [8].

3 ML for OPT

In turn, ML has highly motivated advances in the OPT community by allowing the develop of new effective methods. This original research axe presents modern algorithms including small changes in their underlying core process that enable high computing power. Otherwise, in order to promote the development of OPT, a series of effective learning methods were put forward, which have improved the performance and efficiency of optimization algorithms.

The essence of most OPT algorithms is first to build an optimization model and then to find the best parameters in the objective function w.r.t. some problem constraints. A taxonomy of related concepts and methods exists based on the type of optimization (Integer or discrete), of objective function (single or multiple), of constraints (exact, imperfect or dynamic), etc. Then, as data set size and complexity grows, classical OPT methods become inadequate. New approaches that exploit the ML properties and techniques can outperform most large complex problems.

From the perspective of integrating the ML techniques into optimization methods, research interests can be divided into three categories according to the chosen step integration: i) Before the model mathematical definition, ML techniques can be used to learn for example uncertain and missing data in order to incorporate adequate constraints into the model. ii) During the selection and optimization process in order to enforce the optimality conditions and to converge to optimal solution(s) more efficiently. iii) In the process of algorithms validation by paying attention to the characteristics of their parameters.

Many of the papers blend these different categories and novel methods have been successfully designed and applied to NP-hard problems [2, 17]. Recently, evolutionary machine learning (EML) took the interest of many researchers [14] since they have shown performance in many difficult problems. [15] published a survey about the use of statistics and machine learning for the distributed optimization. [16] reviewed existing literature on the combination of metaheuristics with machine learning methods and then introduces the concept of learnheuristics, a novel type of hybrid algorithms.

4 Conclusions and perspectives

This study, being of an exploratory and analytical survey nature, raises a number of opportunities for future research, both in terms of concept validation and innovative development. More research will in fact be necessary to refine and further elaborate our novel challenges.

An interesting perspective challenge is how to simultaneously exploit combinatorial optimization and machine learning in order to solve a complex decision-making problem encountered in hospital management domain. In fact, the use of both disciplines may help the healthcare systems to be more effective, by improving the quality of in-hospital services and smoothing healthcare performances along with administrative and financial control.

References

1. Amari, S. I., Park, H. and Fukumizu, K.: Adaptive method of realizing natural gradient learning for multilayer perceptrons. Neural computation, 12(6), 1399-1409, (2000)
2. Bennett, K. P. and Parrado-Hernandez, E.: The interplay of optimization and machine learning research. The Journal of Machine Learning Research, 7, 1265-1281, (2006)
3. Sra, S., Nowozin, S. and Wright, S. J.: Optimization for machine learning. Mit Press, (2012)
4. Bubeck, S.: Theory of convex Optimization for Machine Learning. arXiv preprint arXiv. 15, 1405-4980, (2014)
5. Bengio, Y., Lamblin, P., Popovici, D. and Larochelle, H. Greedy layer-wise training of deep networks. Advances in neural information processing systems, 19, 153, (2007)
6. Ciravegna, G., Giannini, F., et al.: A Constraint-based Approach to Learning and Explanation. In Proceedings of the AAAI Conference on Artificial Intelligence, 34(4), 3658-3665, (2020)
7. Duchi, J., Hazan, E. and Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. Journal of machine learning research, 12(7), (2011)
8. Sun, S., Cao, Z., Zhu, H. and Zhao, J.: A survey of optimization methods from a machine learning perspective. IEEE transactions on cybernetics, 50(8), 3668-3681, (2019)
9. Martens, J.: Deep learning via hessian-free optimization. In International Conference on Machine Learning, 27, 735-742, (2010)
10. Scheinberg, K., Peng, J., et al.: Mathematical Programming in Machine Learning and Data Mining, (2007)
11. Hu, J., Jiang, B., et al.: Structured quasi-Newton methods for optimization with orthogonality constraints. SIAM Journal on Scientific Computing, 41(4), 2239-2269, (2019)
12. Gray, G. A., Fowler, K. and Griffin, J. D.: Hybrid optimization schemes for simulation-based problems. Procedia Computer Science, 1(1), 1349-1357, (2010)
13. Yin, W. and Schütze, H.: Multichannel variable-size convolution for sentence classification. arXiv preprint arXiv: 1603-04513, (2016)
14. Al-Sahaf, H., Bi, Y., Chen, Q., et al.: A survey on evolutionary machine learning. Journal of the Royal Society of New Zealand, 49(2), 205-228, (2019)
15. Boyd, S., Parikh, N. and Chu, E.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Now Publishers Inc, (2011)
16. Calvet, L., de Armas, J., Masip, D. and Juan, A. A.: Learnheuristics: hybridizing metaheuristics with machine learning for optimization with dynamic inputs. Open Mathematics, 15(1), 261-280, (2017)
17. Weichert, D., Link, P., Stoll, A., et al.: A review of machine learning for the optimization of production processes. The International Journal of Advanced Manufacturing Technology, 104(5), 1889-1902, (2019)

Time-series Prediction of Flow Field around a Two-Dimensional Cylinder Using Autoencoder and LSTM

Yusuke Yamada[1] and Masahiro Kanazaki[1]

1. Tokyo Metropolitan University

Keywords : CFD, Autoencoder, LSTM.

1 Introduction

Deep learning has been attracting attention as a method for analysis and prediction in Computational Fluid Dynamics (CFD). Deep learning is a multilayer neural network structure consisting of tens of millions of artificial neurons that are layered with nonlinear computational processes. It has been reported that deep learning has higher representational power than conventional modeling methods and can learn complex and abstract features. Convolutional neural networks (CNN) and recurrent neural networks (RNN) [2] are some of the most commonly used methods. By combining these network structures, it is possible to model a variety of objects. Deep learning has a wide range of applications, including image recognition and natural language processing.

In this study, we propose a method for predicting the temporal evolution of a flow field by combining a CNN autoencoder [1] and Long Short-Term Memory (LSTM) [2], which is a method of RNN. The training target is the result of non-stationary computation around a two-dimensional cylinder. In the proposed method, the compressed feature vectors are input to the LSTM for a certain length of time and the feature vectors for the next time are output. The output feature vectors are decoded by the CNN autoencoder to predict the future state of the CFD results. In the proposed method, we try to improve the robustness of the model by repeatedly inputting the decoded output of the model to the same model during the training of the compressed model. In the proposed method, we try to improve the robustness of the model by repeatedly inputting the decoded output of the model into the same model during the training process.

2 Methods

Two types of neural network models are used: a compression model and a prediction model. First, the compression model based on CNN autoencoder is used to convert the data with three dimensions into a one-dimensional array structure vector, which is a compressed feature of the original data. The vector sequence is then inputted to a prediction model based on LSTM. The prediction model is a recursive neural network model that outputs the vector data corresponding to the one-time step to the vector data corresponding to the next time step. Finally, the output vector sequence is decoded by the compression model for each time step to obtain the prediction data for the time ahead of the input.

As the training data, we used the velocity field around a two-dimensional cylinder. The governing equations are the uncompressed Navier-Stokes equation and the continuity equation. The Reynolds number was set to 10000. The target geometry and domain are as shown in Fig. 1.

Learning is performed for the compressive model and the predictive model. In the training of the compressive model, each instantaneous field is input to the autoencoder independently, and the parameters in the model are updated to minimize the sum of squares with the output data. In the training of the prediction model, the training data is compressed using the encoder of the trained compression model. The parameters of the encoder are not learned and are fixed.

In the proposed method, we try to improve the robustness of the model by repeatedly inputting the decoded output of the model to the same model during the training of the compressed model as shown in Fig. 2. In the proposed method, the original input data is given as the supervisory data to the data that has accumulated errors due to repeated input. In the proposed method, the original input data is given to the

model as teacher data, and the model learns to correct the accumulated errors. As for the number of iterations of repeated input, since the reproduction performance of the model is considered to be insufficient in the initial stage of learning, if a large number of iterations of input are performed from the beginning, the output data will differ greatly from the input data, and the learning process will not be efficient. Therefore, in the proposed method, learning is performed without repeated input at the beginning, and the number of repeated inputs is gradually increased as learning progresses. For updating the model parameters in the learning process, error backpropagation is applied only at the last iteration, since the amount of parameter updates increases with the number of iterations if error backpropagation is applied to all iterations. The number of compressed dimensions was set to 50. The number of compressed dimensions was determined by the parametric study, and the number of compressed dimensions with the smallest error was adopted. The CNN that constituted the autoencoder had two dimensions for input and output, and the output of the first layer had 10 dimensions, and the number of dimensions was doubled for each layer. This method was based on VGGNet [3].

Using the encoder and decoder of the trained model, time-series prediction is performed by the process shown in Fig. 3. During training, when the data input length is n, the prediction model is trained to output the feature vector t=2,3,...,n+1 one step ahead of the feature vector t=1,2,...,n. During prediction, each successive instantaneous field of the prediction input length is compressed by an encoder and input to the prediction model in the same way as during training. Next, the last output feature vector of the prediction model is input to the prediction model again, and the operation is repeated to output feature vectors up to an arbitrary time beyond the input range. The last output feature vectors are then decoded by the decoder to obtain the predicted flow field data.

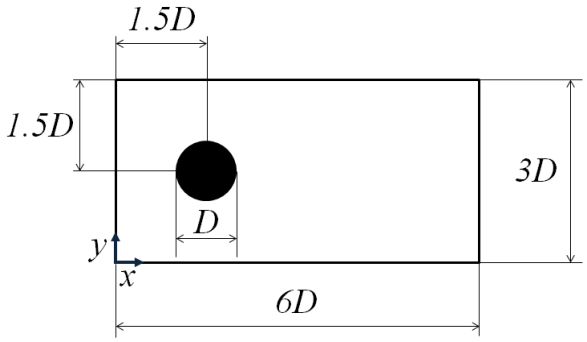


Figure 1. Computational domain and shape of the object to be computed.

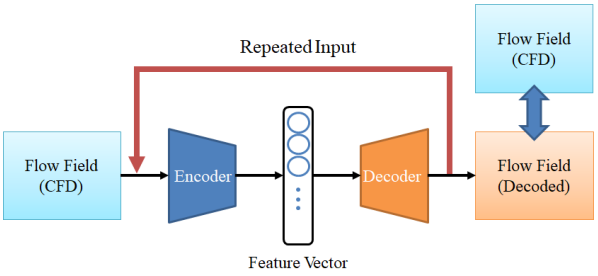


Figure 2. Proposed Learning Method for Repeatedly Inputting Data into Autoencoder.

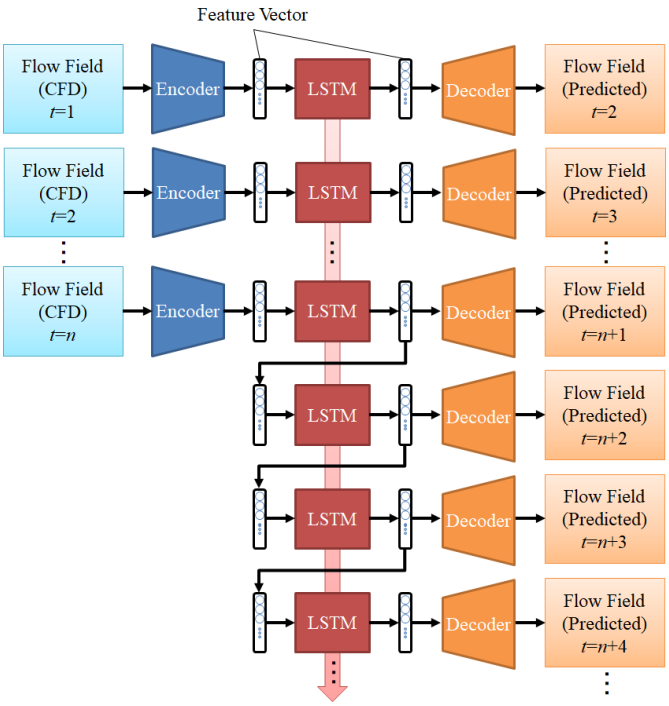


Figure 3. The Flow of Time-series Prediction by LSTM.

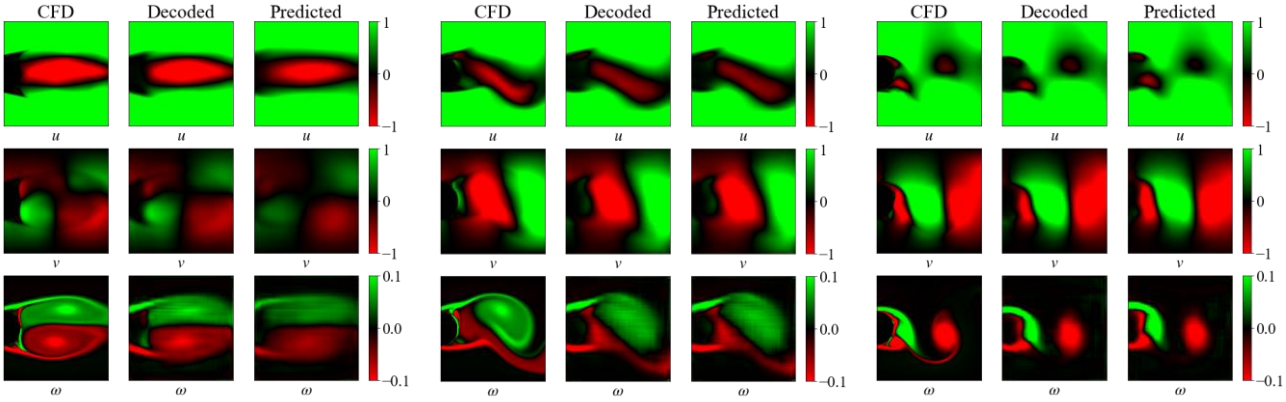
3 Results

The length of the input interval was set to 50 steps, and time-series prediction was performed by varying the input interval. The output results at the start of input, at the start of prediction, and 500 steps from the start of prediction are shown in Fig. 4. The left column CFD represents the instantaneous field output by CFD, the center column Decoded represents the left column encoded and decoded by autoencoder, and the right column Predicted represents the instantaneous field predicted by the prediction model. The

right column, Predicted, represents the instantaneous field predicted by the prediction model. The types of values are, from the top row, velocity u, velocity v, and vorticity ω calculated from u and v.

The CFD and Decoded models were compared to investigate the performance of the compression model, and it was confirmed that the distributions of the velocity and vorticity components were generally reproduced at all times. In the boundary area where the value becomes zero, such as around the object or the contour of the vortex, the accuracy of reproduction is low as if the shape is blurred. Besides, the long and narrow vortex shape shown in Fig. 4(c) was not fully reproduced, probably because the error calculation method used for CNN training is the squared error over the entire space, so the importance of reproducing the detailed shape is relatively low.

Next, comparing CFD and Predicted to investigate the performance of the prediction model, we can see that the prediction is relatively accurate at the start of prediction, while the error of the prediction is larger around the object and in the contour of the vortex at the start of input. This is because the internal parameters of the LSTM are initialized at the start of the input, so the time-series features cannot be applied to the prediction, and the accuracy is reduced. On the other hand, at the start of forecasting, the internal parameters have been updated by the previous inputs, so the model is now ready for forecasting.



(a) Input start time. (b) Prediction start time. (c) 500steps after prediction start.

Figure 4. Comparison of CFD results of the velocity field and vorticity calculated from the velocity field at different times.

4 Conclusions

In this study, time-series prediction was performed for a velocity field with time-series by two-dimensional non-stationary CFD calculations. Two types of networks, a compression model based on autoencoders and a prediction model based on LSTM, were used together to output time-series data from data with a certain time-series length. In the previous studies, the prediction results were greatly corrupted with each time step, making it difficult to make normal predictions. However, by using the compressed model learning method proposed in this study, which is based on repeated input of data, the robustness of the model was improved and normal prediction of data became possible.

We used only one type of data (circular) for training and prediction, but in practical use, it is necessary to predict data that is not used for training. In the future, we would like to develop a method that is capable of predicting time-series of unknown data for the model by using multiple target shape data for model training.

References

[1] Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. science, 313(5786), 504-507.

[2] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

[3] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

A Recommendation System for Efficient PEH Bus Operation Based on Unsupervised Machine Learning

Patricia Ruiz[*1], Juan Carlos de la Torre[1], Renzo Massobrio[1,3], Pablo Moreno[1]
Marcin Seredynski[2], Bernabé Dorronsoro[1]

1. School of Engineering, University of Cadiz, Spain
{patricia.ruiz, juan.detorre, pablo.morenogarcia, bernabe.dorronsoro}@uca.es
2. E-Bus Competence Center, Luxembourg
3. Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay

Keywords : Advanced public transport systems, connected vehicles, machine learning, optimization techniques, sustainability.

1 Introduction

Sustainability and efficient management of the continuously growing urban areas are major challenges to face in the next future. The uncontrolled growth of cities and the services they provide, led to inefficient road transport, traffic congestion, higher pollution levels, noise and stress for citizens. In very dense urban areas with high mobility demands, public transport (PT) is the only solution that can address all the aforementioned problems. Moreover, strategies like the definition of ZE corridors, where noise and tailpipe emissions are eradicated, are being settled in some European countries and will be mandatory soon.

Electric and connected vehicles play a key role in this public transport revolution. In this context, Plug-in Electric Hybrid (PEH) buses are moving from pilot project to small scale deployment [1], because of their high flexibility and their zero emission zone management system that allows to pre-set locations of electric driving thanks to geofencing and connectivity. Currently, the main research lines focus in deploying strategic locations of e-charging stations [2], analysing the potential of the ZE zone –ZEZ– [6], and the benefits of applying cooperative Intelligent Transportation Systems [5]. However, the current ZEZ management of PEH buses is naïve (based on the first-come first-served rule), the assignment is highly conservative and therefore, the full potential of dynamic ZEZ is not exploited [4].

In previous work [3], two different approaches were used for solving the Efficient PH Bus Operation (EPBO) for finding the effective management of electric drive PH buses. The goal was to minimize the energy consumption (i.e. maximize the electric range) while still respecting the mandatory ZE corridors. The two approaches used were: i) genetic algorithm that uses global knowledge of the problem to find an optimal static strategy, and ii) a decentralised recommendation system based on supervised machine learning, that makes use of local knowledge to dynamically take decisions. Results show that the artificial neural network (ANN) was able to learn the right decisions from the GA to build a good strategy, discarding the wrong ones. This is evidenced by the fact that the solutions found by ANN outperformed those of the GA, in general, despite the fact that ANN only uses local information to take decisions while the GA makes use of global information.

Encouraged by these results, in this work we study the suitability of unsupervised machine learning techniques in order to develop a recommendation system able to learn accurate strategies during the bus operation mode through reinforcement learning. The final goal of this approach is the possibility of deployment in any new location, and automatically improve its performance through experience.

2 The Efficient PH Bus Operation Problem

The Efficient PH Bus Operation Problem, or EPBO, was firstly introduced in [3] to optimise the operation strategy of PEH buses. In EPBO, the bus route is divided into a number of segments so that the consumption of the bus in every segment can be considered constant. Therefore, route segments are broken when there is a meaningful change in the road slope, or when there is a bus stop (because it implies that the bus stops and the weight it carries might change). In addition, segments can belong to a Zero Emissions Zone (ZEZ), meaning that the bus is forced to cover its length using the electric engine.

The bus route is divided into n segments $T = \{t_1, \ldots, t_n\}$, each segment t_i being characterized by its length (l_i), its slope (s_i), and whether it belongs to a ZEZ or not (z_i). The EPBO problem is to maximize:

$$f(\vec{x}) = \sum_{i=0}^{n} x_i \cdot g(\vec{x}, t_i) \; ; \quad g(\vec{x}, t_i) = \begin{cases} 2 \cdot l_i & if \; z_i = 1 \wedge \; c_i = l_i \\ c_i & if \; z_i = 0 \\ -K \cdot (l_i - c_i) & if \; z_i = 1 \wedge \; c_i < l_i \end{cases}$$

where $\vec{x}$ is the solution vector, assigning whether route segment t_i should be covered in electric ($x_i = 1$) or explosion ($x_i = 0$) engine, and c_i is the distance covered in segment t_i using the electric engine. Function $g(\vec{x}, t_i)$ contributes to the fitness function with double the length of the segment if it is in a ZEZ and is covered in electric mode, or the length of the segment if it is covered with the electric engine, but it is not in a ZEZ. In the case when the segment is not in a ZEZ and is not completely covered in electric mode, $g(\vec{x}, t_i)$ penalizes the fitness with a negative high constant K multiplied by the number of kilometers not covered in electric mode.

The length covered in electric mode in every segment is computed by our simulator. It is based on PHSim [3], but in this work we make it stochastic by adding some uncertainty on the consumption of the electric engine of the bus.

3 Results and Discussion

We follow in this work the same approach as in [3], to solve the problem, where an ANN chooses whether to use the electric or combustion engines according to the battery level of the bus when entering a given segment, the characteristics of the segment (its length, its elevation, and whether it is a green corridor), and the features of the rest of the route until the end (its length and the number of remaining ZEZ km). However, in this work we analyse the application of unsupervised machine learning techniques to address the problem. In particular, we evolve the weights of the ANN using a GA. Therefore, a solution of the GA is an array of real numbers (in interval [-1.5, 1.5]), each corresponding to the weight of a given connection in the ANN (we consider the same topology of ANN as in [3], 3 hidden layers with 10 neurons each). The solution is evaluated by simulating the strategy defined by the ANN, and computing the fitness function according to the length covered in electric mode and if all green corridors are respected, as it is defined in Section 2.

For evaluating the approach, we took route number 181 from Montevideo (as done in our preliminary work). We chose this route because of its high number of passengers and its length (among the longest routes in the city) with 16.07 km long, making it very challenging. Because the real route lacks of ZEZs, we have considered 5 different percentages of route segments that belong to ZEZ: 2%, 5%, 10%, 15% and 20%, and we have performed 10,000 evaluations to optimize the ANN in each case.

Results with the unsupervised ANN are compared to those obtained in [3] with the supervised one (which outperformed the GA). Figures 1 and 2 show the percentage of valid solutions and their quality, respectively. A solution is considered valid if all the mandatory electric segments (ZE corridors) are fulfilled. As it can be seen in Figure 2, although the quality of the solutions is similar in both, the supervised and unsupervised approaches, the percentage of valid solutions found (Figure 1) is always either the same or higher when applying the unsupervised technique.

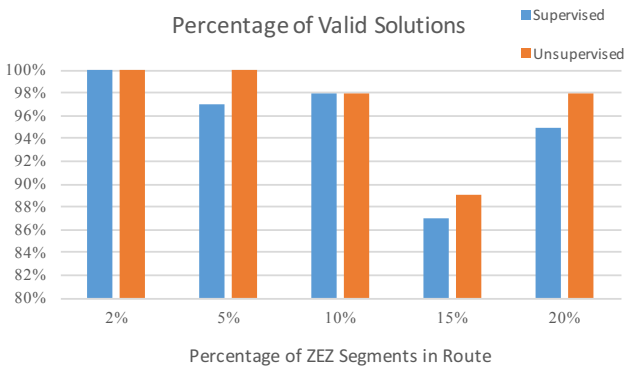


Figure 1. Comparison of the Percentage of valid solutions between supervised and unsupervised approaches

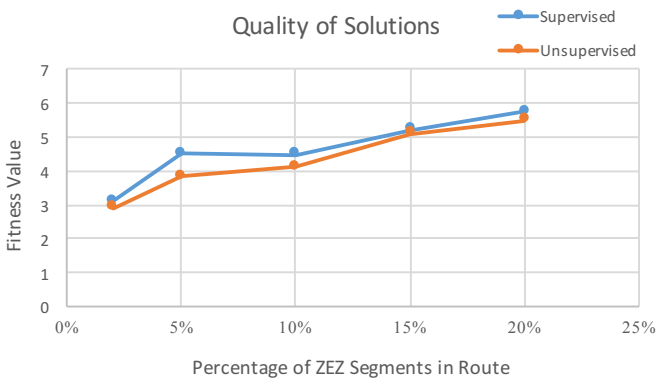


Figure 2. Comparison of the quality of valid solutions between supervised and unsupervised approaches

4 Conclusions

In this work, we propose the use of unsupervised machine learning to reach accurate self-learned strategies for PEH buses to cover their route maximising the electric engine use. A Genetic Algorithm is used to tune the weights of the Artificial Neural Network (ANN), which are evaluated by simulating the performance of the bus with their defined strategy. The performance of the resulting ANN is compared against the state-of-the-art supervised ANN, offering similar quality results but a better reliability (in terms of the number of feasible solutions found).

As future work, we plan to enrich our model with new features that will make our approach more realistic, as taking into account dense traffic situations and the weight carried by the bus.

Acknowledgements

B. Dorronsoro and P. Ruiz would like to acknowledge the Spanish Ministerio de Economía, Industria y Competitividad and ERDF for the support provided under contract RTI2018-100754-B-I00 (iSUN project) and Junta de Andalucía and ERDF for the support provided with P18-2399 (GENIUS) and sol-201800108393-tra (OPTIMALE).

References

[1] Auge, O. 2015. *Keynote 2: TOSA concept: A full Electric Large Capacity Urban Bus System*. In proceedings of the 17th European Conference on Power Electronics and Applications.
[2] Pternea, M., Kepaptsoglou, K., Karlaftis, M. 2015. Sustainable Urban Transit Network Design. *Transportation Research Part A: Policy and Practice*. Vol. 77, pp. 276-291.
[3] Ruiz P., Arias A., Massobrio R., de la Torre J.C., Seredynski M. and Dorronsoro B. 2020. *Intelligent Electric Drive Management for Plug-in Hybrid Buses*. In proceedings of the Third International Conference on Optimization and Learning Conference, 17-19 February, Cádiz, Spain. CCIS Springer, pp 85-98.
[4] Seredynski, M. 2018. Targeted air quality improvement via management of zero emission zones of plug-in hybrid buses. In proceedings of the 25th ITS World Congress.
[5] Seredynski, M., Viti, F. 2017. Novel C-ITS support for electric buses with opportunity charging. In proceedings of the Conference on Intelligent Transportation Systems (ITSC). pp. 1-6.
[6] Seredynski, M., Viti, F. 2019. Towards dynamic zero emission zone management for plug-in hybrid buses. In proceedings of the 26th ITS World Congress.

Towards an hybrid positioning approach for IoT-based smart factories

Andrés Millán-Alcaide, Angel Ruiz-Zafra, and Guadalupe Ortiz

Department of Computer Engineering. University of Cádiz, Cádiz, Spain.
{andres.millan, angel.ruiz, guadalupe.ortiz}@uca.es

Keywords: Internet of Things · Industry 4.0 · Industrial Internet of Things · Hybrid Positioning Systems · Smart Factories

1 Introduction

Nowadays, the Internet of Things (IoT) is one of the most prominent tech trends to have emerged in recent years, enabling interconnected environments through IoT-based devices, dedicated networks and many other technologies [1]. As demanded technology, IoT is being applied in many different areas, such as healthcare, agriculture, smart buildings and, surely, smart manufacturing [2].

Smart factories (also known intelligent factories) is the cornerstone of the fourth industrial revolution (Industry 4.0), a term and concept coined in 2011 to promote the complete computerization of manufacturing (digitization) with the aim to enhance and foster automation, improve communications and self-monitoring, and production of smart machines without the need for human intervention [3].

Industry 4.0 is based in several enabling technologies, that is, Cloud Computing, Computer Vision, Big Data, Cybersecurity and Internet of Things [4]. Although these technologies are crucial for smart manufacturing, IoT stand out from the others as base technology to provide reliable information gathered from the different devices integrated in the ecosystem of a smart factory (e.g. environmental sensors/actuators, integrated IoT-based devices in manufacturing machines). Thus, IoT applied in Industry 4.0 has led to a specific research and application area called Industrial Internet of Things (IIoT) [5].

IIoT is crucial into smart manufacturing as provider of the technology to support multiple functionalities required, such as better management of safety, cybersecurity, full autonomous operations, advanced analytical tools and mobile technologies [6], bringing several benefits. However, many significant challenges still lie ahead, such as IIoT standardization, energy efficiency, real-time performance and interoperability, among others [7].

A relevant challenge faced by industrial partners nowadays as key functionality for smart manufacturing is the positioning and tracking of people and assets [8], usually supported by the so-called *Indoor Positioning Systems* (IPS) or *Indoor Location Systems* (ILS) which enable high precision and dimensional positioning in indoor environments. Even more, the use of services provided by the IPS along with the use of an *Outdoor Positioning Systems* (OPS) (e.g. based on Global Positioning System - GPS-) gives raise to *Hybrid Positioning Systems* (henceforth *HPS*), which could provide an all-in positioning system for an industrial complex or industrial parks [9].

Nevertheless, the design, development and deployment of HPS is complex, where several concerns and issues must be addressed. First of all, a wide range of IoT-based technologies are suitable for an HPS, and choosing the proper technologies, depending of the smart manufacturing scenario, is crucial. For instance, the type of network must be carefully selected due to restrictions in industrial environments, depending of network parameters such as frequency range, physical range, payload size, response time, etc [10]. Beside concerns related to IoT technology, software issues are also involved, such as the use of the proper algorithm, technique or procedure to accurately determine an object position (e.g. Fingerprinting, k-Nearest Neighbors (KNN), Kalman filtering, probabilistic models) [11].

Secondly, smart factories are peculiar environments due to the manufacturing processes carried out. For instance, materials used in the factory (walls, zones) and in some industrial assets may vary regarding more conventional environments (e.g. smart office building), where several problems (e.g. multipath propagation and noise due to metallic environments) can modify the state of the

signal and conduct to errors in the positioning systems [12]. In this way, traditional approaches supported by commonly used technology and techniques (e.g. WiFi, Bluetooth, Fingerprinting basic techniques) are not the most suitable choice to deploy a HPS in a smart factory. Thus, the way to design, develop and deploy adaptable HPS in a smart factory, regarding their application area, is still a challenge for industrial partners as well as for researchers [13].

In this paper we present the first preliminary outcomes of a still on-going research about this challenge, that is, the design, development and deployment of HPS in industrial environments. As preliminary results, we present a methodology to describe the different stages needed, starting from the requirements analysis and concluding with the validation of the proposal through an in-the-wild Proof of Concept (PoC). Through the methodology, we identify the different concerns and issues presented, according to feedback provided by industrial partners as well as a deep review of the literature.

2 Methodology and Work Hypothesis

In the design and deployment of an HPS, many different concerns, challenges and issues are presented. The way to address these concerns and challenges is crucial, and the related tasks involved must be carried out properly, where the order as well as the execution is important.

The methodology, depicted in Figure 1, falls into four different stages or phases. Several tasks are carried out through the different stages and the outcome of each stage is the input information required for the next one.

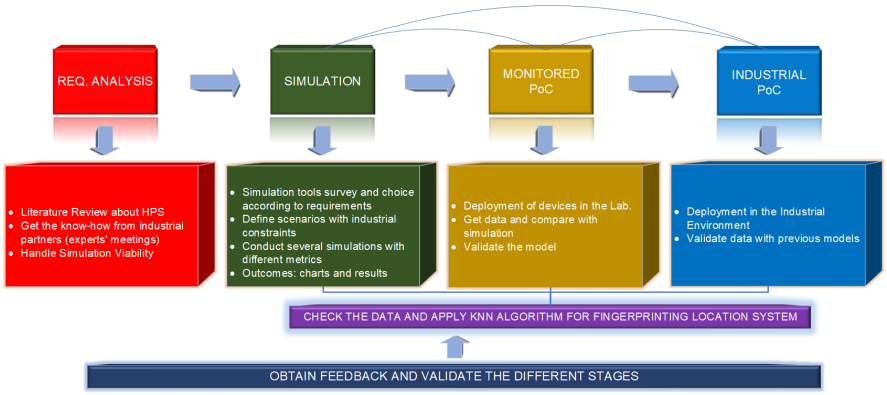


Fig. 1. Proposed methodology for the development of a full 4.0 Industry based HPS.

First of all, it is required a gathering of the requirements, with the aim to know the feasibility of the proposal as well as the technological challenges involved. This is done through a deep literature review and holding meetings with experts to get the know-how.

Secondly, with the requirements collected, we must conduct our research focusing on the proposal of a new HPS system. Due to wide range of technologies and techniques available for indoor/outdoor positioning systems, many different solutions can be proposed. With the aim to determine the most suitable solution for our scenario (smart factories), we design an specific scenario constrained by the requirements and we conduct several simulations to test the different proposals through the proper tools (software). The main aim is to obtain promising results in terms of accuracy in the positioning system into a smart factory or industrial environment.

Thirdly, as in any other conducted experiments or test, the success of the simulation scenario and proposal lead to try to (partially) validate the proposal through a PoC. First of all, in a isolated environment (laboratory, university) and, once the proposal is validated, deploy the PoC in a real smart factory, testing the results (accuracy, efficiency, robustness, etc).

3 Conclusions and Future Work

The deployment of HPS in industrial scenarios (e.g. smart factories) is complex, and traditional technologies, techniques and approaches may not fit in this scope due to the distinctive features presented in this scenarios, such signal frequency in industrial parks, the distance between access points or how material composition affect the signal intensity. In this way, the design, development and deployment of an HPS entail the addressing of many particular concerns and issues, where it is important to define the required stages and related tasks to develop an HPS with reliable and accurate results in terms of positioning system.

In this paper we address this concern proposing a methodology that describes, through four different stages, what features and requirements must be considered in the development of an HPS in industrial environments, as well as the related tasks that must be carried out to fulfill such goal. The methodology starts with a requirement analysis of industrial scenarios and conclude with the need of a PoC deployed in the industrial scenario, passing through simulations and conducted tests.

The proposal described so far in this paper is still in an early stage. Currently, we have completed the stage one (Requirement Analysis) concluding, as one of the outcome of this stage, the use of LoRaWAN as one of the most suitable technologies to support an HPS in industrial scenarios, against common technologies such as WiFi and Bluetooth.

Currently we are focusing on the second stage (Simulation), researching about the construction of set of simulations for further analysis in real environments. Therefore, as for further work, we will work in the remaining stages, addressing and developing both PoC (in-the-lab as well as in-the-wild), once the simulations are completed.

Acknowledgements

The work was supported by the Spanish Ministerio de Ciencia, Innovación y Universidades and the ERDF for the support provided under contract RTI2018-100754-B-I00 (iSUN project). In addition, this work also received inputs by the Universidad de Cádiz under the grant TDI-24-18.

References

1. Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, and Imrich Chlamtac. Internet of things: Vision, applications and research challenges. *Ad hoc networks*, 10(7):1497–1516, 2012.
2. Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.
3. Wolfgang Wahlster Henning Kagermann, Wolf-dieter Lukas. Announcement of the Industrie 4.0 project in the 2011 Hannover fair. https://www.ingenieur.de/technik/fachbereiche/produktion/industrie-40-mit-internet-dinge-weg-4-industriellen-revolution/. Accessed: 2020-12-01.
4. Alejandro Germán Frank, Lucas Santos Dalenogare, and Néstor Fabián Ayala. Industry 4.0 technologies: Implementation patterns in manufacturing companies. *International Journal of Production Economics*, 210:15–26, 2019.
5. Alasdair Gilchrist. *Industry 4.0: the industrial internet of things*. Springer, 2016.
6. John Conway. The industrial internet of things: an evolution to a smart manufacturing enterprise. *Schneider Electric*, 2016.
7. Emiliano Sisinni, Abusayeed Saifullah, Song Han, Ulf Jennehag, and Mikael Gidlund. Industrial internet of things: Challenges, opportunities, and directions. *IEEE Transactions on Industrial Informatics*, 14(11):4724–4734, 2018.
8. Detlef Zuehlke. Smartfactory—towards a factory-of-things. *Annual reviews in control*, 34(1):129–138, 2010.
9. Lionel Reyero and Gilles Y Delisle. A pervasive indoor-outdoor positioning system. *J. Networks*, 3(8):70–83, 2008.
10. Amina Seferagić, Jeroen Famaey, Eli De Poorter, and Jeroen Hoebeke. Survey on wireless technology trade-offs for the industrial internet of things. *Sensors*, 20(2):488, 2020.
11. Zain Bin Tariq, Dost Muhammad Cheema, Muhammad Zahir Kamran, and Ijaz Haider Naqvi. Non-gps positioning systems: A survey. *ACM Computing Surveys (CSUR)*, 50(4):1–34, 2017.
12. Faheem Zafari, Athanasios Gkelias, and Kin K Leung. A survey of indoor localization systems and technologies. *IEEE Communications Surveys & Tutorials*, 21(3):2568–2599, 2019.
13. You Li, Yuan Zhuang, Xin Hu, Zhouzheng Gao, Jia Hu, Long Chen, Zhe He, Ling Pei, Kejie Chen, Maosong Wang, et al. Location-enabled iot (le-iot): A survey of positioning techniques, error sources, and mitigation. *arXiv preprint arXiv:2004.03738*, 2020.

Author Index