8th Int. Conf. on Metaheuristics and Nature Inspired Computing
Marrakech,
~~April, 5-8, 2021~~
Oct 27-30, 2021
META

Proceedings of META'2021

8th International Conference on Metaheuristics and Nature Inspired computing

Sponsors

Table of contents

first_page_proceedings.pdf **1**

A discrete-time Markov Chain model for a healthcare periodic inventory system with stochastic demand, lost sales and required service level., Khoukhi Saâdia **1**

Extended path-relinking method for p-location problem, Janáček Jaroslav [et al.] **9**

Discrete PSO strategies for search in unit lattice of m-dimensional simplex, Kvet Marek [et al.] **16**

Multi phase methodology for solving the multi depot vehicle routing problem with limited supply capacity at the depots, De Prado Javier [et al.] **24**

Multi-Infill Bayesian Optimisation for improving Exploration and Exploitation Trade-off, Sadet Jérémy [et al.] **34**

The Construction of Uniformly Deployed Set of Feasible Solutions for the p-Location Problem, Czimmermann Peter **39**

Adaptive iterative destruction construction heuristic for the firefighters timetabling problem, Ouberkouk Mohamed Amine [et al.] **47**

A Genetic Algorithm for a capacitated lot-sizing problem with lost sales, overtimes, and safety stock constraints, Larroche François [et al.] **57**

A Physarum-inspired approach for influence maximization, Rodríguez-Gómez

Gustavo [et al.] 66

Visualizing Crossover Rate Influence in Differential Evolution with Expected Fitness Improvement, Stanovov Vladimir [et al.] 76

Evolutionary-based Optimization of Hardware Configurations for DNN on Edge GPUs, Bouzidi Halima [et al.] 86

pyParadiseo : a Python-powered Framework for Metaheuristic Optimization, Gmys Jan [et al.] 90

A Large Neighborhood Search for a Cooperative Optimization Approach to Distribute Service Points in Mobility Applications, Jatschka Thomas [et al.] 93

A New Learnheuristic: Binary SARSA - Sine Cosine Algorithm (BS-SCA), Becerra-Rozas Marcelo [et al.] 103

Minimum Rule-Repair Algorithm for Supervised Learning Classifier Systems on Real-valued Classification Tasks, Hamasaki Koki [et al.] 111

MultiObjective Optimization Approach under uncertainty for Workforce Routing and Scheduling Problem in Home Health Care, Belhor Mariem [et al.] 121

Automatic generation of metaheuristic algorithms, Iturra Sergio [et al.] 131

Evaluating private social choice: particle swarm optimisation without panopticon, Torra Vicenc [et al.] 139

A Biased Random-key Genetic Algorithm for the 2-Dimensional Guillotine Cutting Stock Problem with Stack Constraints, Almeida Guimarães Marcos Vinícius [et al.] 149

GA and ILS for optimizing the size of NFA models, Lardeux Frédéric [et al.] 159

Spiking Neural Networks and Audio Classification, Medjahed Farah [et al.] 169

A method for estimating preferences over binary choices, Lando Tommaso 172

Deep Learning and Multi-objective Optimization, Bahri Oumayma [et al.] 173

Energy-efficient buffer allocation : multi-objective study, Alaouchiche Yasmine [et al.] 175

A Study on the Influence of Runtime Uncertainty in the Optimization of Software Programs, Aragon-Jurado Jose Miguel [et al.] 180

Enhancing the Performance of Software Programs for Devices with Limited Computing Capacity, Benito-Jareño Carlos [et al.] 184

ClusterTune : Tuning Metaheuristics Using Clustering forScheduling Problems, Athmani Mohamed Elamine [et al.] 188

Hidden genes genetic algorithm applied to variable-sized design space optimal layout problems, Gamot Juliette [et al.] 191

Recovery-to-Efficiency: A New Robustness Concept for Multi-objective Optimization under Uncertainty, Todosijevic Raca [et al.] 194

A new strategy for collective energy self-consumption in the eco-industrial park : Mathematical modeling and economic evaluation, Gribiss Hamza [et al.] 197

Genetic Algorithm for Neural Network Architecture optimization, Makki Ayman [et al.] 207

Optimization of deep neural network: Application to the forecast of energy consumption, Firmin Thomas [et al.] 218

A review of the Electric School Bus Routing Problem, Bideq Hajar [et al.] 223

Author Index 228

A discrete-time Markov Chain model for a healthcare periodic inventory system with stochastic demand, lost sales and required service level.

S. Khoukhi[1]

Moroccan Laboratory of Innovation and Industrial Performance, Higher School of Technical Education, Mohammed V University, Rabat, Morocco
khoukhi.saadia@gmail.com

1 Abstract

The purchasing and storage costs of pharmaceutical items constitute a large portion of a hospital budget. In the present work, we develop an optimization approach to identify an optimal replenishment policy for a global stock within a hospital. We consider a stochastic demand, a periodic reorder-point order-up-to level inventory control system, lost sales and a set of constraints related to the pharmaceutical supply chain like limited storage capacity, restriction on annual order frequency and required service level. The system is modeled as a Markov chain to minimize the holding cost. The effectiveness of the proposed search procedure is illustrated by using a real case study of Ibn Sina univerity hospital Morocco.

2 Introduction

The main worry of hospital managers concerns the affordability and the accessibility of pharmaceutical items. Therefore, dissimilar to the product-based supply chain, the inventory level in a healthcare structure is calculated based on predefined service performance outcomes. Consequently, a large quantity of pharmaceuticals supplies are stored in hospitals to avoid shortages and permit the healthcare staff to realize efficiently their daily work [1], [2], [3] and [4]. However, healthcare establishments usually have limited storage capacity and an important part of hospital budget goes for medical furniture and their handling that consume approximately 30%-40% of overall hospital net revenue [5], [7], [8] and [9]. Consequently, it is primordial that the pharmacy managers identify a replenishment policy that compromises between the required service level and the limited storage capacity [10], [11] and [12].
Given this background, the present study focus on developing a replenishment model that takes into account the trade-off between the required service level and the restriction on the storage space. We consider a global stock (GS) that delivers pharmaceutical items to care units, which belong to the supply chain network of Ibn Sina hospital, Rabat, Morocco. Our main aim is to optimize inventory policy while considering a set of constraints related to the studied Pharmaceutical supply chain (PSC) as stochastic demand, lost sales, limited storage capacity, annual order frequency, desired quality service and emergency replenishment.
According to their importance in the recovery procedure, pharmaceutical products are classified into three types, namely, vital, essential, and non-essential items. In this work, we focus on vital and essential pharmaceutical items. This choice is motivated by the fact that these medicines are used by several treatments and also they are very expensive which make them critical. It is for these reasons, the GS adopts a nominative distribution to manage inventories related to these items, more particularly, the GS follows a periodic reorder-point order-up-to level policy (R, s, S) firstly describes by [13]. At the end of each review period R the pharmacy staff examine the amount of the storage quantity, if the maintaining inventory level is less than or equal to a predefined service level s, an order is placed to replenish the stock on hand to a pre-established value S. At the time of this study, the values of inventory policy s, S are fixed based on "expert knowledge" that doesn't consider the limited storage capacity or the restriction on the annual number of orders that the GS is allowed to address to the suppliers.
The present work reports our optimization inventory model that minimizes the holding cost while

including the special constraints imposed by the PSC. We start our work with a literature review on the studied topic in section 3, then the operations research program is formulated. Section 4, presents the discrete-time Markov Chain Model. The applicability of our Model is discussed in section 5 followed by conclusions in section 6.

3 Related works

[14] addressed an optimization inventory problem that emerges in a single echelon PSC with stochastic demand (normally distributed) and stochastic lead time. They considered a modern point of use that enables pharmacy managers to adopt an improved replenishment policy, termed, can order inventory policy (R, s, c, S). The authors developed a simple inventory rule based on an extended model of EOQ formula to identify the best values of replenishment parameters that minimize the total cost. Experiments show that the results obtained by this rule are comparable to ones obtained by more sophisticated manners. [15] dealt with a problem closed to the one treated by [14], but easier as they considered a deterministic lead time and known demand. They focused on minimizing the cost related to managing inventories while taking into account the contradictory objectives existing between the stakeholders in the pharmaceutical supply chain and the managerial trade-offs that emerge at the tactical, operational, and strategic levels. Authors developed an iterative search procedure based on EOQ formula and approximations developed by [16] to determine the best near-optimal value of undershoot and order quantity subject to service level, storage space and order frequency.
[17] dealt with an inventory problem that considers the variations of the system behavior over time. They addressed a case of a single depot with a hybrid replenishment policy, stochastic demand and neglected lead time. Authors adopted a resolution methodology firstly introduced by [19] to optimize the expected total cost given by the sum of holding cost, ordering cost and stock-out cost. [18] also examine inventory change over time. they consider a stochastic demand (compound Poisson distribution), limited order frequency and desired service level. Authors developed an innovative approach based on Monte Carlo simulation to identify the near-optimal replenishment policy that optimizes the total cost and satisfies the required service level. It is important to note at this point that the papers reviewed so far proposed approximation techniques based on iterative procedures. However, the development of the exact approach may offer a promising research field.
In this vein,[20] introduced a closed-form model to addressed an inventory problem raised in a point of use within a care unit. They supposed a reorder-point order-up-to level policy replenishment policy (R, s, S) with lost sales, short lead time and stochastic demand (Poison distributed). The authors developed two analytical models, the first one is a service model under storage capacity, while the second one is a capacity model with respect to the desired service level. The authors proposed a two-step solution approach. They started by evaluating the service level value firstly developed by [21], then a knapsack algorithm is implemented to find a good balance between the required service level and the available storage capacity. [23] also addressed a service objective function, by which they looked for identifying the best parameter values of a periodic par level replenishment policy, order frequency and service level. Authors introduced a programming-based algorithm under a set of constraints relates particularly to storage capacity and product criticality.[22] also developed an exact analytical model to address an inventory problem that emerges in a healthcare setting with two sources of uncertainty: demand and material unavailability. They used a continuous-time Markov chain to obtain the different optimal components of the total cost that include holding cost, shortage cost and substitution cost.

4 model and assumptions

In accordance with the present practices and with the primary aim of facilitating the proposed models, we assume that:

- The demand encountered by the GS is stochastic (observed distributed).
- The GS follows, for each product j, a periodic review reorder-point order-up-to level replenishment policy.
- The GS is replenished from trustable sources in fixed lead time (2 days).

– The GS must supply all care units if there is stock, otherwise, the demand is considered as lost and an emergency order can be sent by the GS to suppliers.
– The emergency purchase cost is sent to the supplier at a higher cost than the normal purchasing cost.

With respect to these assumptions and notations listed in table 1, we now proceed to present the proposed model by which we seek to define the optimal replenishment parameters that optimize the expected inventory on hand.

Table 1. Inventory control policy parameters

Indices	
j	Index for drugs ($j = 1, ..., r$).
Parameters	
L	Lead time for GS.
D_j^L	Cumulative demand of item j in the GS in the lead time L.
$D_j^{(R-L)}$	Cumulative demand of item j in the GS in the non lead time $R - L$.
D_j^R	Cumulative demand of item j in the GS in the review period R.
D_j^a	Cumulative demand of product j in the GS in a year.
m_j	Maximum storage capacity of item j at GS (units).
Y_j	Random variable of the stock on hand of item j in the GS.
$\overline{Y_j}$	Expected stock of item j in the GS at the review period.
γ	Annual order frequency.
α	Maximum service level required by hospital complex managers.
h_j	Holding cost of item j in the CP.
π_{lj}	Probability of having l units of product j in the GS at the review period.
η_{lj}	Probability of having l units of product j in the GS at the shifted review period.
Decision variables	
S_j	Order-up-to level for item j.
s_j	Reorder level for item j.

We start by defining a random variable, noted Y_j, as the inventory on hand of pharmaceutical item j in the GS at the review period and $\bar{Y_j}$ its corresponding expected value. Let h_j be the holding cost of item i.The analytical model of the studied inventory structure is presented in the next stochastic program:

$$minFO(s_j, S_j) = \sum_{j=1}^{r} h_j \bar{Y_j} \tag{1}$$

with this program, we look for minimizing the expected holding cost resulting from the average inventory on hand level at the GS. This minimization is subject to a set of constraints related to:

– **Service level:** A minimum inventory position is maintaining, from each item j, to satisfy a required service level of 95% or higher given by the fraction of satisfied demand.

$$Pr(D_j^R \leq S_j) \geq \alpha \qquad \forall j = 1, \ldots, r, \tag{2}$$

– **Storage space:** The amount quantity of product j on hand should not surpass the limited storage capacity of the GS, m_j.

$$S_j \leq m_j \qquad \forall j = 1, \ldots, r, \tag{3}$$

– **Order frequency:** The number of purchases orders sent by the GS is limited by the suppliers.

$$Pr(D_j^a \leq \gamma S_j) \geq \alpha, \qquad \forall j = 1, \ldots, r. \tag{4}$$

– **Integrality and non-negativity constraints.**

$$s_j, S_j \geq 0, \qquad \forall j = 1, \ldots, r. \tag{5}$$

5 Expected stock on hand calculation

To evaluate the expected stock on hand in the GS, we adopt the procedure approach developed by [21]. The first step of this approach is to calculate the transition probabilities of handling n units of items j at the current review point given that there were m units of the same item one review point before.
Providing that the GS follows a periodic reorder-point order-up-to level inventory policy with a fixed lead time L, the balance equations that represent the Markov chain with state $[0, S_j]$ are given by:

$$Y_j(t+1) = \begin{cases} \left[\left(Y_j(t) - D_j^L\right)^+ + (S_j - Y_j(t)) - D_j^{(R-L)}\right]^+ & \text{if} \quad Y_j(t) \le s_j \\ \left(Y_j(t) - D_j^R\right)^+ & \text{if} \quad Y_j(t) > s_j \end{cases} \tag{6}$$

Based on these equations, we obtain the one-period transition probabilities as follows:
For $0 \le m \le s_j$:

$$p_{mn}^j = \begin{cases} \sum_{d=0}^{m-1} P(D_j^L = d)\ P(D_j^{R-L} \ge S_j - d) + P(D_j^L \ge m)\ P(S_j - m \le D_j^{R-L}) & \text{if } n = 0, \\ \sum_{d=0}^{m-1} P(S_j - d - n = D_j^{R-L})\ P(D_j^L = d)\ + P(S_j - m - n = D_j^{R-L})\ P(D_j^L \ge m), & \text{if } 1 \le n \le S_j - m, \\ \sum_{d=0}^{S_j - n} P(S_j - d - n = D_j^{R-L})\ P(D_j^L = d) & \text{if } S_j - m \le n \le S_j, \end{cases} \tag{7}$$

For $s_j \le m \le S_j$:

$$p_{mn}^j = \begin{cases} P(D_j^R \ge m) & \text{if } n = 0, \\ P(D_j^R = m - n) & \text{if } 0 \le n \le m, \\ 0 & \text{if } m \le n \le S_j, \end{cases} \tag{8}$$

Then, like [21], we proceed to calculate the limiting probabilities of handling m units of product j at the review period R by solving the following system of equations:

$$\pi_{mj} = \sum_{l=0}^{S_j} \pi_{lj} p_{lm}^j \tag{9}$$

$$\sum_{m=0}^{S_j} \pi_{mj} = 1 \tag{10}$$

In order to facilitate calculations, we compute the shifted limiting probabilities, η_m, of handling m unit at the start of the shifted review point. These probabilities are calculated with the expressions bellow:

$$\eta_j = \begin{cases} \sum_{l=max(k,s_j+1)}^{S_j} \pi_l \; P(D_j^L \geq l) & \text{if } k = 0, \\ \sum_{l=max(k,s_j+1)}^{S_j} \pi_l \; P(D_j^L = l-k) & \text{if } 1 \leq k < S_j - s_j, \\ \sum_{l=max(k,s_j+1)}^{S_j} \pi_l \; P(D_j^L = l-k) + \pi_{s_j} P(D_j^L \geq s_j) & \text{if } k = S_j - s_j, \\ ml \sum_{l=max(k,s_j+1)}^{S_j} \pi_l \; P(D_j^L = l-k) + \sum_{l=S_j-k+1}^{s_j} \pi_l P(D_j^L = S_j - k) + \pi_{S_j-k,j} P(D_j^L \geq S_j - k) & \text{if } S_j - s_j \leq k < S_j, \\ \pi_{S_j} P(D_j^L = 0) + \sum_{l=1}^{s_j} \pi_l P(D_j^L = 0) & \text{if } k = S_j. \end{cases} \tag{11}$$

Now, we may calculate the expected stock on hand of product j $\bar{Y_j}$, given by the sum of the expected value of product at the end of the shifted review point η_j and the expected inventory level of the product j immediately after the receipt of delivers μ_{η_j}.

$$\bar{Y_j} = \frac{1}{R} \left[\sum_{r=1}^{R-1} \sum_{k=0}^{S_j} \eta_{kj} \sum_{l=1}^{k} l Pr\{D_j^r = k - l\} + \mu_{\eta_j} \right] \tag{12}$$

6 Numerical results

In order to carry out numerical analysis and evaluate the efficiency of the proposed solution, the model was programmed in Java environment and was tested using a 2.67 GHz Intel(R) Core(TM) i5 CPU system processor with 4.00 Go of RAM. The instance was composed of one *GS* and 3 products. To generate demand, we have collected daily orders data faced by the *GS* over five years, then we have used this data as an input of an algorithm, using uniform distribution and cumulative density function, that we have developed to generate the random variable: quantity of pharmaceutical items demanded per day. Based on the search procedure presented in the last section, we proceed to compute the optimal values of the replenishment policy parameters for each pharmaceutical item, which satisfy the required service level ($\alpha = 95\%$). The obtained results are summarized in tables (2-3).

Table 2. Inventory control policy parameters

	s_i^g	S_i^g	α_{m_g}
Product 1	267	1688	30,1%
Product 2	94	678	41,6%
Product 3	101	470	26,6%

To evaluate the performance of the actual inventory structure, we fix the value of the annual order frequency, $\gamma = 1$, and measured the corresponding file rate value. It is clear from the third column of table 2 that the storage space reserve for essential and vital drugs is insufficient which lead to a very low service level 26,6% to 41,6%. To prevent the shortage risk and organizational issues related to managing huge quantities of pharmaceutical items, two main solutions could be proposed, either enhancing the annual order frequency or extending the storage space.

To analyze the efficiency of the first proposal, we examine the file rate variations for several values of annual order frequency. Numerical results revealed that the value of the file rate enhances remarkably, still, the value of the order frequency is also highest for such a solution. Table 4 includes the obtained results. Note that the file rate value increases to 95% for $\gamma_1 = 3,5$. Similarly, the fill rate value of the second product enhances to 96% for $\gamma_2 = 3$ and to 98% for $\gamma_3 = 4$ concerning the third product. At this point,

Table 3. Objective function value

	h_1^g	h_2^g	h_3^g	Total cost
Cost	7106	278	9678	17062

we can deduce that the studied inventory structure, performing under the first proposal, enhances its fill rate by ordering small quantities more frequently, which is not acceptable by hospital managers. Indeed, to avoid managerial risks and rework related to logistics activities, the number of annual orders launched by hospitals should be as small as possible. Therefore, increasing storage capacity is more applicable.

Table 4. The service level and the order frequency for the three products

Product 1			Product 2			Product 3		
S_1	α_1	γ_1	S_2	α_2	γ_2	S_3	α_3	γ_3
1688	0,301	1,000	678	0,416	1,000	471	0,266	1,000
1487	0,354	1,210	644	0,441	1,250	437	0,292	1,200
1457	0,363	1,260	620	0,460	1,300	380	0,346	1,350
1334	0,405	1,360	591	0,485	1,350	350	0,381	1,500
1268	0,430	1,460	562	0,512	1,450	332	0,406	1,650
1209	0,456	1,510	538	0,537	1,500	297	0,462	1,800
1202	0,459	1,560	503	0,578	1,600	295	0,467	1,950
1155	0,481	1,660	481	0,608	1,700	283	0,488	2,050
1053	0,537	1,810	465	0,630	1,750	264	0,528	2,200
981	0,583	1,960	441	0,667	1,850	252	0,566	2,350
919	0,628	2,060	420	0,703	1,900	234	0,607	2,500
902	0,642	2,210	407	0,726	1,950	226	0,629	2,650
823	0,712	2,410	399	0,743	2,050	207	0,692	3,000
804	0,731	2,560	395	0,750	2,250	200	0,719	3,150
728	0,779	2,810	352	0,849	2,500	193	0,750	3,400
698	0,817	2,860	342	0,875	2,650	188	0,770	3,650
652	0,855	3,010	312	0,902	2,900	175	0,833	3,900
616	0,893	3,160	305	0,964	3,150	171	0,905	4,050
580	0,930	3,310	298	1,027	3,400	167	0,977	4,200
545	0,968	3,460	291	1,089	3,650	163	1,049	4,350
509	1,006	3,610	284	1,151	3,900	159	1,122	4,500
473	1,043	3,760	277	1,213	4,150	155	1,194	4,650

7 Conclusion

The primary purpose of this work is to identify the optimal replenishment parameters for inventory at the Ibn Sina hospital complex, Rabat, Morocco. We have focused on this study on managing inventories related to vital and essential pharmaceutical items by considering a stochastic demand, required file rate, limited storage space and restriction on the annual order frequency. An analytical program is proposed to formulate the studied inventory structure and a Discrete-Time Markov chain model is formulated to compute the expected holding cost. Besides its capacity to determine the closed-form solution, the proposed search procedure allows us to analyze the interactions between different performance characteristics.
There are multiple promising research directions to extend this work. Among which the following are noteworthy:

- Consider all types of pharmaceutical items: non-essential and non-vital drugs.
- Treat more constraints related to PSC characteristics such as expiry dates and storage environments.
- Address the distribution decision between the GS and all hospital departments.

References

1. Manuel, D. R., Nebil, B., and Edward, P. ”Chapter 10 Medical Supply Logistics” R. Hall (ed.), Handbook of Healthcare System Scheduling, International Series in Operations Research Management Science 168, DOI: 10/1007=978 ? 1 ? 4614 ? 1734 ? 710.
2. Moon, S. (2004) ”Taking cost off supply shelf. Mod Healthc.” 22;34(47):26-8.
3. Anita, R. V., and Julie, S. I. ”Chapter 17 Managing Supply Critical to Patient Care: An Introduction to Hospital Inventory Management for Pharmaceuticals.” Handbook of Healthcare Operations Management: Methods and Applications, International Series in Operations Research Management Science 184, DOI 10.1007/978-1-4614-5885-2 17.
4. Gregory, D., David, T., Vera, T. (2015). ”Optimizing the timing and number of batches for compounded sterile products in an in-hospital pharmacy.” Decision Support Systems Volume 76, August 2015, Pages 53-62.
5. Little, J., and Coughlan, B. (2008). ”Optimal inventory policy within hospital space constraints.” Health Care Manag Sci 11:177-183.
6. Peter, K., John, W., and Helmut, S. (2012).”Pharmaceutical supply chain specifics and inventory solutions for a hospital case.” Operations Research for Health Care 1 54-63.
7. Sven, A. (2005). ”A simple decision rule for decentralized two-echelon inventory control.”
8. Andersson, J., Marklund, J. (2000). ”Decentralized inventory control in a two-level distribution system.” Eur J Oper Res 127:483-506.
9. Kim, C.O., Jun, J., Baek, J.K., Smith, R.L., and Kim, Y.D. 2004. ”Adaptive inventory control models for supply chain management.” The International Journal of Advanced Manufacturing Technology, Volume 26, Issue 9-10, pp 1184-1192.
10. Khoukhi, S. Bojji, C. and Bensouda, Y. A review of medical distribution logistics in pharmaceutical supply chain. Int. J. Logistics Systems and Management, 34(3), 2019.
11. Qinglin, D., Warren, T. L. (2013). ”Optimization of replenishment policies for decentralized and centralized capacitated supply chains under various demands.” International Journal of Production Economics Volume 142, Issue 1, Pages 194-204.
12. Burns, L. (2002) ?Wharton school colleagues?, in The Health Care Value Chain Producers, Purchasers, and Providers, Jossey-Bass, San Francisco, California, ISBN: 978-0-7879-6021-6, 464pp.
13. Scarf, H., 1959. The optimality of (s,s) policies in the dynamic inventory problem. Mathematical Methods in the Social Sciences, 49.
14. Dellaert, N. and van de Poel, E. (1996). ”Global inventory control in an academic hospital.” Int. J. Production Economics. Vol. 46-47, No. 1, pp. 277-284.
15. Kelle, P., Woosley, J., Schneider, H. (2012). ”Pharmaceutical supply chain specifics and inventory solutions for a hospital case,” Operations Research for Health Care. Vol. 1, No.2, pp. 54?63
16. Schneider, H., Rinks, D.B. and Kelle, P. (1995). ”Power approximations for a two echelon inventory system using service levels.” Production and Operations Management, Vol. 4, No. 4, pp.381.
17. Rosales, C.R., Magazine, M. and Rao, U. (2014). ”Point-of-use hybrid inventory policy for hospitals.” Decis. Sci.Vol. 45, No. 6. pp. 913?937.
18. Khoukhi, S.Bojji, C. and Bensouda, Y. Optimal inventory control policy for a hospital case.Conference Paper ICOA., 2019.
19. Nelder, J.A., and Mead, R.(1965). ”A simplex method for function minimization.” Computer Journal, Vol. 7, No. 4, pp. 308?313.
20. Bijvank, M., and Vis, I. F. A. (2012). ”Inventory control for point-of-use locations inhospitals.” Journal of the Operational Research Society, Vol. 63, No. 4, pp. 497-510.
21. Kapalka, B.A., Katircioglu, K. and Puterman, M.L. (1999). ”Retail inventory control with lost sales, service constraints, and fractional lead times.” Prod Opns Mngt. Vol.8, pp. 393?408.
22. Little J, Coughlan B (2008) Optimal inventory policy within hospital spaceconstraints. Health Care Manag Sci 11:177?183.
23. Saedi, S. Kundakcioglu, O. E. and Henry, A.C.(2016). ”Mitigating the impact of drug shortages for a healthcare facility: An inventory management approach.” European Journal of Operational Research, Vol. 251, pp.107-123.

Extended path-relinking method for p-location problem

Jaroslav Janáček, Marek Kvet[1]

1. University of Žilina, Faculty of Management Science and Informatics
Univerzitná 8215/1, 010 26 Žilina, Slovakia
{jaroslav.janacek, marek.kvet}@fri.uniza.sk

Abstract

Most of the searching strategies based on path-relinking method usage are restricted by a drawback of the method. The drawback of the original path-relinking method consists in its way of processing the pair of input solutions. The path-relinking method applied to zero-one programming problems examines one of the shortest paths connecting the input solutions in the surface of a unit hypercube. This characteristic does not enable to examine any feasible solutions outside the sub-space determined by components, in which the input solutions differ. Within our research directed to heuristics for the public service system design problems, we suggested a new type of the path-relinking method, which is able to overcome the above-mentioned drawback. The novelty consists in determination of an infeasible solution of the *p*-location problem, which corresponds to a hypercube vertex with more than *p*-components, and in projection of a starting feasible solution in the set of the feasible solutions, which are the closest ones to the infeasible solution. The suggested path-relinking projective method was embedded into a simple one-to-all searching strategy and its efficiency dependent on infeasibility level of the infeasible solution was studied.

Keywords: Location problems, heuristics, path-relinking method extension.

1 Introduction

The existence of human society has been always associated with decisions. Making more or less important decisions accompanies us in various areas of everyday life, although many times we are not even aware of it. We often encounter the requirement to find the optimal solution to a particular problem or to improve the current situation as much as possible. The main reasons for such rationalization include reducing costs and increasing efficiency. Choosing the right alternative from all solutions is not easy and involves a great deal of responsibility. The final decision may not affect only our personal lives, but also the lives of a certain group of people or even the whole society [16]. Another factor that needs to be taken into account when making a decision is the time aspect. The consequences of a decision can be very long. In this paper, we focus only on a strategic level of decision-making process. The time lag of strategic decisions is usually in the order of several years. Most often, these are large-scale investment projects, such as the construction of new companies, the location of distribution centers, or the design of various service systems. The research reported in this paper aims at applying the knowledge of Applied Informatics and programming in the location science, mainly to the healthcare segment [2, 4, 13].

The operation of the emergency medical service is one of the basic services by which the state protects its inhabitants and provides them with urgent care in critical situations [13]. The main role of each manager responsible for the efficiency of the service is to decide on the location of service centers. Centers, which can be, for example, warehouses, terminals, or specialized medical facilities, form the structure of the proposed system [16]. This structure plays an essential role in the efficiency of the system performance. Strategic decisions on the location of facilities so that the total costs are kept to be minimal or the service accessibility for patients to be as high as possible, represent a complex combinatorial problem, the solution of which can achieve significant savings or improve the quality of the service provided. Since the resources, which are to be located, are limited, the mathematical model used for the decision/making often follows the weighted *p*-median problem formulation [1, 7, 14]. To make the model more general, the concept of so-called generalized disutility has been introduced to consider also such requirements, which allow providing the service to a patient from more than one nearest located service centers. Even if this model extension makes the problem harder to be solved, it enables us to apply its results into a wider range of systems [9, 11, 15].

Wide range of practical applications of the weighted p-median problem not only in the medical sphere [13, 14, 16] has led to the creation of a large number of solving approaches, which include exact as well as heuristic and metaheuristic methods [1, 5, 6, 11, 19, 20].

Exact algorithms are based mostly on the branch and bound method. Sometimes, they may make use of the principles of duality. Their main disadvantage consists in their capacity limitation caused by commonly available universal optimization environments, to which the exact methods are embedded. Mentioned restriction does not usually allow us to solve problems of practical and real world size. On the other hand, there is a radial formulation of the problem [7, 14], which enables us to overcome this weakness. Other approach consists in developing a special software tool. Therefore, many Operations Research scientists and other authors focus mainly on heuristic and metaheuristic approaches [17, 18, 21].

Currently, the main attention is paid to various metaheuristic approaches, i.e. genetic algorithms, scatter search, path-relinking method and many others, the aim of which can be specified as a task of obtaining a good solution in acceptably short computational time. In this paper we report our research, which was aimed at extending the path-relinking method. This approach proved to be suitable mainly in the case of the generalized weighted p-median problem, in which the demands for service are assumed to occur randomly. It must be noted that the original path-relinking method inspects only the shortest path between two solutions. The scientific effort reported in this paper was aimed at suggesting such a version, which could project a starting solution into a feasible solution, which is the closest one to a given vertex of a unit hypercube regardless of its infeasibility. The suggested path-relinking projective method was embedded into a simple one-to-all search strategy and its efficiency depending on infeasibility level of the infeasible solution was studied.

2 Path-relinking method and its applications

The original path-relinking method was suggested to enable heuristic solution of the problems, which can be described by zero-one mathematical programming tools [8]. A general zero-one programming problem can be formulated by (1).

$$\min\left\{f(\mathbf{x}) : \mathbf{x} \in \mathbf{X} \subseteq \{0,1\}^m\right\} \tag{1}$$

The idea of the method consists in searching one of the shortest paths connecting two input feasible solutions – vertices of an m-dimensional hypercube and returning the best-found-solution, which lies on the path. The hypercube vertices of the path correspond to m-dimensional vectors, components of which take values of one or zero. The sequential search along the shortest path is performed by a move from a currently occupied solution to a neighboring one, which differs from the occupied solution only in a value of one component. In addition, this component must belong to the set of components, which take different values in the vectors describing the input solutions. The original path-relinking method proceeds in accordance to the following algorithm applied to a pair $\mathbf{x}$, $\mathbf{y}$ of input solutions – m-dimensional zero-one vectors.

```
0. Define set $D$ of components, in which $\mathbf{x}$ and $\mathbf{y}$ differ, i.e. $D = \{i= 1, \dots, m: x_i \neq y_i\}$. Initialize $\mathbf{x}^{best}$ by
   $\mathbf{x}^{best} = argmin\{f(\mathbf{x}), f(\mathbf{y})\}$.
1. If $|D| > 1$ go to 2, otherwise go to 3.
2. Determine $d \in D$ by $d = argmin\{f(inv(\mathbf{x}, i)): i \in D\}$ and perform $\mathbf{x} = inv(\mathbf{x}, d)$, $D = D - \{d\}$.
   If $\mathbf{x} \in \mathbf{X}$ then update $\mathbf{x}^{best}$ by $\mathbf{x}^{best} = argmin\{f(\mathbf{x}^{best}), f(\mathbf{x})\}$.
3. Return $\mathbf{x}^{best}$ and terminate.
```

Comment: The operation $inv(\mathbf{x}, i)$ performed with m-dimensional zero-one vector $\mathbf{x}$ and subscript i from the domain 1, …, m returns vector $\underline{\mathbf{x}}$, components of which are defined as follows $\underline{x}_i = x_i$ for $i = 1, \dots, m$, $i \neq d$ and $\underline{x}_d = 1 - x_d$.

The cardinality $|D|$ of the initial set D corresponds to the Hamming or Manhattan distance of the input solutions $\mathbf{x}$ and $\mathbf{y}$, and $|D|$ - 1 of inner vertices is the number of inner vertices on the shortest path connecting the input solutions in the surface of the m-dimensional unit hypercube. Efficiency of the path examination is obviously influenced by the number of feasible solutions inspected during the examination.

If a kind of p-location problem is considered, e.g. the weighted p-median problem or the emergency service system design problem with p service centers, then the set **X** of all feasible solutions is defined by (2).

$$\mathbf{X} = \left\{ \mathbf{x} \in \{0,1\}^m : \sum_{i=1}^{m} x_i = p \right\} \tag{2}$$

Applying the above original version of the path-relinking method to the p-location problem, it can be found that at least every second vertex of the examined path will be inadmissible or infeasible solution. It means that the associated vector **x** will have less or more than p non-zero components. That is why, a more efficient version of the path-relinking method was suggested to solve problem (2). The new version avoids the weird vertices of the hypercube and inspects only feasible solutions of (2).

This adjusted path-relinking mod performs according to the following scheme.

0. Define sets D and E of components, in which take the value of one only in one of the input solutions **x** and **y**. $D = \{i = 1, \dots, m: x_i = 1 \text{ and } y_i = 0\}$ and $E = \{i = 1, \dots, m: x_i = 0 \text{ and } y_i = 1\}$. Initialize $\mathbf{x}^{\text{best}}$ by $\mathbf{x}^{\text{best}} = argmin\{f(\mathbf{x}), f(\mathbf{y})\}$.

1. If $|D| > 1$ go to 2, otherwise go to 3.

2. Determine $d \in D$ and $e \in E$ by $[d, e] = argmin\{f(swap(\mathbf{x}, i, j)): [i, j] \in D \times E\}$ and perform
$\mathbf{x} = swap(\mathbf{x}, d, e)$): $D = D - \{d\}$, $E = E - \{e\}$, and update $\mathbf{x}^{\text{best}} = argmin\{f(\mathbf{x}^{\text{best}}), f(\mathbf{x})\}$. Go to 1.

3. Return $\mathbf{x}^{\text{best}}$ and terminate.

Comment: The operation $swap(\mathbf{x}, d, e)$ performed with m-dimensional zero-one vector **x** and subscripts d and e from the domain 1, …, m, for which $x_d = 1$ and $x_e = 0$ returns vector $\underline{\mathbf{x}}$, components of which are defined as follows $\underline{x}_i = x_i$ for $i = 1, \dots, m$, $i \neq d$ and $i \neq e$. Furthermore $\underline{x}_d = 0$ and $\underline{x}_e = 1$.

The above-described path-relinking method proved to be an excellent tool when embedded into a searching scheme of a discrete version of particle swarm optimization. Nevertheless, the domain of examined solutions stays restricted by the initial deployment of swarm particles and the system of shortest paths among them. To overcome this disadvantage of the method, we suggested an extended version of the path-relinking method described in the next section.

3 Concept of projection and path-relinking method extension

The idea of extension is based on the m-dimensional unit hypercube geometry, where the set of feasible solutions (2) corresponds to a sub-set of the hypercube vertices, which lie in the intersection of the hypercube and a facet of the simplex determined by the constraint in (2).

Let us consider a vertex **v** of the hypercube, which does not belong to set of feasible solutions due to the number of its non-zero components exceeds the value of p. The vertex **v** induces a set $F(\mathbf{v})$ of feasible p-location problem solutions, which are the closest ones to the vertex **v** in terms of Hamming distance. As the vertex **v** has q non-zero components and $q > p$, the minimal Hamming distance equals to $q - p$.

Now, using the path-relinking principle, an input solution **x** will be projected to the set $F(\mathbf{v})$ and the best-found-solution of the shortest path from **x** and the set $F(\mathbf{v})$ will be an output of the procedure. Using the above introduced denotation, the extended path relinking method can be described by the following algorithm, input of which is a feasible solution **x** and an infeasible hypercube vertex **v** with q, $q > p$ components.

ExtendedPathRelinking(**x**, **v**)

0. Define $D = \{i = 1, \dots, m: x_i = 1 \text{ and } v_i = 0\}$ and $E = \{i = 1, \dots, m: x_i = 0 \text{ and } v_i = 1\}$. Initialize $\mathbf{x}^{\text{best}}$ by **x**.

1. If $|D| > 1$ go to 2, otherwise return **x** and terminate.

2. Determine $[d, e] \in D \times E$ by $[d, e] = argmin\{f(swap(\mathbf{x}, i, j)): [i, j] \in D \times E\}$ and update
$\mathbf{x} = swap(\mathbf{x}, d, e)$): $D = D - \{d\}$, $E = E - \{e\}$, and $\mathbf{x}^{\text{best}} = argmin\{f(\mathbf{x}^{\text{best}}), f(\mathbf{x})\}$. Go to 1.

This extended path-relinking method can be employed in a simple version of a discrete particle swarm optimization algorithm [3, 21] with strategy one-to-all as follows.

Let **x** is a starting feasible solution of the solved p-location problem and V is a finite set of hypercube vertices, where each of them has more than p non-zero components. Then the searching strategy follows the next commands:

One-to-allSearch(**x**, V)

While V ≠ ∅ do: Withdraw a **v** from V, update V = V −{**v**} and **x** = *ExtendedPathRelinking*(**x**, **v**). If V = ∅, then terminate the search and return **x**.

4 Numerical experiments

To verify the extended path-relinking method, the medical emergency system design instances were used as benchmarks. The problem is formulated as a task to choose p centers out of the set of m possible center locations so that the objective function f is minimal. The collection of p chosen center locations can be described by an m-dimensional zero-one vector $\mathbf{x}\in\mathbf{X}$. Then, (3) can define the objective function f for the above-described problem. The formula expresses sum of mean distances from a system user j to the nearest available service center.

$$f(\mathbf{x})=\sum_{j=1}^{n} b_j \sum_{k=1}^{r} q_k \min_k \left\{d_{ij} : i=1, ..., m,\ x_i=1\right\} \tag{3}$$

In the formulation (3), the operator $\min_k\{\}$ returns the k-th minimal value of the set {}. The function f is computed for n system users, where b_j denotes a number of user's demands, which are located at j and must be serviced from the nearest available service center. The time-distance between a user location j and a possible service center location i is denoted by symbol d_{ij}. The coefficients q_k, k = 1, …, r stand for probabilities that the k-th nearest service center is the closest available one. This problem description corresponds to the concept of emergency service system design, in which the system operates as a queuing system with p service lines. The system is characterized by a demand assignment strategy following the idea that a randomly emerged demand for service is assigned to the nearest service center only if the center is not occupied by an earlier demand. In the opposite case, the nearest non-occupied center provides the user with service [9, 11, 15].

Computational study reported in this paper was performed on benchmarks derived from real emergency medical service system implemented in eight self-governing regions of the Slovak Republic. These problem instances were used also in our previous research published in [10, 11]. The individual instances are denoted by the names of capitals of the particular regions, which are reported by abbreviations of the region denotations. The list of instances consists of Bratislava (BA), Banská Bystrica (BB), Košice (KE), Nitra (NR), Prešov (PO), Trenčín (TN), Trnava (TT) and Žilina (ZA). The sizes of the individual benchmarks are m and p introduced above. Mentioned basic characteristics of all used benchmarks are reported in the left part of Table I. The coefficients b_j used in the objective function (3) correspond to the number of inhabitants of individual communities rounded up to hundreds. The coefficients q_k for k=1…3 of the generalized objective function (3) were set according to [12] at the values: q_1 = 0,77063, q_2 = 0,16476 and q_3 = 1 - q_1 - q_2. These values were obtained from a simulation model of existing emergency medical system in Slovakia. The middle part of the table consists the objective function value (3) of the optimal solution denoted by *OptSol* together with the computational time in seconds denoted by *CT* [s], in which the optimal solution was obtained. The right part of Table I is devoted to the characteristics of the uniformly deployed sets as described in [10]. We report their cardinalities $|S|$ and minimal Hamming distance h. The uniformly deployed sets of solutions were used in the suggested solving heuristics as a source of feasible solutions of the problem. The process of uniformly deployed set construction and usage are reported in [10] and [11].

Table 1 Basic benchmarks characteristics, the optimal objective function values and uniformly deployed sets sizes

Region	m	p	**Optimal solution**		**Uniformly deployed set**	
			OptSol	*CT* [s]	$\|S\|$	h
BA	87	14	26650	0.35	23	2
BB	515	36	44752	10.57	172	3
KE	460	32	45588	7.58	60	2
NR	350	27	48940	19.21	83	2
PO	664	32	56704	76.53	232	2
TN	276	21	35275	4.04	137	2
TT	249	18	41338	2.79	212	2
ZA	315	29	42110	2.70	112	3

To construct the series V of the infeasible hypercube vertices for individual benchmarks, we ordered the corresponding uniformly deployed set S of p-location solutions according to objective function values. We used the best solution as the initial solution $\mathbf{x}$ and then, we grouped the remaining solutions to disjoint pairs, triples and quadruples. Each created group $\{\mathbf{x}^u: u = 1, \ldots, t\}$of t solutions gave one vertex $\mathbf{v}$, components of which were determined according to $v_i = max\{x_i^u : u = 1, \ldots, t\}$. This way we solved four cases, where the first one did not use the infeasible vertices, but feasible solutions of S. The second case consisted of vertices obtained from pairs and thus $|V| = |S|/2$. In the third and fourth case the infeasible vertices were constructed from triples and quadruples respectively and cardinalities of V equaled to $|S|/t$ for t=3, 4.

The main goal of this computational study is to verify the impact of the cardinality of V on the results measured by computational time in seconds and the solution accuracy. Since the optimal objective function values of all studied benchmarks are available and published in [11], the quality of the resulting system design is here evaluated by *gap*, which expresses a relative difference of the obtained objective function value from the optimal one. Its value is reported in percentage, where the optimal objective function value was taken as the base. Obviously, we provide also the computational time *CT* in seconds.

To achieve the main goal of numerical experiments, a sufficient set of problems and uniformly deployed sets of solutions must be considered. To make the comparison relevant and robust enough, we followed from a very useful property of any uniformly deployed set of solutions. The mentioned useful feature consists in the fact that any arbitrary permutation of $\underline{m}$ locations subscripts brings a new set with the same parameters. This way, we were able to obtain ten different sets for each problem instance. The results are summarized in the following tables.

Table 2 contains the average results of ten instances solved for different uniformly deployed sets of solutions for each self-governing region.

For completeness, let us add the information that the numerical experiments were run on a PC equipped with the Intel® Core™ i7 3610QM 2.3 GHz processor and 8 GB of RAM. The algorithms were implemented in the Java language making use of the NetBeans IDE 8.2 environment.

Table 2 Average results of numerical experiments for the self-governing regions of Slovakia

Region	$\|V\| = \|S\|$		$\|V\| = \|S\| / 2$		$\|V\| = \|S\| / 3$		$\|V\| = \|S\| / 4$	
	gap	*CT*	*gap*	*CT*	*gap*	*CT*	*gap*	*CT*
BA	1.19	0.19	1.81	0.14	2.32	0.09	2.97	0.07
BB	0.30	24.15	0.35	24.60	0.36	21.36	0.30	18.37
KE	0.37	13.90	0.37	13.55	0.36	11.63	0.55	9.68
NR	0.18	6.60	1.52	6.18	0.30	5.27	1.67	4.29
PO	0.45	16.17	0.59	16.86	4.91	15.39	4.89	13.96
TN	1.38	2.54	1.51	2.37	1.71	2.10	1.99	1.77
TT	0.23	1.54	0.10	1.43	0.14	1.21	0.12	1.00
ZA	0.07	7.00	0.05	6.55	0.05	5.47	0.05	4.47

The following Table 3 contains the results of the best run out of ten computations for each benchmark, in which the lowest value of the objective function (3) was achieved.

Table 3 The best results of numerical experiments for the self-governing regions of Slovakia (minimal objective function value of ten runs was taken into account)

Region	$\lvert V\rvert = \lvert S\rvert$		$\lvert V\rvert = \lvert S\rvert / 2$		$\lvert V\rvert = \lvert S\rvert / 3$		$\lvert V\rvert = \lvert S\rvert / 4$	
	gap	*CT*	*gap*	*CT*	*gap*	*CT*	*gap*	*CT*
BA	0.00	0.19	0.65	0.14	0.68	0.08	1.12	0.06
BB	0.00	24.85	0.00	26.37	0.00	22.71	0.00	18.80
KE	0.00	14.36	0.00	13.27	0.00	11.59	0.00	9.40
NR	0.05	6.56	0.09	6.20	0.05	5.31	0.62	4.32
PO	0.03	15.93	0.12	16.77	3.57	15.17	3.57	13.73
TN	0.00	2.40	0.14	2.30	0.54	1.96	0.51	1.66
TT	0.00	1.57	0.00	1.43	0.00	1.24	0.00	1.07
ZA	0.00	7.05	0.00	6.55	0.00	5.57	0.00	4.95

5 Conclusions

The main purpose of this paper was to provide the readers with an effective heuristic method for solving middle and large instances of the weighted *p*-median problem, which finds its application in many different areas including medical sphere and many other subfields of location science. To make the solving approach applicable in a wider range, developed algorithm is able to cope with generalized objective function. The generalization consists in more service centers, which can provide the service to the system user and not only the nearest located center needs to be considered.

Suggested method is based on the former path-relinking method. The drawback of the original path-relinking method consists in its way of processing the pair of input solutions. Mentioned weakness was overcome and the reported computational results prove that most of the instances were solved either to optimality or the resulting solution was very near to the optimal one. The novelty of presented original method extension consists in determination of an infeasible solution of the *p*-location problem, which corresponds to a hypercube vertex with more than p-components, and in projection of a starting feasible solution in the set of the feasible solutions, which are the closest ones to the infeasible solution.

Based on performed numerical experiments we can conclude that we have constructed a very fast and effective heuristic approach to the p-location problems.

Future research in this scientific field could be concentrated on rules, which would enable to reduce the starting set of *p*-location problem solutions.

Acknowledgment

This work was supported by the research grants VEGA 1/0089/19 “Data analysis methods and decisions support tools for service systems supporting electric vehicles”, VEGA 1/0689/19 “Optimal design and economically efficient charging infrastructure deployment for electric buses in public transportation of smart cities”, and VEGA 1/0216/21 “Design of emergency systems with conflicting criteria using artificial intelligence tools”. This work was supported by the Slovak Research and Development Agency under the Contract no. APVV-19-0441.

References

[1] Avella, P., Sassano, A., Vasil'ev, I. (2007). Computational study of large scale p-median problems. Mathematical Programming 109, pp. 89-114.

[2] Current, J., Daskin, M., Schilling, D. (2002). Discrete network location models, Drezner Z. et al. (ed) Facility location: Applications and theory, Springer, pp. 81-118.

[3] Davendra, D., Zelinka, I. (2016). Self-Organizing Migrating Algorithm, Methodology and Implementation. Springer, Studies in Computational Intelligence, 289 p.

[4] Doerner, K. F., Gutjahr, W. J., Hartl, R. F., Karall, M., Reimann, M. (2005). Heuristic Solution of an Extended Double-Coverage Ambulance Location Problem for Austria. Central European Journal of Operations Research, 13(4), pp. 325-340.

[5] Drezner, T., Drezner, Z. (2007). The gravity p-median model. European Journal of Operational Research 179, pp. 1239-1251.

[6] Elloumi, S., Labbé, M., Pochet, Y. (2004). A new formulation and resolution method for the p-center problem. INFORMS Journal on Computing, 16(1), pp. 84-94.

[7] García, S., Labbé, M., Marín, A. (2011). Solving large p-median problems with a radius formulation. INFORMS Journal on Computing, 23(4), pp. 546-556.

[8] Gendreau, M., Potvin, J. (2010). Handbook of Metaheuristics, Springer Science & Business Media.

[9] Janáček, J., Kvet, M. (2016). Min-max Optimization and the Radial Approach to the Public Service System Design with Generalized Utility. In Croatian Operational Research Review, Vol. 7, Num. 1, pp. 49-61.

[10] Janáček, J., Kvet, M. (2019). Uniform Deployment of the p-Location Problem Solutions. In: Operations Research Proceedings 2019: Selected Papers of the Annual International Conference of the German Operations Research Society (GOR), Dresden, Germany, September 4-6, 2019: Springer, 2020, ISBN 978-3-030-48438-5, ISSN 0721-5924, pp. 315-321.

[11] Janáček, J., Kvet, M. (2019). Usage of Uniformly Deployed Set for p-Location Min-Sum Problem with Generalized Disutility. In: SOR 2019 proceedings, pp. 494-499.

[12] Jankovič, P. (2016). Calculating Reduction Coefficients for Optimization of Emergency Service System Using Microscopic Simulation Model. In: 17th International Symposium on Computational Intelligence and Informatics, pp. 163-167.

[13] Jánošíková, Ľ., Žarnay, M. (2014). Location of emergency stations as the capacitated p-median problem. In: Quantitative Methods in Economics (Multiple Criteria Decision Making XVII). pp. 117-123.

[14] Kozel, P. Orlíková, L., Pomp, M., Michalcová, Š. (2018). Application of the p-median approach for a basic decomposition of a set of vertices to service vehicles routing design. In Mathematical Methods in Economnics 2018, Praha: MatfyzPress, 2018, pp. 252-257.

[15] Kvet, M. (2014). Computational Study of Radial Approach to Public Service System Design with Generalized Utility. In: Proceedings of International Conference Digital Technologies 2014, Žilina, Slovakia, pp. 198-208.

[16] Marianov, V., Serra, D. (2002). Location problems in the public sector, Facility location - Applications and theory (Z. Drezner ed.), Berlin, Springer, pp 119-150.

[17] Rybičková, A., Burketová, A., Mocková, D. (2016). Solution to the locating – routing problem using a genetic algorithm. In: SmaRTT Cities Symposiuum Prague (SCSP), pp. 1-6.

[18] Rybičková A., Mocková D., Teichmann D. (2019). Genetic Algorithm for the Continuous Location-Routing Problem, Neural Network World 29(3), pp. 173–187.

[19] Sayah, D., Irnich, S. (2016). A new compact formulation for the discrete p-dispersion problem. European Journal of Operational Research, 256(1), pp. 62-67.

[20] Snyder, L. V., Daskin, M. S. (2005). Reliability models for facility location; The expected failure cost case, Transport Science 39 (3), pp. 400-416.

[21] Zelinka, I. (2016). SOMA-Self –organizing Migrating Algorith, ed. Davendra D., Zelinka, I.: Self-Organizing Migrating Algorithm-Methodology and Implementation, Springer, pp. 3-49.

Discrete PSO strategies for search in unit lattice of m-dimensional simplex

Marek Kvet, Jaroslav Janáček[1]

1. University of Žilina, Faculty of Management Science and Informatics
Univerzitná 8215/1, 010 26 Žilina, Slovakia
{marek.kvet, jaroslav.janacek}@fri.uniza.sk

Abstract

The path-relinking based strategies have proved to be very powerful tool for designing the emergency service systems by deploying a given number of service centers in a finite set of possible center locations. Nevertheless, if the original approach to the emergency service system design is generalized to the case, when more than one facility can be placed at the same possible center location, the question emerges whether the generalized version of the path-relinking method is able to keep its former efficiency. It must be taken into account that the generalized path-relinking method performs its search in nodes of integer lattice of an *m*-dimensional simplex instead of in a sub-set of unit hypercube vertices. This generalization may considerably change characteristics of the path-relinking based searching strategies. This contribution is devoted to studying and comparing two original particle swarm strategies called the shrinking fence and spider search strategies, which employ the generalized path-relinking method.

Keywords: Emergency Medical Service System, heuristics, generalized path-relinking method, discrete PSO strategies

1 Introduction

Applied Informatics belongs to one of the currently fastest developing scientific fields. It deals with creation, collection, processing, storage, transformation, access and usage of any kind of information in natural and artificial, general and special systems. Its content aims at the properties and methods of information processing in terms of their optimal availability and usability. It has purely scientific (theoretical) components that examine the subject regardless of the application, and application (practical) components that contribute to the development of services and products. In this paper, we focus on applying the knowledge of Applied Informatics and programming into the specific family of Operations Research problems. We pay attention to the problem of designing and optimizing a network of rescue service stations in a middle sized geographical region [1, 17, 19]. In other words presented research deals with certain type of location problems, for which suitable heuristics are being developed and studied.

Generally, the locations problems may be divided into two independent groups – they can be either continuous or discrete. When talking about finding the optimal service center deployment for Emergency Medical Service (EMS), then we have to consider such a fact, that the service centers cannot be located anywhere due to certain requirements given by law. Therefore, the problem of finding the optimal locations of EMS stations is usually formulated as median-type problem, which has been recently studied by many researchers [2, 4, 5, 7, 13, 18].

The simplest median-based model is the weighted *p*-median problem with a wide spectrum of applications. Since it belongs to well-known and commonly used optimization models, several authors have analyzed the possibilities of its fast solving either by exact or approximate and heuristic methods [1, 6, 14].

Under the assumption that the service is not possible to be provided to more than one patient simultaneously by the same staff, the EMS systems operates as a queuing system. Obviously, when life is directly endangered or health gets suddenly worse, the rescue service is provided by such a facility, which is the nearest available one. From this point of view, the concept of so-called generalized disutility can be used [10, 12, 15].

Furthermore, if we look at an existing EMS system, we can observe that there are more than one facilities and staff located at some service center locations. Thus, we should consider this feature when a mathematical model of the problem is being formulated [8]. Of course, such an original model modification makes the problem more complex and possible usage of common exact and heuristic approaches designed for the median-type problems is questionable.

Within this paper, we study the path-relinking based strategies, which have proved to be very powerful tool for designing the emergency service systems by deploying a given number of service centers in a finite set of possible center locations. The main goal of presented research consists in answering the question whether the generalized version of the path-relinking method is able to keep its former efficiency. It must be taken into account that the generalized path-relinking method performs its search in nodes of integer lattice of an *m*-dimensional simplex instead of exploring a sub-set of unit hypercube vertices. This generalization may considerably change characteristics of the path-relinking based searching strategies. Therefore, we concentrate on two original Particle Swarm Optimization (PSO) strategies called the shrinking fence and spider search strategies, which employ the generalized path-relinking method [11, 16]. To study suggested heuristic approaches, a computational study with real world middle-sized problem instances was performed and the obtained results are reported in a separate section.

2 Generalized p-facility location problem and path-relinking search

The generalized *p*-facility location problem is formulated as a task to deploy *p* facilities in *m* network nodes so that the mean distance between a user and the nearest available facility is minimal. It is assumed that the system of facilities services demands of *n* users located also at nodes of the transportation network. A user *j* generates his demand randomly with frequency b_j. As the system processes the demands for service similarly to a queuing system equipped with *p* service lines, a current demand is assigned to the nearest available facility, which need not be the closest one. For each user location, *r* nearest facilities is taken into account for the demand satisfaction and a sequence $q_1, \ldots, q_r$ probability values is considered, where q_k is probability that the *k*-th nearest facility to the user is the first available one. Unlike the previous approaches, here we admit that more than one facility can be located in one node of the network. Assuming that d_{ij} denotes the network distance between network nodes *i* and *j* the generalized *p*-facility location problem can be described by the following integer programming model, in which a series of integer location variables $x_i \in Z^+$ will be introduced for each $i = 1, \ldots, m$, to model the number of facilities located at the node *i*.. The value of variable y_i gives the number of facilities located at location *i*. In addition, a series of allocation variables $w_{ijk} \in \{0, 1\}$ will be introduced for $i = 1, \ldots, m$, $j = 1, \ldots, n$ and $k = 1, \ldots, r$, where $w_{ijk} = 1$ if user demand emerged at *j* is assigned to a service node *i* for the *k*-th nearest facility.

$$Minimize \quad \sum_{j=1}^{n} b_j \sum_{k=1}^{r} q_k \sum_{i=1}^{m} d_{ij} w_{ijk} \tag{1}$$

$$Subject\ to \quad \sum_{i=1}^{m} x_i = p \tag{2}$$

$$\sum_{i=1}^{m} w_{ijk} = 1 \quad for \quad j = 1, \ldots, m,\ k = 1, \ldots, r \tag{3}$$

$$\sum_{k=1}^{r} w_{ijk} \le x_i \quad for \quad j = 1, \ldots, n,\ i = 1, \ldots, m \tag{4}$$

$$x_i \in Z^+ \quad for \quad i = 1, \ldots, m \tag{5}$$

$$w_{ijk} \in \{0, 1\} \quad for\ i = 1, \ldots, m,\ j = 1, \ldots, n,\ k = 1, \ldots, r \tag{6}$$

The formula (1) expresses the sum of mean distances from users' locations to the nearest available facility location. As the sequence of $\{q_k\}$ is decreasing, a demand of a user's location *j* is assigned to the nearest facility location *i* for $k = 1$. Similarly, the demand will be assigned to the second nearest facility for the case $k = 2$, etc.

Constraint (2) determines the number of deployed facilities. Series of constraints (3) ensures that demand at location *j* can be allocated to exactly one facility for the case *k*. This means that the *k*-th nearest facility is the first available one. Series of constraints (4) enables to assign a demand at user's location *j* to a possible service node *i* at most x_i times.

The problem (1)-(6) is more complex than the case, when only one facility can be located at a service node. The study reported in [8] has showed that computational time necessary to solve the problem (1)-(6) to optimality using an IP-solver exceeded an acceptable limit. This finding approves usage of a heuristic approaches to the problem solution. We were inspired by discrete particle swarm optimization algorithms [3, 20], which proved to be an efficient tool for this kind of *p*-location problem, but without the possibility to place more than one facility at the same service node.

The mentioned algorithms [11, 16] called the shrinking fence and the spider search are based on systematic examination of a series of the shortest paths connecting pairs of feasible hypercube vertices in a surface of the unit hypercube. To be able to use the above mentioned searching strategies for heuristic solution of the problem (1)-(6), we suggested a new version of the path-relinking method, which is able to examine the shortest path between two nodes of a unit lattice of an *m*-dimensional simplex.

A feasible solution of (1)-(6) is described by an *m*-dimensional vector **x** with integer non-negative components, sum of which equals to *p*. The shortest path between two feasible solutions has a length, which equals to Manhattan distance of the two vectors. The suggested path-relinking method proceeds according to the following steps.

FacetPathRelinking(**x**, **y**)

0. Initialize $\mathbf{x}^{\mathbf{res}} = argmin\{f(\mathbf{x}), f(\mathbf{y})\}$. Define sets M^+ and M^- of component indices by prescriptions $M^+ = \{i = 1, \ldots, m: x_i < y_i\}$ and $M^- = \{i = 1, \ldots, m: x_i > y_i\}$.

1. If $\rho(\mathbf{x}, \mathbf{y}) > 2$ perform step 2, otherwise return $\mathbf{x}^{\mathbf{res}}$ and terminate.

2. Find $[u, v] \in M^+ \times M^-$ using the definition
$[u, v] = argmin\{f(\text{exchange}(\mathbf{x}, i, j)): [i, j] \in M^+ \times M^-\}$ and perform operations
$\mathbf{x} = \text{exchange}(\mathbf{x}, u, v))$; if $x_u = y_u$, then $M^+ = M^+ - \{u\}$; if $x_v = y_v$, then $M^- = M^- - \{v\}$; $\mathbf{x}^{\mathbf{res}} = argmin\{f(\mathbf{x}^{\mathbf{res}}), f(\mathbf{x})\}$.
Having performed the above adjustments, exchange **x** with **y** and M^+ with M^- and go to step 1.

Comments: In the above algorithm, $\rho(\mathbf{x}, \mathbf{y})$ denotes the Manhattan distance of **x** and **y** defined by (7).

$$\rho(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{m} | x_i - y_i | \tag{7}$$

The operation exchange(**x**, *u*, *v*) for $u \in M^+$ and $v \in M^-$ issues the vector $\underline{\mathbf{x}}$, components of which are defined by the following substitutions $\underline{x}_i = x_i$ for $i=1, \ldots, m$, $i \neq u$, $i \neq v$ and $\underline{x}_u = x_u + 1$, $\underline{x}_v = x_v - 1$.

The value of function $f(\mathbf{x})$ for a given **x** is computed according to (1)-(6) after fixing the values of x_i for $i = 1, \ldots, m$.

The algorithm *FacetPathRelinking*(**x**, **y**) examines the shortest path connecting integer points **x** and **y** in an *m*-1 dimensional facet of simplex determined by (2) and (5). The value $\rho(\mathbf{x}, \mathbf{y})$ is obviously even integer and every performance of the step 2 reduces this distance by two. Thanks to the exchange **x** and **y** at the end of step 2, the algorithm constructs and examines the path alternately from the both ends.

3 Particle swarm strategies based on path-relinking method

Principles of the further applied strategies were obtained from [11, 16] and adapted for the search in the set of feasible solutions of the problem (1)-(6) using the above suggested version of the path-relinking method. The both proposed algorithms start with an initial swarm *S* of input solutions-particles and use the path-relinking method as a function *FacetParthRelinking*(**x**, **y**), which returns the best-found-solution in the examined path.

The shrinking fence algorithm imitates building and maintaining a fence, which surrounds a herd of solutions. At the beginning, posts of the fence are represented by known solutions of the initial set *S*. The order of posts in the fence is given by ordering of the associated solutions according their objective function values. It is assumed that the neighboring posts are connected by fence parts. During the optimization process, the individual fence parts are examined, a new, better position of a post is found and the new post replaces one of the neighboring posts. If one of the neighboring posts is closer to the new one more than a given distance, then the unnecessary post is removed. The best-found solution obtained by the inspections of fence parts is output of the algorithm.

The shrinking fence algorithm follows.

0. {Building up phase}

Order the solutions of input swarm *S* increasingly by their objective function values. This way, create a sequence $\mathbf{s}^0, \ldots, \mathbf{s}^{|S|-1}$. Initialize the best-found solution $\mathbf{x^{best}} = \mathbf{s}^0$ and the set of new posts $\underline{S}$ by empty set $\emptyset$.

1. {Maintenance phase}

For $t=|S|-1, \ldots, 1$, inspect the fence part connecting the posts $\mathbf{s}^t$ and $\mathbf{s}^{t-1}$ and define the new post position $\mathbf{x}^{\text{new}}$ by $\mathbf{x}^{\text{new}} = \mathit{FacetPathRelinking}(\mathbf{s}^t, \mathbf{s}^{t-1})$. If $\rho(\mathbf{s}^t, \mathbf{x}^{\text{new}}) < d^{min}$, then put $\underline{S} = \underline{S} \cup \{ \mathbf{x}^{\text{new}} \}$. Replace the best-found solution by $\mathbf{x^{best}} = \mathit{argmin}\{f(\mathbf{x^{best}}), f(\mathbf{x^{new}})\}$.

Inspect the fence part connecting $\mathbf{s}^0$ and $\mathbf{s}^{||S|-1}$ by $\mathbf{x}^{\text{new}} = \mathit{FacetPathRelinking}(\mathbf{s}^0, \mathbf{s}^{||S|-1})$.
If $\rho(\mathbf{s}^0, \mathbf{x}^{\text{new}}) < d^{min}$, then put $\underline{S} = \underline{S} \cup \{ \mathbf{x}^{\text{new}} \}$. Replace the best-found solution by $\mathbf{x^{best}} = \mathit{argmin}\{f(\mathbf{x^{best}}),$ $f(\mathbf{x^{new}})\}$.

{Improving process controlling}

If the termination condition is fulfilled, then terminate and return $\mathbf{x^{best}}$. Otherwise, update $S = \underline{S}$, reorder the elements of *S* according to increasing objective function values and put $\underline{S} = \emptyset$. Go to step 1.

Comment: The termination condition consists of two clauses. The process is terminated whenever the number of updates of the set S reaches the limit maxPop or if the expended computational time exceeds the threshold maxTime.

The next presented heuristic called the spider search is evoked by spider's web creation, which starts with linking fixed points with a center of the web by spider's thread and subsequent linking of the neighboring fixed points. Then some inner web nodes are established and the linking process to web center and then the mutual connections of the neighboring web nodes continue up to the moment, when the web is dense enough.

0. Initialize the starting swarm by a set *S* and order the solutions increasingly according to their objective function values into the sequence $\mathbf{s}^0, \ldots, \mathbf{s}^{|S|-1}$. Initialize the best-found solution $\mathbf{x^{center}} = \mathbf{s}^0$.

1. Process the swarm $\{ \mathbf{s}^0, \ldots, \mathbf{s}^{|S|-1}\}$ and web center $\mathbf{x^{center}}$ in the following way: Update the web center by $\mathbf{x^{center}} = \mathit{argmin}\{f(\mathit{FacetPathRelinking}(\mathbf{x^{center}}, \mathbf{s}^t)): t = 1, \ldots, |S|-1\}$ and insert the final $\mathbf{x^{center}}$ into the new swarm. For $t = 1, \ldots, |S|-1$, determine $\mathbf{x}^* = \mathit{FacetPathRelinking}\ (\mathbf{s}^t, \mathbf{s}^{t-1})$ and if there is no identical solution, insert $\mathbf{x}^*$ into the new swarm, otherwise skip the insertion. Finally perform $\mathbf{x}^* = \mathit{FacetPathRelinking}\ (\mathbf{s}^0, \mathbf{s}^{|S|-1})$ and add $\mathbf{x}^*$ to the new swarm.

2. If the termination condition is fulfilled, then the solving process finishes with the output defined by the best-found solution. Otherwise reorder new swarm, determine new web center $\mathbf{x^{center}}$ and go to step 1.

Comment: The termination condition consists of two clauses. The process is terminated whenever the number of the swarm updates reaches the limit maxPop or if the expended computational time exceeds the threshold maxTime.

4 Computational experiments

The main goal of performed computational study was to verify the efficiency of suggested discrete PSO strategies for search in unit lattice of *m*-dimensional simplex. Note that mentioned heuristics were originally developed and designed for a simple version of the weighted *p*-median problem [11, 16]. Therefore, their quality characteristics may change when the original model gets a more general form (1)-(6).

The numerical experiments reported in this paper were performed on a notebook equipped with the Intel® Core™ i7 3610QM 2.3 GHz processor and 8 GB of memory. The presented algorithms were implemented in the Java language making use of the NetBeans IDE 8.2 environment.

As far as the problem instances used in this computational study are concerned, they originate from real EMS system, which is operated in eight regions of Slovakia. The problem instances were used also in our previous research activities, the results of which are available in [9, 10, 11, 16] and in many others. The cardinalities of the set of possible service center locations and the set of system users vary from 87 to 664 locations. The organization of the Slovak self-governing regions is depicted in Figure 1.

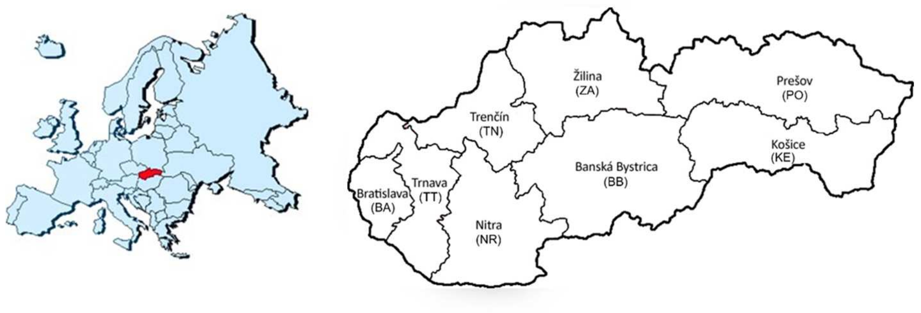


Figure 1 Used benchmarks – self-governing regions of Slovakia.

The parameters of individual benchmarks are summarized in the following Table I. in which also the exact solutions taken from another research [8] are reported. The coefficients q_k, $k = 1, \dots, r$ for r=3 stand for probabilities that the k-th nearest service center is the closest available one. The values of these coefficients were set so that $q_1 = 0{,}77063$, $q_2 = 0{,}16476$ and $q_3 = 1 - q_1 - q_2$. These values were obtained from a simulation model of existing EMS system in Slovakia published in [12].

The first four columns of Table 1 contain the basic characteristics of used problem instances. Column denotations keep the same meaning as used in the model (1)-(6). The last column of the table denoted by *OptObjF* is used to report the objective function value of the exact optimal solution of the model (1)-(6), which was computed in previous research reported in [8].

Table 1 Basic benchmarks characteristics and the optimal objective function values

Region	*m*	*n*	*p*	*OptObjF*
BA	87	87	25	18450
BB	515	515	46	38008
KE	460	460	38	40711
NR	350	350	36	40987
PO	664	664	44	46884
TN	276	276	26	31260
TT	249	249	22	36401
ZA	315	315	36	36929

An individual experiment was organized so that both compared PSO strategies, i.e. the shrinking fence and the spider search employing the generalized path-relinking method were applied to obtain the result of the problem described by the mathematical model (1)-(6). Since the optimal objective function value is available, the suggested algorithms can be compared from the viewpoint of solution accuracy.

Before reporting the achieved results, it must be noted that the basic idea of both solving approaches follows from the fact that the individual strategy starts from a set of feasible solutions, which can be provided by so-called uniformly deployed set. This set can be constructed independently on the solved instance. The process of a uniformly deployed set construction is reported in [9] and its possible usage can be found for example in [10, 11, 16]. The common property of a uniformly deployed set is that an arbitrary permutation of the locations generates a new uniformly deployed set with the same characteristics. We used this property to obtain ten different starting sets for each self-governing region presented in Table 1 and the values plotted in further Table 2 were obtained by averaging ten problem instances. The original uniformly deployed sets of zero-one solutions obtained from [9, 10] were adjusted by a greedy process to include some initial solutions outside the unit m-dimensional hypercube. Both suggested methods were run for stopping rule parameters maxPop = 8 and maxTime = 120 seconds.

The following Table 2 contains the average results. The structure of the table is formed by two parts – separate for each studied PSO strategy. For each heuristic approach we report the objective function value *ObjF* and the computational time *CT* in seconds.

Table 2 Comparison of discrete PSO strategies for the generalized weighted p-median problem – average results of ten runs with different uniformly deployed sets of solutions

Region	*Shrinking fence*		*Spider search*	
	ObjF	*CT*	*ObjF*	*CT*
BA	18751	1.12	18730	2.84
BB	39924	147.68	38094	211.21
KE	40711	92.46	40715	139.55
NR	41062	30.57	41062	58.59
PO	56416	124.73	47005	132.21
TN	31568	16.19	31540	37.28
TT	36768	10.44	36750	21.77
ZA	37030	27.81	37028	51.95

For completeness of reported results, we provide the readers with one additional Table 3, which contains the detailed results for the self-governing region of Žilina. Table 3 has the same structure as the former Table 2.

Table 3 Comparison of discrete PSO strategies for the generalized weighted p-median problem –results of ten runs with different uniformly deployed sets of solutions for the self-governing region of Žilina

Run	*Shrinking fence*		*Spider search*	
	ObjF	*CT*	*ObjF*	*CT*
1	36929	28.32	36929	51.68
2	36929	28.28	36929	53.34
3	36929	27.91	36929	51.86
4	37848	27.58	37828	52.78
5	36929	27.74	36929	51.25
6	36929	28.00	36929	53.00
7	36929	27.69	36929	50.36
8	36993	27.44	36993	52.35
9	36929	27.63	36929	51.46
10	36964	27.46	36964	51.37

All reported results indicate that the quality of obtained results is very satisfactory. From the point of solution accuracy, the strategy of a spider search seems better, because the average gap from the optimal objective function value achieves only the value of 0.54% while the first studied shrinking fence strategy brings worse results. As far as the computational time is concerned, both strategies can achieve the result in acceptably short time and can be used to solve practical real world problems.

5 Conclusions

This contribution was focused on two strategies employing the path-relinking method. The main research goal was aimed at the finding, whether the adjusted shrinking fence and spider search strategies are able to prove the same efficiency as their simple original versions when used for the *p*-location problem solution subject to the assumption that more than one facility can be located at the same possible service center location.

Suggested methods are based on the path-relinking method and they make use of previously developed search strategies. The novelty of presented original method extension consists in adjusting the heuristics for different space, in which the solutions are being explored. It must be realized that the mathematical problem formulation, to which the suggested heuristics were adjusted, makes use of the concept of generalized disutility, which assumes, that the service does not have to be provided by the nearest located service center, because it may be temporarily unavailable. In such a case, the request for rescue service is assigned to the nearest available center. The second modification of the original model consists in significant variables definition scope extension. It means that more than one facilities are allowed to be located in the same possible service center locations. This way, the former binary decision variables change into integers, what can make many available solving tool necessary to be adjusted or rebuilt.

The reported results of numerical experiments aimed at heuristic solving techniques for the multiple *p*-facility location problems with the generalized objective function show that the suggested strategies keep their useful features and both of them can be used for effective solving middle-sized problem instances. The accuracy of the resulting solution is satisfactory and the resulting system design can be obtained in acceptably short computational time. Based on performed numerical experiments we can conclude that we have constructed a very fast and effective heuristic approach to the generalized *p*-location problems.

Future research in this scientific field could be concentrated on rules, which would enable to reduce the starting set of *p*-location problem solutions and on developing other search strategies, which could improve the studied characteristic of the heuristic solving approach.

Acknowledgment

This work was supported by the research grants VEGA 1/0089/19 “Data analysis methods and decisions support tools for service systems supporting electric vehicles”, VEGA 1/0689/19 “Optimal design and economically efficient charging infrastructure deployment for electric buses in public transportation of smart cities”, and VEGA 1/0216/21 “Design of emergency systems with conflicting criteria using artificial intelligence tools”. This work was supported by the Slovak Research and Development Agency under the Contract no. APVV-19-0441.

References

[1] Avella, P., Sassano, A., Vasil'ev, I. (2007). Computational study of large scale p-median problems. Mathematical Programming 109, pp. 89-114.

[2] Current, J., Daskin, M., Schilling, D. (2002). Discrete network location models, Drezner Z. et al. (ed) Facility location: Applications and theory, Springer, pp. 81-118.

[3] Davendra, D., Zelinka, I. (2016). Self-Organizing Migrating Algorithm, Methodology and Implementation. Springer, Studies in Computational Intelligence, 289 p.

[4] Doerner, K. F., Gutjahr, W. J., Hartl, R. F., Karall, M., Reimann, M. (2005). Heuristic Solution of an Extended Double-Coverage Ambulance Location Problem for Austria. Central European Journal of Operations Research, 13(4), pp. 325-340.

[5] Drezner, T., Drezner, Z. (2007). The gravity p-median model. European Journal of Operational Research 179, pp. 1239-1251.

[6] García, S., Labbé, M., Marín, A. (2011). Solving large p-median problems with a radius formulation. INFORMS Journal on Computing, 23(4), pp. 546-556.

[7] Gendreau, M., Potvin, J. (2010). Handbook of Metaheuristics, Springer Science & Business Media.

[8] Janáček, J. (2021). Multiple p-Facility Location Problem with Randomly Emerging Demands. In: Strategic Management and its Support by Information Systems 2021: 14th International Conference, Technical University of Ostrava, in print

[9] Janáček, J., Kvet, M. (2019). Uniform Deployment of the p-Location Problem Solutions. In: Operations Research Proceedings 2019: Selected Papers of the Annual International Conference of the German Operations Research Society (GOR), Dresden, Germany, September 4-6, 2019: Springer, 2020, ISBN 978-3-030-48438-5, ISSN 0721-5924, pp. 315-321.

[10] Janáček, J., Kvet, M. (2019). Usage of Uniformly Deployed Set for p-Location Min-Sum Problem with Generalized Disutility. In: SOR 2019 proceedings, pp. 494-499.

[11] Janáček, J., Kvet, M. (2020). Shrinking fence search strategy for p-location problems. In: CINTI 2020: IEEE 20th International Symposium on Computational Intelligence and Informatics, Budapest, pp. 55-60.

[12] Jankovič, P. (2016). Calculating Reduction Coefficients for Optimization of Emergency Service System Using Microscopic Simulation Model. In: 17th International Symposium on Computational Intelligence and Informatics, pp. 163-167.

[13] Jánošíková, Ľ., Žarnay, M. (2014). Location of emergency stations as the capacitated p-median problem. In: Quantitative Methods in Economics (Multiple Criteria Decision Making XVII). pp. 117-123.

[14] Kozel, P. Orlíková, L., Pomp, M., Michalcová, Š. (2018). Application of the p-median approach for a basic decomposition of a set of vertices to service vehicles routing design. In Mathematical Methods in Economnics 2018, Praha: MatfyzPress, 2018, pp. 252-257.

[15] Kvet, M. (2014). Computational Study of Radial Approach to Public Service System Design with Generalized Utility. In: Proceedings of International Conference Digital Technologies 2014, Žilina, Slovakia, pp. 198-208.

[16] Kvet, M., Janáček, J. (2020). Spider network search strategy for p-location problems. In: CINTI 2020: IEEE 20th International Symposium on Computational Intelligence and Informatics, Budapest, pp. 49-54.

[17] Marianov, V., Serra, D. (2002). Location problems in the public sector, Facility location - Applications and theory (Z. Drezner ed.), Berlin, Springer, pp 119-150.

[18] Rybičková A., Mocková D., Teichmann D. (2019). Genetic Algorithm for the Continuous Location-Routing Problem, Neural Network World 29(3), pp. 173–187.

[19] Snyder, L. V., Daskin, M. S. (2005). Reliability models for facility location; The expected failure cost case, Transport Science 39 (3), pp. 400-416.

[20] Zelinka, I. (2016). SOMA-Self –organizing Migrating Algorith, ed. Davendra D., Zelinka, I.: Self-Organizing Migrating Algorithm-Methodology and Implementation, Springer, pp. 3-49.

Multi phase methodology for solving the multi depot vehicle routing problem with limited supply capacity at the depots

J. de Prado, S. Moscatelli, P. Piñeyro, L. Tansini*, and O. Viera

Department of Operations Research, Computer Science Institute (InCo), Faculty of Engineering, Universidad de la República, Montevideo, Uruguay
*libertad@fing.edu.uy

Abstract. This paper focuses on a capacitated multi depot vehicle routing problem, where each depot has a finite supply capacity to meet the customers demand. To solve this problem we propose a multi phase methodology, that extends the "cluster first, route second" approach. It is based on iterative routings to find and reassign misplaced customers with respect to the depots and with the objective of improving the final routing. Several assignment and routing algorithms are considered to evaluate the proposed methodology under different settings. A mathematical model of the problem is given to perform a comparative study of the methodology against an exact solution method. The results obtained from the numerical experiments carried out allow us to conclude that the methodology can be successfully applied to the capacitated multi depot vehicle routing problem.

Keywords: multi depot vehicle routing problem, heuristics, supply capacity, clustering, assignment

1 Introduction and related works

We address the problem of distribution of goods from several depots to a set of geographically dispersed customers with known coordinates and demand, assuming finite supply capacity at each depot and an unlimited fleet of homogeneous and capacitated vehicles. We refer to this problem as the Capacitated Multi-Depot Vehicle Routing Problem (CMDVRP). The objective is to determine a set of routes starting and ending at each depot, minimizing the total distance traveled and subject to the supply capacity of each depot and the capacities of the vehicles. The CMDVRP can be found in recent real life applications such as emergency facilities location-routing and city logistics problems [20, 22]. The CMDVRP is an NP-hard problem since it can be considered an extension of the Multi-Depot Vehicle Routing Problem (MDVRP), which in turn is an extension of the classical Vehicle Routing Problem (VRP) [8].

We present here a novel multi phase methodology to solve the CMDVRP inspired by the "cluster first, route second" approach. The initial phase consists of the assignment of costumers to depots and the final phase produces the routing of the VRPs related to all depots. Between these two phases, there is an intermediate phase for the reassignment of customers to depots with the aim to obtain a high quality solution in the cluster first part, improving in this way the final routing phase. The detection and reassignment of customers are based on a combination of misplaced-customer criterion and routing algorithm. A misplaced customer is reassigned to another depot, if this reassignment improves the cost of the general solution, which is the objective of the proposed methodology. The main idea behind the proposed methodology is that the complexity of the algorithms used in each phase can be chosen by the decision makers according to their needs and possibilities. The strength of the methodology is to provide good quality solutions in reasonable times, even in the case of using simple algorithms (easy to understand and code).

As far as we know, only few authors focus on the CMDVRP, and in particular, by means of the "cluster first, route second" approach. Giosa et al. [9] describe and compare several assignment algorithms for the clustering phase. Tansini et al. [17] compare the results obtained by a set of heuristic algorithms for the assignment of customers to depots with assignments obtained from solving the Transport Problem. Six heuristics for the clustering problem (assignment of customers to depots) are presented and analyzed in [10]. Also [18] consider this approach for the real-life problem of milk collection. Allahyari et al. [1] tackle the CMDVRP extension in which every customer is satisfied either by visiting the customer or by being located within an acceptable distance from at least one visited customer. Calvet et al. [4] consider the CMDVRP for the case of customers with stochastic demand and supply constraints on the depots due to the limited number of capacitated vehicles assigned to each of them. A collaborative routing problem with shared carriers and multiple depots (wholesalers) with limited storage is tackled in [21].

We note that many authors have considered the multi-depot vehicle routing problem with limited capacity on vehicles and/or route lengths, but not on the supply depots. For instance, Vidal et al. [19] propose a framework to solve the MDVRP, the Periodic VRP (PVRP), and the multi-depot periodic VRP with capacitated vehicles and constrained route duration. Contardo and Martinelli [6] suggest an exact algorithm for the MDVRP under capacity and route length constraints, exploiting the vehicle-flow and set-partitioning formulations. Recently, Pessoa et al. [15] propose a generic branch-cut-and-price solver for different vehicle routing variants and related problems.

The remainder of this paper is organized as follows. In Section 2 we introduce the mathematical formulation for the CMDVRP. The proposed methodology for solving the CMDVRP is further described in Section 3. In Section 4 we present the results of the comparison between the methodology against exact methods and we also analyze the effectiveness of the exploration phase of the methodology. Finally, in Section 5, we provide the conclusions and some directions for future research.

2 The Capacitated Multi-Depot Vehicle Routing Problem (CMDVRP)

The CMDVRP can be formally described as follows, extending that presented in [14] for the MDVRP. Let $G = (V, E)$ be a directed graph, where V denotes the set of nodes $\{1, ..., n\}$ and $E \subseteq V \times V$ the set of arcs. Let D be the set of depot nodes $\{1, ..., m\}$, with $1 \leq m < n$, and U the set of customer nodes $\{(m+1), ..., n\}$. For each node $i \in V$ there is a related quantity $q_i \geq 0$ that represents either the supply capacity for nodes $i \in D$ or the demand requirements in the case of nodes $i \in U$. For each arc $(i, j) \in E$ there is a routing cost $c_{i,j} \geq 0$. Let also consider the set of the possible routes $R = \{1, ..., (n-m)\}$. A route r can be defined as either the empty set or a finite sequence of at least three elements of V satisfying the following conditions: 1) in the extremes there is the same node i, with $i \in D$, 2) the internal nodes are customers nodes j with $j \in U$, and 3) for any pair of nodes $j, k \in U$, we have that $j \neq k$. We assume that for each route r there is a vehicle of capacity $p \geq 0$. Then, the objective is to determine the set of routes r in R in order to fulfill the demand of each customer without exceeding the vehicle and depot capacities, minimizing the total cost of routing. To formulate the CMDVRP as a Mixed Integer Linear Programming (MILP) we define the binary variables x_{ijkr} to be equal to 1 only if the arc (i, j) is in the route r of the depot k; 0 otherwise. Thus, the MILP proposed for the CMDVRP is as follows:

$$\texttt{min} \sum_{i \in V} \sum_{j \in V} \sum_{k \in D} \sum_{r \in R} c_{ij} x_{ijkr} \tag{1}$$

subject to:

$$\sum_{j \in V} \sum_{k \in D} \sum_{r \in R} x_{ijkr} = 1, \qquad \forall i \in U \tag{2}$$

$$\sum_{j \in V \setminus \{i\}} x_{ijkr} = \sum_{j \in V \setminus \{i\}} x_{jikr}, \qquad \forall i \in V, \forall k \in D, \forall r \in R \tag{3}$$

$$\sum_{i \in U} \sum_{j \in V \setminus \{i\}} \sum_{k \in D} q_i x_{ijkr} \leq p, \qquad \forall r \in R \tag{4}$$

$$\sum_{i \in U} \sum_{j \in V \setminus \{i\}} \sum_{r \in R} q_i x_{ijkr} \leq q_k, \qquad \forall k \in D \tag{5}$$

$$\sum_{j \in U} x_{kjkr} \leq 1, \qquad \forall k \in D, \forall r \in R \tag{6}$$

$$\sum_{j \in U} x_{ijkr} = 0, \qquad \forall i, k \in D, i \neq k, \forall r \in R \tag{7}$$

$$y_i - y_j + (n-m)\sum_{k\in D}\sum_{r\in R} x_{ijkr} \leq n-m-1, \qquad \forall i,j \in U, i \neq j \tag{8}$$

$$y_i \geq 0, \qquad \forall i \in U \tag{9}$$

$$x_{ijkr} \in \{0,1\}, \qquad \forall i,j \in V, \forall k \in D, \forall r \in R \tag{10}$$

The objective function (1) is the minimization of the total cost of distance traveled. Constraints (2) state that each customer is included in a single route. Constraints (3) are for the route continuity. Constraints (4) and (5) represent the vehicle and depot capacity, respectively. Constraints (6) and (7) state that one route is assigned at most to a single depot. In (8) are the constraints of Miller-Tucker-Zemlin for subtours elimination [3]. Finally, constraints (9) and (10) are for the domain of values of the decision variables.

Although the main difference between CMDVRP and MDVRP are the constraints of (5), we note that, in general, a more restricted problem is more difficult to solve.

3 Multi-Phase Methodology for the CMDVRP

It is worth to note that the assignment problem and the routing problem in the "cluster first, route second" approach are not independent from each other. A bad assignment solution will result in routes of higher total cost, even if an effective routing algorithm is used. Motivated by this, we consider an improvement to this approach, by means of a multi-phase methodology (MPM) for solving the CMDVRP. It begins from an initial assignment of costumers to depots and in the final phase produces the routing of the VRPs related to all depots. We introduce an intermediate phase in which misplaced costumers are detected and may be reassigned to another depot, if it improves the cost of the overall solution. Successive reassignment of misplaced costumers, based on the routing, will in most cases lead to an improvement of the solution. An outline of the proposed methodology for the CMDVRP is as follows:

1. **Assignment phase**: choose and apply an assignment algorithm of customers to depots taking into account demand and supply restrictions. The choice may depend on computational time and other restrictions.
2. **Exploration phase**: choose and apply a routing algorithm for all VRPs related to the depots. Again, the choice may depend on computational time and other restrictions. Then, repeat until no further improvement can be achieved:
   (a) Detect misplaced customers based on the current assignment and eventually other restrictions.
   (b) Reassign misplaced customers and run the selected routing algorithm. Accept the reassignment if it improves the cost of the overall solution.
3. **Final routing phase**: choose and apply the final routing algorithm for all VRPs related to the depots.

One of the advantages of the suggested MPM is that each phase offers the possibility of choosing different algorithms depending on the specific characteristics of the problem, the problem instances, hardware limitations, time restrictions, etc. They can be exchanged and combined in different manners. Thus, a specific selection of algorithms for each phase produces a particular MPM instantiation that can be considered a heuristic procedure to solve the CMDVRP. Next the MPM phases are explained in more detail and some algorithms that can be used in each one are mentioned.

3.1 Assignment phase

Since each phase of the methodology offers a great variety of possibilities to instantiate and since there are several known methods that can be used to obtain an initial assignment of customers, in this work we narrow down the study to two assignment schemes.

We use the urgency assignment (Ur) which is a simple and fast assignment method that considers an urgency value μ_c for each customer c that determines the order in which customers are assigned to depots with limited supply capacity [9], as follows:

$$\mu_c = \Big[\sum_{d \in D} dist(c,d)\Big] - dist(c,d') \tag{11}$$

where $dist(c,d)$ is the distance of customer c to depot d, and $dist(c,d')$ is the distance to the closest depot d'. This measure accounts for the cost of assigning a customer to a depot other than its closest depot. Customers with more urgency (higher μ_c value) will be assigned first. Once a depot is complete it will no longer be considered for the further assignments and will hence not participate in the urgency calculations. Note that after each assignment the urgency of some customers must be recalculated.

Alternatively in this study the modified urgency assignment (MUr) is used as another assignment method and is defined as the combination of the urgency assignment [9] and the cluster assignment [10]. Customers are assigned to depots with the same criterion as in the urgency assignment until a fraction of them have been assigned (in this case 1/4) and then finalizes by assigning customers to the closest cluster made up of each depot and the already assigned customers, where it is feasible to assign the customer, i.e. will not exceed the total capacity of the depot.

There are other interesting assignment algorithms that could be explored such as the sweep approach [11] or using a grid or Voronoi diagrams [2]. We note that some of them do not consider the capacity of the depots and may require an adaptation or post-processing in order to give an acceptable initial assignment.

3.2 Exploration phase

The exploration phase is the keystone of the proposed methodology. It is characterized by: 1) the definition of misplaced customers; 2) the processing order of the misplaced customers; 3) the routing algorithm used iteratively; 4) the criterion that determines if each misplaced customer should be reassigned or not; and 5) the reassignment strategy. In the following sections we describe the definitions and algorithms to be used in our study for this phase.

Definition of misplaced customers: Here, the definition of misplaced customers, the processing order of the misplaced customers, and the criterion that determines if each misplaced customer should be reassigned or not, all of them depend on the routing algorithm that is used iteratively to obtain the results of the VRPs related to all depots.

It is possible to infer that different definitions of misplaced customers lead to different ways of exploring neighboring solutions. The first approach was to define misplaced customers as those whose two closest customers are assigned to another depot. In general, we can define a misplaced customer as that for which its n closest customers are assigned to other depots (possibly different), for certain positive integer $n > 0$. Thus, a more flexible definition considers a customer to be misplaced if considering its n closest customers, m of them are assigned to other depots, where $m \leq n$. Observe that this definition focuses on the cost of the solution, therefore the reassignment strategy considers the supply capacity of the depots.

Other approaches would be to consider constraints such as capacity and time windows in the definition of misplaced customers.

Processing order of the misplaced customers: In this work, misplaced customers are processed in descending order of the following misplaced criterion:

$$\varphi_c = dist(c,d) - \Big[\sum_{i=1}^{N} dist(c,c_i)\Big] \tag{12}$$

where customer c has been assigned to the depot d and $c_1, ..., c_N$ are its N closest customers not assigned to d, but assigned all to the same depot. The value of φ_c can be positive or negative, where a high positive value of φ_c means that the customer c is very far from the assigned depot compared to the distance to its closest neighbors.

Routing algorithm used iteratively: Three different algorithms for the routing of the customers assigned to each depot were tested in this paper for the Exploration phase: Clarke & Wright algorithm [5], Sweep [11] and JSprit (available at https://jsprit.github.io/) which is a metaheuristic defined by the ruin-and-recreate principle [16]. It is a highly optimized method that consists of a large neighborhood search that combines elements of simulated annealing and threshold-accepting algorithms. Both Clarke & Wright and Sweep are classical and simple routing algorithms for the VRP, and also Clarke & Wright is a very popular constructive heuristic [12] and Sweep is the most elementary version of petal-type constructive heuristics [12]. It is worth noting that in each iteration, the routing algorithm only needs to bee applied for those depots with reassigned customers, since the others remain unchanged.

Reassignment criterion: The reassignment criterion used in this paper is to reassign a customer if it produces a lower routing cost than the current assignment. It is important to note that once the reassignments are decided, it is only necessary to run the routing algorithm for the implicated depots. The routing for the rest of the depots remains unchanged.

Reassignment strategy: Different approaches can be considered for the reassignment strategy. They should describe the conditions and the procedure to assign a misplaced customer to another depot, that will potentially improve the final routing. In general, the demand of customers and the capacities of depots should be considered. In this paper the reassignment strategy is determined by a two-stage procedure executed over an ordered list of misplaced customers. As part of the strategy, it has to be decided the number m of customers with the same target depot that may be considered to be reassigned simultaneously. This section explains the strategy suggested to reassign one misplaced customer ($m = 1$) at a time in the exploration phase.

In the first stage, the reassignment of a misplaced customer i to the depot d' of the closest customer i' is attempted, if d' has enough spare capacity to serve costumer i. If the reassignment produces a better overall routing result, it is accepted and the list of misplaced customers is recalculated. If there is no improvement, the next misplaced customer in order of misplaced criterion is considered to be reassigned. If depot d' does not have enough spare capacity to serve costumer i, then i is reassigned to the closest depot d'' (if it exists) that does have enough spare capacity to serve it. Again, if the reassignment produces a better overall routing result, the reassignment is accepted and the list of misplaced customers is recalculated.

The aim of the second stage is to try reassign the misplaced customers that remain in the list after the first stage. In this stage the same processing is done with the misplaced customers as in the previous stage except in the way of determining the alternative depot d'' and the reassignment moves. Let us assume that depot d' does not have enough spare capacity to serve the misplaced customer i under consideration, with d' as in the first stage. Then, a misplaced customer i'' assigned to d' is determined, that could potentially be reassigned to another depot d'', with d'' the depot of the closest customer to i'', allowing d' to serve the customer i. If misplaced customer i'' exists, a double reassignment is attempted by means of assigning i'' to depot d'' and i to depot d'. If the double reassignment of customers produces a better overall routing result, the reassignment is accepted and the list of misplaced customers is recalculated.

The two stages described above, are repeated until there are no further misplaced customers (the list is empty) or no misplaced customer reassignment results in a cost improvement.

In the case of at least two misplaced customers ($m \geq 2$) being reassigned together, the procedure is similar but the destination depot has to have enough spare capacity to serve the set of misplaced customers under consideration.

3.3 Final routing phase

Several routing algorithms can be used to produce the final routing once the Exploration phase has finished. In this work the same three algorithms used in the Exploration phase were tested for the Final routing phase.

4 Evaluation of the proposed methodology

In this section we provide the results obtained from different numerical experiments of several MPM instantiations. Given the reasonable computational time observed for the MPM methodology, we performed all experiments with all combinations of assignment and routing algorithms for the phases of the methodology (assignment, exploration and final routing phases). The MPM instantiation with the best result obtained is shown in the tables (in the case of equal cost the one with the fastest time is chosen).

The mathematical model for the CMDVRP presented in Section 2 was coded in AMPL and solved with CPLEX 12.6.3.0 on a PC Intel Core i7, 16 CPUs, 64 GB of RAM (DDR4) and CentOS 7. The MPM instantiations were coded in Java and executed in a PC with Intel Xeon CPU E3-1220 V2, 4GB of RAM and Windows 7.

4.1 Comparative study with exact method

Solving the CMDVRP to optimality is extremely costly due to the computational complexity of the problem. Nevertheless, an important aspect of a comprehensive analysis for any proposed heuristic approach is to compare its results against exact methods both regarding objective values and running times.

In https://www.fing.edu.uy/owncloud/index.php/s/XnvURwxKzQUaH1P it is available the benchmark set of instances used to compare different MPM instantiations against CPLEX. Table 1 presents the results obtained.

The first column of Table 1 provides the identification of the instances, with 20 nodes in total, 2 or 3 depots, and a sequential number. The capacity of each depot is in the range $[66, 125]$, and the vehicle capacity in the range $[50, 70]$. The sum of the customers demand is of 180 units for each instance. We note that the distribution of the customers and depots is based on real map coordinates on certain islands of the Pacific ocean. Columns 2 to 4 report the name of the MPM instantiation, the costs and the total running times (in seconds) of all phases of the methodology for each one of the instances in the benchmark set. The name of each MPM instantiation is composed by four terms separated by a simple dash: the assignment algorithm (Ur or MUr), the criterion for misplaced customers and the routing algorithms used for the exploration and final phases, respectively. For example, a misplaced criterion 5c1n2m means that 5 closest customers are considered for determining if certain customer is misplaced, at least 1 of them is assigned to another depot, and 2 can be reassigned simultaneously. For all the experiments performed, we consider 1 to 5 closest customers for the misplaced criterion and between 1 or 2 customers to be reassigned simultaneously. The algorithms used in each phase and the misplaced criteria of the MPM instantations listed in Table 1 were those for which we obtained the best results in the experiments. Columns 5 reports the cost obtained from CPLEX with a running time limited to 3600 seconds (no significant improvements were noticed with higher running times). Last column 6 in Table 1 provides the percentage gap between the cost of the MPM instantiation and CPLEX, calculated as $100 * (MPM_{Cost} - CPLEX_{Cost})/CPLEX_{Cost}$.

From the results in Table 1 we note that MPM outperforms CPLEX in 6 of the 15 instances, and achieves the same objective value in the remaining ones. Thus, we can conclude that MPM is competitive with CPLEX because the gap error is always negative or zero and the running times of all the considered MPM instantiations are significantly lower than CPLEX (less than 60 seconds versus 3600 seconds). We also want to note that the most effective MPM instantiation considering both, costs and running times, is Ur-1c1n1m-C&W-JSprit, since it shows the two lowest percentage cost gaps and less than a half of a second of running time. This MPM instantation makes use of different algorithm approaches for the exploration and final routing phases. This seems to indicate that it would be enough to use a simple and fast algorithm for the exploration phase, and a good and eventually time consuming routing algorithm for the final phase.

4.2 Comparative study with and without exploration phase

A central part of the proposed methodology, is the intermediate exploration phase for the detection of misplaced customers and the reassignment of them to depots using a routing algorithm. In this section we analyze the impact of including the exploration phase in the methodology by means of

Table 1. Comparison of results for MPM instantiations against CPLEX.

Instance	MPM Instantation	Time	Cost	CPLEX Cost	% Gap cost
20n2d01	MUr-5c1n2m-JSprit-JSprit	58.79	3064.9	3085.83	-0.68
20n2d02	Ur-5c1n2m-JSprit-JSprit	24.757	5726.34	5726.34	0.00
20n2d03	Ur-5c1n2m-JSprit-JSprit	22.591	224.18	229.92	-2.50
20n2d04	Ur-1c1n1m-Sweep-C&W	0.001	158.03	158.03	0.00
20n2d05	Ur-5c1n2m-JSprit-JSprit	45.336	354.39	361.26	-1.90
20n2d06	Ur-5c1n2m-Sweep-JSprit	0.266	5808.51	5808.51	0.00
20n2d07	MUr-5c1n2m-JSprit-JSprit	11.337	5873.72	5873.71	0.00
20n2d08	Ur-5c1n2m-JSprit-JSprit	37.826	5062.75	5062.75	0.00
20n2d09	Ur-5c1n2m-Sweep-JSprit	0.298	910.97	924.15	-1.43
20n2d10	Ur-1c1n1m-Sweep-C&W	0.001	292.26	292.26	0.00
20n3d01	Ur-1c1n1m-C&W-JSprit	0.354	2556.36	2726.02	-6.22
20n3d02	Ur-5c1n2m-Sweep-JSprit	0.248	159.84	159.84	0.00
20n3d03	Ur-1c1n1m-C&W-JSprit	0.329	123.98	123.98	0.00
20n3d04	Ur-1c1n1m-C&W-JSprit	0.354	4773.08	4973.73	-4.03
20n3d05	Ur-1c1n1m-C&W-JSprit	0.347	4576.78	4576.78	0.00
Average		13.522			-1.12

a comparative study over ten large instances with different geographical characteristics, available also at the same web repository provided in Section 4.1. Some of them are based on instances of the TSPLIB (http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/vrp/), and others were randomly generated, trying to create clusters of customers with different densities.

Tables 2 and 3 report the results obtained for the MPM instantiations MUr-5c1n2m-Sweep-Sweep and Ur-2c1n1m-Sweep-JSprit without and with exploration phase, respectively. Due to the large size of the instances considered, we chose Sweep for the routing of the exploration phase, since it is a simple and fast routing algorithm, although not very efficient. For this reason, it is not the purpose of the experiments presented here to compare the quality of the solutions obtained of these MPM instantiations. The algorithms of the others phases and the misplaced criteria used for the MPM instantiations were those for which we obtained the best results in the experiments performed. Columns 1 to 6 provide the information about the instances: name, number of total nodes, number of depots, total depot capacity, vehicles capacity and total customer demand, respectively. Columns 7 to 10 show the costs and the total running times (in seconds) of all phases of the MPM methodology, without and with exploration phase, respectively. The last two columns report the percentage of gap for the costs and the time ratio (the ratio between the running times observed with and without exploration).

From Tables 2 and 3 we can appreciate that the exploration phase results in a performance improvement that may depend on the routing algorithms used for the exploration and final phases. In the case of the same algorithm (MUr-5c1n2m-Sweep-Sweep), the inclusion of the exploration phase results in a better final solution for all the instances, with an average improvement of 7.18%. Although the running times increased on average 17 times, they can still be considered very good taking into account the size of the instances. In the case of different routing algorithms (Ur-2c1n1m-Sweep-JSprit), we note from Table 3 that in most instances (7 of 10) the inclusion of the exploration phase results in a better final solution, with an improvement in costs from 0.18% to 6.83%. In addition, empirically it seems that the inclusion of the exploration phase does not cause a significant increase in the execution times. Indeed, in almost half of the instances there is a marked decrease in them. This may be due to the fact that the reassignment of customers to depots of the exploration phase simplifies the final routing, i.e., less effort is needed to obtain a good quality routing. Again, it can be seen that it is enough to use a simple and fast algorithm for the exploration phase, and a good and eventually time consuming routing algorithm for the final phase. However, in some cases using different routing algorithms for the the two phases can result in a higher cost final solution, as it can be seen in Table 3.

Table 2. MUr-5c1n2m-Sweep-Sweep performance without and with exploration phase.

Inst.	Nodes	Dep.	D.Cap.	V.Cap.	Dem.	Without exp.		With exp.		% Gap	Time
						Cost	Time	Cost	Time	cost	ratio
L01	200	5	4250	80	3885	626409.84	0.319	591986.94	1.811	-5.50	5.68
L02	200	5	4250	80	3885	612327.72	0.043	564351.64	1.904	-7.84	44.28
L03	200	8	4400	80	3885	693744.67	0.078	690438.63	0.583	-0.48	7.47
L04	262	13	15920	500	12106	8438.75	0.060	7373.04	1.840	-12.63	30.67
L05	500	6	15000	300	12488	404082.94	0.327	364345.57	4.123	-9.83	12.61
L06	500	6	15000	300	11750	401862.56	0.440	368230.51	15.161	-8.37	34.46
L07	800	7	24500	300	22890	572609.36	0.984	546426.73	11.102	-4.57	11.28
L08	800	7	24500	300	24007	585369.28	1.978	542194.85	22.870	-7.38	11.56
L09	1050	50	50073	500	40801	482949.62	3.818	443298.2	16.027	-8.21	4.20
L10	1050	50	44450	500	40411	452780.95	1.934	420944.92	17.881	-7.03	9.25
Average										-7.18	17.15

Table 3. Ur-2c1n1m-Sweep-JSprit performance without and with exploration phase.

Inst.	Nodes	Dep.	D.Cap.	V.Cap.	Dem.	Without exp.		With exp.		% Gap	Time
						Cost	Time	Cost	Time	cost	ratio
L01	200	5	4250	80	3885	481912.19	10.202	487401.81	8.475	1.14	0.83
L02	200	5	4250	80	3885	468610.92	7.409	467768.58	6.902	-0.18	0.93
L03	200	8	4400	80	3885	578998.86	5.098	578998.86	4.814	0.00	0.94
L04	262	13	15920	500	12106	7022.09	23.496	6542.62	26.43	-6.83	1.12
L05	500	6	15000	300	12488	299702.93	60.413	298152.39	60.57	-0.52	1.00
L06	500	6	15000	300	11750	308062.74	56.938	312336.36	50.678	1.39	0.89
L07	800	7	24500	300	22890	466505.25	166.581	463011.41	157.925	-0.75	0.95
L08	800	7	24500	300	24007	469352.3	123.842	467293.69	125.404	-0.44	1.01
L09	1050	50	50073	500	40801	390585.03	19.974	388445.21	21.17	-0.55	1.06
L10	1050	50	44450	500	40411	414786.54	17.403	409329.19	18.793	-1.32	1.08
Average										-0.80	0.98

5 Conclusions

The Capacitated Multi-Depot Vehicle Routing Problem (CMDVRP) is an extension of the MDVRP that considers limited supply on the depots. As far as we know the CMDVRP problem has received much less attention in the literature than other MDVRP extensions. In order to solve this NP-hard problem, we introduce a Multi-Phase Methodology (MPM) that extends the well-known approach of “cluster first, route second”. The most relevant feature of MPM is an intermediate exploration phase for detecting and reassigning misplaced customers based on VRP algorithms. As the VRP is a well-known and widely studied problem, the strength of the proposed MPM is to give a straightforward an efficient general framework for the direct use of VRP algorithms, in many cases publicly available and free, to solve the CMDVRP. Each MPM phase offers the possibility of choosing different algorithms depending on the specific characteristics of the problem, the problem instances, hardware limitations, time restrictions, etc. A specific selection of algorithms, for each phase, produces a particular MPM instantiation that yields a heuristic procedure to solve the CMDVRP.

From the results obtained of the numerical experiments carried out, we can conclude that the multi-phase methodology suggested can result in competitive heuristics compared to exact methods. In particular, it may be useful for users who often need to find solutions of quality in a reasonable computational time. We point out that it would be enough to use a simple and fast routing algorithm for the exploration phase, and a good and eventually time consuming routing algorithm for the final phase. We also note that in general the exploration phase produces better solutions without causing a significant increase in the execution times but, in many cases, there is a decrease in them. This may be due to the fact that the reassignment of customers to depots during the exploration phase makes that less effort is needed to obtain a good quality final routing.

The proposed multi-phase methodology enables and facilitates the use of different combinations of algorithms and the possibility to define the criterion for misplaced customers that may include geographical information, supply capacity constraints, time windows and others constraints. Therefore, it has a great potential to be adapted to specific MDVRP variants such as Periodic-VRP (PVRP), MDVRPTW or CMDVRPTW. The exploration phase of MPM allows the introduction of randomness for example in the order in which the misplaced customers are considered to be reassigned or in the reassignment strategy. We believe that the methodology could benefit from employing a randomized strategy in order to explore the solution space more extensively and eventually escape from local optimal solutions.

A possible and interesting direction for future research is to compare the proposed methodology against different solution procedures of the literature for related problems, such as MDVRP (the problem without capacity constraints on the depots). In order to make this comparison, we adapted the instances suggested by [7] and available at http://neumann.hec.ca/chairedistributique/data/mdvrp/. We consider those MDVRP instances of [7] without supply capacities on the depots nor time constraints on the routes duration, but do have restrictions on the vehicle fleet size. Preliminary results obtained in tests comparing different instances of MPM and the multiphase SFLA-PLEONS algorithm of [13], which as far as we know is one of the faster and more accurate algorithms in the literature for MDVRP, shown that MPM methodology is competitive with fastest running times. The objective is to continue doing more tests varying the instantiations of MPM methodology and also look for other instances of MDVRP in the literature.

Finally, some of the results of the numerical experiment reported, deserve a further analysis. One of them is to analyze the causes of why the addition of the exploration phase does not increase the execution times of the overall methodology, as we empirically observed in the numerical experiments reported in Tables 2 and 3. Another issue is in which cases and why it is sufficient to use a simple and fast algorithm for the exploration phase, and a good and eventually time consuming routing algorithm for the final phase, to obtain good quality solutions.

References

1. Allahyari, S., Salari, M., Vigo, D.: A hybrid metaheuristic algorithm for the multi-depot covering tour vehicle routing problem. European Journal of Operational Research 242, 756–768, (2015)
2. Aurenhammer, F.: Voronoi diagrams - survey of a fundamental geometric data structure, ACM Computing Surveys 23, 345–405, (1991)

3. Bektas, T., Gouveia, L.: Requiem for the miller-tucker-zemlin subtour elimination constraints?. European Journal of Operational Research 236, 820–832, (2014)
4. Calvet, L., Ferrer, A., Gomes, M.I., Juan, A.A., Masip, D.: Combining statistical learning with metaheuristics for the multi-depot vehicle routing problem with market segmentation. Computers & Industrial Engineering 94, 93–104, (2016)
5. Clarke, G., Wright, J.W.: Scheduling of vehicles from a central depot to a number of delivery points. Operations Research 12, 568–581, (1964)
6. Contardo, C., Martinelli, R.: A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. Discrete Optimization 12, 129-146, (2014)
7. Cordeau, J.F., Laporte, G., Mercier, A.: Improved tabu search algorithm for the handling of route duration constraints in vehicle routing problems with time windows. The Journal of the Operational Research Society 55(5), 542-546 (2004)
8. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman & Co., New York, NY, USA (1979)
9. Giosa, D., Tansini, L., Viera, O.: Assignment algorithms for the multi-depot vehicle routing problem. Proceedings of the 28th Conference of the Sociedad Argentina de Informática e Investigación Operativa, (1999)
10. Giosa, D., Tansini, L., Viera, O.: New assignment algorithms for the multi-depot vehicle routing problem. The Journal of the Operational Research Society 53, 977–984, (2002)
11. Laporte, G.,The vehicle routing problem: An overview of exact and approximate algorithms, European Journal of OperationalResearch59(1992), pp. 345 – 358.URLhttp://www.sciencedirect.com/science/article/pii/037722179290192C
12. Laporte, G.: What you should know about the vehicle routing problem. Naval Research Logistics 54, 811–819, (2007)
13. Luo, J., Chen, M.-R.: Improved shuffled frog leaping algorithm and its multi-phase model for multi-depot vehicle routing problem. Expert Systems with Applications 41, 2535-2545, (2014)
14. Montoya-Torres, J.R., Franco, J.L., Isaza, S.N., Jiménez H.F., Herazo-Padilla, N.: A literature review on the vehicle routing problem with multiple depots. Computers & Industrial Engineering 79, 115–129, (2015)
15. Pessoa, A., Sadykov, R., Uchoa, E., Vanderbeck, F.: A generic exact solver for vehicle routing and related problems. Mathematical Programming 183(1), 483-523, (2020)
16. Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., Dueck, G.: Record breaking optimization results using the ruin andrecreate principle. Journal of Computational Physics 159, 139–171, (2000)
17. Tansini, L., Urquhart, M., Viera, O.: Comparing assignment algorithms for the multi-depot VRP, Technical report, INCO,FING, UDELAR (2001)
18. Urquhart, M., Viera, O.: A vehicle routing system supporting milk collections. OPSEARCH 39, 46–54, (2002)
19. Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., Rei, W.: A hybrid genetic algorithm for multi-depot and periodic vehicle routing problems. Operations Research 60(3), 611-624, (2012)
20. Zhang, B., Li H., Li, S., Peng J.: Sustainable multi-depot emergency facilities location-routing problem with uncertain information. Applied Mathematics and Computation 333, 506–520, (2018)
21. Zhang, W., Chen, Z., Zhang, S., Wang, W., Yang, S., Cai, Y.: Composite multi-objective optimization on a new collaborative vehicle routing problem with shared carriers and depots. Journal of Cleaner Production 274, 1–18, (2020)
22. Zhou, L., Baldacci, R., Vigo, D., Wang, X.: A multi-depot two-echelon vehicle routing problem with delivery options arising in the last mile distribution. European Journal of Operational Research 265, 765–778, (2018)

Multi-Infill Bayesian Optimisation for improving Exploration and Exploitation Trade-off

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

1 Introduction

This contribution concerns the optimisation of expensive black-box functions, where one computation may last more than a day. Due to the time needed to solve these problems, traditional optimisers, which involve multiple calls to the objective function, are not sustainable. Consequently, other strategies have to be invoked to reduce either the number of calls to the objective function or the computational time of the black-box functions.

The Bayesian Optimisation algorithm [1], coupled with Gaussian Processes [2], is an interesting way to deal with this problem of expensiveness. Instead of reducing the computational burden of solving the black-box function, its purpose is to construct a cheaper mathematical model, hinging on a set of evaluated solutions. Then, iteratively, this set of evaluated solutions is incremented whilst aiming for the global optimum of the black-box function. This algorithm was already proved to be efficient in a wide range of applications. To determine the new candidate to add to the set of evaluated solutions, the Bayesian Optimisation algorithm relies on an acquisition function (or infill criterion) [3, 4] of which the most popular is the Expected Improvement [5].

The fundamental aspect not to overlook when carrying Bayesian Optimisation procedure is to maintain a good balance between exploration and exploitation. The Expected Improvement is indeed a compromise between these two objectives but in peculiar conditions, the exploration with this acquisition function might be sub-optimal and some areas of the design space are completely hidden from the Bayesian Optimisation algorithm.

Hence, in this sequel, we suggest a strategy to overcome this defect of the Expected Improvement. This strategy consists in transforming the common mono-infill Bayesian Optimisation into a multi-infill Bayesian Optimisation while using two popular acquisition functions: the Expected Improvement and the variance. The performance of this strategy is assessed with several mathematical problems, which hold multiple local minima.

2 Theoretical background

Gaussian Processes (hereby denoted GP) are a class of surrogate models hinging on a probabilistic definition of the output $\widehat{Y}$. This random vector is given to being Gaussian distributed with mean $\widehat{m}$ and variance $\widehat{s}$ (Eq. 1).

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

where $\mathbf{C}(\cdot, \cdot)$ is the covariance matrix.

Fig. 1 summarizes each step of the Bayesian Optimisation (hereby denoted BO) algorithm, coupled with GP. At first, prior knowledge about the behavior of the black-box function is computed by generating a training set with a Latin Hypercube Sampling. Next, the surrogate model hyperparameters (covariance and likelihood) have to be set. In this communication, we restrict our choice to the Matern 3/2 covariance function and the Gaussian likelihood, which are two traditional choices for GP modelling. Then, the surrogate model is optimised according to the likelihood. Finally, after the prediction, an acquisition function [4, 6, 3] is used to determine the most relevant new candidate to add to the training set.

Fig. 1 relies on the following data: $\mathbf{C}(\cdot,\cdot)$ is the covariance matrix, $\mathbf{x_p}$ and $\mathbf{x_q}$ are samples from the training set input $\mathbf{X}$, σ_k is the signal variance, D the dimension number of the problem, θ_i, the lengthscale associated with the ith dimension, $\mathbf{y}$, the training set output, $\Delta^j := y^j_{min} - \widehat{m}(x)$, with y^j_{min}, the minimum of the training set output at the iteration j, Φ is the gaussian cumulative distribution function, ϕ is the gaussian probability density function.

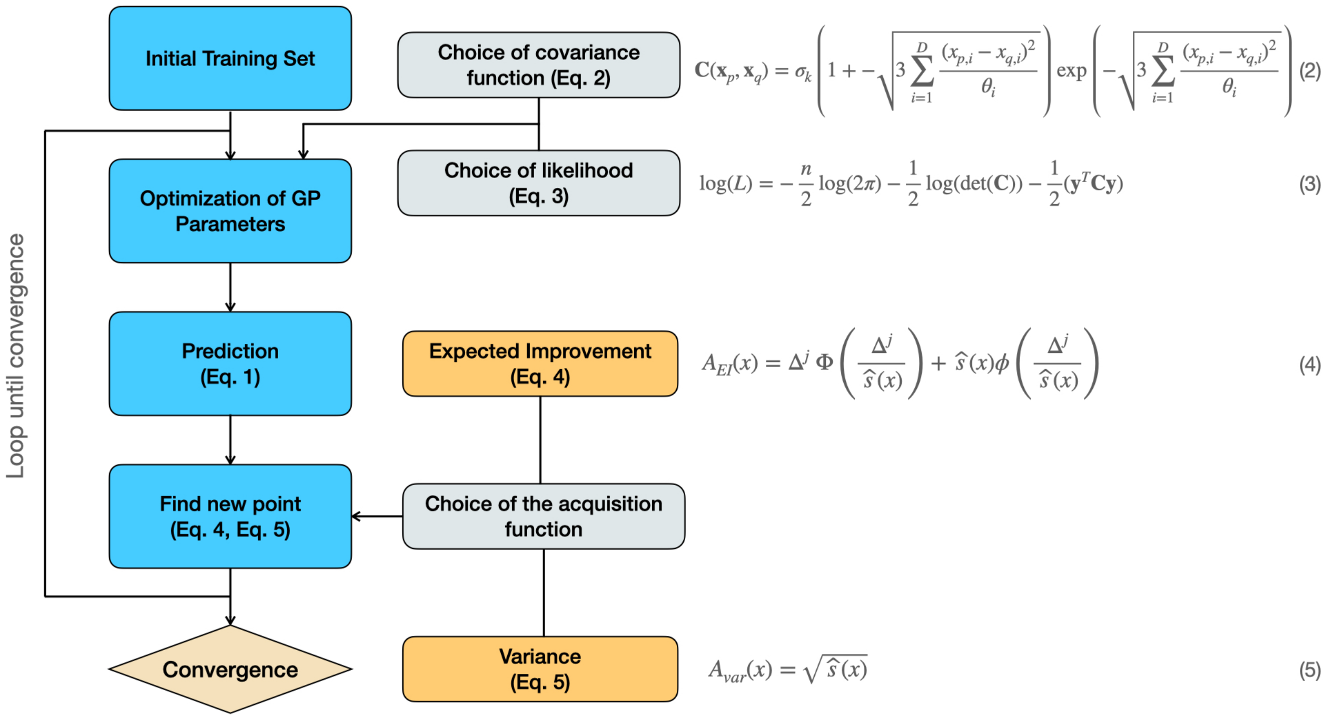

Initial Training Set
Choice of covariance function (Eq. 2)
$\mathbf{C}(\mathbf{x}_p,\mathbf{x}_q) = \sigma_k \left(1 + \sqrt{3\sum_{i=1}^{D}\frac{(x_{p,i}-x_{q,i})^2}{\theta_i}}\right)\exp\left(-\sqrt{3\sum_{i=1}^{D}\frac{(x_{p,i}-x_{q,i})^2}{\theta_i}}\right)$ (2)
Optimization of GP Parameters
Choice of likelihood (Eq. 3)
$\log(L) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log(\det(\mathbf{C})) - \frac{1}{2}(\mathbf{y}^T\mathbf{C}\mathbf{y})$ (3)
Prediction (Eq. 1)
Expected Improvement (Eq. 4)
$A_{EI}(x) = \Delta^j\,\Phi\left(\frac{\Delta^j}{\hat{s}(x)}\right) + \hat{s}(x)\phi\left(\frac{\Delta^j}{\hat{s}(x)}\right)$ (4)
Find new point (Eq. 4, Eq. 5)
Choice of the acquisition function
Loop until convergence
Convergence
Variance (Eq. 5)
$A_{var}(x) = \sqrt{\hat{s}(x)}$ (5)

Fig. 1: Workflow of the Bayesian Optimisation algorithm

As stated before, the Expected Improvement acquisition function (hereby denoted EI) is a compromise between exploration (second term of the sum) and exploitation (first term of the sum) whereas the variance acquistion function (hereby denoted Var) is a pure exploratory criterion.

Fig. 2 puts forward an evaluation of both acquisition functions for a given prediction on a toy function [7]. The maximum of each acquisition function corresponds to the new candidate that will be added to the training set by the Bayesian Optimisation algorithm. Thus, when using EI, the new sample lies at 0.481 whereas for the Var, it lies at 1.

From the reference, it is obvious that samples have to be added near 1 since the global minimum lies here. Adding samples in other locations induces unnecessary calls to the solver. Nevertheless, the EI is completely blind about this area. On the contrary, the Var strongly highlights the interest of this area. This area is emphasized due to a lack of exploration (no samples from the training set are present in this area) and since the prediction mean value is too high, the EI discards this area.

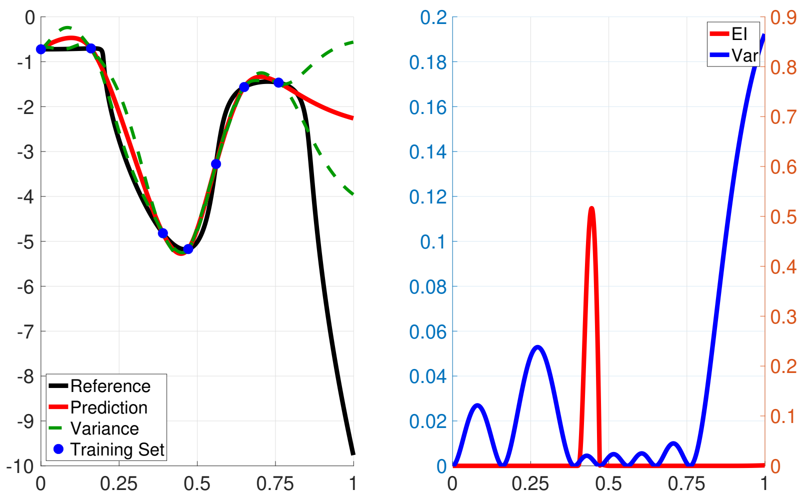


Fig. 2: Common prediction of the Gaussian Process (left figure) and comparison between the two acquisition function on a toy function (right figure)

Instead of questioning completely the EI, we propose to remedy this exploration defect by introducing a sample coming from the Var. Thus, at each iteration, two samples are added: one determined with the Expected Improvement acquisition function and one with the variance acquisition function. We call this new strategy: Multi-Samples Bayesian Optimisation (hereby denoted MS-BO).

3 Numerical applications

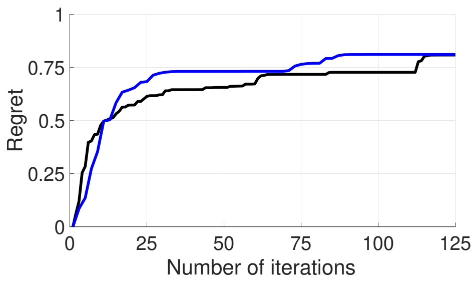
This section aims to highlight the performance of the suggested method. Hence, several functions are considered and exhibited in Appendix A. All functions have multiple local minima and are two-dimensional. The Function n°1 corresponds to a rescaled version of the Schwefel function [8] with zero mean and unitary variance. The other functions are either modified existing functions or new functions that we have designed to add complexity to the Bayesian Optimisation convergence process and challenge the methods.

The performance of the EI and the suggested method are evaluated with the regret function (Eq. 2). This metrics is adimensional and allows to quantify the percentage of improvement from the initial minimum, $\min(\widehat{f}^0)$, to the global minimum, $\min_G$. 1 means that the global minimum is reached.

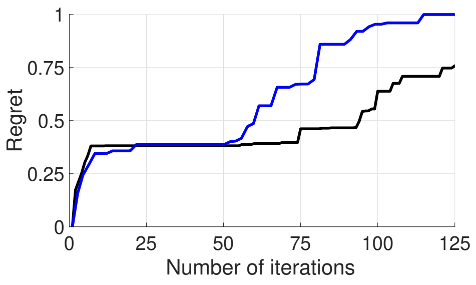
$$r(\widehat{f}^i) = \frac{\min(\widehat{f}^i) - \min(\widehat{f}^0)}{\min_G - \min(\widehat{f}^0)} \tag{2}$$

The regret function is computed after the execution of the BO algorithm for 10 training sets initialized with 6 points per dimension. The algorithm is set for 125 iterations. The results are averaged over the 10 training sets and the evolution of this mean is displayed in Fig. 3.

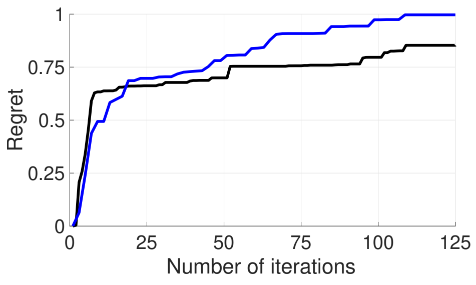
First of all, it is clear that MS succeeds in providing more efficient convergence patterns that EI. Indeed, for 3 functions over 4, the final optimum determined by MS-BO is the global optimum whereas EI-BO does not converge toward the global optimum. Then, the convergence patterns are faster for MS-BO than EI-BO for all considered functions. For 3 functions over 4 (Function n°1, Function n°2 and Rescaled Schwefel), the convergence towards the global optimum is slightly slower at the beginning: during 20 iterations, EI-BO performs better than MS-BO. Nevertheless, the gap is quickly filled and EI-BO is surpassed by MS-BO.



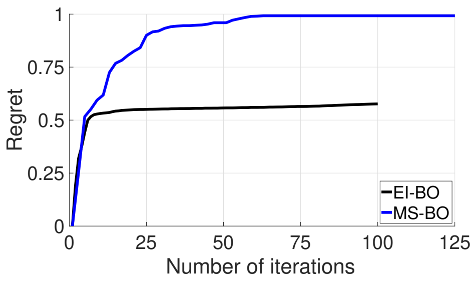
(a) Function n°1



(b) Function n°2



(c) Function n°3



(d) Function n°4

Fig. 3: Convergence plots for the considered functions (black: Bayesian Optimisation with Expected Improvement, blue: Bayesian Optimisation with Multi-Samples Bayesian Optimisation)

4 Conclusions

In this communication, an exploration defect has been highlighted on the Expected Improvement acquisition function in the Bayesian Optimisation algorithm. This flaw induces sub-optimal search of the design space and, thus, reduced the performance of this algorithm.

A new strategy (Multi-Samples Bayesian Optimisation) has been suggested to overcome this defect. Numerical experimentations have shown that this new method outperforms the traditional Expected Improvement both in efficiency and speed.

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. The authors gratefully acknowledge the support of these institutions. Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr)

References

1. Jonas Mockus. *Bayesian Approach to Global Optimization - Theory and Applications*. Kluwer Academic Publishers, 1989.
2. Carl Edward Rasmussen and Chris Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.
3. P. Frazier. A tutorial on bayesian optimization. *ArXiv*, abs/1807.02811, 2018.
4. 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, 2016.
5. Donald Jones, Matthias Schonlau, and William Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 12 1998.

6. Eric Brochu, Vlad M. Cora, and Nando de Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning, 2010.
7. Jeremy Sadet, Franck Massa, Thierry Tison, El-Ghazali Talbi, and Turpin Isabelle. Uncertain friction induced vibration problems: Integration of gaussian process and deep learning methods. In *14th World Congress on Computational Mechanics*, 2021.
8. Manuel Laguna and Rafael Martí. Experimental testing of advanced scatter search designs for global optimization of multimodal functions. *Journal of Global Optimization*, 33(2):235–255, 2005.

A Mathematical functions

Function n°1 defined for $\mathbf{x} \in [-500; 500] \times [-500; 500]$, with $\bar{f} = 837.9658$ and $\bar{\sigma} = 273.75725$

$$f(\mathbf{x}) = \frac{418.9829d - \sum_{i=1}^{d} x_i \sin\left(\sqrt{|x_i|}\right) - \bar{f}}{\bar{\sigma}} \tag{3}$$

Function n°2 defined for $\mathbf{x} \in [0; 1] \times [0; 1]$

$$f(\mathbf{x}) = -0.5\left(\sin\left(40(x_1 - 0.85)^4\right)\cos(2.5(x_1 - 0.95)) + 0.5(x_1 - 0.9x_2) + 1\right) \tag{4}$$

Function n°3 defined for $\mathbf{x} \in [0; 1] \times [0; 1]$

$$f(\mathbf{x}) = \sin(20(x_1 - 1)^4) + \cos^4(x_2 - 0.8) - (x_1 - \frac{7}{6}x_2)^2 + \frac{1}{2}(x_2 - 0.8)^3 \tag{5}$$

Function n°4 defined for $\mathbf{x} \in [0; 5] \times [1; 5]$

$$f(\mathbf{x}) = \left(x_2 - \frac{3107}{333}x_1 + \frac{2839}{230}x_1^2 - \frac{569}{859}x_1^3 - \frac{2850}{1097}x_1^4 + \frac{659}{805}x_1^5 - \frac{177}{2479}x_1^6\right)x_2^2 \tag{6}$$

The Construction of Uniformly Deployed Set of Feasible Solutions for the p-Location Problem

P. Czimmermann[1]

1. Department of Mathematical Methods and Operations Research, Faculty of Management Science and Informatics, University of Zilina, Slovakia
Peter.Czimmermann@fri.uniza.sk

Keywords: p-location problem, uniformly deployed set, Hamming distance, voltage graph.

Abstract. Feasible solutions of the p-location problems can be represented by n-bit binary words with exactly p ones. A set of selected solutions is called a t-uniformly deployed set, if the minimal Hamming distance between each pair of solutions is at least $2p - 2t$ for a given natural number t. The uniformly deployed sets can be used, due to their diversity, as starting population in evolutionary algorithms for p-location problems. In our contribution, we present a method for construction of appropriate t-uniformly deployed sets. The origins of this method trace back to the topological graph theory and we have adapted it to our purpose.

1 Introduction

A lot of public service design problems are represented by weighted p-median and p-center problems. It is known that these problems belong to the family of hard computational problems [1]. Hence, various metaheuristics are used to solve them [2]. An important class of metaheuristics are evolutionary algorithms. These methods involve a set of starting feasible solutions. It is reasonable to suppose in location problems that this set has high diversity. Since the feasible solutions of p-location problems can be represented by n-bit binary words with exactly p ones, the diversity of the solutions can be measured by the Hamming distance [3]. A set of solutions with minimal Hamming distance $2p - 2t$ is called a t-uniformly deployed set ($t \in N$ is the maximum number of overlapping 1's in any two words).
Remark. We notice that the construction of the set, in which the Hamming distance between each pair of solutions is exactly $2p - 2t$, leads to the hard combinatorials problems with no fast algorithms to solve them. For example, we can point to the construction of difference sets or strongly regular graphs with given parameters.

In this paper, we introduce a fast algorithm for the construction of t-uniformly deployed sets from voltage graphs (digraphs). The sets will be given by rows of an adjacency matrix of a digraph derived from a voltage digraph.

2 Definitions

In this section, we provide some important definitions of notions that are used later.

2.1 The p-Location Problem

The p-location problem is a task of locating p-centers at some of the n possible locations from the set I. It can be defined by (1), where the decision variable y_i gets the value one if a center is located at $i \in I$ and it gets zero otherwise.

$$min\{f(\vec{y}); y_i \in \{0,1\}, i \in I, \textstyle\sum_{i \in I} y_i = p\} \quad (1)$$

Where $f(\vec{y})$ is an appropriate objective function. It is known that only few versions of $f(\vec{y})$ lead to the problems solvable in polynomial time [4], [5].

2.2 Hamming Distance

Let two n-bit binary words $\vec{x} = (x_1, \cdots, x_n)$, $\vec{y} = (y_1, \cdots, y_n)$ be given. The Hamming distance of $\vec{x}$ and $\vec{y}$ is

$$H(\vec{x}, \vec{y}) = \sum_{i=1}^{n} |x_i - y_i|.$$

If the words $\vec{x}$ and $\vec{y}$ contain exactly p ones, then their Hamming distance is an even number

$$H(\vec{x}, \vec{y}) \in \{0,2,4, \cdots, 2p\}.$$

Let $\vec{x}$ and $\vec{y}$ represent two feasible solutions of the p-location problem and $H(\vec{x}, \vec{y}) = 2q$ (where $q \leq p$). The expression

$$\frac{2p - 2q}{2} = p - q = t$$

gives the number of locations contained in both solutions.

2.3 The t-Uniformly Deployed Sets

Let I_p be the set of all feasible solutions of a given p-location problem. Hence, I_p contains all n-bits binary words with exactly p ones. The t-uniformly deployed set is a subset $S \subseteq I_p$ such that the inequality $H(\vec{x}, \vec{y}) \geq 2p - 2t$ holds for each $\vec{x}, \vec{y} \in S$. It means that any two words $\vec{x}$ and $\vec{y}$ from S have at most t ones on the same positions [6].

2.4 Digraphs

For our needs, we use a more general definition of digraphs. A digraph D is a pair (V, E), where V is a non-empty set of vertices, and E is a set of directed edges. Every edge has exactly one starting vertex and one end vertex. Multiple edges and loops are also allowed. We say that e_1 and e_2 are multiple edges, if they have the same starting vertex and the same end vertex. We say that edge e is a loop, if it starts and ends at the same vertex. A monopole is a digraph that contains only one vertex, and all its edges are loops. Monopole with p edges is denoted by M_p. The outdegree of vertex u is the number of edges, which start at u. We say that vertex v is a successor of vertex u, if there is an edge from u to v. The adjacency matrix of a digraph is a square matrix $A = (a_{i,j})_{n \times n}$ such that $a_{i,j}$ represents the number of edges from i to j.

2.5 Groups and Modular Arithmetic

A group $(X,*)$ is a non-empty set X with binary operation $*$ defined on X such that

1) $\forall a,b \in X \quad a * b \in X$,
2) $\forall a,b,c \in X \quad (a * b) * c = a * (b * c)$,
3) $\exists e \in X$ such that $\forall a \in X \quad a * e = a = e * a$,
4) $\forall a \in X \quad \exists \bar{a} \in X$ such that $a * \bar{a} = e = \bar{a} * a$.

For example, integers Z with operation + form the group $(Z, +)$. In modular arithmetic, there exists another important class of groups. The set of all remainders of division by k is denoted by Z_k. It means that

$$Z_k = \{0,1, \cdots, k-1\}$$

We can define addition $\oplus_k$ on Z_k by the expression

$$a \oplus_k b = mod(a + b, k)$$

where $a, b \in Z_k$ and $mod(x, y)$ is the remainder after dividing x by y. It is possible to show that $(Z_k, \oplus_k)$ is the group for any $k \in N$. Sometimes, we can omit the operation and parentheses, and the group $(X,*)$ can be denoted by X.

2.6 Voltage Digraphs

The construction of large graphs and digraphs from voltage graphs and digraphs is a method that was invented in topological graph theory [7]. This method was later used in the Degree/diameter problem [8], [9], [10], [11].

Let a graph (digraph) $G = (V, E)$ and a group $(X, *)$ be given. If every edge $e \in E$ has assigned a value $\alpha(e) \in X$, then G is called voltage graph (digraph), and values $\alpha(e)$ on its edges are called voltages.

We say that $G_X = (V_X, E_X)$ is a graph (digraph) derived from G, if

1) its vertex set contains all ordered pairs from $V \times X$, (where the vertex $(u, i) \in V \times X$ is denoted by u_i),

2) a pair of vertices $u_i, v_j \in V_X$ forms an edge from E_X if and only if there is an edge $e \in E$ from u to v in G such that $i * \alpha(e) = j$.

Example: We can consider the voltage digraph G and the group Z_3 in Figure 1. The derived digraph G_{Z_3} can be seen in Figure 2.

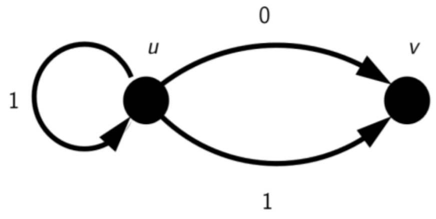


Figure 1 Voltage digraph G.

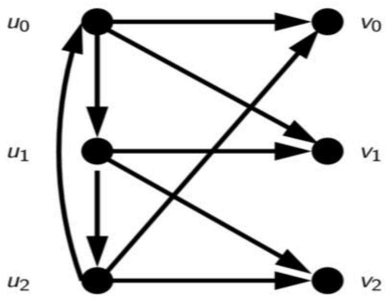


Figure 2 Digraph derived from G.

3 The Construction of Uniformly Deployed Sets from Voltage Digraphs

In [8], there is shown the construction of the Hoffmann-Singleton graph from a voltage graph with two vertices and the group $Z_5 \times Z_5$. The Hoffmann-Singleton graph contains 50 vertices, each vertex has degree seven, each pair of adjacent vertices has no common neighbour, and each pair of non-adjacent vertices has exactly one common neighbour. It follows from these facts that any two rows of its matrix have the Hamming distance 12 or 14. Hence, the rows of this matrix form the 1-uniformly deployed set for $n = 50$ candidates and $p = 7$ locations. However, the Hoffmann-Singleton is a graph with many special properties. It is a strongly regular graph, a Moore graph, and a triangle-free graph. It is known that graphs with these properties occur rarely. This is the reason why we decided to construct digraphs with less constraints by this method. It is possible to show that a t-uniformly deployed set (n, p and t are given) can be obtained from adjacency matrix of a digraph on n vertices, in which every vertex has outdegree p and each pair of vertices has at most t common successors. We denote such digraphs by $D(n, p, \leq t)$. In this paper, we study possible constructions of $D(n, p, \leq t)$ from monopoles M_p and groups Z_n.

Example. We show the construction of 1-uniformly deployed set for $n = 10$ and $p = 3$ by this way. We start with monopole M_3 and group Z_{10}. Let the vertex of M_3 be denoted by v and edges e_1, e_2 and e_3. We assign the following voltages to these edges: $e_1 \to 1$, $e_2 \to 2$, and $e_3 \to 5$. The adjacency matrix of the derived digraph is

$$\begin{pmatrix}
0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0\\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0\\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0\\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1\\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0\\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0\\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1\\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1\\
1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

We can check that the rows of this matrix form the 1-uniformly deployed set. All information about this digraph is in quintuple

$$(M_3, Z_{10}, 1,2,5).$$

In general, for M_p and Z_n, let v_i and v_j be vertices with common successor v_k. It means that the edges

$$(v_i, v_k), (v_j, v_k) \in E_{Z_n}.$$

Hence, there exist voltages $\alpha, \beta \in Z_n$ on edges of G such that $i \oplus_n \alpha = k$ and $j \oplus_n \beta = k$. From these equations, we obtain

$$j = i \oplus_n \alpha \oplus_n \bar{\beta},$$

where $\bar{\beta}$ is the inverse of β in Z_n. The list of all such vertices v_j, which have common successors with a given vertex v_i, can be obtained from the following table. We will call this table a range matrix for $(p+2)$-tuple

$$(M_p, Z_n, \alpha_1, \cdots, \alpha_p).$$

$\oplus_n$	$\overline{\alpha_1}$	$\overline{\alpha_2}$	$\cdots$	$\overline{\alpha_p}$
α_1	0	$\alpha_1 \oplus_n \overline{\alpha_2}$	$\cdots$	$\alpha_1 \oplus_n \overline{\alpha_p}$
α_2	$\alpha_2 \oplus_n \overline{\alpha_1}$	0	$\cdots$	$\alpha_2 \oplus_n \overline{\alpha_p}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
α_p	$\alpha_p \oplus_n \overline{\alpha_1}$	$\alpha_p \oplus_n \overline{\alpha_2}$	$\cdots$	0

The number of occurrences of value $\gamma \in Z_n$ in the range matrix is the answer to the question: how many common successors do the vertices v_i and $v_{i \oplus \gamma}$ have?

Example. The range matrix for $(M_3, Z_{10}, 1,2,5)$ is

$\oplus_{10}$	9	8	5
1	0	9	6
2	1	0	7
5	4	3	0

Each value from Z_{10}, except the zero, occurs in the range matrix at most once. It means that a vertex v_i has exactly one common successor with vertices $v_{i\oplus 9}$, $v_{i\oplus 6}$, $v_{i\oplus 1}$, $v_{i\oplus 7}$, $v_{i\oplus 4}$, $v_{i\oplus 3}$, and no common successor with vertices $v_{i\oplus 2}$, $v_{i\oplus 5}$, $v_{i\oplus 8}$, since the range matrix does not contain values 2, 5, and 8.

Example. A 1-uniformly deployed set for $n = 80$, $p = 8$ can be represented by 10-tuple

$$(M_8, Z_{80}, 1,2,4,12,21,27,34,39).$$

The corresponding range matrix is

$\oplus_{80}$	79	78	76	68	59	53	46	41
1	0	79	77	69	60	54	47	42
2	1	0	78	70	61	55	48	43
4	3	2	0	72	63	57	50	45
12	11	10	8	0	71	65	58	53

21	20	19	17	9	0	74	67	62
27	26	25	23	15	6	0	73	68
34	33	32	30	22	13	7	0	75
39	38	37	35	27	18	12	5	0

Each value from Z_{80}, except the zero, occurs in the range matrix at most once. Hence, this 10-tuple represents a 1-uniformly deployed set.

4 How to Construct the Set of Voltages

The main computational problem is the construction of an appropriate set of voltages. We present this problem for the Bratislava Region, where we have 87 candidates for emergency stations and we need to choose 14 locations. We have $\binom{87}{14}$ possibilities how to do it and we also have the same number of possibilities for the set of voltages.

Hence, in this section, we present the algorithm for choosing the voltages to obtain the derived digraph with parameters $D(n,p,\leq t)$. We define for these purposes an increasing sequence $\{a_i\}_{i=1}^{\infty}$ of nonnegative integers. We will call it a t-sequence and it can be stated recursively:
1. $a_1 = 0$, $a_2 = 1$.
2. For $k > 2$, a_k is the minimum value such that for all $i \in \{1,\cdots,k-1\}$, the value $a_k - a_i$ occurs between values $a_j - a_i$ (where $1 \leq i < j < k$) at most $t-1$ times.
The first q members of t-sequence can be computed by the following algorithm:

```
Let $a_1 := 0; a_2 := 1;$
For $k = 3,\cdots,q$
   $x := a_{k-1} + 1;$
   $A_k := \{a_j - a_i; 1 \leq i < j < k\};$
   While $a_k = 0$
      $y := 1;$
      For $i = 1,\cdots,k-1$
         If $x - a_i \in A_k$ at most $t-1$ times
            Then $y := y \cdot 1;$
            Else $y := y \cdot 0;$
      If $y = 1$ Then $a_k = x;$
              Else $x := x + 1;$
```

Where A_k is multiset.
Examples of the first $q = 15$ members of t-sequences for $t = 1,2,3,4$ can be seen below:

$t = 1$	0,1,3,7,12,20,30,44,65,80,96,122,147,181,203
$t = 2$	0,1,2,4,7,11,16,22,30,38,48,61,73,86,103
$t = 3$	0,1,2,3,5,8,12,16,21,27,33,40,48,57,71
$t = 4$	0,1,2,3,4,6,9,13,17,22,27,33,39,46,53

If we want to construct a digraph $D(n,p,\leq t)$ from monopole M_p, group Z_n and t-sequence for appropriate t, then we can use the following procedure:

1) If $a_p < n/2$, then the voltages on edges are the members of t-sequence. It follows from the properties of t-sequences that the digraph derived from $(M_p, Z_n, a_1, \cdots, a_p)$ is $D(n,p,\leq t)$.
2) If $a_{p-x} < n/2$, and $a_{p-x+1} \geq n/2$ (for small $x \in N$, for example $x \in \{1,2,3,4\}$), then the voltages are $\alpha_i = a_i$ for $i = 1,2,\cdots,p-x$.
For $k \in \{p-x+1,\cdots,p\}$, $\alpha_k \in \{\alpha_{k-1}+1,\cdots,n-1\}$ is the minimum value such that for all $i \in \{1,\cdots,k-1\}$, the values $\alpha_k \oplus_n \overline{\alpha_i}$ and $\alpha_i \oplus_n \overline{\alpha_k}$ occurring in the multiset

$$\{\alpha_i \oplus_n \overline{\alpha_j}; \forall i,j \text{ such that } 1 \leq i,j \leq k, i \neq j\}$$

at most t times.

3) If we still do not have p voltages, then we can increase $t \coloneqq t+1$ and repeat step 2 to complete the set of voltages.

Example. A 4-uniformly deployed set for the Bratislava Region can be constructed from

$$(M_{14}, Z_{87}, 0,1,2,3,4,6,9,13,17,22,27,33,39,46)$$

where all voltages are computed by previous procedure from 4-sequence.

5 Limitations

What are the limits of parameters n, p, and t, when we construct t-uniformly deployed sets from M_p and Z_n? From the range matrix, we have inequality

$$p(p-1) \leq t(n-1),$$

where $p(p-1)$ is the number of non-zero elements in a range matrix and $n-1$ is the number of non-zero elements in Z_n. From this inequality, we have some upper and lower bounds for n, p, and t. Lower bounds for n computed from inequality

$$\frac{p(p-1)}{t} + 1 \leq n$$

can be seen in table

$p \setminus t$	1	2	3	4
10	91	46	31	24
20	381	191	128	96
30	871	436	291	219

Lower bounds for t computed from inequality

$$\frac{p(p-1)}{n-1} \leq t$$

can be seen in table

$p \setminus n$	100	200	300
10	1	1	1
20	4	2	2
30	9	5	3

From inequality

$$p^2 - p - t(n-1) \leq 0,$$

we have interval

$$p \in \left\langle \frac{1-\sqrt{1+4t(n-1)}}{2}, \frac{1+\sqrt{1+4t(n-1)}}{2} \right\rangle$$

and some upper bounds for p can be found in table

$n \setminus t$	1	2	3	4
100	10	14	17	20
200	14	20	24	28
300	17	24	30	35

6 Computational Results

Colleagues J. Janacek and M. Kvet tested the efficiency of using the UDS in various heuristics to solve the weighted p-median problem and its generalised version. Their results can be found in [3], [6], [12], and [13]. We tested Swap and Path-relinking heuristics for the weighted p-median problem in [14]. Some numerical results (for generalised p-median problem) can be seen in the following table (taken over from [15]), which shows the tests of the discrete self-organizing migrating algorithm (DSOMA) with and without UDS extension. Benchmarks for the tests are derived from the self-governing regions of Slovakia.

Regions	n	p	Opt Sol	DSOMA$_U$	Time$_U$ [s]	DSOMA	Time [s]
BB	515	36	44752	44907	30.3	44923	30.1
KE	460	32	45588	45733	17.6	46099	17.5
NR	350	27	48940	48996	8.5	49986	8.3
PO	664	32	56704	56936	2.8	60476	20.5
TN	276	21	35275	35789	3.4	49260	3.2
TT	249	18	41338	41432	2.0	44090	2.0
ZA	315	29	42110	42140	8.7	42145	8.7

The columns of the table mean:

Regions – shortcuts of the self-governing regions of Slovakia (Bratislava region is omitted),

n – the number of candidates for placing an emergency station,

p – the number of emergency stations that need to be located,

Opt Sol – optimal value of the objective function for the generalised weighted p-median problem,

DSOMA$_U$ – values of the objective function obtained by DSOMA with UDS extension,

DSOMA – values of the objective function obtained by basic version of DSOMA,

Time$_U$, Time – computation time.

7 Conclusions

In our contribution, we introduce the construction of t-uniformly deployed sets from voltage graphs. We study possible constructions from monopoles with elements from Z_n as voltages. We also derive some limitations for these classes of graphs and groups. The constructions from more complicated voltage graphs and groups will follow in our next paper. The effect of using t-uniformly deployed sets in genetic algorithms is tested in [13], and [14], where the authors present its efficiency on real data from regions of Slovakia. The solutions that can be obtained by this method could have applications in real-life contexts, such as the location of emergency stations within certain environs.

Acknowledgements

This work was supported by the research grants VEGA 1/0342/18 Optimal dimensioning of service systems, VEGA 1/0089/19 Data analysis methods and decisions support tools for service systems supporting electric vehicles. This work was supported by the Slovak Research and Development Agency under the Contract no. APVV-19-0441.

References

[1] O. Kariv, S. Hakimi (1979). An Algorithmic Approach to Network Location Problems. *SIAM Journal on Applied Mathematics*, 37, pp.513-560.

[2] J. Janacek, L. Janosikova, L. Buzna (2012). *Optimized Design of Large-Scale Social Welfare Supporting Systems on Complex Networks.* Chapter 12 in Handbook of Optimization in Complex Networks: Theory and Applications, Springer, editors: Thai, M.T., Pardalos, P.M.

[3] J. Janacek, M. Kvet (2019). Usage of Uniformly Deployed Set for p-Location Min-Sum Problem with Generalized Disutility. In: SOR 2019: *Proceedings of the 15th International Symposium on Operational Research,* ISBN 978-961-6165-55-6, pp. 494-499.

[4] C. Klein, R. Kincaid (1994). The Discrete Anti-p-center Problem. *Transportation Science*, 28(1).

[5] P. Czimmermann, S. Pesko (2015). A Polynomial Algorithm for a Particular Obnoxious Facility Location Problem. In: SOR 2015*: Proceedings of the 13th International Symposium on Operational Research,* pp. 427-432.

[6] M. Kvet, J. Janacek (2019). Population Diversity Maintenance using Uniformly Deployed Set of p-Location Problem Solutions. In: SOR 2019: *Proceedings of the 15th International Symposium on Operational Research,* ISBN 978-961-616555-6, pp. 354-359.

[7] J. Gross, T. Tucker (2012). *Topological Graph Theory*. Dover Publications; Reprint edition 2012, ISBN 978-0486417417.

[8] J. Siagiova (2001). A note on the McKay-Miller-Siran graphs. *Journal of Combinatorial Theory Ser.B* 81.

[9] T. Vetrik (2008). An upper bound for graphs of diameter 3 and given degree obtained as Abelian lifts of dipoles. *Discussiones Mathematicae Graph Theory*, Vol.28, No.1.

[10] O. Czimmermannova, P. Czimmermann (2008). On the number of voltages on walks in dipoles. *Journal of Information, Control and Management Systems,* Vol. 6, No. 2.

[11] E. Loz (2009). Graphs of given degree and diameter obtained as abelian lifts of dipoles. *Discrete Mathematics,* 309.

[12] J. Janacek, M. Kvet (2019). Uniform Deployment of the p-Location Problem Solutions. *Operations Research Proceedings 2019,* Springer, in print.

[13] J. Janacek, M. Kvet (2020). Efficient Incrementing Heuristics for Generalized p-Location Problems, *Central European Journal of Operations Research,* accepted for publication.

[14] J. Janacek, M. Kvet, P. Czimmermann (2021). Kit of Uniformly Deployed Sets for p-Location Problems. Submitted in *Annals of Operations Research*.

[15] M. Kvet, J. Janacek (2021). *Unpublished results.*

Adaptive iterative destruction construction heuristic for the firefighters timetabling problem

Mohamed-Amine Ouberkouk, Jean-Paul Boufflet and Aziz Mourkim

Université de technologie de Compiègne, CNRS, Heudiasyc (Heuristics and Diagnosis of Complex Systems), CS 60 319 - 60 203 Compiègne Cedex
{mohamed-amine.ouberkouk,jean-paul.boufflet,aziz.moukrim}@hds.utc.fr

Abstract. Every year, wildfires accentuated by global warming, cause economic and ecological losses, and often, human casualties. Increasing operating capacity of firefighter crews is of importance to better face the forest fire period that yearly occurs. In this study, we investigate the real-world firefighters timetabling problem (FFTP) of the INFOCA institution in Andalusia (Spain) with the aim of increasing operating capacity while taking into account work regulation constraints. We propose an Integer Linear Programming model and an Adaptive Iterative Destruction Construction Heuristic solution approache to address the problem. We report on experiments performed on datasets generated using real-world data of the INFOCA institution. The work was initiated as part of the GEO-SAFE project[1].

Keywords. Timetabling, Firefighters, ILP, Adaptive destruction/contruction heuristic

1 Introduction

Timetabling problems [1, 7, 9] involve allocating resources within time slots considering a predefined planning horizon while respecting precedence, duration, capacity, disjunctive and distribution (spacing, grouping) constraints. Staff planning aims at building timetables so that an organization can meet demands for goods or services. For each staff member, working and rest days are scheduled in a timetable while taking into account work regulation constraints and local regulation constraints, if any.

The first works on personnel scheduling can be traced back to Edie's work on traffic delays at toll booths [5]. Since then, scheduling algorithms have been applied in a lot of areas like transportation systems (airlines, railways), healthcare systems, emergency services (police, ambulances), call centers and other services (hotels, restaurants, commercial stores).

Comprehensive literature reviews covering a wide area of problems with many references on personnel scheduling can be found in [6, 10]. The works are classified by type of problem, application area and solution method. As an example, the nurse rostering [4] is a scheduling issue in health systems. The objective is to build a daily schedule for nurses with the aim of obtaining a full timetable over few weeks for the institution. The rosters should provide suitably qualified nurses to cover the demand of working shifts arising from the numbers of patients in the wards. The resulting schedule should comply with regulatory constraints and should ensure that night and weekend shifts are fairly distributed while accommodating nurse preferences.

Staff scheduling is known as crew scheduling in transportation systems areas such as market/airlines, railways, mass transit and buses [2]. For these problems, there are two common features. The first is that both temporal and spatial constraints are involved. Each task is characterized by its starting time and location, and, its ending time and location. The second is that all tasks to be performed by employees are determined from a given timetable. The tasks are determined following a decomposition of the different duties that the company must ensure within a planning period. A task may be assuring a flight leg in airlines or ensuring a trip between two segments in a train.

The firefighters problem that we address consists in providing the INFOCA's daily schedule within a fixed planning horizon for a number of firefighter crews. Each firefighter is assigned to a crew for a year. These firefighters crews can be assigned to several types of shifts such as helicopter

[1] https://geosafe.lessonsonfire.eu/

work, night work, work on demand (24 hour on call). The planning period is the high-risk period from 1st June to 15th October where wildfires yearly occur (forest fire period).

The objective is to build a schedule for every crew of firefighters, hence a full timetable that covers all the forest fire period. The aim is to maximize the overall operating capacity while respecting the minimum demands for each shift, the regulatory constraints imposed by the institution as well as other soft constraints of good practice in order to make the schedules adequate to the preferences of the institution. The constraints of good practice relate to the grouping of assignments of same shifts within consecutive days, the allocation of compensations after rest days while maximizing of the number of operational crews a day.

The application of various metaheuristics to employee scheduling problems is presented in the reviews mentioned above. In this study, we choose to investigate an algorithm mainly based on an Adaptive Iterative Destruction/Construction Heuristic (AIDCH) [3]. An initial feasible solution that only complies with the minimum demands is build first by applying a constructive heuristic. Then, the AIDCH approach that we propose aims at increasing the overall operating capacity by first partly destroying a solution, next it is completed by inserting as many crews as possible, that can be easily done through a Destruction/Construction Heuristic approach. While completing the solution to increase the overall operational capacity, we make work together adaptive diversification mechanisms and parallel independent searches to avoid to be trapped in a local optimum.

In this paper we propose an Integer Linear Programming (ILP) formulation together with an Adaptative Iterative Destruction Construction Heuristic (AIDCH) to address the firefighters timetabling problem (FFTP) of the INFOCA institution. The ILP is designed for modeling purposes and with the aim of giving lower bounds useful for the tuning analysis of the AIDCH solution approach. The Adaptive Iterative Destruction/Construction Heuristic is composed of an adaptive diversification mechanism at the destruction phase followed by an adaptive construction phase, based on a Best Insertion Algorithm, which performs parallel independent searches. The initial parameter values are adjusted by the algorithm according to the solution progress throughout the resolution process. The AIDCH is appropriate to generate solutions of good quality for the larger instances. The remainder of the paper is organized as follows. Section 2 provides a description of the FFTP, then the ILP formulation is presented in Section 3. The proposed AIDCH solution approach is described in Section 3. Computational experiments performed on a benchmark that we generated using real data of the INFOCA firefighter institution are reported in Section 4. Conclusion and future works are given in Section 5.

2 Problem description

In this section we present a global overview of the real-world firefighter planning problem that we address. We gives the set of daily working shifts to be considered, we introduce the hard constraints to be respected and the soft constraints used to assess the quality of a solution.

The notations used for the types of shifts and their brief descriptions are the following:

(T12) from 8 am to 4 pm at fire station, regular daily shift;
(T16) from 3 pm to 10 pm at fire station, regular daily shift;
(H) from 8 am to 4 pm at fire station, regular daily shift, assigned to a helicopter;
(N) from 10 pm to 8 am at fire station, regular night shift;
(G7) from 7 am to 3 pm at fire station, stand-by to face instantly any extra urgent request;
(G24) 24h guard, crew stay at home but may be mobilized to face any urgent situation;
(A3) from 8 am to 6 pm at fire station (or elsewhere) for training purposes;
(R) rest day;
(C) additional compensation day granted when a number of hours have been worked.

For the considered firefighters timetabling problem, the hard constraints relating to work regulation and to local regulation of the INFOCA institution are the following:

(H1) one shift a day: a firefighter crew can only be assigned to one shift a day;
(H2) minimum demands: each daily shift has a minimum demand of firefighter crews;
(H3) forbidden shift successions: some shift assignments on consecutive days are forbidden;
(H4) maximum workload: over the planning horizon, a maximum workload for every crew should not be exceeded;

(H5) compensation: compensation days are granted according to the hours worked, they should be used;
(H6) maximum consecutive working days: every firefighter crew have a maximum number of consecutive working days.

Some consecutive shift assignment are forbidden for a crew (H3), for instance a night shift ends at 8 am and cannot be followed by an helicopter shift which begins at 8 am, this forbidden consecutive shift assignment is denoted as (N, H).

Soft constraints are constraints of good practice that should be satisfied as best as possible. The violation of any soft constraint induces a penalty. A weighted sum of the penalties measures the quality of the solution produced. For the studied firefighters timetabling problem, the soft constraints are the following:

(S1) shift grouping: assignments of a crew to the same shift should be grouped. Each shift assignment change between two consecutive days is penalized;
(S2) same start time: start times should be the same whatever the working shifts over consecutive working days. Each starting time change for working shifts between two consecutive days is penalized;
(S3) compensation assignments: compensation day assignments should be right after the rest days, the aim is to allow firefighters to have a short vacation during the planning period. Each assignment of compensation not right after rest days is penalized.
(S4) period fairness: for the sake of fairness the workload should be balanced between the crews over the planning period. The unbalance of workload between crews should be minimized;
(S5) preferences: each crew assignment to an undesired shift is penalized;
(S6) evenly balance extra daily shifts: assigning of extra crews to the different shifts should be balanced each day. The unbalance on extra assignment to different shifts should be minimized each day.

Provided the minimum demand (H2) is respected, the idea beyond (S6) is to ensure a balance between shift assignments. If we can assign three extra crews for a day, we had better to assign a crew to three different shifts to balance operating capacity rather than assigning the three crews to a same shift.

3 ILP model for FFTP

In this section we present the ILP model for minimizing the criteria detailed in Section 2. The ILP has a twofold objective, first a modeling purpose for investigating the problem we face, second we aim at obtaining optimal values whether possible for the smaller instances within a reasonable time limit (or lower/upper bounds). This allows to get reference values to make comparisons with the AIDCH solution approach that we propose. We present data and parameters prior to the decision variables, we then give the model.

The data and parameters are the following:

$Days$ set of days of the planning period, a day $d \in [1, \cdots, l_d]$, size n_d;
$Shifts$ set of types of shifts, a shift $s \in \{T12,\ T16,\ H,\ N,\ G7,\ G24,\ A3, R,\ C\}$, size n_s;
$Crews$ set of firefighter crews, size n_c;
l_d last day of the planning period;
F set of couples of forbidden consecutive shift assignment, e.g. $(N, H) \in F$;
r_s daily minimum demand for a working shift $s \in \{Shifts \setminus \{R,\ C\}\}$;
l_s duration of shift s (length in hours);
L maximum workload for any crew over the planning period;
t_s start time of shift s;
w_{oc} **o**perating **c**apacity weight;
w_{sg} shift **g**rouping violation weight (S1);
w_{sst} same **s**tart **t**ime change violation weight (S2)
w_{ca} **c**ompensation **a**ssignments violation weight (S3);
w_p **p**references violation weight (S5);

p_{csd} if crew c does not prefer to work on shift s on day d $p_{csd} = w_p$, zero otherwise (S5);
MAX_d maximum number of consecutive work days for a crew (H6);
WHC number of worked hours giving a compensation day.

The primary boolean variables are X_{csd}, if the crew c works on shift s in day d then $X_{csd} = 1$, zero otherwise. The secondary boolean variables used in the model are the followings:

$\alpha_{css'd} = 1$ if crew c works on shift s in day d and works on a different shift s' in day $d+1$, zero otherwise;
$\beta_{css'd} = 1$ if crew c works on shift s in day d and works on a different shift s' in day $d+1$ with $t_s \neq t_{s'}$, zero otherwise;
$\gamma_{css'd} = 1$ if the crew c works on shift s in day d with $s \neq' R'$ and is assigned to shift $s' =' C'$ in day $d+1$, zero otherwise.

$\alpha_{css'd} = 1$ if a shift change violation occurs (S1, shift grouping), $\beta_{css'd} = 1$ if a working time change violation occurs (S2, same start time) and $\gamma_{css'd} = 1$ if a compensation assignment violation occurs (S3, compensation assignment).

The integer variables used in the model are the followings:

λ_d daily difference between the maximum number of assignable crews (n_c) and those assigned;
δ_c total number of worked shifts for crew c over the planning period;
θ_c total working time of crew c over the planning period;
ρ_{cd} number of worked hours of crew c from the first day to day d;
$\phi_{cc'}$ number of shift assignment difference between the crews c and c' (S4);
$\varphi_{cc'}$ working time difference between the crews c and c' (S4);
$\psi_{ss'}$ unbalance of assignments between the shifts s and s' (S6).

The aim is to maximize operating capacity over the planning period while minimizing the soft constraint violations. We propose the following ILP to address this problem:

Min

$$w_{oc} \cdot \sum_{d \in Days} \lambda_d \tag{1a}$$

$$\sum_{c \in Crews} \sum_{s \in Shifts \setminus \{R,\, C\}} \sum_{s' \in Shifts \setminus \{R,\, C\}} \sum_{d \in Days} (w_{sg} \cdot \alpha_{css'd} + w_{sst} \cdot \beta_{css'd} + w_{ca} \cdot \gamma_{css'd}) \tag{1b}$$

$$+ \sum_{c \in Crews} \sum_{c' \in Crews} (\phi_{cc'} + \varphi_{cc'}) \tag{1c}$$

$$+ \sum_{c \in Crews} \sum_{s \in Shifts \setminus \{R,\, C\}} \sum_{d \in Days} p_{csd} \cdot X_{csd} \tag{1d}$$

$$\sum_{s \in Shifts \setminus \{R,\, C\}} \sum_{s' \in S \setminus \{R,\, C\}} \psi_{ss'} \tag{1e}$$

Subject to:

$$\sum_{s \in Shifts} X_{csd} = 1 \quad \forall c \in Crews,\ \forall d \in Days \tag{2}$$

$$\sum_{c \in Crews} X_{csd} \geq r_s \quad \forall d \in Days,\ \forall s \in \{Shifts \setminus \{R,\, C\}\} \tag{3}$$

$$X_{csd} + X_{cs'(d+1)} \leq 1 \quad \forall (s, s') \in F,\ \forall c \in Crews,\ \forall d \in Days \setminus \{l_d\} \tag{4}$$

$$\sum_{s \in \{Shifts \setminus \{R,\, C\}\}} \sum_{d \in Days} l_s \cdot X_{csd} \leq L \quad \forall c \in Crews \tag{5}$$

$$\sum_{s \in \{Shifts \setminus \{R,\, C\}\}} \sum_{d' \in Days, d' \leq d} l_s \cdot X_{csd} = \rho_{cd} \quad \forall c \in Crews,\ \forall d \in Days \tag{6}$$

$$\sum_{d' \in Days, d' \leq d} X_{csd} \leq \frac{\rho_{cd}}{WHC} \quad s = \text{'C'},\ \forall c \in Crews,\ \forall d \in Days \tag{7}$$

$$\sum_{d \in Days} X_{csd} = \left\lfloor \frac{\rho_{c(l_d)}}{WHC} \right\rfloor + 1 \quad s = \text{'C'}, \forall c \in Crews \tag{8}$$

$$\sum_{s \in \{Shifts \setminus \{R,\ C\}\}} \ \sum_{d' \leq (1+MAX_d),(d+d') \leq l_d} X_{csd} \leq MAX_d \quad \forall c \in Crews,\ \forall d \in Days \tag{9}$$

$$\sum_{c \in Crews} \ \sum_{s \in \{Shifts \setminus \{R,\ C\}\}} X_{csd} = n_c - \lambda_d \quad \forall d \in Days \tag{10}$$

$$X_{csd} + X_{cs'(d+1)} \leq 1 + \alpha_{css'd} \quad \begin{cases} \forall s, s' \in \{Shifts \setminus \{R,\ C\}\}, s \neq s' \\ \forall c \in Crews, \forall d \in \{Days \setminus \{l_d\}\} \end{cases} \tag{11}$$

$$X_{csd} + X_{cs'(d+1)} \leq 1 + \beta_{css'd} \quad \begin{cases} \forall s, s' \in \{Shifts \setminus \{R,\ C\}\},\ s \neq s',\ with\ t_s \neq t_{s'} \\ \forall c \in Crews, \forall d \in \{Days \setminus \{l_d\}\} \end{cases} \tag{12}$$

$$X_{csd} + X_{cs'd+1} \leq 1 + \gamma'_{css'd} \quad \begin{cases} s \in \{Shifts \setminus \{R,\ C\}\},\ s' = \text{'C'} \\ \forall c \in Crews,\ \forall d \in \{D \setminus \{l_d\}\} \end{cases} \tag{13}$$

$$\sum_{s \in Shifts \setminus \{R,\ C\}} \ \sum_{d \in Days} X_{csd} = \delta_c \quad \forall c \in Crews \tag{14}$$

$$\sum_{s \in Shifts \setminus \{R,\ C\}} \ \sum_{d \in Days} l_s \cdot X_{csd} = \theta_c \quad \forall c \in Crews \tag{15}$$

$$\delta_c - \delta_{c'} \leq \phi_{cc'} \quad \forall c, c' \in Crews,\ c \neq c' \tag{16}$$

$$\theta_c - \theta_{c'} \leq \varphi_{cc'} \quad \forall c, c' \in Crews,\ c \neq c' \tag{17}$$

$$\left(\sum_{c \in Crews} X_{csd} - r_s \right) - \left(\sum_{c \in Crews} X_{cs'd} - r_{s'} \right) \leq \psi_{ss'} \quad \begin{cases} \forall s, s' \in \{Shifts \setminus \{R,\ C\}\} \\ \forall d \in Days \end{cases} \tag{18}$$

$$X_{csd},\ \alpha_{css'd},\ \beta_{css'd},\ \gamma_{css'd} \in \{0,1\} \tag{19}$$

$$\delta_c, \theta_c, \rho_{cd}, \phi_{cc'}, \varphi_{cc'}, \psi_{ss'} \in \mathbb{N} \tag{20}$$

The five terms of the objective function aims at maximizing operating capacity while minimizing the soft constraint violations. The first term (1a) aims at maximizing operating capacity. The weighted sum (1b) assesses the (S1, shift grouping), (S2, same start time) and (S3, compensation assignments) soft constraint violations. The period fairness (S4) soft constraint relates to the number of shift assignment differences and to the working time differences between crews, they are considered using the (1c) term. The preferences of the firefighters (S5) are considered using the (1d) term. The evenly balance of extra daily shifts (S6) is considered using the (1e) term.

The hard constraints **one shift a day** (H1) are enforced by Equation (2). The hard constraints **minimum demands** (H2) are enforced by Equation (3). The hard constraints **forbidden shift successions** (H3) are enforced by Equation (4). The hard constraints **maximum workload** (H4) are enforced by Equation (5). The hard constraints **compensation** (H5) are enforced by Equations (6)-(8). For a crew c and a day d, Equation (6) count ρ_{cd}, the number of worked hours of crew c from the first day of the planning period to day d, and links variables X_{csd} and ρ_{cd}. For a crew c and a day d, Equation (7) forces the number of compensation days ($s =' C'$) being assigned to be less or equal to (ρ_{cd}/WHC) since one compensation day is granted when WHC worked hours are made. For a crew c, all the compensation days must be assigned over the planning horizon (until $d = l_d$), this is enforced by Equation (8). The hard constraints **maximum consecutive working days** (H6) are enforced by Equation (9). For a crew c and a day d, the crew is assigned to at most MAX_d consecutive working shifts (rest and compensation days are not to be considered).

The daily differences between the maximum number of assignable crews (n_c) and those assigned are to be minimized to optimize the overall operating capacity, the λ_d values are assessed by Equation (10).

Consider a crew c, two days d and $d+1$, if the crew is assigned to two different shifts ($s \neq s'$) a **shift grouping** (S1) soft constraint violation occurs and Equation (11) sets $\alpha_{css'd} = 1$. Consider a crew c, two days d and $d+1$, if the crew is assigned to two different shifts ($s \neq s'$) and the start times of these shifts are different ($t_s \neq t_{s'}$) a **same start time** (S2) soft constraint violation occurs and Equation (12) sets $\beta_{css'd} = 1$. Consider a crew c, two day d and $d+1$, if the crew is assigned to a working shift ($s \neq$ 'R') on day d, and if this crew is assigned to a compensation day ($s' =$ 'C') on day $d+1$ a **compensation assignment** (S3) soft constraint violation occurs and Equation (13) sets $\gamma_{css'd} = 1$. Every compensation day assignment will be right after a rest day (constraints of good practice imposed by the institution).

Consider a crew c, Equation (14) counts δ_c the total number of worked shifts over the planning period and Equation (15) counts θ_c the total working time over the planning period. Hence, Equation (16) gives $\phi_{cc'}$ the number of shift assignment differences. Given that $\phi_{cc'} \in \mathbb{N}$, a negative difference involves $\phi_{cc'} = 0$, so for any couple of crews only positive differences are counted. The same rationale applies on Equation (17) for $\varphi_{cc'}$, the number of working time differences. These variables $\phi_{cc'}$ and $\varphi_{cc'}$ are used for the **period fairness** (S4) soft constraint violations assessment.

We recall that **preferences** (S5) soft constraint violations are assessed by Equation (1d).

Consider a day d and two shifts s and s', Equation (18) aims at **evenly balance extra daily shifts** (S6). Minimum demands (H2) are enforced by Equation (3), assigning of extra crews to shifts should be balanced each day within the forest fire period to increase operating capacity.

Equation (19) defines variables X_{csd}, $\alpha_{css'd}$, $\beta_{css'd}$ and $\gamma_{css'd}$ as boolean. Equation (20) defines variables δ_c, θ_c, ρ_{cd}, $\phi_{cc'}$, $\varphi_{cc'}$ and $\psi_{ss'}$ as integers.

4 Adaptive iterative destruction/construction heuristic

We propose an Adaptive Iterative Destruction/Construction Heuristic (AIDCH) to compute solutions of good quality for larger instances of the FFTP. The Algorithm 1 gives the global scheme of the AIDCH proposed approach. We use the adaptive construction approach *BuildFeasibleSchedule()* to build an initial solution which respects the hard constraints. The initial solution complies with *minimum demands* (H2) but there is room for improvement in operating capacity.

Algorithm 1: General structure of AIDCH

```
Input      : An instance of FFTP
Output     : $S_{best}$ best solution found
Parameters: $D_{limit}$ limit for diversification degree, $n_c$ number of crews
             $n_s$ number of type of shifts
Variables  : $iter$ number of iterations, $MaxIter$ maximum iteration
             $D_{max}$ diversification degree, $S_{cur}$ current solution
$iter := 0$
$MaxIter := n_c$
$D_{max} := 3$
$D_{limit} := \left\lceil \frac{n_c}{n_s} \right\rceil$
$S_{cur}$ := BuildFeasibleSchedule()
$S_{best} := S_{cur}$
while $iter < MaxIter$ do
    $k$ := rand(1,$D_{max}$)
    AdaptativeDestruction($S_{cur}$,$k$)  /* adaptive diversification */
    AdaptativeConstruction($S_{cur}$)  /* insert as many crews as possible in $S_{cur}$ */
    if $S_{cur} > S_{best}$ then
        $S_{best} := S_{cur}$
        $iter := 0$
        $D_{max} := 3$
    else
        $iter++$
        $D_{max}$ := min($D_{max}$+1,$D_{limit}$)
    end
end
```

Provided a feasible solution, at each iteration, a part of the solution is destroyed by removing at random a number k of crews, then it is completed by inserting as many crews as possible in order to increase the operating capacity (while respecting the hard constraints). At each overall iteration at most D_{max} crews are removed ($k \leq D_{max}$). Therefore, we define D_{max} as the degree of

Algorithm 2: Best Insertion Algorithm

```
Input      : $S_{cur}$ a partial solution
             $(\alpha, \beta, \gamma, \theta, \omega, \mu)$ parameter set
Output     : $S_{best}$ best solution found
Variables  : (d,s,c)* best triplet, success boolean
$S_{best} := S_{cur}$ /* store reference solution for BIA */
success := true
while success do
    (d,s,c)* := $(\emptyset, \emptyset, \emptyset)$
    foreach $d \in Days$ do
        foreach $s \in Shifts$ do
            foreach $c \in Crews$ do
                ComputeBIC(d,s,c)
                UpdateBestTriplet (d,s,c)*
            end
        end
    end
    success := Insert($S_{cur}$, (d,s,c)*) /* if no feasible insertion, Insert returns false */
    /* Comparing $S_{cur}$ and $S_{best}$, all terms of the objective function are assessed */
    if $S_{cur} > S_{best}$ then
        $S_{best} := S_{cur}$
    end
end
```

diversification. The D_{max} value is initialized to 3, next incremented after each non-improving overall iteration up to D_{limit}. We set $D_{limit} = \lceil n_c/n_s \rceil$ which represents the average number of crews that can be assigned to shifts. Provided an improvement is found, D_{max} is reset to 3 to entirely explore the neighborhood of the new solution. We perform an adaptive construction procedure to complete the solution. This process is reiterated and it stops when $MaxIter$ overall iterations have been performed without improving the quality of the solution. We set $MaxIter = n_c$. The final result is the best solution found over all iterations.

The proposed AIDCH algorithm makes use of an adaptive diversification mechanism with the aim to escape from local optima. We explore the neighborhood of the new solution as soon as an improvement is found. We explore more distant zones by increasing D_{max} whenever the search is trapped in a local optimum.

The main component of the AIDCH heuristic is the *AdaptativeConstruction(S_{cur})* procedure, an adaptive construction heuristic based on a Best Insertion Algorithm (BIA) shown in Algorithm 2. The BIA algorithm considers a partial solution S_{cur}, and tries to insert as many crews as possible in S_{cur}, one by one. At each iteration, the BIA assesses all feasible insertions that respect the hard constraints and scores them according to a Best Insertion Criterion (BIC). The best insertion is then performed and the quality of S_{cur} is assessed considering all terms of the objective function (1a)-(1e). This process is iterated until no more valid insertion is possible. The algorithm returns the updated S_{cur}, the best solution over all the BIA iterations.

To evaluate the insertion of a crew in the planning (day, shift), we propose to compute the Best Insertion Criterion (BIC) as follows:

$$(SG^{\alpha} * SST^{\beta} * CA^{\gamma} * PF^{\theta} * P^{\omega} * EB^{\mu})$$

The aim is to minimize the soft constraints violation whether the insertion is performed. In case a hard constraint is violated (e.g. maximum workload (H4)), the BIC is set to $+\infty$. The criterion is composed of 6 terms, one for each soft constraints: SG is for the **S**hift **G**rouping (S1), SST is for the **S**ame **S**tart **T**ime (S2), CA is for the **C**ompensation **A**ssignments (S3), PF is for the **P**eriod **F**airness (S4), P is for the **P**references (S5) and EB is for **E**venly **B**alance extra daily shifts (S6). The terms are weighted with parameters α, β, γ, θ, ω and μ in order to control their relative importance.

At each iteration i of the AIDCH heuristic, *AdaptativeConstruction(S_{cur})* works as follows. Four constructive heuristics launch separately BIA with different values of the parameter set $(\alpha, \beta, \gamma, \theta, \omega, \mu)$ on the current solution. During each launch, α, β, γ, θ, ω and μ are chosen randomly in the 6 dimension space having the center $(\alpha_{i-1}, \beta_{i-1}, \gamma_{i-1}, \theta_{i-1}, \omega_{i-1}, \mu_{i-1})$ and the side length ϕ, where α_{i-1}, β_{i-1}, γ_{i-1}, θ_{i-1}, ω_{i-1} and μ_{i-1} are the best parameters obtained by the method at previous iteration. All four BIA being applied, the parameter set that produces the best solution is stored to be used in the next iteration.

Finally, the best solution obtained among the four methods is retained as the current solution. This aims at performing parallel independent searches in the solutions space and at choosing the best values of the parameters to better explore the solutions space to speed-up the convergence of the AIDCH algorithm toward a good solution.

5 Computational experiments

In our experiments, our objectives were: (i) to show the adaptive construction impact, by comparing ϕ together with the best parameter set that produces the best solution at previous iteration to compute the next parameter set, versus a fully randomized parameter set; (ii) to show the efficiency of the adaptive destruction, impact of an adaptive D_{max} for perturbations versus a constant one; (iii) to compare performances between the ILP model and the AIDCH approach within a 3600 seconds time limit.

Tests were done using C++ compiled with gcc version 7.5.0, using STL, using a CPLEX 12.10 [8] solver with a single thread and the *MipEmphasis* parameter set to *feasibility*, on a machine with an Intel(R) Xeon(R) X7542 CPU @ 2.6 GHz and 64 GB of RAM.

Datasets overview and performance metric

We tested the ILP and AIDCH approaches on a benchmark composed of 4 datasets, each having 7 instances, that we generated using real data of the INFOCA firefighter institution. Datasets have been created to be of increasing difficulty, the firsts of reasonable sizes given that the ILP may face difficulty to get a solution within the time limit. The instances in datasets are ranged according to the number of crews n_c and to the total daily number of working shifts demands (i.e. $\sum r_s$). So, instances are denoted as $cXXrYY(a/b)$, the (a/b) notation is used whether n_c and $\sum r_s$ equals for two distinct instances which are different in minimum demands distributions.

For each instance, the AIDCH algorithm is run 10 times. We recorded the **R**elative **P**ercentage **E**rror, we defined as $RPE = 100 * (Z_{best} - Z_{max})/Z_{best}$ and the **A**verage **R**elative **P**ercentage **E**rror, we defined as $ARPE = 100 * (Z_{best} - Z_{avg})/Z_{best}$ where Z_{max} is the best result obtained among the ten executions, Z_{avg} is the average result obtained among the ten runs and Z_{best} is the best solution found by the AIDCH approach for the according instance. The ARPE criterion aims at investigating whether the AIDCH is stable over the runs.

To compare the solutions found by the AIDCH approach against the solutions attained by the ILP approach, we define the **R**elative **P**ercentage **G**ap as $RPG = 100*(Z_{ILP} - Z_{max})/Z_{ILP}$ where Z_{ILP} represents the solution value attained, if any, by the ILP approach for an instance.

For our experiments using the ILP, we set w_{oc} to 2, w_{sg} to 1, w_{sst} to 1, w_{ca} to 1 and w_p to 2.

Impact of the adaptive construction mechanism

We first carried out preliminary experiments to choose the best value of ϕ that is necessary to show the impact of the adaptive construction mechanism, because of lack of space those experiments are not reported here. According to these experiments, the parameter value $\phi = 0.1$ provides the best results considering RPE.

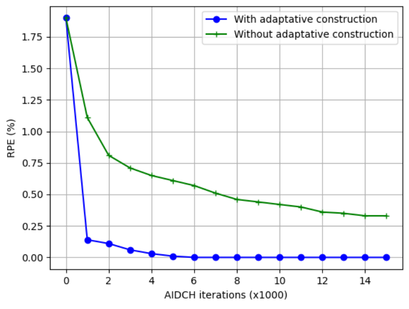


Fig. 1. Adaptive construction impact

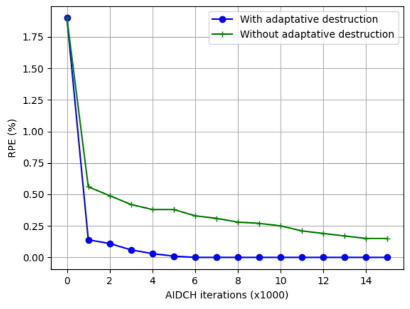


Fig. 2. Adaptive destruction impact

The adaptive construction mechanism aims to guide the search by computing at each time the best trade-off between the different terms of the BIC representing soft constraints violations. To show whether it is efficient, we conducted experiments with the adaptive construction mechanism and without the adaptive construction mechanism. In that latter case, the parameters of BIC are chosen randomly in $[0, 1]$ at each iteration. In these experiments, for each instance, the algorithm is launched and we record the best solution for the first 15000 iterations. We performed these tests using 2 instances chosen at random from each dataset. We report in Figure 1 the average of RPE values computed for the 8 chosen instances against the number of iterations.

As it can be shown in Figure 1, the adaptive construction mechanism permits to converge faster toward good solutions rather than without adaptive construction mechanism.

Impact of the diversification mechanism

To evaluate the effectiveness of the adaptive destruction, we tested a version of AIDCH where the diversification degree D_{max} is set to 3. As aforementioned, we record the best solution for the first 15000 iterations using this fixed value. We proceed in the same way using the adaptive diversification mechanism that makes use of D_{max} to explore the neighborhood of the new solution as soon as an improvement is found and also to explore more distant zones whenever the search is trapped in a local optimum.

Figure 2 shows the average of RPE values recorded against the number of iterations for these two versions. The adaptive diversification mechanism, achieved using the management of D_{max}, permits to converge faster toward good solutions rather than without its use.

Based on these two graphs, we can easily notice that the average of RPE values with the adaptive mechanisms is always below the average of RPE values with the standard perturbation at each iteration, which shows the effectiveness of our proposed technique.

Instance	ILP	t (s)	gap	AIDCH	t (s)	RPG	ARPE	Instance	ILP	t (s)	gap	AIDCH	t (s)	RPG	ARPE
c18r09a	1325	1443	0	1325	341	0	0	c50r22a	ns	-	nc	3765	741	nc	0.43
c18r10a	1359	1409	0	1359	352	0	0	c50r23a	ns	-	nc	3783	754	nc	0.31
c18r10b	1344	1526	0	1344	348	0	0	c50r26a	ns	-	nc	3799	759	nc	0.27
c18r11a	1378	1886	0	1378	372	0	0	c50r28a	ns	-	nc	3823	783	nc	0.58
c18r11b	1420	2786	0	1420	401	0	0	c50r31a	ns	-	nc	3947	849	nc	0.52
c18r12a	1422	2103	0	1422	391	0	0	c50r33a	ns	-	nc	3931	817	nc	0.56
c18r12b	1440	2209	0	1440	413	0	0	c50r35a	ns	-	nc	4097	831	nc	0.34
c30r15a	1767	-	0.74	1758	553	-0.51	0.1	c70r31a	ns	-	nc	4913	943	nc	0.67
c30r16a	1801	-	1.18	1811	561	0.56	0.15	c70r33a	ns	-	nc	4957	954	nc	0.71
c30r17a	1818	-	0.44	1860	582	2.31	0.22	c70r37a	ns	-	nc	5102	995	nc	0.69
c30r18a	1834	-	0.11	1867	593	1.80	0.08	c70r40a	ns	-	nc	5151	1034	nc	0.65
c30r19a	1889	-	0.48	1934	612	2.38	0.13	c70r44a	ns	-	nc	5213	1067	nc	0.71
c30r20a	2144	-	12.9	1947	661	-9.19	0.26	c70r47a	ns	-	nc	5557	1113	nc	0.83
c30r21a	1966	-	2.77	1936	657	-1.53	0.16	c70r50a	ns	-	nc	5401	1158	nc	0.67

Table 1. Performances of ILP and AIDCH approaches

ILP versus AIDCH

Table 1 compares the results obtained by the ILP solver against those obtained by the AIDCH approach. In Table 1, **ns** stands for **n**o **s**olution, **nc** stands for **n**ot **c**alculable, and **-** shows that the 3600 seconds time limit has been attainted. For the sake of compactness, datasets are grouped by two then tabulated side by side. Column *Instance* gives the instance label. The next tree columns, *ILP*, *t (s)* and *gap* show the performances of the ILP. They report the objective function value, the computing time and the gap found by the CPLEX solver. Then, the next four columns, *AIDCH*, *t (s)*, RPG, and *ARPE* show the performances of the AIDCH approach. They report the objective function value, the computing time, the gap between the solutions found by the AIDCH approach and the solution provided by the ILP solver and the average of RPEs over the 10 runs for an instance.

The ILP approach attains optimal solutions for all $n_c = 18$ instances. It faces difficulty for the second dataset having $n_c = 30$, however feasible solutions are obtained within the 3600s time limit.

For the third and the fourth datasets having $n_c = 50$ and $n_c = 70$, the ILP approach fails to find a feasible solution within the time limit.

For the first dataset, the AIDCH approach succeeded in obtaining all the optimal solutions found by the ILP approach. We also notice that all the ARPE values are equal to 0: which means that the AIDCH approach was able to attain the optimal solutions.

For the second dataset, the AIDCH approach attains solutions closed to or better than the solutions obtained by the ILP approach within a 3600s time limit. For four instances the RPG values are between 0.56 and 2.38. For the three other instances, the AIDCH approach obtains better solutions than the ones provided by the ILP approach, with an RPG values from -0.51 up to -9.19. ARPE values are less than 0.26 for all instances which shows the stability of our proposed heuristic approach for this dataset.

For the third and the fourth datasets, the AIDCH approach was able to find solutions in a reasonable time. The ARPE values are less than 0.83, the proposed heuristic behaviour is stable over the last two datasets. Unfortunately, the quality of the solutions found by the AIDCH approach cannot be assessed since the ILP approach fails to provide solutions for these datasets within the one hour time limit.

6 Conclusion and future work

We presented in this paper both an ILP model and a AIDCH heuristic to address the real-worl firefighters timetabling problem (FFTP) of the INFOCA institution. The proposed approaches were tested over four datasets with different sizes of increasing difficulty that we generated using real data from INFOCA. The ILP approach obtained optimal or near optimal solutions for the first two datasets, but it faced difficulty in obtaining feasible solutions for the larger instances of the two other datasets. The AIDCH approach obtained good solutions for all the instances of the first two datasets, those are either optimal or closed to the ones obtained by the ILP approach. The proposed heuristic approach was able to find feasible solutions for the larger instances within a reasonable computation time. Future works aim at investigating a metaheuristic solution approach to improve the quality of the solutions obtained over the datasets and aim at reducing the computation time. We also plan to obtain lower bounds for the larger instances for comparison purposes.

Acknowledgments

This work was initiated as part of the GEO-SAFE project. The GEO-SAFE project has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie RISE grant agreement No 691161. This work is carried out in the framework of the Labex MS2T, which was funded by the French Government, through the program "Investments for the future" managed by the National Agency for Research (Reference ANR-11-IDEX-0004-02).

References

1. Aggarwal, S. C. (1982). A focussed review of scheduling in services. European Journal of Operational Research, 9(2), 114-121.
2. Barnhart, C., Cohn, A. M., Johnson, E. L., Klabjan, D., Nemhauser, G. L., Vance, P. H. (2003). Airline crew scheduling. In Handbook of transportation science (pp. 517-560). Springer, Boston, MA.
3. Ben-Said, A., El-Hajj, R., Moukrim, A. (2016). An adaptive heuristic for the Capacitated TeamOrienteering Problem. In 8th IFAC Conference on Manufacturing Modelling, Management and Control (pp 1662-1666).
4. Burke, E. K., De Causmaecker, P., Berghe, G. V., Van Landeghem, H. (2004). The state of the art of nurse rostering. Journal of scheduling, 7(6), 441-499.
5. Edie, L. C. (1954). Traffic delays at toll booths. Journal of the operations research society of America, 2(2), 107-138.
6. Ernst, A. T., Jiang, H., Krishnamoorthy, M., Owens, B., Sier, D. (2004). An annotated bibliography of personnel scheduling and rostering. Annals of Operations Research, 127(1), 21-144.
7. Ernst, A. T., Jiang, H., Krishnamoorthy, M., Sier, D. (2004). Staff scheduling and rostering: A review of applications, methods and models. European journal of operational research, 153(1), 3-27.
8. IBM, CPLEX, User's Manual, https://www.ibm.com, (2020)
9. Tien, J. M., Kamiyama, A. (1982). On manpower scheduling algorithms. SIAM review, 24(3), 275-287.
10. Van den Bergh, J., Beliën, J., De Bruecker, P., Demeulemeester, E., De Boeck, L. (2013). Personnel scheduling: A literature review. European journal of operational research, 226(3), 367-385.

A Genetic Algorithm for a capacitated lot-sizing problem with lost sales, overtimes, and safety stock constraints

Benoît Le Badezet[1], François Larroche[12], Odile Bellenguez[1], and Guillaume Massonnet[1]

IMT Atlantique, LS2N, La Chantrerie, 4 rue Alfred Kastler, 44307 Nantes France
VIF, 10 Rue de Bretagne, 44240 La Chapelle-sur-Erdre
benoit.le-badezet@etu.univ-nantes.fr, {francois.larroche, odile.bellenguez, guillaume.massonnet}@imt-atlantique.fr

Abstract. This paper deals with a complex production planning problem with lost sales, overtimes, safety stock and sequence dependent setup times on parallel and unrelated machines. The main challenge of this work is to propose a solution approach to obtain a good feasible plan in a short execution time (around 2 minutes) for large industrial instances. We develop a genetic algorithm that combines several operations already defined in the literature to solve the problem. Preliminary numerical results obtained with our algorithm are presented and compared to a straightforward MIP resolution. The method appears to be an appealing alternative on large instances when the computational time is limited.

1 Introduction

The problem presented in this paper is related to practical cases encountered in the food industry for production planning. In this context, manufacturers can generally use several production lines, each able to make several types of items. This complexity usually leads to problems that are too large to be solved optimally by off-the-shelf solvers. In addition, the models we consider in this paper also combines constraints from the lot-sizing and the scheduling literature, by assuming that the setup times between different types of items depends on the production sequence. This further limit the applicability of standard methods in practice, when the planners need to obtain "good" feasible solutions in reasonable time to test several machine configurations or shifts assignments and obtain quick insights to support their decisions.

This problem extends the field of lot-sizing, which has been extensively studied since the work of Wagner and Whitin [1]. Motivated by the physical constraints found in practical applications, the finite production capacity version of the problem (CLSP) has received a lot of attention, see [2] and [3] for a review of extensions and solution approaches. The problem we consider is an extension of the industrial problem with lost sales and shortage costs presented in [4], for which the authors introduce new classes of valid inequalities. The safety stock is seldom considered in the deterministic production and inventory literature. [5] define the safety stock as a lower bound on the number of units that must be held in the inventory at each period when [6] choose to penalize the missing units from the safety stock. The latest version is studied here. Versions of the problem with parallel machines and sequence dependent setups are less common in the literature. [7] develop new heuristics on a parallel machines problems. [8] present an industrial problem in which setup times depend on the sequence of production and propose a solution procedure based on subtour elimination and patching. [9] use a small bucket formulation to compute the sequence of production. [10] also present an extensive review of this extension and compare the efficiency of several methods to solve it. The possibility to exceed the production capacity is not common in the literature, see [11] for an overtime extension of a capacitated lot-sizing problem.

In terms of metaheuristics, various researches have been done on the previously detailed extensions of our problem. [12] propose a Genetic Algorithm (GA) to tackle a multi-items CLSP and on multiple production lines, using various crossovers and mutation. The authors also use a new operator called "siblings" that consists in a local search using a ranking system on the neighbours. [13] propose a Tabu-Search (TS) to solve the same problem. [14] and [15] propose hybridized GA to solve the CLSP with an overtime constraints. The hybridization introduces elements of Tabu-search and Simulated Annealing into the GA in order to improve the efficiency of the algorithm. On top of that, they also use multi-population on different version of the algorithm to tackle their instances. On the single-machine CLSP with sequence dependent setup times, [16] and [17] propose a Threshold Accepting whereas [18] develop a Tabu-Search and [19] propose a GA. To the best of our knowledge however, none of the previous problems incorporate a target-stock constraint similar to our case.

In the following, we denote CLSSD-PM the *multi-item capacitated lot-sizing problem with lost sales, safety stock, overtimes, and sequence dependent setups on parallel machines*. A previous work in [20] focus on a part of this problem without safety stock. To the best of our knowledge, this whole problem has never been studied in the literature before.

2 Problem Definition

The CLSSD-PM is a extensive version of the capacitated lot-sizing problem which is proven to be NP-hard ([21]). The goal is to plan the production of N different items, over T time periods and on M parallel unrelated production lines. There is a demand d_t^i for each item $i \in \{1, ..., N\}$ in each period $t \in \{1, ..., T\}$ that must be satisfied if units of i are available in stock. When that is not the case, the demand can be (partially or totally) lost, incurring a per-unit lost sales cost l_t^i. Any production of item i in period t on line $m \in \{1, ..., M\}$ is an integral number of batches, i.e. a multiple of a fixed quantity Q^i of units. The production of one such batch incurs a cost p_{mt}^i and requires a production time τ_m^i. In addition, the production of items of type k immediately after items of type $i \neq k$ during a given period on machine m induces a setup time γ_m^{ik}.

Each line m at each period t has a (planned) time capacity of C_{mt}, but production overtimes are allowed up to a maximum total production time $\bar{C}_{mt}$. When production occurs during the planned capacity, the corresponding cost of line usage is c_{mt} per unit of time, but this cost increases to $c_{mt} + \bar{c}_{mt}$ when the production needs overtime, i.e. for any usage that exceeds C_{mt}.

We model item storage by the mean of a target stock S_{it} for each item i in each period t. Any unit of stock of i in period t that exceeds S_{it} induces an excess storage cost of h_{it}^+, while missing inventory to reach the target stock incurs a per-unit penalty equal h_{it}^-.

We also make the following hypothesis on our problem :

- Demand and inventory are satisfied and consumed following a FIFO rule. This implies that it is impossible to choose to loose some demand of an item that is held in stock.
- Setup times between items follow the triangle inequality rule.
- At the beginning of each period, each line is in a neutral state, and the setup time to start the production of the first item in any period is null.

The objective of our problem is to minimize the total cost of the production planning (line usage, production, storage and lost sales combined). For conciseness reasons, we do not present the MILP formulation here and instead refer the interested readers to the [22].

3 Genetic Algorithm

We now develop a genetic algorithm to address the CLSSD-PM. We start by introducing the general structure of the procedure, before presenting in mode details the chromosome representation, crossover and mutation operators.

3.1 Genetic Algorithm Pseudo-Code

We use a generational genetic algorithm (GA), which creates successive generations of a population of individuals, by using specific operators inspired by nature and called crossovers and mutations. We keep some overlapping between consecutive generations, i.e. some of the best elements obtained in the current population are retained for the next generation, to keep the most interesting information of what has been done in previous iterations.

To avoid being in a local optimum for too long, the algorithm sometimes performs a reset that re-generates randomly a large portion of the current population. This operation is done only after a long period without improvement of the best known solution. A pseudo-code of the procedure is presented in Algorithm 1.

Algorithm 1: *Genetic Algorithm*

```
1  curGen ← GeneratePopulation() ;
2  $s^*$ ← bestIndividual ;
3  while stopping criterion not met do
4      nextGen ← overlap(curGen);
5      while |nextGen| < maxPopSize do
6          if condCrossover then
7              parent1, parent2 ← selectionCross(curGen);
8              nextGen.add(crossover(parent1, parent2));
9          end
10         if condMutation then
11             mutated ← selectionMutation(curGen);
12             nextGen.add(mutation(mutated));
13         end
14     end
15     if condReset then
16         reset();
17     end
18     if cost(nextGen.bestIndividual) < cost($s^*$) then
19         $s^*$ ← nextGen.bestIndividual ;
20     end
21     curGen ← nextGen ;
22 end
23 return $s^*$
```

3.2 Chromosome representation

The problem requires two type of decisions: The first one assigns the production of items to periods and machines, while the second one aims at designing the production sequences. As a consequence, we propose the following independent variables that serve as chromosomes:

- x_{mt}: Set of tuples ⟨ item ; quantity ⟩ produced on m during t.
- w_{mt}: Contains the ordered sequence of production on m during t.

Other necessary information to represent a solution are deduced from these two variables, using the dependent variables below:

- $cost$: Total cost of the solution.
- u_{mt}: Time usage of line m in period t.
- $prod^i_t$: Number of batches of i produced in period t.
- $stock^i_t$: Stock of item i available at the end of period t.
- L^i_t: Number of lost sales for item i in period t.

To ensure diversity within the population, we start a completely random chromosome generation. For each line in each period we draw randomly a subset of items and affect to each of them a random production quantity. The sequence is determined as the items are drawn. Since the goal is to minimize the objective function of our problem, we keep a fitness parameter $fitness = \frac{1}{cost(solution)}$ updated to ensure that the gaps between the costs of different solutions are proportional. Finally, the selection is made based on a roulette wheel mechanism, applied to the fitness of the population.

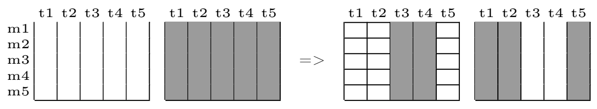
3.3 Crossover

In order to explore a large variety of solutions, we apply several crossovers from one generation to the next, in a similar fashion as the GA presented in [12]. In our case, we have three different crossover :

- On periods.
- On items.
- On sequences.

Crossover on periods. This crossover is heavily inspired by [12]. It basically consists of a two-point crossover applied on the periods of the solutions. The concept is to choose randomly a subset of periods and exchange all the production quantities of the two parents in the selected periods. Table 1 illustrates a crossover on periods 3 and 4.

Table 1. Illustration of the period crossover



Crossover on items. This crossover is inpired by [12] For a given machine m and a given time period t, this crossover iterates following the item information stored in the chromosomes x_{mt} of both parents. For each m and t, we consider the union the items produced by the two parents and draw for each of them a random boolean. If we draw 0 then the first child takes the first parent's production, and the second child takes the second parent's. Otherwise the first child takes the second parent's production, and the second child takes the first parent's. Table 2 shows a practical example of this crossover for a given period and line.

Table 2. Example of item crossover for a given period and line

Parent 1 :

item	1	4	3
quantity	15	13	9

Parent 2 :

item	2	9	4
quantity	17	10	5

Union and random draws :

union items	1	4	3	2	9
random draws	1	1	0	0	0

Child 1 :

item	4	3
quantity	5	9

Child 2 :

item	1	4	2	9
quantity	15	13	17	10

Crossover on the sequences. This crossover is inspired by [19] This crossover enables us to change the sequence of production. For each line and in each period, we form the set containing the common items from the two parent solutions. We then create a new sequence in the following manner:

1. Draw a random integer X between 1 and the number of common items
2. Order the X first item as they are in the sequence of the first parent. The remaining items follow the same order they have in the sequence of the second parent.
3. Create the sequences of the children using the parents sequences in which the common items are reordered.

Table 3 shows a practical example of this crossover for a given period and line.

3.4 Mutation

We consider a mutation that swaps the positions of two randomly selected items in the sequence of production, as represented in Table 4. As we also do not want to alter the totality of the individual we will add a parameter to describe the amount of of information that will be altered in a mutated individual.

Table 3. Example of sequence crossover for a given period and line

Parent 1 :

sequence	1	2	3	4	5	6	7	11

Parent 2 :

sequence	10	9	8	7	6	5	4	3

Intersection :

common items	3	4	5	6	7

Random number draw X : 3

Order from the parents :

Order of the first X items on parent 1	3	4	5
Order of remaining items on parent 2	7	6	

New order :

New order	3	4	5	7	6

Child 1 :

sequence	1	2	3	4	5	7	6	11

Child 2 :

sequence	10	9	8	3	4	5	7	6

Table 4. Example of a mutated sequence on items 2 and 6

sequence before mutation	1	2	3	4	5	6
sequence after mutation	1	6	3	4	5	2

3.5 Repair

Note that such movements may result in infeasible solution since some line usage may exceed its maximum capacity. When this situation arise, we repair them by removing the production of one or more items until we don't exceed the hard capacity anymore and then replace it if possible on previous periods. In order to have a minimum impact on the quality of the solution, we chose to remove the item having the highest ratio $\frac{prod_t^i}{demand_t^i}$ so that we can avoid most of the lost sales. The quantity the remove in order to make the period feasible, is stored and will be spread on the previous periods where the item was already in production.

4 Experimentation Results

The instances that we use for our numerical experiments are derived from practical applications defined by VIF, a software company specialised in solutions for the food industry.

4.1 Parameters

Our algorithm is tuned through 9 parameters that have been tested to choose the best possible values.

- Size of the population: 200 individuals.
- Number of generations: 15000 generations, limited to 2 minutes of execution.
- Percentage of overlapping population between generations: 10%.
- Percentage of rested population: 50%.
- Number of non-improving iterations needed to reset: 200.
- Crossover ratio: 90%.
- Mutation ratio: 10%.
- Percentage of information of an individual that will be mutated: 20%.
- Ratio between the different crossovers: 60% period crossover, 20% item crossover, and 20% sequence crossover.

4.2 Experimentation

Implementation and tests of the algorithms have been done in Java. Tests have been realised on a personal computer with the following characteristics :
OS : Ubuntu 18.04.4 LTS

Processor : Intel i5-7600K @ 4.200GHz × 4
GPU : NVIDIA GeForce GTX 1070
RAM : 16 Gb
Type : 64-bit

We tested our GA on 168 instances that combine the following parameters: Number of items $\in \{20, 30, 40, 50, 75, 100, 125\}$, number of lines $\in \{1, 2, 4, 6\}$ and number of periods $\in \{15, 30\}$. The lower bound and upper bound considered are based on the results computed by CPLEX in 4 hours using a MIP formulation of the problem. We compare our results with the best lower bound (LB) obtained by CPLEX using settings presented in [22] and compute the gap achieved by our procedure with the following formula:

$$Gap = \frac{GA.cost - LB}{LB} \times 100$$

4.3 Results

We tested the GA presented in this paper with a maximum computational time of 2 minutes and compared the solutions obtained with the ones found by CPLEX in 4 hours. Note that the latter are used as a baseline and do not represent a viable option for practitioners to do its large computational time. In fact except for the smallest instances, CPLEX rarely even finds a feasible solution within 2 minutes, which already gives the GA an edge in the specific application that is targeted. In addition we observe that in 45 out of the 168 tested instances, our GA obtains a better solution in 2 minutes than the one obtained by CPLEX in 4 hours. The distribution of these 45 instances is as follows:

- 0 case for 20 items.
- 1 case for 30 items (0 for 15 periods, 1 for 30 periods).
- 7 cases for 40 items (0 for 15 periods, 7 for 30 periods).
- 10 cases for 50 items (2 for 15 periods, 8 for 30 periods).
- 9 cases for 75 items (3 for 15 periods, 6 for 30 periods).
- 11 cases for 100 items (7 for 15 periods, 4 for 30 periods).
- 7 cases for 125 items (5 for 15 periods, 2 for 30 periods).

The table 5 compares the gaps obtained by CPLEX and our GA on groups of 6 instances of same size. For each group we retain the minimal gap obtained, the maximal gap and the mean gap for all 6 instances.

This table also shows clearly the great differences that can appear between solutions found by CPLEX on 2 instances of same size (example for instance of size 50-1-30 where we have a minimal gap of 292% and a maximal gap of 14 145%) whereas our GA shows closer values (min : 999%, max : 3 453%). In general, the consistency of the results obtained by the GA is better across instances of the same size: In particular it appears that the solutions from CPLEX seem more sensitive to the number of periods that our procedure. Even if the results obtained by our GA are far behind the ones obtained by the MIP for the smallest instances, they become competitive on larger ones. For the largest instances, our heuristic consistently outperforms in 2 minutes the feasible solution computed by CPLEX in 4 hours.

These results clearly demonstrate the tendency of metaheuristics, in this case a genetic algorithm, to deal quickly with complex problems, and their usefulness in practice to tackle large industrial instances compared to MIP formulations and commercial solvers. Finally, note that the two approaches can also be used in combination, where the solution find by the GA can serve as a first feasible solution for the MIP solver, in an attempt to speed up the its convergence towards an optimal solution.

5 Conclusion

In this work, we apply the well-known genetic algorithm paradigm to develop a dedicated algorithm that is able to run quickly on large industrial instances of a complex practical production planning

Table 5. Comparison of gaps obtained by CPLEX (4 hours) and our GA (2 minutes) by group of same size instances

Instances	CPLEX Gap(%)			GA Gap(%)		
items-lines-periods	Min	Max	Mean	Min	Max	Mean
20-1-15	0.1	4.1	**1.7**	381.9	625.8	**534.0**
20-1-30	1.8	19.9	**8.9**	492.1	836.4	**731.4**
20-2-15	0.1	10.0	**2.5**	258.0	520.3	**374.2**
20-2-30	2.4	13.2	**7.0**	527.8	1108.7	**707.8**
30-1-15	1.5	12.0	**5.0**	602.4	1 203.1	**907.3**
30-1-30	0.8	1 163.3	**277.1**	1 000.8	1 911.9	**1 243.8**
30-2-15	1.8	18.8	**9.5**	513.2	1 122.8	**792.2**
30-2-30	7.4	544.4	**190.9**	379.9	1 298.2	**975.5**
40-1-15	7.6	75.1	**40.2**	685.7	1 471.5	**1 094.8**
40-1-30	533.8	2 853.7	**1 198.4**	685.0	2 245.4	**1 352.4**
40-2-15	6.0	789.9	**145.7**	391.5	1 553.2	**1 025.3**
40-2-30	322.2	7 127.8	**3 177.8**	640.1	1 768.2	**1 383.7**
50-1-15	83.0	3 573.4	**842.4**	968.8	1 778.6	**1 411.2**
50-1-30	292.0	14 145.4	**4 063.2**	998.6	3 452.6	**2 033.9**
50-2-15	4.9	1 305.0	**735.0**	1 014.6	1 598.5	**1 351.5**
50-2-30	530.9	4 277.8	**2 543.6**	1 681.4	2 312.7	**1 914.2**
75-2-15	774.5	2 811.1	**1 713.8**	2 070.9	3 459.9	**2 891.2**
75-2-30	432.6	6 829.5	**2 79.3**	2 139.9	4518.3	**3 328.3**
75-4-15	464.6	2 134.6	**1 744.8**	1 883.3	3 079.3	**2 582.1**
75-4-30	3 141.6	7 866.4	**4 537.1**	1 912.0	4 281.5	**3 387.3**
100-2-15	1 163.1	4 824.4	**2 494.2**	2 055.4	5 426.2	**3 864.5**
100-2-30	4 406.2	8 608.9	**6 371.8**	3 921.6	5 789.1	**4 449.2**
100-4-15	742.3	5 212.6	**2 569.0**	2 787.8	4 832.3	**4 034.2**
100-4-30	3 243.8	6 843.9	**5 090.9**	4 273.1	6 017.9	**5 241.9**
125-4-15	3 126.3	11 547.6	**5 756.7**	3 945.9	6 101.4	**4 935.5**
125-4-30	4 960.9	19 640.6	**8 306.3**	5 348.4	7 110.8	**6 402.2**
125-6-15	2 437.1	6 862.7	**4 358.6**	3 169.0	6 314.2	**5 301.0**
125-6-30	5 021.4	17 563.0	**7 579.5**	5 423.7	7 130.9	**6 295.3**

problem. The main contributions of this study can be partitioned in two broad categories. First, the heuristic developed is the first one that takes into account several industrial extensions of classical lot-sizing problems, such as the combination of multiple unrelated machines and sequence-dependent setup times. Second, it provides a viable alternative to commercial solvers to deal with large industrial instances that displays a robust behavior with respect to the size of the problem considered. Note that the solution obtained using our procedure may serve as a warm start for an exact method.

While the first results obtained show that such metaheuristics are a viable alternative on large instances, additional work is necessary to improve the overall performances. In particular, the method would become a lot more reliable if the solutions on small instances were comparable to the ones computed by commercial solvers. Local search methods or more advanced concepts such as hybridization or multi-population could help reduce the gap in such cases. We could also seek to find dominance properties to reduce the search space and speed up the resolution.

Another research direction to achieve this goal is to apply the procedure to a simpler problem that approximates the original one. In a recent paper [22], we developed a procedure that computes clusters of items with small switching times, which enables the algorithm to primarily focus on positioning clusters in the production sequence rather than items. This approximation greatly reduces the size of the original problem and was proven successful when used in combination with classical heuristics from the lot-sizing literature. It is likely that the GA presented in this paper would also benefit from this reduction of the problem size to converge faster to good quality solutions.

References

1. Wagner, H.M., Whitin, T.M.: Dynamic Version of the Economic Lot Size Model. Management Science **5** (1958) 89–96
2. Quadt, D., Kuhn, H.: Capacitated lot-sizing with extensions: a review. 4OR **6** (2008) 61–83
3. Karimi, B., Fatemi Ghomi, S., Wilson, J.: The capacitated lot sizing problem: a review of models and algorithms. Omega **31** (2003) 365–378
4. Absi, N., Kedad-Sidhoum, S.: The multi-item capacitated lot-sizing problem with setup times and shortage costs. European Journal of Operational Research **185** (2008) 1351–1374
5. Loparic, M., Pochet, Y., Wolsey, L.A.: The uncapacitated lot-sizing problem with sales and safety stocks. Mathematical Programming **89** (2001) 487–504
6. Absi, N., Kedad-Sidhoum, S.: The multi-item capacitated lot-sizing problem with safety stocks and demand shortage costs. Computers & Operations Research **36** (2009) 2926–2936
7. Beraldi, P., Ghiani, G., Grieco, A., Guerriero, E.: Rolling-horizon and fix-and-relax heuristics for the parallel machine lot-sizing and scheduling problem with sequence-dependent set-up costs. Computers & Operations Research **35** (2008) 3644–3656
8. Clark, A.R., Morabito, R., Toso, E.A.V.: Production setup-sequencing and lot-sizing at an animal nutrition plant through atsp subtour elimination and patching. Journal of Scheduling **13** (2010) 111–121
9. Gicquel, C., Minoux, M., Dallery, Y., Blondeau, J.M.: A tight mip formulation for the discrete lot-sizing and scheduling problem with parallel resources. In: 2009 International Conference on Computers & Industrial Engineering, IEEE (2009) 1–6
10. Guimarães, L., Klabjan, D., Almada-Lobo, B.: Modeling lotsizing and scheduling problems with sequence dependent setups. European Journal of Operational Research **239** (2014) 644–662
11. Özdamar, L., Bozyel, M.A.: The capacitated lot sizing problem with overtime decisions and setup times. IIE Transactions **32** (2000) 1043–1057
12. Hung, Y.F., Shih, C.C., Chen, C.P.: Evolutionary algorithms for production planning problems with setup decisions. The Journal of the Operational Research Society **50** (1999) 857
13. Hung, Y.F., Chen, C.P., Shih, C.C., Hung, M.H.: Using tabu search with ranking candidate list to solve production planning problems with setups. Computers & Industrial Engineering **45** (2003) 615–634
14. Özdamar, L., Birbil, Ş.İ.: Hybrid heuristics for the capacitated lot sizing and loading problem with setup times and overtime decisions. European Journal of Operational Research **110** (1998) 525–547
15. Özdamar, L., Bilbil, Ş.İ., Portmann, M.C.: Technical note: New results for the capacitated lot sizing problem with overtime decisions and setup times. Production Planning & Control **13** (2002) 2–10
16. Fleischmann, B., Meyr, H.: The general lotsizing and scheduling problem. Operations-Research-Spektrum **19** (1997) 11–21
17. Meyr, H.: Simultaneous lotsizing and scheduling by combining local search with dual reoptimization. European Journal of Operational Research **120** (2000) 311–326

18. Laguna, M.: A heuristic for production scheduling and inventory control in the presence of sequence-dependent setup times. IIE Transactions **31** (1999) 125–134
19. Sikora, R.: A genetic algorithm for integrating lot-sizing and sequencing in scheduling a capacitated flow line. Computers & Industrial Engineering **30** (1996) 969–981
20. Larroche, F., Bellenguez-Morineau, O., Massonnet, G.: Approche de résolution d'un problème industriel de lot-sizing avec réglages dépendant de la séquence. In: ROADEF 2020 : 21ème Congrès Annuel de la Société Française de Recherche Opérationnelle et d'Aide à la Décision, Montpellier, France (2020)
21. Florian, M., Lenstra, J.K., Rinnooy Kan, A.H.G.: Deterministic Production Planning: Algorithms and Complexity. Management Science **26** (1980) 669–679
22. Larroche, F., Bellenguez-Morineau, O., Massonnet, G.: Clustering-based solution approach for a capacitated lot-sizing problem on parallel machines with sequence-dependent setups. To appear in International Journal of Production Research (2020)

A Physarum-inspired approach for influence maximization

Álvaro O. López-García, Gustavo Rodríguez-Gómez, Aurelio López-López

Instituto Nacional de Astrofísica, Óptica y Electrónica
Luis Enrique Erro No. 1, Sta. María Tonantzintla, 72840, Puebla, México
{alvarolopez, grodrig, allopez}@inaoep.mx

Abstract. The influence maximization problem (IM) is an open problem in graph theory, and also is identified as a NP-hard, so there has been a lot of developments in order to solve or approximate a solution. In this paper, we present an approach for pointing to a solution of the IM problem, by leveraging k-shell decomposition analysis, and combining it with a *Physarum*-inspired model. Additionally, this procedure was tested on five data-sets both synthetic and real, showing encouraging results.

1 Introduction

Social media is a global phenomenon that changed our lives forever, altering the way we see the world and also is a phenomenon that has an important influence in our opinions, ideas, and decisions. They are an important and fundamental way of expressing ourselves to the world. For 2020, it is estimated that 3.8 billion of users worldwide are using actively social media, that is the 84% of overall web population [1].

Studying social media is an important field of interest to researchers of various fields such as sociology, psychology, mathematics and computer science. The study of social media is important because this is an information goldmine for advertisers and reaches a very wide audience, providing additionally a lot of personal user data.

One successful strategy adopted for dissemination of information and products are the so-called *influencers*. They are Internet personalities, quite popular with a lot of followers, and consequently called "opinion leaders". The concept of influencer is created around the idea of influential marketing, which identifies people with a lot of influence over potential buyers, and all the marketing is constructed around these influencers [2].

From a computer science viewpoint, the identification of influencers is an instance of a bigger problem, called influence maximization on graphs (IM). This problem is an open problem in graph theory [7], and is identified as a NP-hard problem (a problem whose solution can not be computed with a polynomial-time algorithm), so there have been several proposals in this area, with new algorithms and heuristics for approximating a solution [3]. Note that those proposals have limitations, either with the size of the instances that they can solve or with the graph topologies.

One of the newest and interesting approaches is that based on a biological being and its behavior in the natural habitat. This organism is *Physarum Polycephalum*, and scientists are interested on it because of its intelligent behavior, lack of a central brain, and also given that can solve a maze to find food and nutrients [4]. Also, it can optimize energy consumption when transfering nutrients into its body, so it can solve problems like the optimal traffic network problem [5].

In this paper, we present an approach for solving the IM problem, by leveraging k-shell decomposition analysis, and combining it with a *Physarum Polycephalum* model. The $k-$shell decomposition is an useful technique because it performs an initial selection of nodes with a high degree (and potentially high propagation ability), and then the *Physarum Polycephalum* model is employed for evaluating nodes and determines which nodes are the most influential.

The contributions of this paper are: a new procedure for obtaining the influential nodes in a social network graph by using $k-$shell decomposition in conjunction with a Physarum model. Additionally, this procedure is tested on both synthetic and real data-sets. This is of importance as a part of an approach for solving the IM problem using a bio-inspired algorithm.

This paper is organized as follows. Section 2 shows some related work for both the IM problem and some developments with *Physarum Polycephalum*. In Section 3, we show the theoretical basis of the IM problem, $k-$shell decomposition, the Physarum model, and the employed degree index. The proposed method is shown and expanded on Section 4. Section 5 details the data-sets, experiments and results of our approach. Finally, in Section 6 we provide our conclusions and future work.

2 Related Work

When first computer networks were created, their high potential for communicating with people was perceived, since distance was no longer a limitation for communicating with friends and relatives overseas. In particular, the massification of the Internet in 1995 [6] catalyzed the development of new online platforms for communicating with friends, one of them was the social networks.

A social network service is an Internet platform, employed for creating networks or relationships among people with similar interests. In addition, such platforms allow to create friend lists and meet new people. Examples of social networks are Facebook, Twitter, QQ, or TikTok. A social network can be modeled as a graph, since this is based on establishing relations among people, and they, at the same time, can also have their own friends. So, we can build a huge graph of people connected by their relationships. Each node is a person, and each vertex represents a given relation (e.g. friendship, or con-generic) between two. Open problems in social networks [7] are: community detection, recommendation systems, trust prediction, opinion mining, and influence maximization.

Influence maximization (IM) was formulated by Kempe et al. [3], as a combinatory optimization problem (i.e., we need to search a near object from a finite object set), also proposing a greedy algorithm for solving it. This algorithm is initialized with an empty seed set, and then searches for nodes that maximizes influence. The downside of this algorithm is that on big graphs, computing time grows fast, because of the number of calculations needed. Kempe et al. identified two diffusion models: the Independent Cascade (IC) and Lineal Threshold (LT) [8].

Leskovec et al. [9] proposed a new algorithm called Cost Effective Lazy Forward (CELF), and it promised faster speed (marginaly) on solving the problem, compared to the traditional greedy algorithm. This algorithm is based on the modularity function of the diffusion model, so it can select nodes faster and more precisely. Also, the algorithm can prevent unnecessary calculations for diffusion, so is faster than the traditional greedy algorithm [10].

CELF also has its own limitations, so Goyal et al. [11] proposed an improvement on this algorithm, called CELF++, showing an increase of $35-55\%$ of performance, compared to CELF.

On other line, scientists have been trying to solve IM from an heuristic approach with algorithms such as Local Directed Acyclic Graph (LDAG), proposed by W. Chen et al.[12] This algorithm only works with LT diffusion model, but the authors argue that an improvement in time is achieved, decreasing it from hours to seconds (or from days to minutes).

Other important heuristic is PageRank, developed by Larry Page for the Google search engine [13], adapted for the IM problem by Li Q et al. [14], calling it Group-PageRank. This algorithm only works on graphs with IC diffusion model. The original idea of PageRank is that each node of the graph is given a score, according to a probability of being activated by a user on the web [10].

The *Physarum Polycephalum* approach is a recent development in bio-inspired computation. Adamatzky et al. [16] proposed one of the earliest problems solved by this model, which was facing the optimization of traffic network. But also *Physarum Polycephalum* model has been explored for solving other problems such as solving mazes [17], the Steiner tree problem in networks [18], the graph coloring problem [19], among other graph-related tasks [20].

As part of our bio-inspired optimization Physarum-based approach, there are some related works, such as that proposed by Gao C. et al. [21], who developed a new method for obtaining the centrality degree of a node, which is important for establishing how probable a node is going to be influential in a social network. This new centrality degree is called Physarum centrality, based on properties of *Physarum Polycephalum*, and also is supported by $k-$shell decomposition for identifying nodes. This algorithm tackles both weighted and unweighted networks.

3 Theoretical framework

A social network is a structure formed with entities and a group of $2-$way links among them (for example, friendship or family relationships). A social network can be modeled as a graph G both directed or undirected, and weighted or unweighted.

3.1 Influence Maximization

An open and important problem in graph theory is influence maximization (IM) [7]. This problem has been studied and showed to be NP-Hard [10], indicating that is a problem that does not have a verification algorithm in polynomial time.

A social network is studied and represented as a graph $G = (V, E)$, where V is the set of nodes in G (e.g. users), E is the set of edges in G (the relationships between the users). The aim is to find the set of users with the maximum influence in G [10].

IM is an optimization problem, which consists in maximizing the spread of information or influence in a social network graph. Formally speaking, according to [10]:

Definition 1. *Given a social graph $G = (V, E)$ and a user set $S \subseteq V$, a diffusion model M captures the stochastic process for S spreading information on G. Influence spreading of S, denoted as $\sigma_{G,M}(S)$, is the number of expected users influenced by S.*

Definition 2. *Given a social graph $G = (V, E)$, a diffusion model M and a positive integer k, influence maximization selects a set S^* of k users of V as a seed set for maximizing influence diffusion, such as $S^* = \arg\max_{S \subseteq V \wedge |S| \leq k} \sigma_{G,M}(S)$.*

For the IM problem there are various diffusion models, but the most common are Independent Cascade (IC) and Lineal Threshold (LT). The aim of these models is to associate each user in G a status (i.e active or inactive) and the conditions of activating or infecting them. Independent Cascade (IC) states that there is a probability of infection for each edge, namely P_{ij}, where P is the probability of i infecting j. Once j is infected, it can infect neighbours on the next step, according to the probability assigned to next edge. Linear Threshold (LT) is different, because each node is infected if neighbour nodes are infected by reaching a threshold, according to their weights [22].

3.2 K-shell Decomposition

$K-$shell decomposition is a technique for decomposing and studying the structure of large graphs. This method is also noted for showing the importance of certain nodes in regards of their hierarchies.

The following concepts are basic for this decomposition, as given by [23].

Definition 3. *The k–Core of a graph G is the maximal subgraph of G having minimum degree at least k.*

Definition 4. *The k–Shell of a graph G is the set of all nodes belonging to the k–Core of G but not to the (k+1)–Core.*

The $k-$shell index of a node is denoted as K_s.

Kitsak et al. [24] proposed the use of $k-$shell decomposition as a means for identifying node spreaders in a graph network. So, the higher the $k-$shell index of a node, the more it can spread information in the network. The $k-$shell decomposition works better on static networks, where topologies do not change over time [23].

3.3 Physarum model

Our model for *Physarum Polycephalum* is based on the works of Z. Cai et al. [5] and Gao C. [21].

This model simulates the foraging behavior of *Physarum Polycephalum.* Its body is a single cell made up of interconnected tubes forming networks, that can stretch from centimeters to meters, and can store and recover information when searching for food. [15]. On laboratory experimental setups, the model consists of a Petri dish, a map (usually made of agar), external food sources (usually oat flakes) and a live Physarum [16].

Physarum Polycephalum consists of the following components [5]:

1. **Plasmodium and Myxamoebas**: The plasmodium is the moving part of the organism, and the tentacle-shaped myxamoebas are the deformed part of the plasmodium. They are used for foraging and consuming food and nutrients, and expand and contract accordingly.
2. **Nucleus**: The nucleus is the central and critical part of the organism, which moves and feeds around it.

3. **Nutrients**: They are the source of energy, and come from external food sources.

These all parts work collaboratively in order to solve problems, and the solving process includes two stages: food searching, and feeding [5]. The first stage is when the myxamoebas start growing around the nucleus in order to find external food sources. The second stage is when the myxamoebas contract in order to transport all the found nutrients into the Physarum's body. In terms of optimization, these two stages correspond to exploration and exploitation of the search space.

Multiple myxamoebas m grow around the nucleus, by expanding in multiple directions [5]. This growth is constrained by the topology of its environment, and in our case the topology of the social network graph, i.e. its adjacency matrix, which is a time-varying structure, denoted as follows:

$$\mu = [\mu_{ij}(t)]_{n\times n} \tag{1}$$

where $\mu_{ij} = 1$ when there is a direct edge from node i to j, or $\mu_{ij} = 0$ otherwise.

Another important part of our model is the nutrient concentration matrix on the edges, which is also a time-varying structure, and defined as [5]:

$$\tau(t) = [\tau_{ij}(t)]_{n\times n} \tag{2}$$

There are two operations related to nutrient consumption, namely the enhancing operation ($\varDelta > 0$) and the decreasing operation ($\sigma > 0$). The first one is used to simulate the nutrient transportation through the Physarum body, and the latter is intended to simulate the nutrient consumption by other life activities [5]. The nutrient concentration on each edge is updated at time t and m number of myxamoebas, with the following formula:

$$\tau_{ij}(t) = \tau_{ij}(t-1) + \sum_{k=1}^{m} \mu_{ij}^{k}(t)\varDelta_{ij}^{k}(t) - \sum_{k=1}^{m} \sigma_{ij}^{k} \tag{3}$$

3.4 Node selection by propagation capability

One detail to observe is that node selection is correlated with identifying the importance of each node or edges on the social network graph. There are various node indices for doing this selection, namely the degree of a node, the importance degree index, the closeness index, the betweenness degree index, and the redundancy rate index, among others [22].

In our research, we employ the betweenness index, since this considers the myxamoebas passing through node i and reaching outer nodes, and that is critical for information spreading. Betweenness measures the extent to which a node lies in the path between others, so it can measure the influence a node has over the spread of information through the network [28]. The betweenness index takes into account the number of myxamoebas passing through node i, so the more of them go by, the more important node i is.

Let λ_{jk} be the number of myxamoebas from node j to k passing through a node i. The betweenness index Bt_i of a node i can be calculated as [26]:

$$Bt_i = \sum_{j=1}^{n}\sum_{k=1}^{n} \lambda_{jk} \tag{4}$$

In a social network with n nodes, at most $n-1$ neighbor nodes can connect to a node i, so the betweenness degree index of a node i is calculated by:

$$\overline{Bt_i} = \frac{Bt_i}{\sum_{j=1}^{n} Bt_j} \tag{5}$$

In consequence, the total betweenness index of all the myxamoebas can be calculated by:

$$Bt_i^{m} = \sum_{k=1}^{m} \overline{Bt_i} \tag{6}$$

4 Proposed Method

Our proposed method follows most of the steps proposed by Gao C. et al.[21], in which they apply a Physarum model based on a Poisson equation, and after that, they use $k-$shell decomposition for calculating the called Physarum Centrality. However, we employ the method used by Z. Cai et al.[5], in which the most important part of the model are the growth of the myxamoebas and the nutrient consumption by the organism.

The first step is to initialize the procedure, having the social network graph $G = (N, E)$ as the only input parameter. So the adjacency matrix μ is defined in this first step (see expression (1)). Then, thereafter there is a $n \times n$ matrix, with n being the total number of nodes in the social network graph G.

The nutrient matrix concentration τ at starting time is a zero matrix of size $n \times n$ (with n being the total number of nodes in the social network graph G), since at the start time there is no nutrient flowing through the body of the organism.

The increment ($\varDelta$) and decrement (σ) parameters are also initialized, with the constraints of $\varDelta > 0$, $\sigma > 0$, and following the rule of thumb $\varDelta > \sigma$, given that the nutrients consumed should be higher than the nutrients consumed for growing and foraging.

Once all the parameters are initialized, we proceed to compute the $k-$shell decomposition on the social graph G, for obtaining the K_s value for each node. This process basically consists on the following [25]:

```
def kShell(G):
    h = G.copy()
    it = 1
    tmp = []
    buckets = []
    while (1):
        flag = kShell_check(h, it)
        if (flag == 0):
            it += 1
            buckets.append(tmp)
            tmp = []
        if (flag == 1):
            node_set = kShell_find_nodes(h, it)
            for each in node_set:
                h.remove_node(each)
                tmp.append(each)
        if (h.number_of_nodes() == 0):
            buckets.append(tmp)
            break
    return buckets
```

Once each node has its K_s value, the nodes with higher index value are selected to function as nucleus and grow myxamoebas, so the next step is grow the myxamoebas on the whole social network graph G. Each nucleus can grow m number of myxamoebas on it, depending on the topology of the social network graph G. This m number of myxamoebas for each nucleus can be determined as follows:

$$m = \begin{cases} K_s, & \text{if } K_s < 5 \\ \left\lceil \frac{K_s}{2} \right\rceil, & \text{otherwise} \end{cases} \tag{7}$$

where K_s is the $k-$shell decomposition index value of the selected node acting as the nucleus. The idea is to explore as many connecting paths as possible when the nodes are not highly connected but administer resources when the node is heavily connected.

For growing the myxamoebas, a recursive function is applied, with the following parameters: the adjacency list μ, the node acting as nucleus, the number of m myxamoebas, and an array used for keeping track of the visited nodes. This function is defined as follows:

```
def growMyxo(adjList, node, totalMyx, visited=[]):
    neighbours = getNeighbours(node, adjList)
    if (totalMyx>=len(neighbours)):
        selected_nodes = neighbours
    else:
        selected_nodes = random.sample(neighbours, totalMyx)
    for i in range(0,len(selected_nodes)):
        if (not selected_nodes[i] in visited):
            visited.append(selected_nodes[i])
            growMyxo(adjList, selected_nodes[i], totalMyx,
             visited)
    return visited
```

This function returns all the visited nodes by the myxamoeba while searching for food, and the nutrient matrix is updated according to the expression (3).

The feeding stage of the organism consists of a loop iterating over the m myxamoebas while updating the nutrient concentration matrix with (3). The more myxamoebas pass through a node, the higher concentration of nutrients this will have, and that will be reflected on the nutrient matrix τ.

After the feeding stage, the betweenness degree index of each node of the myxamoebas grown on the selected nucleus will be calculated using expression (5).

For output of the procedure, the total betweenness degree index for all the nodes on the social network graph G is calculated by formula (6).

5 Experiments and Results

Five experiments were done in order to validate the proposed approach, using two synthetic examples and three real data-sets from social network graphs.

For running the experiments, the number of myxamoebas assigned to each social graph was defined by (7). For each data-set, the particular defined value is detailed later on. The total number of iterations for the feeding stage was set to 10. Since each myxamoeba grows differently in each run, the method was run 10 times on each data-set, and then average and standard deviation were calculated.

5.1 Synthetic data-sets

The first synthetic graph is based on the graph used in [21] and this has 15 nodes and 21 edges, with a density value of 0.2. The reported maximum degree of a node is 6, shown in Fig. 2. For this graph, the m value was set to 3.

To illustrate the process of growing myxamoebas to explore the first graph, Figure 1 includes two examples of sub-graphs obtained when taking two different nodes as nucleus.

The results of running our method on the first synthetic graph are shown on Table 1. For comparison, we report also the Physarum centrality index (Ck_p) as described in [21]. The Top-4 nodes selected by the Physarum centrality are the same as those selected by our method, showing that nodes 7, 11, 12 and 5 have the potential of spreading the information the most in the social network graph. Also the table shows that the nodes with the least propagation ability are again the same (nodes 0, 1, 3 and 2).

The second synthetic graph is based on the graph employed in [8], having 8 nodes and 20 edges, with a density value of 0.444. The reported maximum degree of a node is 8, as shown in Fig. 3. For this second synthetic data-set, the value of m was set to 2.

The results of running our method on the second graph are shown in Table 2. As we can notice, nodes 0 and 1 are reported as those having the best capability to spread information, and they are also the nodes selected by the CELF algorithm in [8]. The topology of this graph shows that nodes 0 and 1 are the best spreaders, since this is pretty clear because they are in the center and have the most outer connected nodes.

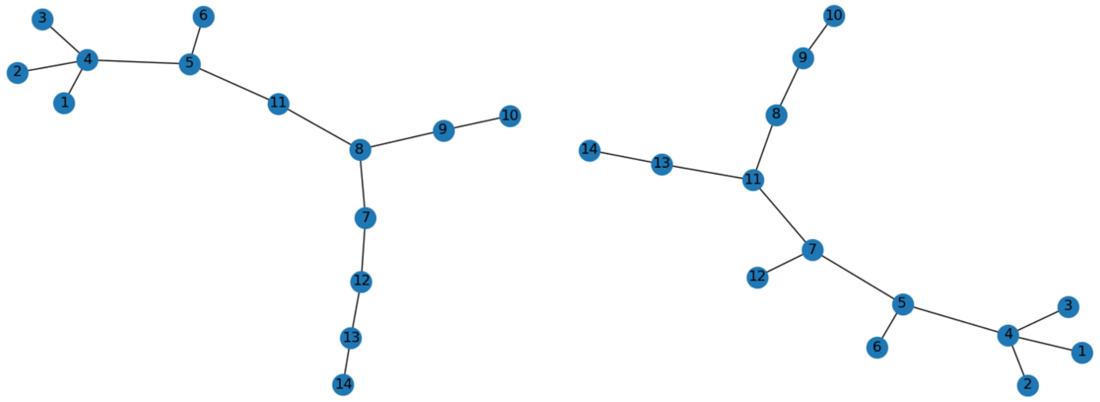


Fig. 1. Examples of growing myxamoebas on graph of Fig. 2. The graph on the left had node 11 as nucleus, and the graph on the right took node 5 as nucleus.

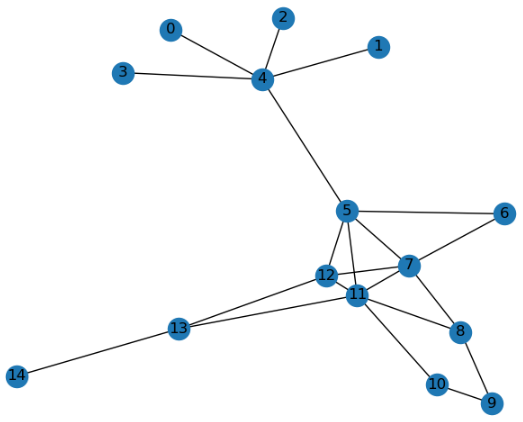


Fig. 2. Network with 15 nodes and 21 edges. After k-shell decomposition ($K_s = 3$), nodes 5, 7, 11 and 12 are selected

Node	B.d.i. (avg.)	Std deviation	Ck_p
5	0.350	0.018	0.265
7	0.350	0.018	0.113
11	0.350	0.018	0.182
12	0.350	0.018	0.093
8	0.347	0.018	0.052
9	0.347	0.018	0.031
10	0.340	0.019	0.038
13	0.320	0.017	0.064
14	0.310	0.023	0.011
6	0.241	0.029	0.024
4	0.196	0.029	0.084
0	0.177	0.026	0.011
1	0.133	0.026	0.011
3	0.114	0.024	0.011
2	0.077	0.035	0.011

Table 1. Selected nodes for graph shown in Fig. 2 ordered by betweenness degree index (B.d.i.)

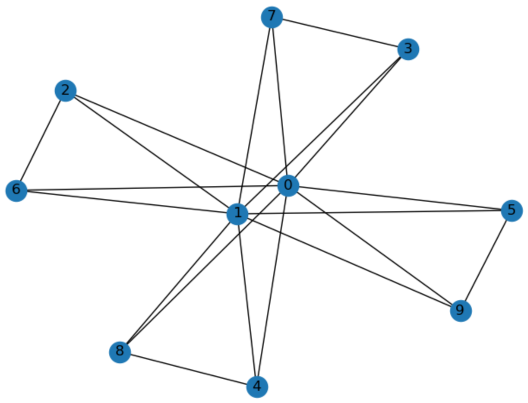


Fig. 3. Network with 8 nodes and 20 edges. During the k-shell decomposition, all nodes (0, 1, 2, 3, 4, 5, 6, 7 and 8) are selected with $K_s = 2$

Node	B.d.i. (avg.)	Std deviation
0	1.086	0.018
1	1.086	0.018
4	1.024	0.026
8	1.024	0.026
2	0.986	0.038
6	0.971	0.024
3	0.933	0.025
7	0.986	0.038
5	0.971	0.024
9	0.933	0.025

Table 2. Selected nodes for graph shown in Fig. 3 sorted by degree index (B.d.i.)

5.2 Real Data-sets

The first real data-set is the Zachary's karate club network of 1977, containing the social ties between the members of a university karate club [27]. This graph consists of 34 nodes, 78 edges and a density value of 0.139037. The reported maximum degree of a node is 17. This graph is shown in Fig. 4. Using k-shell decomposition nodes 1, 2, 3, 4, 8, 9, 14 31, 33 and 34 are selected with $K_s = 3$. In consequence, the value for m was set to 3.

The Top-10 results for the karate club network are shown in Table 3, and these are similar to those reported in [21]. They show that nodes 1, 34, 3, 33 and 14 (in bold in the table) have the greatest Physarum centrality index, while our method shows the same nodes (except by node 14) in addition of nodes 2 and 4, are those with highest betweenness degree index.

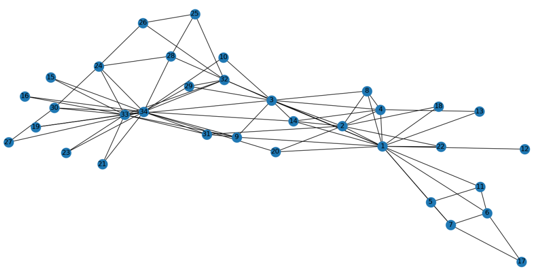


Fig. 4. Zachary's karate club social network graph.

Node	B.d.i. (avg.)	Std deviation
34	0.403	0.012
33	0.403	0.015
2	0.403	0.012
3	0.403	0.012
4	0.403	0.015
1	0.403	0.006
28	0.389	0.029
24	0.389	0.030
26	0.374	0.017
32	0.374	0.034

Table 3. Top-10 nodes for the karate club graph, sorted by betweenness degree index (B.d.i.)

The second real data-set is the bottlenose dolphins social network, containing a list of links, where each link is a frequent association between dolphins [27]. This graph consists of 62 nodes, 159 edges and a density value of 0.0840825. The reported maximum degree of a node is 12. This graph is shown in Fig 5. After k-shell decomposition, thirty six nodes were selected with $K_s = 3$ (i.e. nodes 1, 2, 7, 8, 9, 10, 11, 14, 15, 16, 17, 18, 19, 20, 21, 22, 25, 29, 30, 31, 34, 37, 38, 39, 41, 42, 43, 44, 46, 48, 51, 52, 53, 55, 58 and 60), so also for this graph the value for m was set to 3.

This data-set, the dolphin social network graph, showed an interesting behavior. The results of running our method on this graph are summarized in Table 4. After growing the myxamoebas on all the thirty six nodes selected after the k-shell decomposition, acting as nucleus and performing the feeding stage, the list of possible influential nodes narrowed down to nodes 15, 21 and 46. An implementation of the CELF algorithm applied on this data-set, selected nodes 15 and 46.

The third real dataset is the public figures network, gathered from Facebook [27]. This graph consists of 11.6K nodes, 67K edges and a density value of 0.00100253. The reported maximum degree of a node is 326. Using k-shell decomposition, 170 nodes were selected with $K_s = 41$. In consequence, the value for m was set to 22.

The implementation of the CELF algorithm applied to this data-set, selected 191 nodes. Our method selected 2963 nodes as top ranked, with a recall of 122 (64%) of those selected by CELF. The range of standard deviation is between 0.006 and 0.000004. Among the 10 runs of our proposed algorithm, the ninth showed the best behavior and the second showed the worst. It is important to note that because of the non-deterministic nature of the approach, this might vary over time.

5.3 Discussion

The experiments on synthetic data-sets allow to verify that the approach was working adequately. The further validation on real data-sets led to similar results as previously reported or as identified by a previous algorithm. In all the data-sets, the standard deviation was relatively small, indicating that most of the executions tend to converge to the same set of nodes.

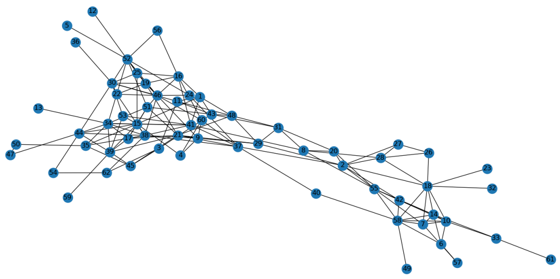


Fig. 5. Bottlenose dolphins social network graph.

Node	B.d.i. (avg.)	Std deviation
15	0,719	0,001
21	0,719	0,001
46	0,719	0,001
38	0,718	0,004
30	0,714	0,003
34	0,713	0,007
43	0,712	0,006
48	0,712	0,008
41	0,711	0,008
39	0,711	0,010

Table 4. Top-10 nodes for the dolphin graph, sorted by betweenness degree index (B.d.i.)

The approach starts with a k-shell decomposition, which has a complexity of $O(n)$, with n the number of vertices of the social media graph. Overall the proposed approach has a complexity of $O(\mu n^3)$, where μ is the total number of grown myxamoebas, and n is again the number of vertices of the social media graph. The adjacency matrix is of size $n \times n$, i.e. a square matrix representing the edges between all the n vertices.

Also, our algorithm has a straightforward implementation for the experiments, so we employed big matrices for doing all the computation, and this constrains the size of the data-sets. One improvement can be handling graphs as linked lists, which would demand less memory and work with larger data-sets.

6 Conclusion and future work

In this paper, a method for obtaining the influential nodes in a social network graph is proposed, by using k-shell decomposition and by simulating the behavior of a *Physarum Polycephalum*. The growth of its myxamoebas and its nutrient consumption are illustrated and used in our method, in order to filter which nodes are the most influential. After simulating the food searching and feeding stage, the method will help determine which nodes are the most influential in the social network graph. For this, the betweenness degree index and the myxamoebas that pass through a particular node are considered. The experiments showed that the approach was working well and reached similar results as those reported in earlier works.

The proposed method has to be further tested on larger graphs, but since IM is a NP-hard problem, we might need to recur to improved computing infrastructure, in order to operate on such graphs. Also a further complexity analysis has to be done, in order to have a better idea of the efficiency and applicability of the proposed method.

Acknowledgements

First author thanks the support of Conacyt, México through scholarship number 1008092. The other authors had partial support of SNI.

References

1. We are Social, Hootsuite. (2020, January). Digital in 2020. Retrieved from https://wearesocial.com/digital-2020.
2. Agencia de Publicidad Endor. (2020, March 5). ¿Qué es un Influencer y cuál es su función? Retrieved July 1, 2020, from https://www.grupoendor.com/influencer-funcion/
3. Kempe, D., Kleinberg, J., Tardos, E. (2003). Maximizing the Spread of Influence through a Social Network. ACM, 1. Retrieved from https://www.cs.cornell.edu/home/kleinber/kdd03-inf.pdf.

4. Tang, C.-B., Zhang, Y., Wang, L., Zhang, Z. (2019). What can AI learn from bionic algorithms? Physics of Life Reviews, 29, 41–43. https://doi.org/10.1016/j.plrev.2019.01.006.
5. Cai, Z., Xiong, Z., Wan, K., Xu, Y., Xu, F. (2020). A Node Selecting Approach for Traffic Network Based on Artificial Slime Mold. IEEE Access, 8, 8436–8448. https://doi.org/10.1109/access.2020.2964002
6. Schafer, V., Thierry, B. G. (2018). The 90s as a turning decade for Internet and the Web. Internet Histories, 2(3–4), 225–229. https://doi.org/10.1080/24701475.2018.1521060
7. Problems in Graph Theory and Combinatorics. (n.d.). Retrieved July 1, 2020, from https://faculty.math.illinois.edu/
8. Kingi, H. (2018, September 7). Influence Maximization in Python -Greedy vs CELF. Retrieved June 29, 2020, from https://hautahi.com/im_greedycelf.
9. Leskovec, Jure Krause, Andreas Guestrin, Carlos Faloutsos, Christos Vanbriesen, Jeanne Glance, Natalie. (2007). Cost-effective outbreak detection in networks. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 420-429. https://doi.org/10.1145/1281192.1281239.
10. Li, Y., Fan, J., Wang, Y., Tan, K.L. (2018). Influence Maximization on Social Graphs: A Survey. IEEE Transactions on Knowledge and Data Engineering, 30(10), 1852–1872. https://doi.org/10.1109/tkde.2018.2807843
11. Goyal, A., Lu, W., Lakshmanan, L. V. S. (2011). CELF++. Proceedings of the 20th International Conference Companion on World Wide Web -WWW '11,1. https://doi.org/10.1145/1963192.1963217.
12. W. Chen, Y. Yuan and L. Zhang, Scalable Influence Maximization in Social Networks under the Linear Threshold Model, 2010 IEEE International Conference on Data Mining, Sydney, NSW, 2010, pp. 88-97, doi: 10.1109/ICDM.2010.118.
13. Page, L., Brin, S., Motwani, R., Winograd, T. (1998). The PageRank Citation Ranking: Bringing Order to the Web. Stanford University Info Lab, 1–17. http://ilpubs.stanford.edu:8090/422/
14. Liu, Q., Xiang, B., Chen, E., Xiong, H., Tang, F., Yu, J. X. (2014). Influence Maximization over Large Scale Social Networks. Proceedings of the 23rd ACM International Conference on Information and Knowledge Management CIKM '14, 171–180. https://doi.org/10.1145/2661829.2662009.
15. Staff, S. X. (2021, February 23). Researchers find a single-celled slime mold with no nervous system that remembers food locations. Phys.Org. https://phys.org/news/2021-02-single-celled-slime-mold-nervous-food.html
16. Adamatzky, A., Jones, J. (2010). Road Planning with Slime Mold: If Physarum Built Motorways it would Route M6/M74 through Newcastle. International Journal of Bifurcation and Chaos, 20(10), 3065–3084. https://doi.org/10.1142/s0218127410027568
17. Tero, A., Kobayashi, R., Nakagaki, T. (2006). Physarum solver: A biologically inspired method of road-network navigation. Physica A: Statistical Mechanics and Its Applications, 363(1), 115–119. https://doi.org/10.1016/j.physa.2006.01.053
18. Liang Liu, Yuning Song, Haiyang Zhang, Huadong Ma, Vasilakos, A. V. (2015). Physarum Optimization: A Biology-Inspired Algorithm for the Steiner Tree Problem in Networks. IEEE Transactions on Computers, 64(3), 818–831. https://doi.org/10.1109/tc.2013.229
19. Lv, L., Gao, C., Chen, J., Luo, L., Zhang, Z. (2019). Physarum-Based Ant Colony Optimization for Graph Coloring Problem. LNCS, pp.210–219. https://doi.org/10.1007/978-3-030-26369-0_20
20. Zhang, X., Mahadevan, S., Deng, Y. (2015). Physarum-Inspired Applications in Graph-Optimization Problems. Parallel Processing Letters, 25(01), 1540005. https://doi.org/10.1142/s0129626415400058
21. Gao, C., Lan, X., Zhang, X., Deng, Y. (2013). A Bio-Inspired Methodology of Identifying Influential Nodes in Complex Networks. PLoS ONE, 8(6), e66732. https://doi.org/10.1371/journal.pone.0066732.
22. Shakarian, P., Bhatnagar, A., Aleali, A., Shaabani, E., Guo, R. (2015). The Independent Cascade and Linear Threshold Models. SpringerBriefs in Computer Science, 35–48. https://doi.org/10.1007/978-3-319-23105-1_4
23. Bickle, Allan, The k-Cores of a Graph (2010). Dissertations. 505. https://scholarworks.wmich.edu/dissertations/505
24. Kitsak, M., Gallos, L. K., Havlin, S., Liljeros, F., Muchnik, L., Stanley, H. E., Makse, H. A. (2010). Identification of influential spreaders in complex networks. Nature Physics, 6(11), 888–893. https://doi.org/10.1038/nphys1746
25. GeeksforGeeks. (2020, October 1). K-shell decomposition on Social Networks. https://www.geeksforgeeks.org/k-shell-decomposition-on-social-networks/
26. Bauer, B., Jordán, F., Podani, J. (2010). Node centrality indices in food webs: Rank orders versus distributions. Ecological Complexity, 7(4), 471–477. https://doi.org/10.1016/j.ecocom.2009.11.006
27. Rossi, A. A. N. R. K. (2015). Social Networks — Network Data Repository [Data-sets]. http://networkrepository.com/
28. Newman, M. J. (2005). A measure of betweenness centrality based on random walks. Social Networks, 27(1), 39–54. https://doi.org/10.1016/j.socnet.2004.11.009

Visualizing Crossover Rate Influence in Differential Evolution with Expected Fitness Improvement

V. Stanovov[1], Sh. Akhmedova[2], and E. Semenkin[1]

[1]Siberian Federal University, 79 Svobodny pr., 660041, Krasnoyarsk, Russian Federation
vladimirstanovov@yandex.ru, eugenesemenkin@yandex.ru
[2]Reshetnev Siberian State University of Science and Technology, 31 Krasnoyarsky rabochy pr. 660037, Krasnoyarsk, Russian Federation
shahnaz@inbox.ru

1 Introduction

The area of heuristic optimization methods, which includes evolutionary algorithms and biology-inspired methods, is currently under rapid development of due to the high efficiency of such approaches in various domains and availability of large computational resources. However, the proposal of new ideas is always strongly connected to the level of understanding of these algorithms' inner functioning principles, which is often not as high as desired. Because of this, the development of methods which allow better understanding of algorithms behaviour, for example, the influence of parameter values, would promote intuition of researches and lead to new ideas and directions of investigation.

One of the most popular evolutionary optimization techniques today is the differential evolution (DE) algorithm, originally proposed in [9]. The DE has shown its superior properties compared to other approaches in numerous competitions and found a large variety of real-life applications, which makes this method an interesting research topic. However, one of the disadvantages of DE is its high sensitivity to parameter values, such as scaling factor F and crossover rate Cr [4]. Better understanding of these parameters' influence is one of the most important directions of studies about DE.

In this paper the expected fitness improvement (EFI) metric is proposed to visualize the parameter search space of crossover rates of modern DE modification, NL-SHADE-RSP algorithm. The expected fitness improvement shows the possible improvement that could be achieved with different Cr values, highlighting the areas of interest at different stages of search process. Based on the EFI heatmap profiles, the conclusions about crossover rate importance are made for different benchmark scenarios, such as biased, shifted and rotated goal functions, taken from the Congress on Evolutionary Computation (CEC) 2021 competition for single-objective optimization. The performed experiments shows that efficient control strategies could be applied for NL-SHADE-RSP crossover rate change.

The rest of the paper is organized as follows: section 2 provides the related work and describes DE basics, section 3 contains the description of EFI metric calculation method, section 4 contains the experimental setup and results, as well as their discussion, and section 5 concludes the paper, outlining the possible directions of further studies.

2 Related Work: Differential Evolution

Differential Evolution or DE is the population based evolutionary algorithm for solving real-valued optimization problems firstly introduced by R. Storn and K. Price in [9]. This algorithm became one of the most popular among researchers due to its simplicity (it is easy to emplement and has just three parameters, which will be discussed later) and high efficiency [7]. The differential evolution algorithm is based on the idea that to find the optimal solution only the difference vectors between candidate solutions should be used.

The basic DE approach has two main phases: the initialization and search conducted by mutation, crossover and selection operators. During the initialization a set (or population) of candidate solutions (also called individuals) $x_i = (x_{i,1}, x_{i,2}, ..., x_{i,D})$, $i = 1, ..., NP$, $j = 1, ..., D$, is randomly generated in the search space:

$$S = \left\{x_i \in R^D | x_i = (x_{i,1}, x_{i,2}, ..., x_{i,D}) : x_{i,j} \in [x_{lb,j}, x_{ub,j}]\right\} \tag{1}$$

using the uniform distribution with D being the dimensionality of that space and NP or population size is the first parameter of the DE algorithm.

After initialization individuals iteratively change their position in the search space with aim to find the best solution (optimum). For this purpose three operators are used: mutation, crossover and selection. The search process starts with mutation and in the original DE approach the *rand/1* mutation strategy was introduced:

$$v_{i,j} = x_{r1,j} + F \times (x_{r2,j} - x_{r3,j}), \tag{2}$$

where $x_{i,j}$ is the j-th coordinate of i-th individual, index i is different from indexes $r1$, $r2$ and $r3$, which are also mutually different. It should be noted that in this formula the second parameter of the DE algorithm, namely the scaling factor F, chosen from $[0, 2]$, is used. Mentioned parameter has to be adjusted for an optimization problem in hand.

After mutation the crossover operator is applied to mutant vectors v_i, $i = 1, ..., NP$. One of the most commonly used crossover operators is the binomial crossover, where each gene of the mutant vector v_i is exchanged with the corresponding gene of x_i with a uniformly distributed random number from $[0, 1]$ and additional condition:

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } rand(0,1) < Cr \text{ or } j = jrand \\ x_{i,j}, & \text{otherwise} \end{cases}. \tag{3}$$

Here $Cr \in [0, 1]$ or crossover rate is the last parameter of the DE algorithm, while the $jrand$ is a randomly chosen index from $[1, D]$. Thus, the genetic information of both parent-individual x_i as well as the mutant vector v_i are combined to generate the trial vector u_i. That additional condition is required to make sure that at least one coordinate of the trial vector u_i is taken from the mutant vector v_i, otherwise there is a chance that the trial vector and the parent individual will be the same, which will then lead to unnecessary calculations during the selection step.

The second crossover operator often used in DE is the exponential crossover, which performs crossover of adjacent components of the vector. In the exponential crossover first an integer n_1 is chosen randomly in range $[1, D]$ to act as a starting point for crossover, and then the second index n_2 indicating the number of components to be taken from the mutant vector is determined by incrementing n_2 with Cr probability. The exponential crossover is then performed using indexes n_1 and n_2 as follows:

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } j \in [n_1, n_1 + n_2) \\ x_{i,j}, & \text{otherwise} \end{cases}. \tag{4}$$

To keep individuals in the search space, namely each j-th coordinate of the i-th mutant vector in the interval $[x_{lb,j}, x_{ub,j}], j = 1, ..., D$, the midpoint target bound constraint handling method [1] was applied. In this method if the component of the obtained vector is greater than the upper boundary or smaller than the lower boundary, its parent x_i is used to set the new value for the mutant vector.

Finally, during selection either the trial vector u_i or the parent individual x_i is carried to the next iteration. It is done according to their fitness values, which are usually determined by calculating the objective function values: if the trial vector u_i is better or equal to the parent individual x_i in terms of fitness, then the i-th individual in the population is replaced. The selection step is performed in the following way:

$$x_i = \begin{cases} u_i, & \text{if } f(u_i) \le f(x_i) \\ x_i, & \text{if } f(u_i) > f(x_i) \end{cases}. \tag{5}$$

Nowadays, there are a lot of modifications of the differential evolution approach developed for solving various optimization problems, including one- or multi-objective constrained or unconstrained optimization problems. Most of these modifications are focused on its parameters adjustment or proposing new mutation strategies [5]. The following several well-known mutation strategies are commonly applied to the DE algorithm: *rand/2*, *best/1*, *best/2*, *current-to-best/1* and *current-to-pbest/1*.

The last mutation strategy mentioned here, namely *current-to-pbest/1*, is to be of particular interest. It was introduced in the JADE algorithm [13] and later used in the SHADE algorithm [10] and also in its various modifications. The *current-to-pbest/1* mutation strategy works as follows:

$$v_{i,j} = x_{i,j} + F \times (x_{pbest,j} - x_{i,j}) + F \times (x_{r1,j} - x_{r2,j}), \tag{6}$$

where *pbest* is the index of one of the $pb * 100\%$ best individuals, different from i, $r1$ and $r2$. Thus, to use this mutation strategy the pb parameter should be chosen.

It was established that the scaling factor F as well as the crossover rate CR affect algorithm's efficiency and should be chosen carefully for specific optimization problems. Therefore, using them for mutation and crossover operators with fixed values may cause poor results. In the JADE algorithm [13] parameters F and Cr are adjusted automatically, to be more specific, firstly for each individual x_i at each iteration t, the crossover probability Cr_i is independently generated according to a normal distribution of mean μ_{Cr} and standard deviation 0.1; obtained value is then truncated to $[0, 1]$. In the same manner the mutation factor F_i for each individual x_i at each generation t is independently generated according to a Cauchy distribution with location parameter μ_F and scale parameter 0.1. If the obtained value $F_i \leq 0$ then it is generated again, and if $F_i \geq 1$ then it is set to 1.

Similar ideas were used in the SHADE algorithm [10], and its mechanism for parameter adaptation can be described as follows. The historical memory of H cells $(M_{F,h}, M_{Cr,h})$ is maintained, each containing a couple of F and Cr values (in the SHADE approach the memory size was set to $H = 5$ and the current memory index was denoted as h). Thus, for mutation and crossover operators new parameter values are sampled with Cauchy distribution $F = randc(M_{F,k}, 0.1)$, and normal distribution $Cr = randn(M_{CR,k}, 0.1)$, k is chosen in range $[1, D]$ for each candidate solution. Both obtained values are then truncated to $[0, 1]$ the same way as it is done in the JADE algorithm.

Additionally, two arrays S_F and S_{Cr} are generated: if there was an improvement in terms of the fitness value, then the corresponding values of parameters F and Cr as well as the fitness value difference Δf are stored in these arrays. They are used at the end of the iteration to update the memory cells with weighted Lehmer mean [3]:

$$mean_{wL} = \frac{\sum_{j=1}^{|S|} w_j S_j^2}{\sum_{j=1}^{|S|} w_j S_j}, \tag{7}$$

where $w_j = \frac{\Delta f_j}{\sum_{k=1}^{|S|} \Delta f_k}$, $\Delta f_j = |f(u_j) - f(x_j)|$ and S is either S_{Cr} or S_F.

And finally the new memory cell values are updated: $M_{F,k}^{t+1} = mean_{wL}(F)$, $M_{Cr,k}^{t+1} = mean_{wL}(CR)$, where t is the current iteration number.

The JADE and SHADE algorithms as well as their modifications (for example, the L-SHADE approach [11]) also use an external archive A, which size is usually equal to NP. Solutions replaced during the selection step are stored in that external archive. The archive A is empty during the initialization and it is filled as the algorithm works: if the newly generated candidate solution is better than the parent individual in terms of the fitness value, then the parent is saved in the archive. If the archive is full, the new individuals replace randomly selected ones. The individuals from the archive A are used during the mutation step, namely individuals used to calculate new coordinates can be randomly selected as from the population so from the external archive.

It should be noted that in the L-SHADE algorithm additionally the population size NP changes from iteration to iteration: the linear reduction strategy was proposed for the population size adaptation [11]. The population size NP is recalculated at the end of each generation, and the worst individuals in terms of fitness are eliminated. The population size is calculated with the linear function depending on current number of function evaluations:

$$NP_{g+1} = round(\frac{NP_{min} - NP_{max}}{NFE_{max}} NFE + NP_{max}), \tag{8}$$

where $NP_{min} = 4$ and NP_{max} are the minimal and initial population sizes, NFE and NFE_{max} are the current and maximal number of function evaluations.

3 Expected Fitness Improvement Metric

Every optimization method mainly relies on the fitness values, as long as the goal is to minimize/maximize these values. The described parameter adaptation techniques are designed to adjust parameter values so that higher fitness improvements are achieved. So, the parameter setting is highly dependent not only on the fact of the improvement, but also on the improvement value, like in SHADE algorithm. The problem of setting the parameters represents an optimization problem itself, so a better understanding of this problem structure is highly desirable.

To perform the visualization of the possible fitness improvements at every step of the search process with different crossover rates Cr the Expected Fitness Improvement (EFI) metric is proposed. The EFI is based on the following idea: at every iteration where EFI should be calculated a large set of solutions is generated using mutation and crossover steps with different Cr values from a grid, and for every Cr the average improvement is measured. For example, to estimate the expected fitness improvements for the full range of crossover rates at a given generation g, the values of $Cr = 0, 0, Cr_{st}, ..., 1 - Cr_{st}, 1$ are tested, where Cr_{st} is the step size. The number of steps is defined as $NCr_{st} = \frac{1}{Cr_{st}}$. The result is an array EFI_{g,Cr_k}, $k = 0, ..., NCr_{st}$ for all $g = 1, ..., NG$ generations. The pseudocode of the EFI estimation for different Cr values is presented in Algorithm 1.

Algorithm 1 EFI computation

```
1: Initialize Differential Evolution
2: Set grid with $Cr_{st}$, $NCr_{st}$
3: Initialize matrix $EFI[NCr_{st}, NG] = 0$
4: Generation number $g = 0$
5: while $NFE < NFE_{max}$ do
6:   for $s = 0$ to $NCr_{st}$ do
7:     Set $AImp = 0$
8:     for $i = 1$ to $NP$ do
9:       Sample $F$ value
10:      if $s == NCr_{st}$ then
11:        Sample $Cr$ value
12:      else
13:        Set $Cr = s * Cr_{st}$
14:      end if
15:      Mutation
16:      Crossover
17:      if $f(u_i) < f(x_i)$ then
18:        $AImp = AImp + f(x_i) - f(u_i)$
19:        Save new solution
20:      end if
21:    end for
22:    $EFI[s, NG] = AImp/NP$
23:  end for
24:  $g = g + 1$
25:  Update algorithm specific parameters
26: end while
27: Return matrix $EFI[NCr_{st}, NG]$
```

The EFI array containing the measured possible improvements could be visualized to estimate the distribution of promising crossover rate Cr values and the efficiency of parameter tuning technique used. However, there is a problem of values scale, which arises from the fact that at every next generation the average fitness improvements are gradually decreasing, i.e. if initially the EFI values could be around 10^{10} or even more, at the end of the search they could be around 10^{-10} or even exactly zero. To overcome this issue, the distribution of EFI values should be visualized at every generation separately.

The described EFI metric is applied to the NL-SHADE-RSP algorithm, developed for the CEC 2021 benchmark. It which contains several important improvements compared to the well-known

L-SHADE, namely the non-linear population size reduction, adaptive archive usage, and modified historical memory size depending on the problem dimension. The population size in NL-SHADE is controlled in the following way:

$$NP_{g+1} = round((NP_{min} - NP_{max})NFE_r^{1-NFE_r} + NP_{max}), \tag{9}$$

where $NFE_r = \frac{NFE}{NFE_{max}}$ is the ratio of current number of fitness evaluations. This population size control scheme was taken from the Adaptive Gaining-Sharing Knowledge (AGSK) algorithm [6], which implements the concept of knowledge exchange between experienced and non-experienced individuals in the population. The main operators and algorithm structure is still similar to DE, although there is a difference in trial vector generation - the algorithm generates them using two populations of mutant vectors. Although the paper does not describe it, according to the available source code, AGSK implements non-linear population size reduction presented above.

The NL-SHADE-RSP uses automatic tuning of archive usage probability, originated from the strategy adaptation implemented in IMODE algorithm [8]. The probability p_A of archive usage in the last index $r2$ in current-to-pbest strategy is initially set to 0.5, unless the archive is empty. It is then automatically tuned based on the number of usages n_A, which is incremented every time an offspring is generated using archive and sum of fitness improvements achieved with the archive Δf_A and without it Δf_P. The archive usage probability is recalculated at the end of each generation as follows:

$$p_A = \frac{\Delta f_A/n_A}{\Delta f_A/n_A + \Delta f_p/(1-n_A)}. \tag{10}$$

After this the probability p_A is checked to be within [0.1, 0.9] by applying the following rule: $p_A = min(0.9, max(0.1, p_A))$, similar to the rule used in IMODE algorithm [8].

The *pb* value for current-to-pbest mutation in NL-SHADE-RSP is controlled in a similar manner to the jSO algorithm [2], with the initial pb_{max} set to 0.4 and the final $pb_{min} = 0.2$. The same linear reduction of *pb* parameter is used, allowing wider search at the beginning and better convergence at the end.

The NL-SHADE-RSP algorithm used both exponential and binomial crossovers with equal probability, and the type of crossover to be used was randomly chosen for each individual.

In addition to the proposed EFI metric, the pairwise distance distribution is analyzed in this study. For this purpose, the Euclidean distance between all individuals in the population is estimated, and the histogram of distances is built at every generation separately. The distributions of EFI and distances for several scenarios of DE optimization are presented in the next section.

4 Experimental Setup and Results

The experiments in this study were performed using the benchmark functions presented for the Congress on Evolutionary Computation 2021 competition on single-objective optimization [12] because this framework considers eight cases of the same goal functions, i.e. basic, biased, shifted, rotated functions, and combinations of these modifications, e.g. biased, shifted and rotated at the same time. The set of functions contained 10 functions, which should be tested with the optimization method across dimensions 10 and 20. The maximum number of function evaluations $maxFE$ is set to 2×10^5 and 10^6 for $10D$ and $20D$ functions respectively.

For every function and every benchmark type the step size for checked crossover rates was set to 0.01, i.e. there were 100 crossover rates tested from 0 to 0.99. The initial population size was set to $30D$, as this appeared to be a reasonable setting in previous studies. The memory size was set to $20D$. The EFI arrays, as well as distance histograms are visualized as heatmap profiles. In addition, the best, average, worst fitness values, average EFI, average distance and average of parameter values in memory cells M_{Cr} are shown in the figures. As long as the population size was constantly changing and the number of generations spent at the beginning of the search and at the end of the search to evaluate the same number of solutions is different, the EFI was calculated not every generation, but every $0.002\frac{NFE}{maxFE}$ function evaluations, resulting in 500 iterations. The first experiment was performed for the bent cigar function (F1) without any modifications, $10D$, the EFI heatmap is shown in Figure 1. Better values are shown in yellow.

Figure 1 shows that for the relatively simple bent cigar function, where the achieved function values are around 10^{-200}, during most of the search process larger crossover rates were dominating,

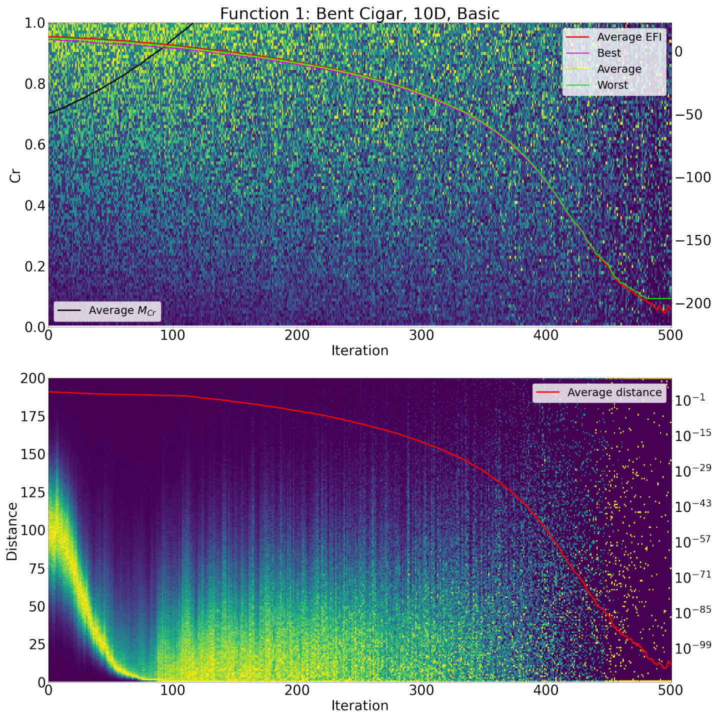


Fig. 1. EFI heatmap and distance histogram profiles, F1, basic benchmark, 10D

i.e. the expected improvement for Cr values was larger when $Cr > 0.5$ than for $Cr < 0.5$. This due to the fact that bent cigar is a non-separable function, i.e. it requires steps along more then one axis to perform the search. The distance histogram profile shows that there are no groups of points, which indicates that there is only one optimum. The average crossover rate in the memory cells quickly converges to 1, which seems to be a valid strategy in this case. Figure 2 shows the results for shifted Schwefel's function, $10D$.

The Schwefel's function has multiple local optima, and in non-rotated case could be efficiently solved by one-variable-at-a-time strategy, which is clearly seen on the EFI heatmap, where smaller Cr values perform better, as larger Cr lead to more "risky" moves in the search space, which not always result in efficient search. However, this is true only for the main part of the search process, i.e. from iteration 10 to iteration 300. During the first 10 iterations large Cr values are better, probably because in this period the initial exploration of the search space happens, capturing the most interesting areas to exploit later. Similar to this, the final convergence, which happens at around generation 300, and also requires larger Cr values, as at this moment it is important to find the optimum in a bowl-like landscape, so diagonal steps could be helpful. Also, the pairwise distance histogram shows that the algorithm identifies multiple local optima, and then deletes some of them thanks to the population size reduction. Despite the fact that smaller Cr are better, the memory cells values are dragged up to 1 due to the biased parameter adaptation with Lehmer mean. Figure 3 shows the results for the same Schwefel's function, $10D$, but for the rotated case.

In the rotated case the EFI heatmap changes for the Schwefel's function, but not in the way which could be expected. The first 10 iterations are almost the same, however, later the search efficiency drops, because it is difficult for the algorithm to tackle the function landscape. Although the function is rotated, large Cr values do not lead to significant improvements, while small Cr lead to some improvements, which makes the algorithm move the memory cells values towards zero. It could have been expected that for non-rotated functions smaller Cr would be more efficient, same as larger Cr for rotated, but the EFI in this cases shows that the opposite happens. In the rotated case the algorithm stopped without reaching the global optimum. Figure 4 demonstrates the EFI heatmap profiles for the next function, Lunacek bi-Rastrigin, which has two large areas of attration.

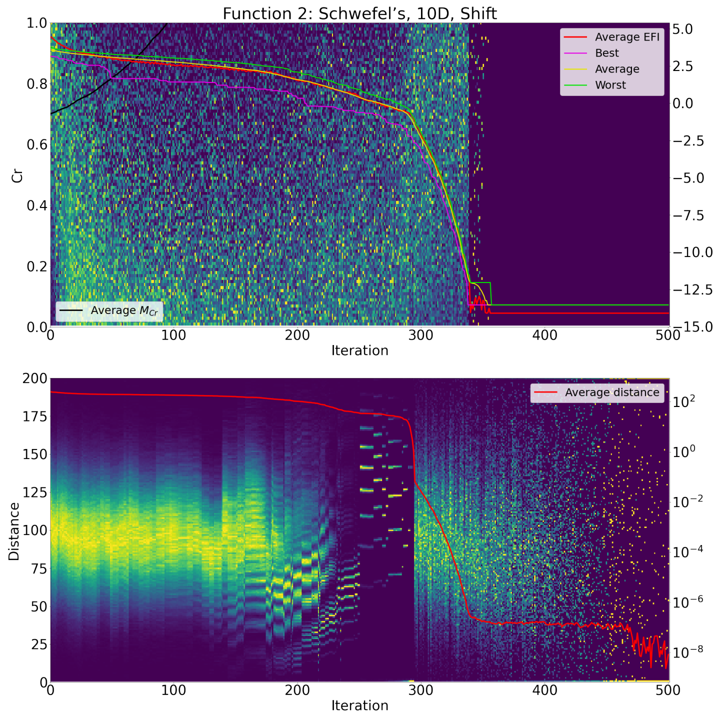


Fig. 2. EFI heatmap and distance histogram profiles, F2, shifted, 10D

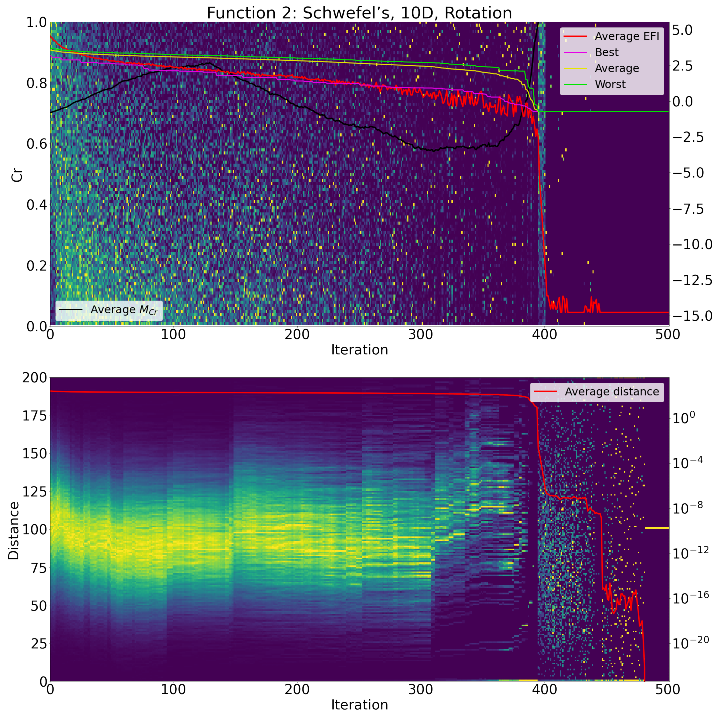


Fig. 3. EFI heatmap and distance histogram profiles, F2, rotated, 10D

Figure 4 represents a particular interest, as here there are several switches between small and large Cr values being better for the function improvements. At the initial stage for the first around

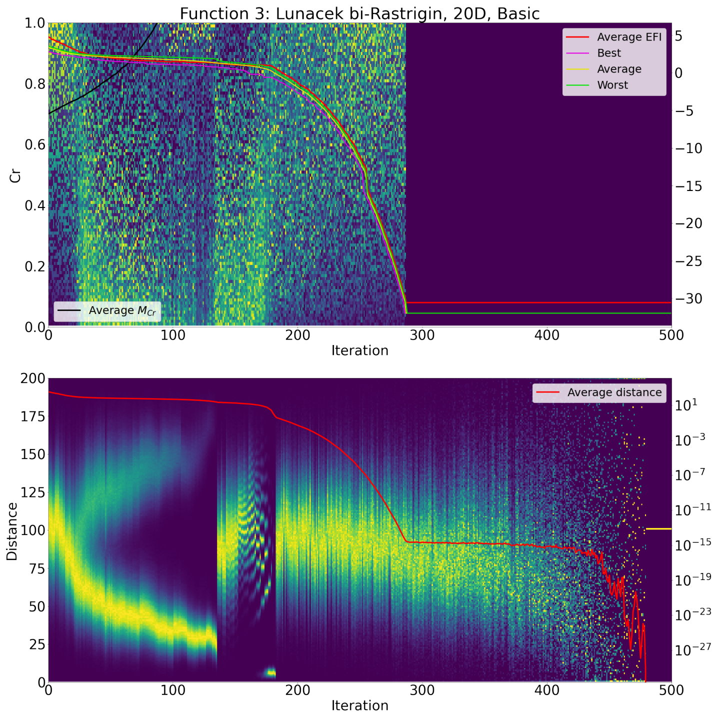


Fig. 4. EFI heatmap and distance histogram profiles, F3, basic benchmark, 20D

20 generations the large Cr are dominating, and at the distance histogram it is clearly seen that all points have almost similar distances. After this period, the population is split in two groups, reach rushing towards one of the areas of attraction, and at this period smaller Cr would result in more larger improvements - this could be because of the convergence to local optima, which is easily done by one-variable-at-a-time manner for non-rotated Rastrigin function. However, at the end of this period, at around iteration 100, large Cr start dominating again, when two parts of the population are far from each other. After the population size reduction cancels one of them out, small Cr values are better again, and finally, when the global optimum is almost found, Cr close to 1 are good again, allowing fast convergence to the optimum. These several switches demonstrate that even for such relatively simple problems the behaviour of Cr values could be quite complicated.

It is important to notice here how the average improvement, i.e. average EFI is related to the Cr switches and average fitness values in the population. When the red line, average EFI is above the yellow (average fitness) and green (worst fitness), $Cr > 0.5$ are dominating, and vice versa. This it true for all periods except the one around iteration 100, where all three lines are close to each other. Similar behaviour could be observed on all previous figures: if average EFI is closer or even larger than worst, then large Cr are better, and when average EFI is close to best fitness, smaller Cr appear to deliver more improvements. The mechanism behind such dependence remains unclear. Figure 5 considers one of the more complicated cases, hybrid function with bias, shift and rotation applied altogether.

In the case shown on Figure 5 there are two clearly seen stages of search: initial convergence and exploration with $Cr > 0.8$ being the best choice, and the more difficult and inefficient search, where smaller Cr allow better improvements. Same as for previous functions, the switch between these two stages, happening after iteration 100, coincides with the average EFI curve hitting worse and average fitness curves. The averaged memory cells M_{Cr} values in this case still keep the Cr close to 0.9, although the EFI shows that this is the region of smallest efficiency. This could be one of the reasons of algorithms low efficiency for this function. It is important to mention, that although smaller Cr are more efficient at the middle of the search process, this does not meant that the function is separable, is actually shows that performing the search along the axis at this stage would bring more benefits.

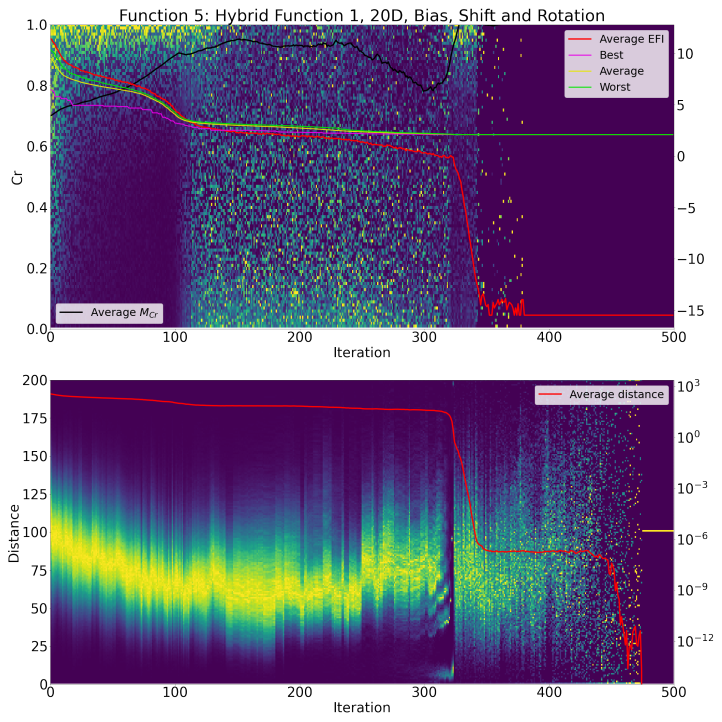


Fig. 5. EFI heatmap and distance histogram profiles, F5, bias, shift and rotation, 20D

Considering the discovered dependency between the crossover rate, current fitness and average fitness improvement, a simple parameter control scheme was tested:

$$Cr = \begin{cases} 1, & \text{if } \frac{1}{|S|}\sum_{j=1}^{|S|} \Delta f_j > \frac{1}{NInds}\sum_{i=0}^{NInds} f_i \\ 0, & \text{otherwise} \end{cases}. \tag{11}$$

The crossover rate was updated every generation. In this case the EFI calculation was switched off. The comparison between NL-SHADE-RSP with and without fitness-based crossover rate control is presented in Table 1. The Mann-Whitney statistical test with significance level $p = 0.01$ is used for comparison, with the number of wins (+), ties (=) and losses (-) over 10 functions for every benchmark set and both 10D and 20D.

Table 1. Mann-Whitney tests of NL-SHADE-RSP against modified with crossover control

Benchmark (code)	10D	20D
Basic (000)	1+/9=/0-	2+/8=/0-
Bias (100)	1+/9=/0-	3+/7=/0-
Shift (010)	1+/9=/0-	2+/8=/0-
Rotation (001)	1+/8=/1-	1+/8=/1-
Bias, Shift (110)	1+/9=/0-	2+/8=/0-
Bias, Rotation (101)	0+/9=/1-	2+/7=/1-
Shift, Rotation (011)	0+/8=/2-	2+/7=/1-
Bias, Shift, Rotation (111)	0+/8=/2-	2+/8=/0-
Total	5+/69=/6-	16+/61=/3-

The results in Table 1 show that such relatively simple control strategy could be competitive, or even better than the success-history based adaptation used in L-SHADE based algorithms. The improvements were observed for function 4 (Expanded Rosenbrock plus Rastrigin) for 10D

and losses were for functions 5 and 6, i.e. hybrid functions. For 20 the wins were for functions 3 (Lunacek bi-Rastrigin), 4 and 6, while few performance deteriorations were for functions 8 and 9, i.e. composition functions. The improvements were mainly found at the end of the search, while for most of the computational resource both modified and non-modified NL-SHADE-RSP had similar performance - this is mainly due to the fact that Cr has much less influence on the algorithm performance, then, for example, scaling factor F.

5 Conclusion

In this study the expected fitness improvement metric was proposed to visualize the parameter search space of the crossover rate of the NL-SHADE-RSP algorithm, allowing to indicate more promising regions for Cr at different stages of the optimization process. The EFI heatmap profiles allowed revealing several important properties, such as switching behaviour of more promising Cr values. Based on the dependence between the improvements and the average fitness values a simple parameter control strategy is proposed, which was shown to be more efficient the standard parameter adaptation in some cases. The EFI calculation represents a general framework, which could be applied to other optimization algorithms to analyze different parameters' dynamics during the search process.

References

1. R. Biedrzycki, J. Arabas, and D. Jagodziski. 2019. Bound constraints handling in Differential Evolution: An experimental study. Swarm and Evolutionary Computation 50 (2019), 100453. https://doi.org/doi.org/10.1016/j.swevo.2018.10.004
2. J. Brest, M.S. Mauec, and B. Bokovic. 2017. Single objective real-parameter optimization algorithm jSO. In Proceedings of the IEEE Congress on Evolutionary Computation. IEEE Press, 13111318. https://doi.org/doi.org/10.1109/CEC.2017. 7969456
3. P.S. Bullen. 2003. Handbook of Means and Their Inequalities. Springer Netherlands. https://doi.org/doi.org/10.1007/978-94-017-0399-4
4. S. Das, S.S. Mullick, and P.N. Suganthan. 2016. Recent advances in differential evolution an updated survey. Swarm and Evolutionary Computation 27 (2016), 130. https://doi.org/doi.org/10.1016/j.swevo.2016.01.004
5. S. Das and P.N. Suganthan. 2011. Differential evolution: a survey of the stateof-the-art. IEEE Transactions on Evolutionary Computation 15, 1 (2011), 431. https://doi.org/doi.org/10.1109/TEVC.2010.2059031
6. A.W. Mohamed, Hadi A. A., Mohamed A., and Awad Noor H. 2020. Evaluating the Performance of Adaptive GainingSharing Knowledge Based Algorithm on CEC 2020 Benchmark Problems. In Proceedings of the IEEE Congress on Evolutionary Computation, CEC. 18. https://doi.org/doi.org/10.1109/CEC48606.2020.9185901
7. K. Price, R.M. Storn, and J.A. Lampinen. 2005. Differential evolution: a practical approach to global optimization (1st. ed.). Springer.
8. K. M. Sallam, Elsayed S., Chakrabortty R. K., and Ryan M. 2020. Improved Multioperator Differential Evolution Algorithm for Solving Unconstrained Problems. In Proceedings of the IEEE Congress on Evolutionary Computation, CEC. 18. https: //doi.org/doi.org/10.1109/CEC48606.2020.9185577
9. R. Storn and K. Price. 1997. Differential evolution a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11, 4 (1997), 341359. https://doi.org/10.1023/A:1008202821328
10. R. Tanabe and A.S. Fukunaga. 2013. Success-history based parameter adaptation for differential evolution. In Proceedings of the IEEE Congress on Evolutionary Computation. IEEE Press, 7178. https://doi.org/doi.org/10.1109/CEC.2013.6557555
11. R. Tanabe and A.S. Fukunaga. 2014. Improving the search performance of SHADE using linear population size reduction. In Proceedings of the IEEE Congress on Evolutionary Computation, CEC. Beijing, China, 16581665. https://doi.org/doi. org/10.1109/CEC.2014.6900380
12. A. Wagdy, A. A Hadi, A. K. Mohamed, P. Agrawal, A. Kumar, and P. N. Suganthan. 2021. Problem Definitions and Evaluation Criteria for the CEC 2021 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization. Technical Report. Nanyang Technological University, Singapore.
13. J. Zhang and A. C. Sanderson. 2007. JADE: Self-adaptive differential evolution with fast and reliable convergence performance. In 2007 IEEE Congress on Evolutionary Computation. 22512258. https://doi.org/doi.org/10.1109/CEC. 2007.4424751

Evolutionary-based Optimization of Hardware Configurations for DNN on Edge GPUs

H. Bouzidi[1], H. Ouarnoughi[1], E-G. Talbi[2], A. Ait El Cadi and[1] S. Niar[1]

Université Polytechnique Hauts-de-France, LAMIH/CNRS, Valenciennes, France[1]
Université de Lille, CNRS/CRIStAL INRIA Lille Nord Europe[2]

Abstract. Performance and power consumption are major concerns for Deep Learning (DL) deployment on Edge hardware platforms. On the one hand, software-level optimization techniques such as pruning and quantization provide promising solutions to minimize power consumption while maintaining reasonable performance for Deep Neural Network (DNN). On the other hand, hardware-level optimization is an important solution to balance performance and power efficiency without changing the DNN application. In this context, many Edge hardware vendors offer the possibility to manually configure the Hardware parameters for a given application. However, this could be a complicated and a tedious task given the large size of the search space and the complexity of the evaluation process. This paper proposes a surrogate-assisted evolutionary algorithm to optimize the hardware parameters for DNNs on heterogeneous Edge GPU platforms. Our method combines both metaheuristics and Machine Learning (ML) to estimate the Pareto-front set of Hardware configurations that achieve the best trade-off between performance and power consumption. We demonstrate that our solution improves upon the default hardware configurations by 21% and 24% with respect to performance and power consumption, respectively.

1 Introduction and related works

Deep Neural Networks (DNN) are known for their intensive computations and memory operations. Thus, they need a careful tuning of both software and hardware, especially for resource-constrained Edge platforms. Modern Edge Graphical Processing Unit (GPU) accelerators provide outstanding performances for Deep Learning (DL) applications [1]. Nevertheless, this comes at the cost of considerable power consumption. Adjusting hardware parameters such as processing cores and operating frequencies according to the DNN execution requirements, represents a different way to improve performance and power efficiency. However, it is hard to decide the best Hardware configuration because of the heterogeneous complexity of the GPU architecture and the wide range of possible configurations. The contradictory nature of the two objectives, increasing performance and decreasing power consumption, makes the optimization even more complex. Hence, this issue can be formulated as a multi-objective optimization problem where we search for an optimal Pareto set of hardware configurations that achieve the best trade-off between the two objectives for a given DNN application. This paper proposes a surrogate-assisted multi-objective optimization that incorporates both Machine Learning (ML) and metaheuristics to approximate an optimal Pareto set of hardware operating frequencies for DNNs on Edge GPU accelerators. The resulted Pareto set will help the user to choose adequate operating frequencies according to the application requirements and system budget constraints.

Some works have been proposed in the literature to address the hardware tuning issue in heterogeneous GPUs. Authors in [2] propose a prediction model based on Support Vector Regression (SVR) for power consumption of GPU kernels for different GPU core and memory frequencies. In [3] and [4], the authors propose a cross-domain modeling approach for power consumption that models both the application and the GPU micro-architecture under variable GPU core and memory frequencies. [5] conducts an empirical study on the impact of frequency scaling on performance and energy consumption of DNNs training and inference on high-performance GPUs. This study shows that GPU DVFS has a significant improvement on both performance and energy consumption of DNNs. [6] proposes a ML based prediction methodology for performance and power consumption of OpenCL kernels on GPU platforms. They combine the two prediction models to approximate a Pareto-set of frequency configurations on GPUs. Where the works mentioned above only focus on tuning GPUs, [7] considers both CPU and GPU tuning in heterogeneous devices. However, the authors use neither prediction models nor optimization algorithms. They rely on empirical observations of profiling results, which may lead to sub-optimal solutions.

2 Problem formulation

Given a fixed DNN application and Edge GPU platform, adjusting the hardware parameters can be formulated as a multi-objective optimization problem where we search for the optimal hardware configurations that provide the best trade-off between performance and power consumption. Let $X = \{x_1, x_2, \ldots, x_n\}$ be a set of hardware configurations, where each x_i represents one instance of the hardware operating frequencies. For instance, a x_i can represent the frequency value of a CPU, GPU cores or memory. Let $F = (f_1, f_2)$ be a vector of objectives to minimize, where $f_i \in \{\text{execution\_time, power\_consumption}\}$. A real evaluation of these objectives is a tedious and time-consuming task. Thus, instead of directly measure F on the Hardware platform, we rely on prediction functions as surrogate-models for F that we denote $\hat{F}$. Our problem is defined as follows:

$$MOP = \begin{cases} \min \hat{F(x)} = (\hat{f_1}(x), \hat{f_2}(x)) \\ s.t. \quad x \in X \end{cases} \tag{1}$$

In this paper, we study the case of optimizing the hardware configurations of a modern Edge GP-GPU platform: NVIDIA Jetson AGX Xavier [8] for a state-of-the-art DNN: AlexNet [9]. We tune four hardware parameters: CPU, GPU, PVA, and memory frequencies. We set the lower and upper bounds for each parameter according to the reported minimum and maximum values in the configuration file of Jetson Tegra system [10].

3 Proposed Approach

We propose a surrogate-assisted evolutionary algorithm that leverages both metaheuristics and Machine Learning. We speed up the optimization process using ML-based prediction models to estimate $\hat{F}$. Our proposed methodology is composed of two main steps:

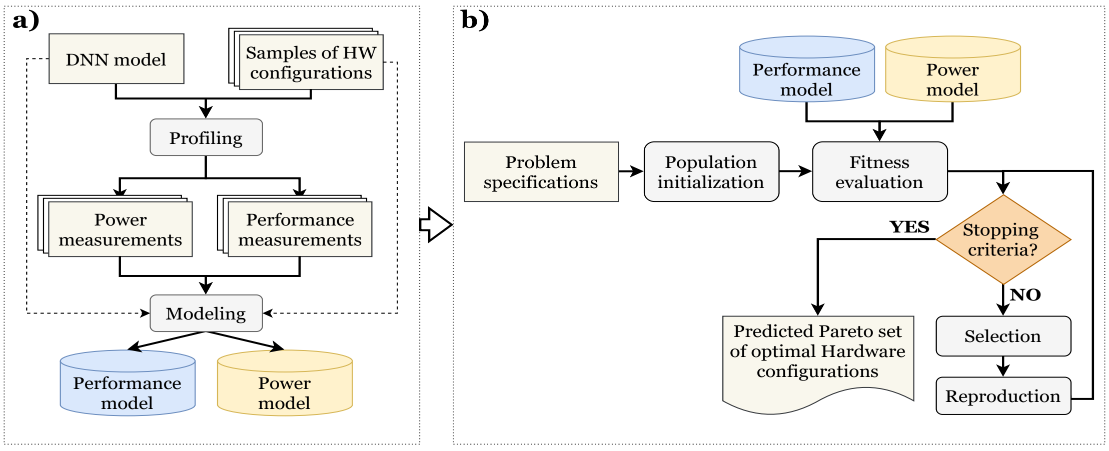


Fig. 1: Overview of the proposed methodology: **a)** corresponds to the training phase of the prediction models for performance and power consumption. **b)** depicts the optimization phase of the hardware parameters using both the trained prediction models and evolutionary-based multi-objective metaheuristic

a. Prediction models training: The training phase is illustrated in figure 1.a. First, we collect training data by profiling the DNN application on randomly sampled Hardware configurations. We denote the resulting training datasets for performance by $D_l = \{(x_1, l1), (x_2, l_2), \ldots, (x_n, l_n)\}$ and for power consumption by $D_p = \{(x_1, p1), (x_2, p_2), \ldots, (x_n, p_n)\}$, where l_i and p_i refer to the measured values of performance and power consumption, respectively, under the hardware configuration x_i. Second, we train SVR-based prediction models for performance and power consumption on D_l and D_p. The trained prediction models are used in the optimization step for a rapid evaluation. The following prediction models are defined:

$$\begin{cases} M_{performance}(D_l) = \sum_{j=1}^{n}(\alpha_j - \hat{\alpha_j})K_{RBF}(d_{jl}, D_l) + b \\ M_{power}(D_p) = \sum_{j=1}^{n}(\alpha_j - \hat{\alpha_j})K_{RBF}(d_{jp}, D_p) + b \end{cases} \tag{2}$$

where b, α_j, and $\hat{\alpha_j}$ refer to the bias and training coefficients of the trained instance of SVR. K_{RBF} is a *radial basis kernel function.* D_l, D_p are the datasets used to train the prediction models for performance and power consumption, respectively.

b. Optimization: Figure 1.b gives an overview of the optimization phase. To efficiently explore the search space of the hardware configurations, we implement MOEA/D, a decomposition-problem-based metaheuristic, as a multi-objective evolutionary optimization algorithm. It uses different evolutionary operators to combine good solutions of neighboring problems, resulting in quick and accurate convergence. We adapt MOEA/D for our problem by leveraging the normalization technique as both performance and power consumption have different scales. We choose the Tchebycheff method as a problem decomposition technique. To generate an ensemble of uniformly distributed weight vectors, we use the Das and Dennis technique. The trained prediction models from figure 1.a are used to evaluate the fitness in the MOEA/D algorithm.

4 Experimental Results

Figures 2 and 3 provide an overview of the estimated Pareto front and set by our proposed method. In figure 2, the blue points represent the predicted Pareto front, while orange ones report the measured values of performance and power consumption of the Pareto front. The seven default hardware configurations of NVIDIA Jetson AGX GPU are marked with the other point types and colors.

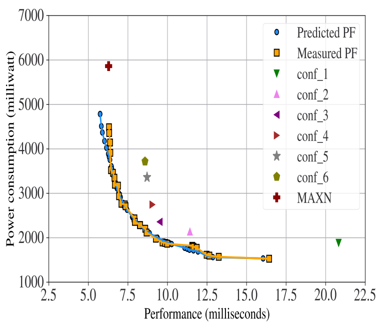


Fig. 2: Predicted vs measured PF

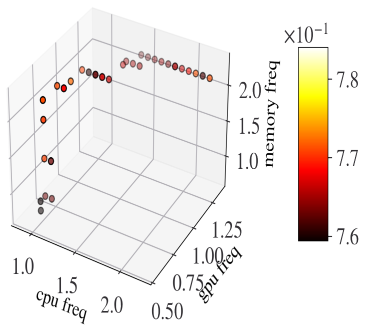


Fig. 3: Obtained Pareto set

Metric	Performance	Power
MAPE	4.21%	5.75%
RMSPE	4.83%	6.69%
Kendall's tau	0.971	0.970

Table 1: Prediction errors and rank order persevering

Figure 2 shows that in addition to the small gap between predictions and measurements, our approach gives configurations that dominate the default suggested NVIDIA configurations. Details of the prediction errors are given in table 1. Obtained MAPE and RMSPE values are small for both performance and power predictions. Moreover, the rank order is highly respected between predicted and measured metrics according to the reported Kendall's τ coefficients in table 1. For configurations that give high performance and low power consumption, we have obtained a configuration with the same performance as the MAXN power mode of NVIDIA with a power-saving of 24%. Similarly, for performance, we have obtained a configuration that gives similar power consumption as the minimum power mode suggested by NVIDIA (i.e., conf_1), with a performance gain of 21%. Figure 3 presents the Pareto set in the decision space. We notice that most configurations maximize the memory frequency. This is explained by the architecture of AlexNet that holds a large number of parameters, which results a high memory activities. This also corroborate our motivation to adjust the hardware configuration according to the DNN requirements.

5 Conclusion

In this paper we introduced a multi-objective optimization approach that leverages both metaheuristics and Machine Learning to optimize the Hardware configurations for Deep Neural Networks on GPU heterogeneous accelerators. The optimization approach incorporates prediction models for approximating the fitness functions to speed up the evaluation of the sampled configurations by the optimization algorithm. Experimental results on AlexNet and Jetson AGX Xavier GPU demonstrated that a higher accurate prediction and a more energy-efficient configurations that outperform the predefined ones can be obtained. As a future work, we plan to develop a cross-surrogate-based multi-objective optimization approach that models both DNN architecture and Hardware configuration. We also propose to enhance the optimization process by injecting the knowledge on the execution requirements of the DNN.

References

1. Hassan Halawa, Hazem A Abdelhafez, Andrew Boktor, and Matei Ripeanu. Nvidia jetson platform characterization. In *European Conference on Parallel Processing*, pages 92–105. Springer, 2017.
2. Qiang Wang and Xiaowen Chu. Gpgpu power estimation with core and memory frequency scaling. *ACM SIGMETRICS Performance Evaluation Review*, 45(2):73–78, 2017.
3. Joao Guerreiro, Aleksandar Ilic, Nuno Roma, and Pedro Tomas. Gpgpu power modeling for multi-domain voltage-frequency scaling. In *IEEE International Symposium on High Performance Computer Architecture*, pages 789–800. IEEE, 2018.
4. João Guerreiro, Aleksandar Ilic, Nuno Roma, and Pedro Tomás. Modeling and decoupling the gpu power consumption for cross-domain dvfs. *IEEE Transactions on Parallel and Distributed Systems*, 30(11):2494–2506, 2019.
5. Zhenheng Tang, Yuxin Wang, Qiang Wang, and Xiaowen Chu. The impact of gpu dvfs on the energy and performance of deep learning: An empirical study. In *10th ACM International Conference on Future Energy Systems*, pages 315–325, 2019.
6. Kaijie Fan, Biagio Cosenza, and Ben Juurlink. Predictable gpus frequency scaling for energy and performance. In *48th International Conference on Parallel Processing*, pages 1–10, 2019.
7. Ourania Spantidi, Ioannis Galanis, and Iraklis Anagnostopoulos. Frequency-based power efficiency improvement of cnns on heterogeneous iot computing systems. In *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, pages 1–6. IEEE, 2020.
8. Jetson AGX xavier developer kit. https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit. Accessed: 2021-02-01.
9. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.
10. Jetson developer kits and modules. https://docs.nvidia.com/jetson/l4t/. Accessed: 2021-02-01.

pyParadiseo : a Python-powered Framework for Metaheuristic Optimization

J. Gmys[1], N. Melab[2,1] and E-G. Talbi[2,1]

[1] Inria Lille Nord-Europe, France
jan.gmys@inria.fr
[2] Université de Lille, France
nouredine.melab@univ-lille.fr
el-ghazali.talbi@univ-lille.fr

1 Introduction

Paradiseo is an open-source white-box object-oriented framework dedicated to the reusable design of metaheuristics. It allows the design and implementation of single-solution and population-based metaheuristics for mono- and multi-objective, continuous, discrete and mixed optimization problems. Built on the core module Evolving Objects (EO) [1], the development of Paradiseo started in the early 2000's [2], becoming since then the largest codebase of existing components for stochastic optimization algorithms.

In 2011, Parejo *et al.* performed a comparative study of metaheuristic optimization frameworks according to 271 features grouped in six areas of interest [4]. Among 33 frameworks, 10 (including Paradiseo) have been selected using well-defined filtering criteria and analyzed. Paradiseo ranked 2^{nd} in terms of metaheuristic feature support and 3^{rd} in terms of adaptation to the optimization problem and its structure. As one of the rare frameworks that provide the most common parallel and distributed models, Paradiseo ranks 1^{st} in terms of advanced characteristics. Figure 1 gives an overview of the modules provided in Paradiseo and E.G. Talbi's book [3] covers their design in detail.

2 The Paradiseo design and motivation for its "pythonization"

Implemented in C++, Paradiseo is based on a clear conceptual separation of the solution methods from the problems they are intended to solve. This separation confers to the user a maximum code and design reuse. Furthermore, the fine-grained nature of the classes provided by the framework allow a higher flexibility compared to other frameworks [3]. At the implementation level, this separation is expressed by dividing classes into two categories: (1) provided classes that implement problem-invariant parts and (2) required abstract classes that have to be specialized by the user to implement the problem-specific parts. The heavy use of small-sized classes allows to change

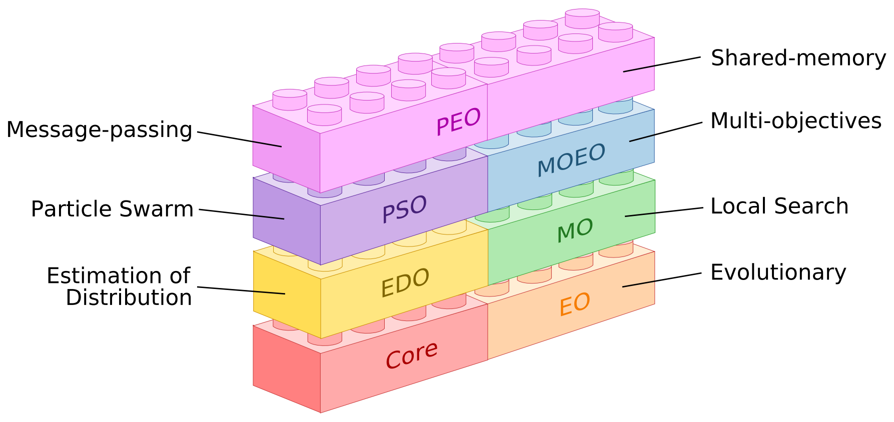


Fig. 1. Paradiseo's modules

existing components and to add new ones easily, without impacting the rest of the application. Templates are used to model metaheuristic features: coding structures, transformation operators, stopping criteria and so on. These templates can be instantiated by the user according to the problem-dependent parameters.

The object-oriented mechanisms such as inheritance, polymorphism, and so on are powerful ways to design new algorithms or evolve existing ones. However, while the use of templates allows to write generic reusable code, it also increases compile times, makes error messages hard to understand and forbids the compilation of Paradiseo into an object library file. Designed as a white-box framework, an efficient usage requires a certain level of knowledge of Paradiseo's inheritance tree from the user. Even the application of a basic genetic algorithm to a custom optimization problem involves deriving user-defined classes from a few abstract base classes. Therefore, users of the framework must be quite comfortable with object-oriented concepts of C++ and Paradiseo's build system (cmake).

Clearly, there is a steep learning curve one has to climb when getting started, which is one of the main impediments to the use of Paradiseo. Even users familiar with the framework may not want to bear long compilation times and potentially complicated error messages when simply checking whether a problem can be solved with metaheuristics or testing some new ideas. While this might just be the inevitable price to pay for achieving such a high level of speed *and* flexibility, another weakness of Paradiseo pointed out in the study of Parejo *et al.* [4] is its relative lack of optimization process support (statistical analysis, user interface, interoperability, experiments design, visualization)—in other words, Paradiseo could be a lot more user-friendly!

3 The pyParadiseo framework

For the reasons mentioned above, we decided to develop a "pythonized" pyParadiseo framework. The objective of this talk is to present an early version of pyParadiseo and to outline its development roadmap for the next two years. The three key objectives laid out for the new framework are:

– **Reducing the cost of the "entry ticket" to the platform.** pyParadiseo exposes a large number of algorithms and components provided by Paradiseo to Python through a high-level C++-Python interface. This allows users to specify required user-defined components and articulate the algorithm selection (composition of components) in Python without having to go through the C++ build process and learn Paradiseo's internals. Parts of the framework will be rewritten entirely in Python, as accessing Paradiseo through a high-level interface reduces the ability for the user to customize *all* components of an algorithm. Specific tutorials for pyParadiseo will be made available to facilitate its usage.
– **Improving interoperability** with machine learning tools, statistics, data science and visualization software, other (exact) optimization software, and so on. Since the early days of Paradiseo (over 20 years ago) Python has evolved into a dominant language in application domains which are closely related to metaheuristic optimization, such as machine learning, data science and scientific computing. The new pyParadiseo framework aims at providing a high level of interoperability with popular tools such as tensorflow, scikit-learn, scipy, R, pandas, and others.
– **Maintaining and extending parallel processing capabilities** of the framework. The outstanding feature of Paradiseo is its parallel processing support and pyParadiseo aims at unlocking these features for Python users, notably though Numba (`https://numba.pydata.org`) which uses the LLVM compiler library for just-in-time compilation and provides support for shared memory parallelization and GPU programming. pyParadiseo will extend Paradiseo's unique support for GPU-accelation [5] and make this feature much easier to use.

Paradiseo, previously hosted at GForge, has been recently migrated to GitLab where it is now co-located next to pyParadiseo in the project repository:

`https://gitlab.inria.fr/paradiseo/`

References

1. Keijzer, K., Merelo, J.J., Romero, G., Schoenauer, M.: Evolving Objects: A general purpose evolutionary computation library, Artificial Evolution, 2310, 829–888 (2002).
2. Cahon, S., Melab, N., Talbi, EG.: ParadisEO: A Framework for the Reusable Design of Parallel and Distributed Metaheuristics. Journal of Heuristics 10, 357–380 (2004).
3. Talbi, E.G.: Metaheuristics: from Design to Implementation. John Wiley & Sons Book, 587 pages (2009).
4. Parejo, J.A., Cortés, A.R., Lozano, S., Fernandez, P.: Metaheuristic optimization frameworks: a survey and benchmarking. Soft Computing, 16(3):527-561 (2012).
5. Melab, N., Luong, T-V., Boufaras, K., Talbi, E-G.: ParadisEO-MO-GPU: a framework for parallel GPU-based local search metaheuristics. In Proc. of ACM GECCO, pages 1189-1196, (2013).

A Large Neighborhood Search for a Cooperative Optimization Approach to Distribute Service Points in Mobility Applications[⋆]

Thomas Jatschka[1], Tobias Rodemann[2], and Günther R. Raidl[1]

[1] Institute of Logic and Computation, TU Wien, Austria
{tjatschk,raidl}@ac.tuwien.ac.at
[2] Honda Research Institute Europe, Germany
tobias.rodemann@honda-ri.de

Abstract. We present a large neighborhood search (LNS) as optimization core for a *cooperative optimization approach* (COA) to optimize locations of service points for mobility applications. COA is an iterative interactive algorithm in which potential customers can express preferences during the optimization. A machine learning component processes the feedback obtained from the customers. The learned information is then used in an optimization component to generate an optimized solution. The LNS replaces a mixed integer linear program (MILP) that has been used as optimization core so far. A particular challenge for developing the LNS is that a fast way for evaluating the non-trivial objective function for candidate solutions is needed. To this end, we propose an *evaluation graph*, making an efficient incremental calculation of the objective value of a modified solution possible. We evaluate the LNS on artificial instances as well as instances derived from real-world data and compare its performance to the previously developed MILP. Results show that the LNS as optimization core scales significantly better to larger instances while still being able to obtain solutions close to optimality.

1 Introduction

The traditional approach for solving service point placement problems, such as distributing charging stations for electric vehicles or vehicle sharing stations in a geographic area, essentially is to first estimate the demand that may be fulfilled at potential locations and then to select actual locations either manually or by some computational optimization. However, estimating the customer demand that may be fulfilled by certain stations is an intricate task in which erroneous assumptions may result in heavy economic losses for the service point provider. Also, estimating demand upfront requires specific data which can be challenging and/or expensive to collect. As an alternative approach, in [1] we introduced a *cooperative optimization approach* (COA) for optimizing the locations of service points in mobility applications. In contrast to the traditional approach, COA is an iterative interactive algorithm that solves the demand data acquisition and optimization in a single process by allowing customers to express their preferences intertwined with the optimization. A machine learning component processes the feedback obtained from the customers and provides a surrogate objective function. This surrogate objective is then used in an optimization component to generate an optimized solution. This solution is then a basis for further interaction with the users to obtain more relevant knowledge, and the whole process is repeated until some stopping criterion is met. So far, COA uses a mixed integer linear program (MILP) in the optimization core for determining solutions [2] or, in a former version [3], basic metaheuristic approaches that treated the problem as black box model and hence do not make significant use of structural properties of the problem. For an exact optimization core, the generated solutions are optimal w.r.t. to the so far known information derived from the customer feedback. However, this optimality comes at the cost of large computation times, especially for large-scale instances with thousands of customers and hundreds of potential service point locations. In contrast, a heuristic optimization core may feature better scalability towards larger instances. To this end we present here a large neighborhood search (LNS) that can reduce computation times by orders of magnitudes with only small

[⋆] Thomas Jatschka acknowledges the financial support from Honda Research Institute Europe.

losses in final solution quality. Due to the nature of the non-trivial objective function of our service point distribution problem, an efficient way for evaluating said objective is necessary to make this speedup possible. Therefore, our LNS features a data structure, referred to as *evaluation graph* for modeling the evaluation of solutions. We show how the evaluation graph can be used to efficiently keep track of small changes in the solution, such as opening or closing a service point. Based on this evaluation graph, the LNS is able to quickly repair partially destroyed solutions in a promising heuristic way. We evaluate the LNS on artificial instances as well as instances derived from real-world data and compare its performance to the previously developed MILP-based approach.

In the next section we review related work. Section 3 formally defines the General Service Point Distribution Problem (GSPDP), as it is referred to, while an overview on the COA framework is given in Section 4. Our main contribution, the LNS with its evaluation graph, is presented in Section 5. Section 6 explains the benchmark scenarios, and Section 7 discusses experimental results. Finally, Section 8 concludes this article and gives an outlook on future work.

2 Related Work

The basic concept of COA was presented in [1]. In interactive optimization algorithms, such as COA, humans are used to (partially) evaluate the quality of solutions and to guide the optimization process. For a survey on interactive optimization, see [4]. Interactive algorithms are often combined with surrogate-based approaches [5, 6], in which a machine learning model is trained to evaluate intermediate solutions approximately in order to reduce user interactions and to avoid user fatigue [7]. In contrast to COA, most approaches from literature only allow a single user to interact with the algorithm, e.g., [8, 9]. Hence, in [10] COA's surrogate function is based on a matrix factorization model [11], a popular collaborative filtering technique [12] in which unknown ratings of items are derived from users with similar preferences.

In [3] two heuristic black box optimization approaches were suggested for COA to generate new candidate solutions w.r.t. to the current surrogate model: a variable neighborhood search as well as a population-based iterated greedy approach. In [2] COA was substantially extended to also be applicable in use cases where the satisfaction of demands relies on the existence of two or more suitably located service stations, such as car and bike sharing systems.

More generally, there exists a vast amount of literature regarding the location planning of service points for mobility applications, see, e.g., [13] for electric vehicle charging stations or [14] for stations of a bike sharing system. However, to the best of our knowledge no further work on interactive optimization approaches for location planning in mobility applications exists.

3 The General Service Point Distribution Problem

In this section we give a formal description of the *Generalized Service Point Distribution Problem* (GSPDP) introduced in [2], which is the problem to be solved at the core of COA and for which we will then propose the LNS. Service points may be set up at a subset of locations $V = \{1, \ldots, n\}$. Establishing a service point at a location $v \in V$ is associated with costs $z_v^{\text{fix}} \geq 0$ and the total setup costs of all stations must not exceed a maximum budget $B > 0$. Additionally, the expected costs for maintaining this service point over a defined time are $z_v^{\text{var}} \geq 0$. Given a set of users U, each user $u \in U$ has a certain set of *use cases* C_u, such as going to work, visiting a recreational facility, or going shopping.

Each user's use case $c \in C_u$ is associated with a demand $D_{u,c} > 0$ expressing how often the use case is expected to happen within some defined time period. The demand of each use case may possibly be satisfied by subsets of service points to different degrees, depending on the concrete application and the customer's preferences. Hence, we associate each use case c of a user u with a set of *Service Point Requirements* (SPR) $R_{u,c}$ with which a user can express the dependency on multiple service points to fulfill the needs of the use case. For example, for the use case of visiting a fitness center using a bike sharing system, one SPR may represent the need of a rental station close to home or work and a second SPR a rental station close to some fitness center. We denote the set of all different SPRs over all use cases of a user u by $R_u = \bigcup_{c \in C_u} R_{u,c}$. Moreover, let $R = \bigcup_{u \in U} R_u$ be the set of all SPRs over all users.

For now, let us further assume we know values $w_{r,v} \in [0, 1]$ indicating the suitability of a service point at location $v \in V$ to satisfy the needs of user $u \in U$ concerning SPR $r \in R_{u,c}$ in the use case

$c \in C_u$. A value of $w_{r,v} = 1$ represents perfect suitability while a value of zero means that location v is unsuitable; values in between indicate partial suitability. For each unit of satisfied customer demand a prize $q > 0$ is earned.

A solution to the GSPDP is a subset of locations $X \subseteq V$ indicating where service points are to be set up. It is feasible if its total fixed costs do not exceed the maximum budget B, i.e.,

$$z^{\mathrm{fix}}(X) = \sum_{v \in X} z_v^{\mathrm{fix}} \leq B. \tag{1}$$

The objective function of the GSPDP is to maximize

$$f(X) = q \cdot \sum_{u \in U} \sum_{c \in C_u} D_{u,c} \cdot \min_{r \in R_{u,c}} \left(\max_{v \in X} w_{r,v} \right) - \sum_{v \in X} z_v^{\mathrm{var}}. \tag{2}$$

In the first term, the obtained prize for the expected total satisfied demand is determined by considering for each user u, each use case c, and each SPR r a most suitable location $v \in V$ at which a service point is to be opened. Over all SPRs of a use case, the minimum of the obtained suitability values is taken. The second term of the objective function represents the total maintenance costs for the service stations. In [2] we have shown that the GSPDP is NP-hard.

By linearizing the objective function, the GSPDP can be modeled by the following MILP.

$$\max \quad q \cdot \sum_{u \in U} \sum_{c \in C_u} D_{u,c}\, y_{u,c} - \sum_{v \in V} z_v^{\mathrm{var}}\, x_v \tag{3}$$

$$\sum_{v \in V} o_{r,v} \leq 1 \qquad \forall r \in R \tag{4}$$

$$o_{r,v} \leq x_v \qquad \forall v \in V,\ r \in R \tag{5}$$

$$y_{u,c} \leq \sum_{v \in V} w_{r,v} \cdot o_{r,v} \qquad \forall u \in U,\ c \in C_u,\ r \in R_{u,c} \tag{6}$$

$$\sum_{v \in V} z_v^{\mathrm{fix}}\, x_v \leq B \tag{7}$$

$$x_v \in \{0,1\} \qquad \forall v \in V \tag{8}$$

$$0 \leq y_{u,c} \leq 1 \qquad \forall u \in U,\ c \in C_u \tag{9}$$

$$0 \leq o_{r,v} \leq 1 \qquad \forall r \in R,\ v \in V \tag{10}$$

Binary variables x_v indicate whether or not a service point is deployed at location $v \in V$. Continuous variables $o_{r,v}$ are used to indicate the actually used location $v \in V$ for each SPR $r \in R$; these variables will automatically become integer. The degree to which a use case $c \in C_u$ of a user $u \in U$ can be satisfied is expressed by continuous variables $y_{u,c}$. The objective value is calculated in (3). Inequalities (4) ensure that at most one location with the highest suitability value is selected for each SPR. Inequalities (5) and (6) ensure that use cases are only satisfied if there are suitable locations with opened service points for each SPR of the respective use case. Inequalities (6) additionally determine the degree to which a use case is satisfied. Last but not least, Inequality (7) ensures that the budget is not exceeded.

4 Cooperative Optimization Algorithm

A crucial aspect of COA's general approach is that the suitability values $w_{r,v}$ are not explicitly known a priori. A complete direct questioning would not only be extremely time consuming but users would easily be overwhelmed by the large number of possibilities, resulting in incorrect information. For example, users easily tend to only rate their preferred options as suitable and might not consider certain alternatives as also feasible although they actually might be on second thought when no other options are available.

Hence, interaction with users needs to be kept to a minimum and should be done wisely to extract as much meaningful information as possible. Therefore, COA does not ask a user to directly provide best suited station locations for the SPRs but creates meaningful *location scenarios*, i.e., subsets of locations, and asks the users to evaluate these. More specifically, a user u returns as

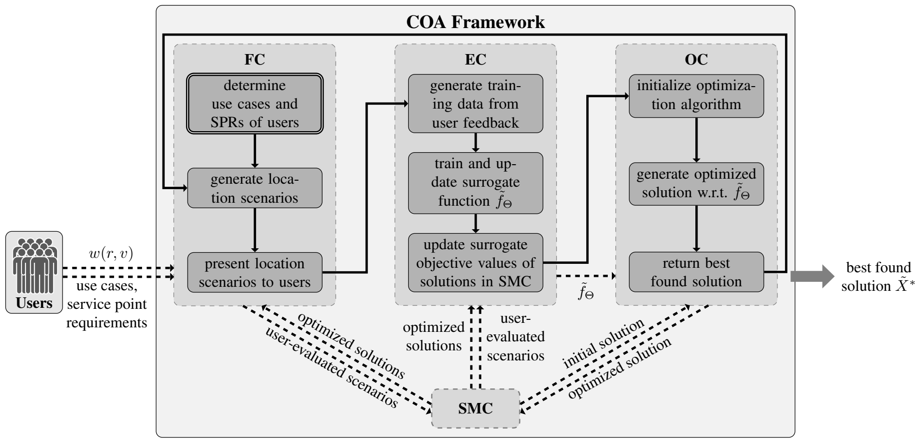

COA Framework
FC
determine use cases and SPRs of users
generate location scenarios
present location scenarios to users
EC
generate training data from user feedback
train and update surrogate function $\tilde{f}_\Theta$
update surrogate objective values of solutions in SMC
OC
initialize optimization algorithm
generate optimized solution w.r.t. $\tilde{f}_\Theta$
return best found solution
Users
$w(r, v)$
use cases, service point requirements
$\tilde{f}_\Theta$
best found solution $\tilde{X}^*$
optimized solutions
user-evaluated scenarios
optimized solutions
user-evaluated scenarios
initial solution
optimized solution
SMC

Fig. 1: Components of COA and their interaction.

evaluation of a location scenario S w.r.t. one of the user's SPRs $r \in R_u$ a best suited location $v_{r,S} \in S$ and the corresponding suitability value $w(r, v_{r,S}) > 0$ or the information that none of the locations of the scenario S is suitable. We assume here that the suitability of a location w.r.t. an SPR can be specified on a five valued scale.

The COA framework consists of a *Feedback Component (FC)*, an *Evaluation Component (EC)*, an *Optimization Component (OC)*, and a *Solution Management Component (SMC)*. Figure 1 illustrates the fundamental principle and communication between these components. During an initialization phase, the FC first asks each user $u \in U$ to specify her or his use cases C_u with their associated SPRs $R_{u,c}$, as well as corresponding demands $D_{u,c}$, $c \in C_u$. Then, the FC is responsible for generating individual location scenarios for each user which are presented to the user in order to obtain her/his feedback.

The obtained feedback is processed in the EC. A crucial assumption we exploit is that in a large user base some users typically have similar preferences about the locations of service points w.r.t. to some of their use cases. Hence, by identifying these similarities and learning from them, the EC maintains and continuously updates a *surrogate suitability function* $\tilde{w}_\Theta(r, v)$ approximating the real and partially unknown suitability values $w_{r,v}$ of service point locations $v \in V$ w.r.t. SPR $r \in R$ without interacting with the respective user. Based on this surrogate function, the EC also provides the surrogate objective function

$$\tilde{f}_\Theta(X) = q \cdot \sum_{u \in U} \sum_{c \in C_u} D_{u,c} \cdot \min_{r \in R_{u,c}} \left(\max_{v \in X} \tilde{w}_\Theta(r, v) \right) - \sum_{v \in X} z_v^{\text{var}} \tag{11}$$

with which a candidate solution X can be approximately evaluated.

A call of the OC is supposed to determine an optimal or close-to-optimal solution to the problem with respect to the EC's current surrogate objective function $\tilde{f}_\Theta$. In [2] this is achieved by solving the MILP (3)–(10) in which the suitability values are approximated by the surrogate suitability function $\tilde{w}_\Theta$.

The SMC stores and manages information on all generated solutions as well as suitability values obtained by the FC.

The whole process is repeated until some termination criterion is reached. In the end, COA returns a solution $\tilde{X}^*$ with the highest surrogate objective value of all of the so far generated solutions. For more details, in particular on how meaningful solution scenarios are derived in the FC and how a matrix factorization is utilized to determine the approximated values $\tilde{w}_\Theta(r, v)$ in the EC, we refer the interested reader to [2].

5 Large Neighborhood Search

We now propose a large neighborhood search (LNS) as a faster replacement for the original MILP-based optimization core in COA. The LNS follows the classical scheme from [15]. The key idea of LNS is to not search neighborhoods in a naive enumerative way but instead to identify via some problem-specific more effective procedure either best or promising solutions within larger neighborhoods. To this end, LNS frequently follows an iterative destroy and repair scheme: First, a given solution is partially destroyed, typically by freeing a subset of the decision variables and fixing the others to their current values. Afterwards this partial solution is repaired by finding best or at least promising values for the freed variables. If the obtained solution is better than the previous one, it is accepted, otherwise the previous solution is kept.

In our LNS a solution to a GSPDP instance is destroyed in a uniform random fashion by adding k^{dest} new locations to the solution, where k^{dest} is a parameter that is varied.

To repair a solution X, we make use of a randomized greedy approach: Let $\Delta(v, X)$ denote by how much the objective value of a solution X would decrease when removing location v from X. Note that, it is discussed later how $\Delta(v, X)$ can be efficiently calculated for all $v \in X$. In each iteration we first generate a restricted candidate list of k^{rep} locations $v \in V$ for which $\Delta(v, X)$ is lowest, i.e., the candidate list contains the locations that have the lowest impact on objective value of X. Hereby, k^{rep} is another strategy parameter. Ties are broken randomly. A location is then chosen uniformly at random from this restricted candidate list and removed from X.

To construct an initial solution in the first iteration of COA, we also make use of the repair heuristic, starting from $X = V$ and then sequentially removing locations from X for which $\Delta(v, X)$ is lowest until the solution becomes feasible, i.e. $k^{\text{rep}} = 1$ for constructing an initial solution. In subsequent iterations of COA, the LNS is warm-started with COA's current best solution $\tilde{X}^*$.

Our LNS makes use of two destroy operators with $k^{\text{dest}} = 10$ and $k^{\text{dest}} = 20$, respectively, and two repair operators with $k^{\text{rep}} = 2$ and $k^{\text{rep}} = 4$, respectively. These settings have shown to yield a robust convergence behavior across the kinds and sizes of instances in our benchmark sets. In each iteration a repair and destroy operator is chosen uniformly at random. Moreover, each LNS run terminates after 40 iterations without improvement.

A crucial aspect for developing an effective heuristic for solving the GSPDP is that computing the surrogate objective value $\tilde{f}_\Theta$ of a solution in a straight-forward way from scratch is time consuming. Hence, in order to accelerate this task we maintain for a GSPDP instance a directed graph $G = (LL \cup SL \cup CL \cup \{l_{\text{obj}}\}, A_{LL} \cup A_{SL} \cup A_{CL})$ referred to as *evaluation graph.* This graph represents the objective function calculation and stores intermediate results for a current solution, allowing for an effective incremental update in case of changes in the solution. The evaluation graph consists of four layers of nodes, which are the location layer (LL), the SPR layer (SL), the use case layer (CL), and the evaluation layer containing a single node l_{obj}. The location layer contains n nodes corresponding to the locations in V, i.e., $LL = \{l_v \mid v \in V\}$. The use case layer consists of one node for each use case C_u of each user $u \in U$, i.e., $CL = \{l_c \mid c \in C_u,\ u \in U\}$, and the SPR layer contains one node for each SPR in $\in R_{u,c}$, for each use case $c \in C_u$ and user $u \in U$, i.e., $SL = \{l_{u,r} \mid r \in R_{u,c},\ c \in C_u, u \in U\}$.

There exists an arc in G from a node of the location layer l_v to a node of the SPR layer $l_{u,r}$ if $\tilde{w}_\Theta(v, r) > 0$, i.e., $A_{LL} = \{(l_v, l_{u,r}) \mid l_v \in LL,\ l_{u,r} \in SL,\ \tilde{w}_\Theta(v, r) > 0\}$. A node of the SPR layer is connected to a node of the use case layer if the corresponding SPR is an SPR of the corresponding use case, i.e., $A_{SL} = \{(l_{u,r}, l_c) \mid l_{u,r} \in SL,\ l_c \in CL,\ r \in R_{u,c}\}$. Finally, each node l_c of the use case layer is connected to l_{obj}, i.e., $A_{CL} = \{(l_c, l_{\text{obj}}) \mid l_c \in CL\}$.

The location layer gets as input a binary vector $(x_v)_{v \in V}$ with $x_v = 1$ if $v \in X$ and $x_v = 0$ otherwise, w.r.t. a solution X. Moreover, each node in G has an activation function $\alpha()$ that decides its output value which is propagated to its successor nodes in the next layer as their input, i.e.,

$$\alpha_{LL}(l_v, X) = \begin{cases} 1 & \text{if } v \in X \\ 0 & \text{otherwise,} \end{cases} \qquad \forall l_v \in LL, \tag{12}$$

$$\alpha_{SL}(l_{u,r}, X) = \max_{(l_v, l_{u,r}) \in A_{LL}} (\alpha_{LL}(l_v, X) \cdot \tilde{w}_\Theta(v, r)) \qquad \forall l_{u,r} \in SL, \tag{13}$$

$$\alpha_{CL}(l_c, X) = \min_{(l_{u,r}, l_c) \in A_{SL}} \alpha_{SL}(l_{u,r}, X) \qquad \forall l_c \in C_L, \tag{14}$$

$$\alpha_{eval}(l_{\text{obj}}, X) = \sum_{(l_c, l_{\text{obj}}) \in A_{CL}} \alpha_{SL}(l_c, X) - \sum_{v \in X} z_v^{\text{var}}. \tag{15}$$

The evaluation graph stores all output of the activation functions from the last evaluated solution and is therefore especially efficient for evaluating subsequent solutions that only differ in a single location $v \in V$ as not everything needs to be calculated from scratch but just the modified value v w.r.t. the current solution X needs to be propagated. Note that A_{LL} needs to be updated in each iteration of COA as the EC recalculates the surrogate suitability values $\tilde{w}_\Theta$ in each iteration with newly obtained user feedback.

Additionally, the evaluation graph also makes it possible to efficiently keep track of how much each location v contributes to the objective value of a solution. For this purpose, we introduce the following new notations. Let X be a current solution and $c \in C_u$ be a use case of a user $u \in U$ that is satisfied (to some degree) in X, i.e., for each $r \in R_{u,c}$ there exists at least one location $v \in X$ such that $\tilde{w}_\Theta(r,v) > 0$. Let $v^{\max}(r,X)$ refer to a location in the solution for which $\tilde{w}_\Theta(r, v^{\max}(r,X)) = \max_{v\in X} \tilde{w}_\Theta(r,v)$. For the sake of readability we further refer to $\tilde{w}_\Theta(r, v^{\max}(r,X))$ as $\tilde{w}_\Theta^{\max}(r,X)$. Additionally, let $\tilde{w}_\Theta^{\text{fallback}}(r,X)$ denote the second highest suitability value for an SPR r w.r.t. to the locations in X, i.e., $\tilde{w}_\Theta^{\text{fallback}}(r,X) = \max\{\tilde{w}_\Theta(r,v) \mid v \in X \setminus \{v^{\max}(r,X)\} \cup \{0\}\}$. Note that $\tilde{w}_\Theta^{\text{fallback}}(r,X)$ is zero if $X \setminus \{v^{\max}(r,X)$ is empty. Finally, let $\tilde{w}_\Theta^{\min}(u,c,X) = \min_{r\in R_{u,c}} \tilde{w}_\Theta^{\max}(r,X)$.

From the definition of the surrogate objective function, it follows that the degree to which a use case c is satisfied in a solution X is only determined by the set of locations $\{v^{\max}(r,X) \mid r \in R_{u,c}\}$. Hence, let $\Delta(u,c,v,X)$ denote by how much the degree to which a use case $c \in C_u$ of a user $u \in U$ is satisfied w.r.t. a solution X would decrease when removing v from X, i.e.,

$$\Delta(u,c,r,X) = \begin{cases} q \cdot D_{u,c} \cdot (\tilde{w}_\Theta^{\max}(r,X) - \tilde{w}_\Theta^{\text{fallback}}(r,X)) & \tilde{w}_\Theta^{\text{fallback}}(r,X) < \tilde{w}_\Theta^{\min}(u,c,X) \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

$$\Delta(u,c,v,X) = \max\{\Delta(u,c,r,X) \mid r \in R_{u,c}, v = v^{\max}(r,X)\} \cup \{0\} \tag{17}$$

Generally speaking, the removal of a location v from a solution X only has an impact on a use case $c \in C_u$ if it results in a change of $\tilde{w}_\Theta^{\min}(u,c,X)$. Additionally, note that the GSPDP also allows cases in which one service point location can be associated to multiple SPRs of the same use case. Such a case would for example correspond to situations in which a customer returns a vehicle at the same station at which the vehicle was picked up. Therefore, the removal of a location from X may affect a use case w.r.t. more than one of its SPRs. However, only the change that affects $\tilde{w}_\Theta^{\min}(u,c,X)$ the most is relevant for calculating by how much the degree to which a use case is satisfied changes.

Hence, the amount $\Delta(v,X)$ by how much the objective value of a solution would decrease when removing location v from X is calculated as

$$\Delta(v,X) = -z_v^{\text{var}} + \sum_{u\in U} \sum_{c\in C_u} \Delta(u,c,v,X). \tag{18}$$

Note that the time required for determining $w^{\max}, w^{\text{fallback}}$, and $w^{\min}$ is negligible if the domain of the rating scale by which users can specify suitability values is small. Moreover, $\Delta(v,X)$ does not need to be calculated from scratch every time a location is added or removed from the solution. Let $X \circ \{v\}$ refer to the modification of a solution, by either adding or removing a location $v \subseteq V$ to/from X. Then $\Delta(v', X \circ \{v\})$ with $v' \in X$ can be determined from $\Delta(v',X)$ as follows:

$$\Delta(v', X \circ \{v\}) = \Delta(v',X) - \sum_{u\in U} \sum_{c\in C_u} \Delta(u,c,v',X) + \Delta(u,c,v',X \circ \{v\}). \tag{19}$$

Additionally, $\Delta(v,X)$ needs to be updated only w.r.t. use cases that are actually affected by the modification of the solution, i.e., only if $\tilde{w}_\Theta^{\max}, \tilde{w}_\Theta^{\text{fallback}}$, or $\tilde{w}_\Theta^{\min}$ of a use case change. Finally, for each use case $c \in C_u$ at most $2 \cdot |R_{u,c}|$ locations need to updated in the worst case.

6 Benchmark Scenarios

Benchmark scenarios for our experiments were generated as described in detail in [2] and are available at `https://www.ac.tuwien.ac.at/research/problem-instances/#spdp`.

The considered test instances are of two groups. One group of instances is inspired by the location planning of car sharing systems and hence referred to as CSS. Locations are randomly generated on a grid in the Euclidean plane. The number of use cases for each user is chosen randomly, but each use case always has two SPRs. To generate suitability values for locations w.r.t. SPRs, ten *attraction points* are randomly placed on the grid, and each SPR is then associated with a geographic location sampled from a normal distribution centered around a randomly chosen attraction point. The actual suitability value is then calculated via a sigmoid function based on the distance between the SPR's geographic location and the respective service point location and afterwards perturbed by Gaussian noise. Six sets of 30 benchmark instances were generated for CSS, considering different combinations of the number of potential service point locations and the number of users.

The second group of instances is derived from real-world taxi trip data of Manhattan and referred to as MAN. The underlying street network of the instances corresponds to the street network graph of Manhattan provided by the Julia package LightOSM[3]. The Taxi trips have been extracted from the 2016 Yellow Taxi Trip Data[4]. For the generation of the instances all trips within the ten taxi zones with the highest total number of pickups and drop-offs of customers were considered, resulting in a total of approximately two million taxi trips. The set of potential service point locations has been chosen randomly from vertices of the street network that are located in the considered taxi zones. Each use case of a user is associated with two SPRs representing the origin and destination of a trip chosen uniformly at random. Suitability values for locations w.r.t. SPRs are again calculated via a sigmoid function based on the distance between the SPR's geographic location and the respective service point location. The MAN benchmark group also consists of 30 instances in total with each instance having 100 potential service point locations and 2000 users. Additionally, each instance will be evaluated with different budget levels $b\,[\%] \in \{30, 50, 70\}$ such that about b percent of the stations can be expected to be opened.

7 Computational Results

All test runs have been executed on an Intel Xeon E5-2640 v4 2.40GHz machine in single-threaded mode. Gurobi 9.1[5] was used to solve the MILP models in the OC. We compare our COA with the LNS, denoted in the following as COA[LNS], to the COA from [2] that uses the MILP (3)–(10) as optimization core and henceforth denoted as COA[MILP]. Since COA[LNS] always uses the current best solution $\tilde{X}^*$ as initial solution, we also set $\tilde{X}^*$ as starting solution in the MILP solver.

We present the results of COA by providing snapshots at different levels of performed user interactions. In [2] we have argued that at most $I_u^{\mathrm{UB}} = \sum_{r \in R_u} (|\{v \mid w(r,v) > 0\}| + 1)$ interactions per user are required to completely derive all suitability values of user $u \in U$. Let I_u be the number of user interactions of user $u \in U$ performed within COA to generate some solution. Then, $I = 100\% \cdot (\sum_{u \in U} I_u / I_u^{\mathrm{UB}})/m$, refers to the relative average number of performed user interactions relative to I_u^{UB} over all users. Results are presented in an aggregated way at various *interaction levels* ψ by selecting for each instance the COA iteration at which I is largest but does not exceed ψ.

First, we provide some general information about the performance of COA[LNS]. Table 1 shows for each instance group at different interaction levels the average number of performed destroy and repair iterations n_{iter}, the average time in seconds required for finding the best solution $t^*[s]$, and the average total time in seconds until the LNS terminated $t[s]$. We can see that the LNS terminates within 43 to 80 iterations on average and usually terminates within three seconds for the CSS instances and within eight seconds for the MAN instances. While the total number of iterations is relatively low, we later show in Table 2 that the solutions generated by the LNS are almost optimal w.r.t. the presented instances. The number of iterations performed tends to decrease as the number of performed user interactions increases while the total runtime increases in each iteration for the MAN instance but stays almost constant for the CSS instances. The decreasing number of iterations can be explained by the LNS being warm-started with the so far best found solution $\tilde{X}^*$. Moreover, as the number of user interactions increases, COA is able to identify more locations relevant to the SPRs of the use cases of the users, resulting in a higher number of arcs between the nodes in the service point layer and the nodes in the SPR layer of the respective evaluation graph. Therefore, the number of iterations until the LNS converges decreases while the time for performing one iteration increases.

Next, we investigate COA runs in which we apply in each iteration both, the LNS and the MILP, for solving the exact same GSPDP instances w.r.t. $\tilde{w}_\Theta$ as well as the initial solution $\tilde{X}^*$.

[3] `https://github.com/DeloitteDigitalAPAC/LightOSM.jl`
[4] `https://data.cityofnewyork.us/Transportation/2016-Yellow-Taxi-Trip-Data/k67s-dv2t`
[5] `https://www.gurobi.com/`

Table 1: Results of COA[LNS].

	CSS																	
(n, m)	(100, 500)			(100, 1000)			(200, 1000)			(200, 2000)			(300, 1500)			(300, 3000)		
ψ	n_{iter}	$t^*[s]$	$t[s]$	n_{iter}	$t^*[s]$	$t[s]$	n_{iter}	$t^*[s]$	$t[s]$	n_{iter}	$t^*[s]$	$t[s]$	n_{iter}	$t^*[s]$	$t[s]$	n_{iter}	$t^*[s]$	$t[s]$
40	50	0.21	0.87	62	0.96	2.08	60	0.21	0.61	76	1.37	2.31	59	0.35	0.66	75	1.32	1.99
50	51	0.27	1.09	67	1.18	2.82	67	0.48	1.05	68	1.03	2.42	65	0.50	0.94	71	1.32	2.34
60	46	0.22	1.17	58	1.09	3.09	58	0.41	1.17	59	1.01	2.74	66	0.61	1.19	65	1.47	2.96
70	47	0.25	1.39	53	0.79	3.09	50	0.30	1.27	58	1.18	3.07	64	0.56	1.22	64	1.45	3.27
80	45	0.18	1.51	48	0.44	2.78	45	0.16	1.16	50	0.59	2.80	56	0.47	1.33	59	1.14	2.98
90	43	0.10	1.48	44	0.25	2.64	45	0.17	1.19	49	0.60	2.73	46	0.22	1.17	44	0.43	2.51

	MAN								
b	30%			50%			70%		
ψ	n_{iter}	$t^*[s]$	$t[s]$	n_{iter}	$t^*[s]$	$t[s]$	n_{iter}	$t^*[s]$	$t[s]$
40	78	2.19	3.85	74	1.58	3.35	59	0.65	1.76
50	80	3.70	6.12	75	2.76	5.25	55	1.10	3.30
60	78	4.22	8.20	72	3.72	7.21	63	2.00	5.02
70	65	3.40	8.18	64	3.10	7.74	54	1.51	5.62
80	55	2.62	7.65	58	2.93	8.12	54	1.93	6.74
90	49	1.40	7.24	48	1.27	7.35	46	0.73	6.09

The MILP solver is able to find optimal solution in all cases, but at the expense of typically much longer running times. Note however that only the solution generated by the LNS is further used for the next iteration in COA. Table 2 shows the average percentage gaps between the objective values of the best solutions found by the LNS and respective optimal solutions w.r.t. $\tilde{f}_\Theta$, denoted by gap$_{\tilde{f}_\Theta}$[%], the average total running times in seconds of the LNS $t[s]$, the average times $t^\circ_{\mathrm{M}}[s]$ needed by the MILP solver required for reaching a solution with at most the same objective value as the solution obtained by the LNS, as well as the average total times $t_{\mathrm{M}}[s]$ in seconds of the MILP solver for determining a proven optimal solution. Bold values indicate best times w.r.t. t, t°_{M}, and t_{M}. First, we can see that the solutions generated by the LNS are on average only about 1% worse than an optimal solution for most instance groups. Next, the table shows that for CSS instances with a n/m ratio of 1/10, the MILP solver needs significantly more time for finding good solutions. Note that these instances have been designed in such a way that users behave less similar resulting in more complex instances. Nonetheless, the LNS significantly outperforms the MILP w.r.t. all instance groups. For all instance groups the LNS requires significantly less time on average to terminate than the MILP needs to reach a solution of the same quality as the solution obtained by the LNS. Additionally, Table 2 especially highlights how much more time the MILP requires for improving a solution at the same quality as the best found LNS solution to a provable optimal solution. Moreover, further tests have shown that most of the time the LNS is able to identify its best found solution while the MILP solver has still not yet solved the root relaxation in the same amount of time.

Finally, we want to compare independent COA[MILP] and COA[LNS] runs, and thus the impact of the in general slightly worse intermediate solutions of the LNS on the overall results of the two COA variants. For this purpose Table 3 shows for each interaction level the average optimality gaps between the best found solution during the optimization to an optimal solution w.r.t. the original objective f for COA[LNS] (gap$_{\mathrm{L}}$[%]) as well as COA[MILP] (gap$_{\mathrm{M}}$[%]). The table shows that small differences in the solution quality w.r.t. $\tilde{f}_\Theta$ translate to slightly larger differences w.r.t. f. With the exception of the MAN instance group with $b[\%] = 30$, the solutions generated by COA[LNS] are usually at most 3% off from the values obtained by COA[MILP]. In most cases, the average differences are around 1% or less. Hence, in general it can be concluded that the LNS substantially outperforms the MILP in terms of computation time while still being able to generate almost optimal solutions.

Table 2: Times required by the LNS, times the MILP solver needed to obtain a solution with at least the same quality as the solution of the LNS, as well as the total time required by the MILP to find a proven optimal solution. Additionally, the optimality gaps between the LNS solutions and respective optimal solutions are also shown.

	CSS											
	(100, 500)				(200, 1000)				(300, 1500)			
ψ	$t[s]$	$t^{\circ}_{\mathrm{M}}[s]$	$t_{\mathrm{M}}[s]$	$\mathrm{gap}_{\tilde{f}_\Theta}[\%]$	$t[s]$	$t^{\circ}_{\mathrm{M}}[s]$	$t_{\mathrm{M}}[s]$	$\mathrm{gap}_{\tilde{f}_\Theta}[\%]$	$t[s]$	$t^{\circ}_{\mathrm{M}}[s]$	$t_{\mathrm{M}}[s]$	$\mathrm{gap}_{\tilde{f}_\Theta}[\%]$
40	**0.87**	4.58	6.50	0.91	**0.61**	2.41	3.90	0.65	**0.66**	2.97	4.01	0.08
50	**1.09**	4.03	7.68	0.90	**1.05**	3.60	5.27	0.27	**0.94**	3.59	4.31	0.10
60	**1.17**	5.50	7.47	0.78	**1.17**	3.32	5.12	0.19	**1.19**	3.67	4.62	0.07
70	**1.39**	6.65	8.10	0.64	**1.27**	3.75	4.81	0.12	**1.22**	3.14	3.74	0.07
80	**1.51**	5.74	7.04	0.44	**1.16**	4.48	5.97	0.08	**1.33**	3.28	4.40	0.04
90	**1.48**	5.48	6.73	0.33	**1.19**	4.11	5.06	0.06	**1.17**	4.58	5.30	0.03

	CSS											
	(100, 1000)				(200, 2000)				(300, 3000)			
ψ	$t[s]$	$t^{\circ}_{\mathrm{M}}[s]$	$t_{\mathrm{M}}[s]$	$\mathrm{gap}_{\tilde{f}_\Theta}[\%]$	$t[s]$	$t^{\circ}_{\mathrm{M}}[s]$	$t_{\mathrm{M}}[s]$	$\mathrm{gap}_{\tilde{f}_\Theta}[\%]$	$t[s]$	$t^{\circ}_{\mathrm{M}}[s]$	$t_{\mathrm{M}}[s]$	$\mathrm{gap}_{\tilde{f}_\Theta}[\%]$
40	**2.08**	21.87	37.42	2.15	**2.31**	32.79	92.15	1.24	**1.99**	26.04	85.61	0.81
50	**2.82**	28.03	50.51	1.97	**2.42**	37.61	90.11	1.07	**2.34**	39.84	101.52	0.57
60	**3.09**	35.60	59.04	1.45	**2.74**	36.47	126.67	0.89	**2.96**	38.34	130.05	0.47
70	**3.09**	42.95	67.34	1.74	**3.07**	40.48	111.96	0.84	**3.27**	43.41	136.93	0.36
80	**2.78**	43.57	69.94	1.83	**2.80**	40.98	120.07	0.90	**2.98**	43.78	137.76	0.37
90	**2.64**	40.09	74.98	1.37	**2.73**	41.33	123.32	0.78	**2.51**	63.56	149.78	0.37

	MAN											
	30%				50%				70%			
ψ	$t[s]$	$t^{\circ}_{\mathrm{M}}[s]$	$t_{\mathrm{M}}[s]$	$\mathrm{gap}_{\tilde{f}_\Theta}[\%]$	$t[s]$	$t^{\circ}_{\mathrm{M}}[s]$	$t_{\mathrm{M}}[s]$	$\mathrm{gap}_{\tilde{f}_\Theta}[\%]$	$t[s]$	$t^{\circ}_{\mathrm{M}}[s]$	$t_{\mathrm{M}}[s]$	$\mathrm{gap}_{\tilde{f}_\Theta}[\%]$
40	**3.85**	67.21	326.46	2.15	**3.35**	17.24	53.54	0.87	**1.76**	6.36	10.71	0.21
50	**6.12**	80.31	328.53	1.36	**5.25**	16.76	95.43	0.59	**3.30**	10.20	15.29	0.11
60	**8.20**	131.28	368.28	1.19	**7.21**	24.36	89.15	0.43	**5.02**	14.59	21.54	0.07
70	**8.18**	140.34	375.46	1.06	**7.74**	24.86	108.59	0.35	**5.62**	13.22	21.73	0.06
80	**7.65**	160.13	414.39	1.12	**8.12**	27.70	108.01	0.34	**6.74**	18.00	24.43	0.05
90	**7.24**	154.44	411.55	1.29	**7.35**	43.43	102.70	0.27	**6.09**	13.03	17.46	0.03

Table 3: Quality of solutions generated by COA[LNS] and COA[MILP].

	CSS											
(n,m)	(100, 500)		(100, 1000)		(200, 1000)		(200, 2000)		(300, 1500)		(300, 3000)	
ψ	$\mathrm{gap}_{\mathrm{L}}[\%]$	$\mathrm{gap}_{\mathrm{M}}[\%]$	$\mathrm{gap}_{\mathrm{L}}[\%]$	$\mathrm{gap}_{\mathrm{M}}[\%]$	$\mathrm{gap}_{\mathrm{L}}[\%]$	$\mathrm{gap}_{\mathrm{M}}[\%]$	$\mathrm{gap}_{\mathrm{L}}[\%]$	$\mathrm{gap}_{\mathrm{M}}[\%]$	$\mathrm{gap}_{\mathrm{L}}[\%]$	$\mathrm{gap}_{\mathrm{M}}[\%]$	$\mathrm{gap}_{\mathrm{L}}[\%]$	$\mathrm{gap}_{\mathrm{M}}[\%]$
40	3.46	**2.98**	11.92	**9.57**	1.64	**1.21**	4.34	**2.81**	0.54	**0.51**	2.81	**2.36**
50	2.06	**1.50**	7.34	**4.72**	0.81	**0.62**	2.86	**1.90**	0.43	**0.31**	1.87	**1.27**
60	1.62	**0.63**	4.31	**2.29**	0.45	**0.36**	2.21	**1.20**	0.30	**0.20**	1.31	**0.72**
70	1.20	**0.27**	4.11	**1.61**	0.22	**0.12**	1.56	**0.47**	0.19	**0.11**	0.81	**0.28**
80	0.66	**0.18**	2.72	**0.92**	0.15	**0.05**	1.30	**0.18**	0.09	**0.04**	0.58	**0.15**
90	0.43	**0.01**	1.95	**0.08**	0.08	**0.02**	0.95	**0.05**	0.06	**0.01**	0.44	**0.03**

	MAN					
b	30%		50%		70%	
ψ	$\mathrm{gap}_{\mathrm{L}}[\%]$	$\mathrm{gap}_{\mathrm{M}}[\%]$	$\mathrm{gap}_{\mathrm{L}}[\%]$	$\mathrm{gap}_{\mathrm{M}}[\%]$	$\mathrm{gap}_{\mathrm{L}}[\%]$	$\mathrm{gap}_{\mathrm{M}}[\%]$
40	8.61	**3.46**	3.58	**3.46**	**1.32**	3.46
50	5.14	**1.88**	2.19	**1.88**	**0.77**	1.88
60	3.32	**1.14**	1.41	**1.14**	**0.46**	1.14
70	2.53	**0.63**	0.86	**0.63**	**0.25**	0.63
80	2.03	**0.26**	0.54	**0.26**	**0.12**	0.26
90	1.77	**0.10**	0.33	**0.10**	**0.05**	0.10

8 Conclusion and Future Work

We presented a large neighborhood search (LNS) to be used as optimization core in a cooperative optimization approach (COA) for the general service point distribution problem (GSPDP) in mobility applications. While the LNS follows the traditional destroy and repair principle, a major challenge was to (a) effectively guide the repair heuristic to produce promising new solutions and to (b) efficiently calculate the surrogate objective function for modified solutions in an incremental way. Both was achieved by introducing the evaluation graph, which stores relevant intermediate results allowing efficient updates when stations are added to or removed from the current solution. In particular, the evaluation graph provides an effective way to keep track of how much impact each location in the solution has on its respective objective value. The efficient update possibility also allows to consider a larger amount of locations during the destroy procedure. The performance of the LNS within COA was tested on artificial instances as well as instances derived from real-world data and was compared to the original COA with its MILP-based optimization core. Results show that at the cost of a slight deterioration of usually not more than one percent in the quality of the solutions, the LNS can outperform the MILP w.r.t. to computation times by orders of magnitudes. In future work it seems promising to also consider other metaheuristic approaches, such as an evolutionary algorithm that uses the evaluation graph for efficiently recombining solutions. Moreover, the GSPDP it is still a rather abstract problem formulation, and it would be important to extend it as well as the solving approach to cover further relevant practical aspects such as capacities of stations and time dependencies of users.

References

1. Jatschka, T., Rodemann, T., Raidl, G.R.: A cooperative optimization approach for distributing service points in mobility applications. In Liefooghe, A., Paquete, L., eds.: Evolutionary Computation in Combinatorial Optimization. Volume 11452 of LNCS., Springer (2019) 1–16
2. Jatschka, T., Raidl, G., Rodemann, T.: A general cooperative optimization approach for distributing service points in mobility applications. Technical Report AC-TR-21-006, TU Wien, Vienna, Austria (2021) submitted.
3. Jatschka, T., Rodemann, T., Raidl, G.R.: VNS and PBIG as optimization cores in a cooperative optimization approach for distributing service points. In Moreno-Díaz, R., et al., eds.: Computer Aided Systems Theory – EUROCAST 2019. Volume 12013 of LNCS., Springer (2020) 255–262
4. Meignan, D., Knust, S., Frayret, J.M., Pesant, G., Gaud, N.: A review and taxonomy of interactive optimization methods in operations research. ACM Transactions on Interactive Intelligent Systems **5** (2015) 17:1–17:43
5. Sun, X., Gong, D., Jin, Y., Chen, S.: A new surrogate-assisted interactive genetic algorithm with weighted semisupervised learning. IEEE Transactions on Cybernetics **43** (2013) 685–698
6. Sun, X.Y., Gong, D., Li, S.: Classification and regression-based surrogate model-assisted interactive genetic algorithm with individual's fuzzy fitness. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, ACM (2009) 907–914
7. Llorà, X., Sastry, K., Goldberg, D.E., Gupta, A., Lakshmi, L.: Combating user fatigue in iGAs: Partial ordering, support vector machines, and synthetic fitness. In: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, ACM (2005) 1363–1370
8. Kim, H.S., Cho, S.B.: Application of interactive genetic algorithm to fashion design. Engineering applications of artificial intelligence **13** (2000) 635–644
9. Dou, R., Zong, C., Nan, G.: Multi-stage interactive genetic algorithm for collaborative product customization. Knowledge-Based Systems **92** (2016) 43–54
10. Jatschka, T., Rodemann, T., R. Raidl, G.: Exploiting similar behavior of users in a cooperative optimization approach for distributing service points in mobility applications. In Nicosia, G., et al., eds.: Machine Learning, Optimization, and Data Science. Volume 11943 of LNCS., Springer (2019) 738–750
11. Bell, R.M., Koren, Y., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42** (2009) 30–37
12. Ekstrand, M.D., Riedl, J.T., Konstan, J.A.: Collaborative filtering recommender systems. Foundations and Trends in Human–Computer Interaction **4** (2011) 81–173
13. Frade, I., Ribeiro, A., Gonçalves, G., Antunes, A.: Optimal location of charging stations for electric vehicles in a neighborhood in Lisbon, Portugal. Transportation Research Record: Journal of the Transportation Research Board **2252** (2011) 91–98
14. Kloimüllner, C., Raidl, G.R.: Hierarchical clustering and multilevel refinement for the bike-sharing station planning problem. In: International Conference on Learning and Intelligent Optimization. Volume 10556 of LNCS., Springer (2017) 150–165
15. Gendreau, M., Potvin, J.Y., et al.: Handbook of Metaheuristics. Volume 3. Springer (2019)

A New Learnheuristic: Binary SARSA - Sine Cosine Algorithm (BS-SCA)

Marcelo Becerra-Rozas[1], José Lemus-Romani[1], Broderick Crawford[1], Ricardo Soto[1], Felipe Cisternas-Caneo[1], Andrés Trujillo Embry[1], Máximo Arnao Molina[1], Diego Tapia[1], Mauricio Castillo[1], and José-Miguel Rubio[3]

Pontificia Universidad Católica de Valparaíso, Valparaíso, Chile
{broderick.crawford,ricardo.soto}@pucv.cl
{marcelo.becerra.r,jose.lemus.r,felipe.cisternas.c,andres.trujillo.e;
maximo.arnao.m,diego.tapia.r,mauricio.castillo.d}@mail.pucv.cl
Universidad Bernardo O'Higgins, Santiago, Chile
josemiguel.rubio@ubo.cl

1 Introduction

Optimization problems have been growing in a big way in the last decades, causing the emergence of more metaheuristics (MH) that try to solve NP-Hard combinatorial optimization problems. The premise of the No Free Lunch Theorem [1, 2] incentives us to develop increasingly robust optimization algorithms that present and high feasible, quality solutions in reasonable computational times.

To develop more robust algorithms, different techniques used in MH can be distinguished. First, there is the hybridization of mathematical programming with MH or also known as "Matheuristics" [3] There are methods that interrelate MH with simulation problem, also known as "Simheuristics" [4]. There are also hybridization methods between MH techniques that combine their exploration-exploitation components [5]. Currently, the area that is in constant development and will continue to develop for a couple of years more, learnheuristic is the interaction of MH with learning techniques, where it has been observed in several studies that these techniques support operators in various ways to improve their performance [6–8].

From this particular case, this work presents the incorporation of SARSA to determine a specific action: the selection of binarization schemes when solving binary domain problems. A comparison is made between the proposed implementation of the new Binary SARSA-Sine Cosine Algorithm (BS-SCA) and the work presented by Cisternas-Caneo F. et al. in [9], where in this occasion the binarization selector is Q-Learning. The problem to be solved is Set Covering Problem with 45 instances to be evaluated, it can be determined that the proposed implementation with SARSA has a statistically significant better performance.

The paper is organized as follows: In section 2 we raise points in favor of why it is worthwhile to implement reinforcement learning techniques with swarm intelligence algorithms. In section 3 we present the reinforcement learning techniques belonging to the machine learning area: Q-Learning and SARSA. Our proposed BS-SCA as a new algorithm is presented in section 4. Finally, the results obtained are evaluated and analyzed and a conclusion is drawn in sections 5 and 6.

2 Swarm-Intelligence Algorithms

Swarm Intelligence Algorithms are regularly based on interesting behaviors found in nature. In particular, in those situations that involve behaviors carried out collectively by some biological systems, such as animals or insects. That is why this type of algorithms are based on the study of self-organized and distributed systems, since they manipulate a population of agents with limited individual capacity where each of which reacts with its environment and can modify it in order to carry out an intelligent collective behavior. This ability allows communication between agents, and when they perceive changes in their environment, they interact locally with other agents. This leads to the emergence of a global behavior giving agents the ability to solve highly complex problems.

MH have different elements depending on the metaphor they represent. Although they are generally composed of an instance, parameters, operators, population, local search, evaluation, initialization, and decision variables [7]. For the definition of the parameters a considerable number of experiments are carried out that will allow their values to be adjusted. This requires the dedication of considerable time and an imbalance between the exploration and exploitation of MH. That is why the need arises for the integration of dynamic elements in the algorithms so that these are adjusted during the execution of the iterations. In this paper, SARSA and Q-Learning are used to perform a dynamic selection of operators.

2.1 Hybrid-Metaheuristics

A hybrid MH is described as the combination of a metaheuristic algorithm and a different learning algorithm, for instance, matheuristics, Machine Learning Programming, Reinforcement Learning (RL) techniques [5]. For this work we will focus on the hybrids generated with RL, where we find two groups: RL supporting MH, or MH supporting RL.

Focusing on the first group mentioned above, two lines of research are shown in the work of García et al. [10]. First, we find the integration of RL techniques as the replacement of an operator, such as the handling of a population, local search, and parameter tuning. Second, is to use RL as a selector of a set of MH, choosing the most appropriate one depending on the problem to be approached.

When using RL as a selector, we can divide this category into three groups. The first is algorithm selection that chooses from a set of techniques for the problem, in order to obtain better performance for a set of similar instances [11]. Secondly we find the hyperheuristic strategies, where their goal is to use the MH to cover a set of problems. And finally we find cooperative strategies, which combine algorithms sequentially with the objective of improving the robustness of the solution.

We consider worthy of mention the following reasons of why hybridization is advantageous:

- It makes the MH adaptive allowing the algorithm to be applicable to different problems.
- It does not require complete information about the problem, since reinforcement learning models learn by collecting experience [12].
- By using independent learning agents [13] allows, in some cases, the computational cost to be lower. Since only one update formula is used in each step.
- If general features are used, the information learned by the reinforcement learning can be used in other parts of the same problem [14].
- The behavior of various reinforcement learning methods end in optimal action state pairs [15] which can be exploited. For example, one can see how the policy choosing the next action make progress at each step.

3 Reinforcement-Learning Techniques

The learning process in this technique lies in maximizing the value function, as it tells us how fruitful the consequence of the action taken by the agent will be from a state. The expected reward function R_t is composed of both the current rewards obtained and the discounted future rewards. The future reward from the passage of time t is given by the following equation (1).

$$R_t = \sum_{j=0}^{n} \gamma^j \cdot r_{t+j+1} \tag{1}$$

Where $\gamma \in [0, 1]$ is the discount factor, r_t is the reward when an action is taken at time t, and n is often regarded as the time when the process terminates. Hence, the agent's goal is to learn a policy capable of maximizing long-run rewards by interacting with the environment based on one's own experience. Thus, at each step t, starting from the state s_t, the agent has to compute as follows the value of the action-value function: $Q^{\pi}(s, a)$ for each possible action a_t on the basis of the policy π. The policy π, can be defined as in eq. (2):

$$Q^{\pi}(s, a) = \mathbb{P}_{\pi}\{R_t \mid s_t = s, a_t = a\} \tag{2}$$

where $\mathbb{P}_{\pi}\{R_t \mid s_t = s, a_t = a\}$. Now the agent aims at obtaining the optimal state-action function Q*(s,a). Then two methods called Q-learning and SARSA will be compared to get the optimal state-action value function.

Q-Learning is the best known reinforcement learning algorithm [9, 16] and is an off-policy method, i.e., the agent selects action a independent of the environment where it is executed. When the agent selects an action and executes it in the environment a perturbation is generated. The impact of this perturbation is judged through the reward or punishment (r) to decide which is the next state s_{t+1} of the environment. The way to represent the update equation mathematically is eq. (3):

$$Q(s_t, a_t) \longleftarrow (1-\alpha) \cdot Q(s_t, a_t) + \alpha \cdot [r + \gamma \cdot maxQ(s_{t+1}, a_{t+1})] \tag{3}$$

where α represents the learning rate, γ is the discount factor, r is the immediate reward or penalty received, and $maxQ(s_{t+1}, a_{t+1})$ tells us that a_{t+1} is the best action for state s_{t+1}. This tells us that the new value of the state-action function is not only related to the reward or punish received, but also to the best estimated action for the next state.

Instead, SARSA is an on-policy method [15], the agent learns the value of the state-action pair based on the performed action, in other words, when the value of the current state-action is updated, the next action a_{t+1} will be taken. Unlike Sarsa, in Q-Learning the action a_{t+1} is completely greedy. Based on this, the state-action value update equation is defined as in eq. (4):

$$Q(s_t, a_t) \longleftarrow Q(s_t, a_t) + \alpha \cdot [r + \gamma \cdot Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \tag{4}$$

In fact, the procedure of forming the Q-table is the same for both algorithms. The only difference between them is the update rule that is being followed in every step, equations (3) and (4). The different update rule enables SARSA to learn faster than the Q-learning algorithm. However, this makes SARSA a more conservative algorithm and the probability of finding the optimal policy is higher for Q-learning.

3.1 Reward Function

The big question when using reinforcement learning methods is: How to reward or punish the actions performed by the agent?. The balance between reward and punishment achieves an equal variation of the selection of actions, so that the best action found is more reliable.

Different learnheuristics have been found in the literature in which MH incorporate reinforcement learning techniques as a machine learning technique. The classical reward function used by these learnheuristics is adapted to the behavior of the MH.

For instance, we will use a simplified version of the version proposed by Yue Xu and Dechang Pi [17] for optimal topology selection in particle swarm optimization. The simplified performance-oriented version of the MH considers as reward value +1 when fitness is improved or 0 otherwise. As a result, the reward or penalty visible in equations (5) and is born where only the reward is given. The type of reward mentioned above is shown in table (1).

Table 1. Types of Rewards

Reference	Reward Function
[17]	$\mathrm{r}_n = \begin{cases} +1, & if\ the\ current\ action\ improves\ fitness \\ 0, & otherwise. \end{cases} \quad (5)$

4 Binary SARSA - Sine Cosine Algorithm

Sine Cosine Algorithm (SCA) [18] is a swarm metaheuristic of recent interest to researchers for solving complex optimisation problems. While it is a metaheuristic that provides great results, it still falls into the classic problem of swarm metaheuristics, falling into premature convergences which implies falling into local optima [19]. Recent works [9, 20, 16], the authors propose ambidextrous metaheuristics [21, 22] where their main objective is to improve decision making during the optimisation process, which translates into improving the exploration and exploitation balance, i.e. avoiding premature convergence and thus improving the solutions obtained.

The authors in [9, 20] propose the incorporation of Q-Learning to Sine Cosine Algorithm as an intelligent binarization schemes selector mechanism to solve discrete optimisation problems [23]. Our proposal is to replace Q-Learning by another intelligent selector mechanism such as SARSA and compare both techniques performing the same task, selecting binarisation schemes with the aim of improving the major problem of SCA, premature convergence.

As proposed in [20], the states used in SARSA are the phases of the MH, i.e., exploration and exploitation. The estimation of these states is done by means of diversity metrics which allow quantifying the dispersion of individuals in the search space. The metric used in this work is the Dimensional-Hussain Diversity [24] and is defined as follows:

$$Div = \frac{1}{l \cdot n} \sum_{d=1}^{l} \sum_{i=1}^{n} |\bar{x}^d - x_i^d| \tag{6}$$

Where n is the number of search agents in the population X, $\bar{x}^d$ is average of the d-th dimension, and l is the number of dimension of the optimization problem.

This diversity quantification is calculated iteration by iteration and to determine whether the population has an exploration or exploitation behavior the equations proposed by Morales-Castañeda et. al. in [25] are used. There they propose that the percentage of exploration (XPL%) and the percentage of exploitation (XPT%) is given as follows:

$$XPL\% = \left(\frac{Div_t}{Div_{max}}\right) \times 100 \;\;,\;\; XPT\% = \left(\frac{|Div_t - Div_{max}|}{Div_{max}}\right) \times 100 \tag{7}$$

By obtaining these percentages, the phase in which the MH is found is determined as follows:

$$next\ state = \begin{cases} Exploration\ if\ XPL\% \geq XPT\% \\ Exploitation\ if\ XPL\% < XPT\% \end{cases} \tag{8}$$

The proposal of this work is shown in Algorithm (1). In line 1 we initialise the Q-values of the Q-Table, in lines 4-5 we determine the initial state (exploration or exploitation) of SARSA, in line 7 we select an action from the Q-Table for the corresponding state, in line 16 we execute the selected action and observe its consequences from the obtained fitness, in lines 17-18 we determine the next state of SARSA, and finally in line 19 we update the Q-value of the selected action from the SARSA equation (4).

5 Experimental Results

The results are shown in the table (2), which displays the name of each instance resolved in the first column and the optimum value of each instance in the second column. While the following 6 columns present the best results (Best), average results (Avg), and RPD according to Eq. 9 for each of the instances and both versions: BS-SCA and BQ-SCA. The comparison is performed using the algorithm proposed by Cisternas-Caneo et al. [9], using a version of SCA hybridized with Q-Learning.. The last two rows of the table show the average values of all cases and the p-value of the test of Wilcoxon Mann-Whitney [26]. In order to establish which of the two hybridized versions is superior for this collection of situations, the test lets us to assess if the outcomes achieved differ considerably.

$$\text{RPD} = \frac{100 \cdot (Best - Opt)}{Opt}. \tag{9}$$

The total number of instances used to solve the Set Covering Problem with Beasley's OR-Library instances was 45. These cases were run with 40 populations and 1000 iterations, with a total of 40,000 calls to the objective function, as used in [27]. The code was written in Python 3.8 and executed using the free Google Colaboraty service [28]. The following parameters were specified for the SARSA and Q-Learning algorithms: $\gamma = 0.4$ and $\alpha = 0.1$.

The exploration-exploitation graphs obtained: Fig. 1 and 2 according to section 4, do not show similar behaviors to those presented by Morales-Castañeda et al. in [25], despite the fact that our

Table 2. Results obtained by BS-SCA and BQ-SCA solving SCP

		BS-SCA			BQ-SCA		
Inst.	Opt.	Best	Avg	RPD	Best	Avg	RPD
4.1	429	**432**	434.0	0.7	435	442.72	1.4
4.2	512	**527**	535.9	2.93	537	553.71	4.88
4.3	516	**524**	529.0	1.55	534	552.03	3.49
4.4	494	**502**	515.12	1.62	514	530.44	4.05
4.5	512	**524**	530.14	2.34	537	553.17	4.88
4.6	560	**564**	570.56	0.71	573	588.68	2.32
4.7	430	**435**	439.25	1.16	441	449.77	2.56
4.8	492	**500**	502.57	1.63	509	516.39	3.46
4.9	641	**665**	677.14	3.74	683	697.48	6.55
4.10	514	**518**	519.57	0.78	521	533.88	1.36
5.1	253	**256**	266.22	1.19	264	272.75	4.35
5.2	302	**318**	326.27	5.3	327	335.58	8.28
5.3	226	**230**	231.1	1.77	**230**	235.62	1.77
5.4	242	**247**	250.22	2.07	250	254.6	3.31
5.5	211	**213**	215.62	0.95	218	221.46	3.32
5.6	213	**218**	222.12	2.35	221	231.26	3.76
5.7	293	**297**	305.11	1.37	304	316.4	3.75
5.8	288	**290**	294.56	0.69	296	301.32	2.78
5.9	279	**283**	285.14	1.43	284	293.42	1.79
5.10	265	**271**	273.0	2.26	274	281.35	3.4
6.1	138	**143**	146.0	3.62	144	148.16	4.35
6.2	146	**151**	152.56	3.42	152	159.06	4.11
6.3	145	**148**	149.5	2.07	149	151.29	2.76
6.4	131	**131**	133.6	0.0	133	136.03	1.53
6.5	161	**165**	171.82	2.48	173	183.26	7.45
a.1	253	**260**	264.22	2.77	266	269.42	5.14
a.2	252	**254**	266.2	0.79	267	273.8	5.95
a.3	232	**238**	244.25	2.59	245	248.87	5.6
a.4	234	**241**	246.5	2.99	245	252.61	4.7
a.5	236	**242**	245.0	2.54	247	251.27	4.66
b.1	69	**69**	70.9	0.0	71	72.68	2.9
b.2	76	**76**	77.8	0.0	78	81.35	2.63
b.3	80	**80**	84.4	0.0	82	83.87	2.5
b.4	79	**82**	83.3	3.8	83	84.9	5.06
b.5	72	**72**	73.12	0.0	73	75.03	1.39
c.1	227	**237**	240.5	4.41	246	251.85	8.37
c.2	219	**230**	235.44	5.02	237	242.89	8.22
c.3	243	**252**	254.88	3.7	259	263.25	6.58
c.4	219	**228**	232.25	4.11	230	236.1	5.02
c.5	215	**222**	225.5	3.26	229	234.2	6.51
d.1	60	**62**	64.6	3.33	64	65.97	6.67
d.2	66	**67**	73.27	1.52	69	69.97	4.55
d.3	72	**75**	81.8	4.17	76	78.86	5.56
d.4	62	**62**	64.6	0.0	63	64.16	1.61
d.5	61	**62**	69.89	1.64	64	66.35	4.92
Average		**259.18**	263.88	2.11	264.38	271.27	4.23
p-value							0.00

Algorithm 1 Binary S-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 $X_{best}$
 1: Initialize Q-Table with $q_0$
 2: Initialize random population X
 3: Set initial $r_1$
 4: Calculate Initial Population Diversity (X) using equation (6)
 5: Define the initial state using equation (8)
 6: for iteration $(t)$ do
 7:     $a$ : Select action from Q-Table
 8:     for solution $(i)$ do
 9:         Evaluate solution $X_i$ in the objective function
10:         for dimension $(j)$ do
11:             Update $P^t_j$, where $P^t_j = X_{best,j}$
12:             Randomly generate the value of $r_2$, $r_3$, $r_4$
13:             Update the position of $X_{i,j}$
14:         end for
15:     end for
16:     Binarization $X$ with action $a$ and apply reward function
17:     Calculate Population Diversity (X) using equation (6)
18:     Define the next state using equation (8)
19:     Update Q-Table using SARSA equation (4)
20:     Update $r_1$
21:     Update $X_{best}$
22: end for
23: Return the updated population X where $X_{best}$ is the best result
```

proposal does not have similarities to the graphs presented by them, when observing the results obtained it is determined that they are not random algorithms since they present variations in their exploration percentages.

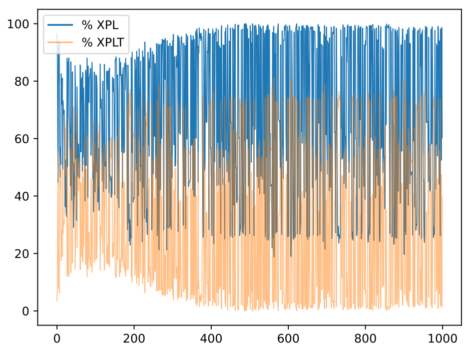


Fig. 1. SCP - Exploration and Exploitation Graphic of instance 4.7 version BS-SCA

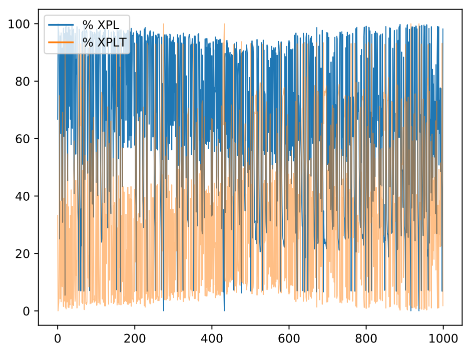


Fig. 2. SCP - Exploration and Exploitation Graphic of instance 4.7 version BQ-SCA

6 Conclusion

The results are encouraging because the performance of a binarization selector reduces tuning times by not having to assess the combinations of various binarization plans in the literature.

The application of learning techniques to metaheuristics has made a significant contribution to both the improvement of fitness and the attainment of a better exploration-exploitation balance. SARSA has been used as a binarization selector in the BS-SCA algorithm resolving 45 instances of the OR-Library of the Set Covering Problem. When compared to its more closely related BQ-SCA version, it has been demonstrated that BS-SCA achieves higher quality results on all the instances, and that these results are statistically significant better under the Wilcoxon-Mann-Whitney test.

On the other hand, it was observed that, when observing in detail the graphs used for exploration and exploitation there is a similar convergence, but with BS-SCA the values tend to have smaller magnitude changes, with greater occurrence, and much more defined which may indicate that they are more efficient in solving this problem.

As future works, in addition to the implementation of SARSA in other metaheuristic techniques, we should parameterize the results obtained by exploration-exploitation graphs. Although it provides valuable and useful information about the search process, it still needs a comparative metric and in turn, that this same metric can be incorporated in the agent learning process.

7 Acknowledgements

Broderick Crawford is supported by Grant CONICYT/FONDECYT/REGULAR/1210810. Ricardo Soto is supported by Grant CONICYT/FONDECYT/REGULAR/1190129. José Lemus-Romani is supported by National Agency for Research and Development (ANID)/Scholarship Program/DOCTORADO NACIONAL/2019-21191692. Marcelo Becerra-Rozas is is supported by National Agency for Research and Development (ANID)/Scholarship Program/DOCTORADO NACIONAL/2021-21210740.

References

1. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE transactions on evolutionary computation **1** (1997) 67–82
2. Wolpert, D.H., Macready, W.G.: Coevolutionary free lunches. IEEE Transactions on evolutionary computation **9** (2005) 721–735
3. Voss, S., Maniezzo, V., Stützle, T.: Matheuristics: Hybridizing metaheuristics and mathematical programming (annals of information systems). (2009)
4. Juan, A.A., Faulin, J., Grasman, S.E., Rabe, M., Figueira, G.: A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. Operations Research Perspectives **2** (2015) 62–72
5. Talbi, E.G.: Combining metaheuristics with mathematical programming, constraint programming and machine learning. Annals of Operations Research **240** (2016) 171–215
6. Talbi, E.G.: Machine learning into metaheuristics: A survey and taxonomy of data-driven metaheuristics. (2020)
7. Song, H., Triguero, I., Özcan, E.: A review on the self and dual interactions between machine learning and optimisation. Progress in Artificial Intelligence **8** (2019) 143–165
8. Calvet, L., de Armas, J., Masip, D., Juan, A.A.: Learnheuristics: hybridizing metaheuristics with machine learning for optimization with dynamic inputs. Open Mathematics **15** (2017) 261–280
9. Cisternas-Caneo, F., Crawford, B., Soto, R., de la Fuente-Mella, H., Tapia, D., Lemus-Romani, J., Castillo, M., Becerra-Rozas, M., Paredes, F., Misra, S.: A data-driven dynamic discretization framework to solve combinatorial problems using continuous metaheuristics. In: Innovations in Bio-Inspired Computing and Applications, Cham, Springer International Publishing (2021) 76–85
10. García, J., Moraga, P., Valenzuela, M., Crawford, B., Soto, R., Pinto, H., Peña, A., Altimiras, F., Astorga, G.: A db-scan binarization algorithm applied to matrix covering problems. Computational intelligence and neuroscience **2019** (2019)
11. de León, A.D., Lalla-Ruiz, E., Melián-Batista, B., Moreno-Vega, J.M.: A machine learning-based system for berth scheduling at bulk terminals. Expert Systems with Applications **87** (2017) 170–182
12. Sutton, R.: Advances in neural information processing systems: Vol. 8. generalization in reinforcement learning: Successful examples using sparse coarse coding (1996)
13. Sutton, R.S.: Learning to predict by the methods of temporal differences. Machine learning **3** (1988) 9–44
14. Taylor, M.E., Stone, P., Liu, Y.: Transfer learning via inter-task mappings for temporal difference learning. Journal of Machine Learning Research **8** (2007)
15. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT press (2018)
16. Tapia, D., Crawford, B., Soto, R., Palma, W., Lemus-Romani, J., Cisternas-Caneo, F., Castillo, M., Becerra-Rozas, M., Paredes, F., Misra, S.: Embedding q-learning in the selection of metaheuristic operators: The enhanced binary grey wolf optimizer case. In: 2021 IEEE International Conference on Automation/XXIV Congress of the Chilean Association of Automatic Control (ICA-ACCA). (2021) 1–6

17. Xu, Y., Pi, D.: A reinforcement learning-based communication topology in particle swarm optimization. Neural Computing and Applications (2019) 1–26
18. Mirjalili, S.: Sca: a sine cosine algorithm for solving optimization problems. Knowledge-based systems **96** (2016) 120–133
19. kai Feng, Z., Liu, S., jing Niu, W., jian Li, B., chuan Wang, W., Luo, B., min Miao, S.: A modified sine cosine algorithm for accurate global optimization of numerical functions and multiple hydropower reservoirs operation. Knowledge-Based Systems **208** (2020) 106461
20. Crawford, B., Soto, R., Cisternas-Caneo, F., Tapia, D., de la Fuente-Mella, H., Palma, W., Lemus-Romani, J., Castillo, M., Becerra-Rozas, M.: A comparison of learnheuristics using different reward functions to solve the set covering problem. In: International Conference on Optimization and Learning, OLA 2021. (2021) ARTICLE IN PRESS
21. Crawford, B., León de la Barra, C.: Los algoritmos ambidiestros. https://www.mercuriovalpo.cl/impresa/2020/07/13/full/cuerpo-principal/15/ (2020) Acceded 12-02-2021.
22. Lemus-Romani, J., Crawford, B., Soto, R., Astorga, G., Misra, S., Crawford, K., Foschino, G., Salas-Fernández, A., Paredes, F.: Ambidextrous socio-cultural algorithms. In: International Conference on Computational Science and Its Applications, Springer (2020) 923–938
23. Crawford, B., Soto, R., Astorga, G., García, J., Castro, C., Paredes, F.: Putting continuous metaheuristics to work in binary search spaces. Complexity **2017** (2017)
24. Hussain, K., Zhu, W., Salleh, M.N.M.: Long-term memory harris' hawk optimization for high dimensional and optimal power flow problems. IEEE Access **7** (2019) 147596–147616
25. Morales-Castañeda, B., Zaldivar, D., Cuevas, E., Fausto, F., Rodríguez, A.: A better balance in metaheuristic algorithms: Does it exist? Swarm and Evolutionary Computation (2020) 100671
26. Mann, H.B., Whitney, D.R.: On a test of whether one of two random variables is stochastically larger than the other. The annals of mathematical statistics (1947) 50–60
27. Lanza-Gutierrez, J.M., Crawford, B., Soto, R., Berrios, N., Gomez-Pulido, J.A., Paredes, F.: Analyzing the effects of binarization techniques when solving the set covering problem through swarm optimization. Expert Systems with Applications **70** (2017) 67–82
28. Bisong, E.: Google colaboratory. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform. Springer (2019) 59–64

Minimum Rule-Repair Algorithm for Supervised Learning Classifier Systems on Real-valued Classification Tasks

Koki Hamasaki[1] and Masaya Nakata[1]

Department of Electrical Engineering and Computer Science, Yokohama National University, Japan

1 Introduction

Learning Classifier Systems [14] (LCSs) are a paradigm of evolutionary rule-based learning methods. LCSs intend to produce accurate, maximally general, and thus explainable rules [15, 8]. Relying on this advantage, many works have applied LCSs to data-mining tasks [5, 6, 21]. Technically, LCSs are designed to generate a minimal rule-set that determines plausible outputs for given inputs. Thus, each rule should accurately predict an output while covering as many inputs as possible. For this purpose, rule-based learning evaluates a rule-fitness through interaction with an environment, and then evolutionary computation, e.g., GA, generatively refines rules with fitness guidance.

While many branches of LCSs have been proposed thus far [22, 26], most works have extended either one of the two basic LCSs: the XCS classifier system [28] and the UCS classifier system [4]. XCS is based on a reinforcement learning (RL) approach, and thus it is suitable for RL problem domains, e.g., online-control [23]. XCS evaluates the rule-fitness with reward signals. In contrast, UCS is an extension of XCS, and it uses a supervised learning approach, where both an input and the correct output for it are sent to the system. Thus, UCS can be suitable for supervised learning tasks, e.g., classification [4, 12]. Note that XCS and UCS commonly use the steady-state GA as a rule-evolution scheme. Consider the LCS's advantage aforementioned, this paper studies UCS as a data-mining tool for classification.

However, a restriction of LCSs (including UCS) is in less scalability of the system performance against the input space size. For instance, the LCS performance significantly degrades when dealing with high-dimensional and/or real-valued inputs [20, 17, 24]. To tackle this issue, Debie provided a theoretical insight for UCS on high-dimensional problems [10]. He also proposed an ensemble learning scheme of UCS to boost the performance on real-valued classification tasks [11]. Urabanowicz introduced some heuristics for UCS to improve the efficiency of rule-evolution (i.e., ExSTraCS) [27]. ExSTraCS successfully solves the 135-bit multiplexer problem with binary inputs. Some modern works revealed the impacts of the lexicase selection [3] and the fine-tuning for hyper-parameters [18] on the UCS framework. In addition, dimensional reduction techniques were used in XCS, e.g., feature selection [1, 2] and deep auto-encoder approaches [17, 24]; those approaches can be extended to the UCS framework.

Although various extensions of UCS have been considered as aforementioned, there are very few works that intend to repair inaccurate rules for the real-valued UCS framework. A possible reason is that UCS is originally designed to evolve only accurate rules to construct a *best action map* [4]. However, as another critical reason common to XCS, UCS should be designed to maintain a low frequency of the rule-production to avoid a problematic cover-delete cycle [9, 7]. For instance, the steady-state GA produces only two offspring rules per generation, which results in a fundamental inefficiency of UCS. Here, the cover-delete cycle is one of the major difficulties to design online learning-based LCSs, meaning that insufficiently-trained rules may be deleted due to a high frequency of rule-production; and this cycle may frequently occur when each rule is less general under a limited population size. Thus, a rule-repair strategy can be a possible reason to provoke the cover-delete cycle.

Note that there are some rule-repair algorithms for XCS. In [16], Lanzi proposed a *specify* operator for XCS with binary-input problems. His concept is to repair inaccurate rules identified by the XCS's reinforcement learning scheme. In detail, the specify operator replaces some don't care bits involved in a rule-condition with specific values of the input. This operator was applied to the UCS framework [4]. In [13], Iqbal presented a GP-based XCS and its rule-repair algorithm for GP-based rule expression. Tadokoro introduced a local covering operator [25]. While this operator does not intend to repair inaccurate rules, it produces new initial rules with a similar concept to the specify operator. Although those related works show the effectiveness of rule-repair algorithms, they have not been designed for UCS with real-valued inputs.

Accordingly, this paper presents a minimum rule-repair algorithm for UCS with real-valued inputs on classification problems. Our rule-repair algorithm intends to improve the performance by 1) boosting the classification accuracy (i.e., the rule-fitness) of inaccurate rules and by 2) increasing the frequency of rule-reproductions. Besides, we design our algorithm based on a minimum rule-repair concept to avoid the problematic cover-delete cycle. That is, our algorithm repairs the rule-condition with the minimum reduction of its rule-generality; the rule-condition is repaired so that it excludes one incorrect input from a subspace covered by its rule-condition.

This paper is organized as follows. Section 2 describes the UCS framework for real-valued inputs. Section 3 introduces our rule-repair algorithm. Section 4 tests UCS with our rule-repair algorithm on real-valued benchmark classification problems. Section 5 empirically validates our hypothetical insights. Finally, in Section 6, we summarize our contributions with future directions.

2 UCS for real-valued inputs

This section gives a description of the UCS framework for real-valued inputs $\boldsymbol{x} = [x_1, x_2, \cdots, x_d]$, where d is the problem dimension and $x_i \in [0,1] (i = \{1, 2, \cdots, d\})$. This paper employs a lower-upper representation as a rule-condition for real-valued inputs [30]. The rule-condition with this coding represents a d-dimension hyperrectangle as its matching sub-space on the input space; and thus its rule-generality can be measured with a volume of its hyperrectangle. Note that this paper denotes a uniformly-sampled random value as r, and $r \in [0, 1]$, if not stated differently; and all rs used in equations are independently sampled. Note also that we introduce new mathematical notations for the UCS framework, which is exactly the same as in the original working of UCS [4].

2.1 Rule parameters

A rule cl consists of a condition $C = \{c_1, c_2, \cdots c_d\}$ and an action A, where a sub-condition c_i involves a lower l_i and an upper u_i both used for x_i, i.e., $c_i = [l_i, u_i]$ ($l_i \leq u_i$, $l_i, u_i \in [0, 1]$). A rule cl can be matched to $\boldsymbol{x}$ if and only if $l_i \leq x_i \leq u_i, \forall i \in \{1, 2, \cdots, d\}$, simply denoted by $\boldsymbol{x} \in C$ in this paper. The action A represents a class when its rule is executed.

The rule cl also has the following five main parameters; the number of correct classification $ct \in \mathbb{N}_0$, which represents how many times cl belongs to $[C]$; the accuracy $acc \in [0, 1]$, which is a classification accuracy of cl; the rule-fitness $F \in [0, 1]$, which is calculated from acc; the experience $exp \in \mathbb{N}_0$, which represents the number of parameter-update times; the numerosity $num \in \mathbb{N}_0$; which is the number of subsumed rules to cl by a subsumption operator (see Section 2.2).

Suppose two rules cl_1 and cl_2 both having the same action, cl_1 can be more general than cl_2 if and only if $l_{1,i} \leq l_{2,i} \wedge u_{2,i} \leq u_{1,i}, \forall i \in \{1, 2, \cdots, d\}$, simply denoted by $cl_2.C \subset cl_1.C$ in this paper. All rules are contained in a population $[P]$ with the maximum population size N.

2.2 Framework

The UCS framework is composed of the training phase and the test phase. During training, UCS activates rule-parameter updates and the steady-state GA in order to produce the optimal rule-set as a solution. During the test phase, it only determines an output based on the trained rule-set.

Training phase.

At the initial iteration $t = 0$, UCS builds the population $[P]$ as an empty set. For $t \leftarrow t+1$, UCS receives an input $\boldsymbol{x}$ together with its correct class A^*. Then, it builds a match set $[M]$ consisting rules matched to $\boldsymbol{x}$, given by;

$$[M] = \{cl \in [P] \mid \boldsymbol{x} \in cl.C\}. \tag{1}$$

Then, UCS further builds a correct set $[C]$ and an incorrect set $[!C]$, given by;

$$\begin{cases} [C] &= \{cl \in [M] \mid cl.A = A^*\} \ , \\ [!C] &= \{cl \in [M] \mid cl.A \neq A^*\} \ . \end{cases} \tag{2}$$

Thus, $[C]$ and $[!C]$ are composed of (temporarily) accurate rules and inaccurate rules, respectively. If $[C]$ is empty, the covering operator takes place to produce a new rule with an initial setting $\{A = A^*, ct = 0, exp = 0, F = 0.01, num = 1\}$; and l_i and u_i for $c_i \in C$ are initialized as;

$$\begin{cases} l_i & = x_i - r \cdot s_0 \ , \\ u_i & = x_i + r \cdot s_0 \ , \end{cases} \tag{3}$$

where $s_0 \in [0, 1]$ is a hyperparameter that controls the initial rule-generality. Thus, $\boldsymbol{x} \in C$ is always satisfied. Note that $0 \leq l_i \leq u_i \leq 1$.

Next, UCS updates rule parameters. First, ct is updated as $ct \leftarrow ct + 1$ for each rule in $[C]$. Next, for all rules in $[M]$, exp, acc, and F are updated. In detail, exp is updated as $exp \leftarrow exp + 1$ to count the update time; then, acc is updated by;

$$acc = \frac{ct}{exp} \ . \tag{4}$$

Thus, acc represents the classification accuracy of cl. Then, the fitness F is updated with exponential reduction, given by;

$$F = (acc)^{\nu} \ , \tag{5}$$

where ν controls a selection bias in the steady-state GA.

Finally, the steady-state GA is applied to $[C]$ to generate plausibly better rules. First, UCS selects two parent rules from $[C]$; and it produces two offspring rules cl_1, cl_2 as copies of the corresponding parent rules except for $\{ct = 0, exp = 0, F = 0.01, num = 1\}$. Then, a crossover operator is activated with a probability χ; and this paper employs the uniform crossover. If the crossover is activated, it may swap $c_{1,i} \in cl_1.C$ for $c_{2,i} \in cl_2.C$ with a probability 0.5 for each $i = \{1, 2, \cdots, d\}$. Next, the mutation operator is also applied to each sub-condition of $cl_*.C$ (i.e., $c_{*,i}$, with a probability μ ($*$ can be 1 and 2)). In detail, $l_{*,i}$ and $u_{*,i}$ for $c_{*,i}$ may be mutated as;

$$l_{*,i} \leftarrow \begin{cases} l_{*,i} - r \cdot m_0 & r < 0.5 \ , \\ l_{*,i} + r \cdot m_0 & \text{otherwise} \ , \end{cases} \tag{6}$$

$$u_{*,i} \leftarrow \begin{cases} u_{*,i} - r \cdot m_0 & r < 0.5 \ , \\ u_{*,i} + r \cdot m_0 & \text{otherwise} \ , \end{cases} \tag{7}$$

where m_0 is a hyperparameter that controls a degree of the rule-generality; again, $0 \leq l_{*,i} \leq u_{*,i} \leq 1$. Then, two offspring rules are inserted to $[P]$; and rules may be deleted if the population size $|[P]|$ exceeds N. This paper uses the tournament selection with a tournament size τ.

A subsumption operator may be applied to the rules in $[C]$ after updating the rule parameters or to offspring rules after the steady-state GA. A rule can be subsumed by a more general rule than it, provided that the more general rule is reliably accurate and sufficiently updated (i.e., $acc > acc_0 \wedge exp > \theta_{sub}$); $acc_0 \in [0, 1]$ defines the minimum classification accuracy the maximally accurate rules must have; and $\theta_{sub} \in \mathbb{N}$ defines the minimum update time. In detain, for each rule cl in $[C]$ except for maximally accurate, maximally general rules cl^*, cl^* subsumes cl if $cl.C \subset cl^*.C \wedge cl.A = cl^*.A$; then, the numerosity of cl^* is updated as $cl^*.num \leftarrow cl^*.num + cl.num$, and cl is deleted from $[P]$.

Test phase.

For a given input $\boldsymbol{x}$, UCS builds the match set $[M]$, where all actions a existed in $[M]$ are contained in $[A_M]$. Then, for each action $a \in [A_M]$, it builds a subset $[M_a]$ which consists of rules having the action a, given by $[M_a] = \{cl \in [M] \mid cl.A = a\}$. Finally, it outputs the best action A' having the highest fitness, that is,

$$A' = \arg\max_{a \in [A_M]} \sum_{cl \in [M_a]} cl.F \ . \tag{8}$$

3 Minimum rule-repair algorithm

In this section, we first introduce our concept of the minimum rule-repair algorithm. Then, the detailed algorithm is described.

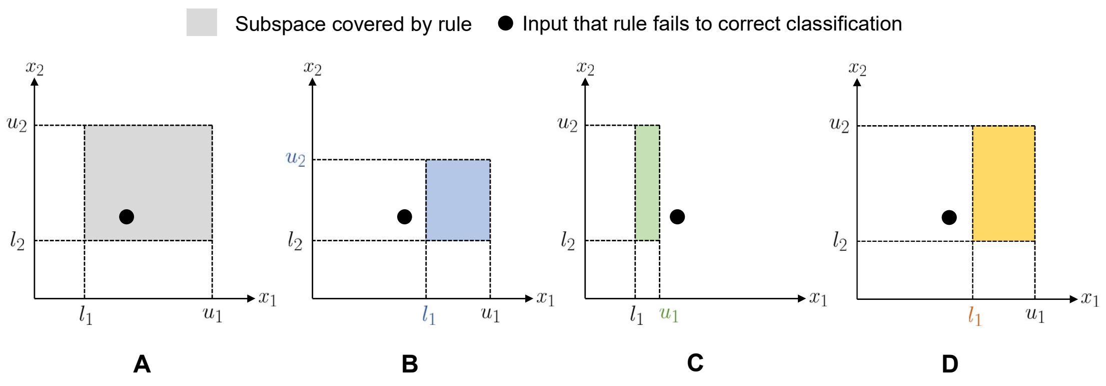


Fig. 1. Examples of possible repair patterns of the rule-condition to eliminate a misclassified input (denoted by the black dot). "A" represents the original subspace covered by a rule; "B", "C", and "D" represent repaired subspaces by repairing $\{l_1, u_2\}$, u_1, and l_1, respectively. "D" can have the hightest rule-generality in those examples.

3.1 Concept

As described in the previous section, given $\boldsymbol{x}$ at iteration t, the UCS framework does not intend to utilize rules temporarily identified as inaccurate,(i.e., $cl' \in [!C]$). However, such inaccurate rules may contribute to the correct classification for other matched inputs. Consider cl' misclassifies some inputs $\boldsymbol{x}'$, its classification accuracy ($cl'.acc$), certainly improves if $cl'.C$ is repaired to match inputs except for $\boldsymbol{x}'$s. Thus, acc tends to improve by reducing the rule-generality.

However, this strategy(i.e., to reduce the rule-generality), provokes the problematic cover-delete cycle under a restricted population size, as noted in Section 1. Thus, we here consider a conservative approach to design our rule-repair algorithm. Our algorithm is designed to repair cl''s rule-condition with a possible minimum reduction of the rule-generality. Specifically, given $\boldsymbol{x}$ at t, we repair $cl'.C$ to eliminate $\boldsymbol{x}$ from its matching sub-space represented by its rule-condition. That is, we target one single input for each repair to avoid a drastic reduction of the rule-generality. In this case, we can still consider various repair patterns of the rule-condition. Fig. 1 shows possible examples of repair patterns on two-dimensional inputs $\boldsymbol{x} = [x_1, x_2]$; As shown in this figure, we can suppose possible repair patters ("B", "C", and "D"); however, "D" can have the highest rule-generality in those patterns. Technically, we do not need to repair both l_i and u_i and/or more than one dimension x_i to achieve the minimum reduction of the rule-generality.

Thus, our minimum rule-repair algorithm is designed to satisfy the following two conditions; 1) to repair the rule-condition to eliminate $\boldsymbol{x}$ from its matching sub-space, and 2) to repair either l_i or u_i for one dimension x_i.

3.2 Algorithm

Our rule-repair algorithm is activated after the rule-parameter update in the UCS framework. First, we add a new rule-parameter $rt \in \mathbb{N}_0$, which denotes the latest update time its rule was repaired. The initial value of rt is set to 0 when a rule is generated by the covering operator or the steady-state GA. Then, it selects and then repairs only sufficiently-updated rules. In detail, UCS with our algorithm builds a repair set $[R]$, given by;

$$[R] = \{cl \in [!C] \mid cl.exp - cl.rt > \theta_{sub}\} . \tag{9}$$

Thus, once a rule is repaired, we temporarily remove its rule from candidates for repair until its rule-quality is estimated trustworthy. In other words, this boosts a stable convergence of rule-parameters, and it also prevents a drastic reduction of the rule-generality.

Next, for each $cl \in [R]$, we randomly select a dimension index k from $[1, 2, \cdots, d]$ to decide a target sub-condition c_k for repair; and then it generates new upper/lower candidates $\hat{l}_k, \hat{u}_k$ as;

$$\begin{cases} \hat{l}_k & = x_k + D \ , \\ \hat{u}_k & = x_k - D \ , \end{cases} \tag{10}$$

where a hyperparameter $D \in [0, 1]$ controls the minimum distance between a specific input value x_k and $\hat{l}_k/\hat{u}_k$. Thus, cl's rule condition can be guaranteed that it does not match $\boldsymbol{x}$ at t when either $l_k = \hat{l}_k$ or $u_k = \hat{u}_k$. Note that D should be set to a relatively small value, e.g., $D \leq 0.01$, to maintain a small reduction of the rule-generality; in Section 5, we empirically reveal the impact of D. Next, we further select either $\hat{l}_k$ or $\hat{u}_k$ to maintain the rule-generality as possible. Since the other sub-conditions $c_i (i \neq k)$ are not repaired, a volume of the hyperrectangle specified by $cl.C$ changes dependent on the length of c_k (i.e., $u_k - l_k$). Thus, to maximize the rule-generality, it is sufficient that we can use the candidate maximizing the length of c_k, that is;

$$\begin{cases} l_k \leftarrow \hat{l}_k & \text{if } u_k - \hat{l}_k > \hat{u}_k - l_k \ , \\ u_k \leftarrow \hat{u}_k & \text{otherwise} \ . \end{cases} \tag{11}$$

Note that as an exceptional case, we forcedly set u_k to $\hat{u}_k$ if $\hat{l}_k > u_k$, and vice versa; if $\hat{l}_k > u_k$ and $\hat{u}_k < l_k$, we skip to repair the rule-condition, but this is the extremely rare case in our experiments. Finally, rt is updated as $rt \leftarrow exp$. Algorithm 1 shows the pseudo-code of our algorithm.

Algorithm 1 Minimum rule-repair algorithm

```
1: Input: $\boldsymbol{x}$, $[!C]$
2: for each $cl \in [!C]$ do
3:     if $cl.exp - cl.rt > \theta_{sub}$ then
4:         $k \leftarrow$ randomly sampled from $[1, 2, \cdots, d]$
5:         $\hat{l}_k = x_k + D$
6:         $\hat{u}_k = x_k - D$
7:         if $u_k - \hat{l}_k > \hat{u}_k - l_k$ then
8:             $l_k \leftarrow \hat{l}_k$
9:         else
10:            $u_k \leftarrow \hat{u}_k$
11:        $cl.rt \leftarrow cl.exp$
```

4 Experiment

This section tests our rule-repair algorithm on two real-valued benchmark classification tasks: the real-valued multiplexer problem (RMUX) and the real-valued majority-on problem (RMOP).

4.1 Benchmark problems

Real-valued multiplexer problem.

The n-bit multiplexer problem (n-MUX) is originally used as a binary input classification problem to validate the generalization capacity of XCS [28]. It has been extended for real-valued classification tasks [29, 20]. The n-MUX is defined over a binary string of $n = k + 2^k$; a decimal number of the first k bits represents a position of one of the remaining 2^k bits. Then, the correct class is the bit pointed to by the first k bits. In n-RMUX, each attribute x_i is binarized at the common boundary 0.5(i.e., 0 if $x_i < 0.5$ or 1 if $x_i \geq 0.5$) and then a correct class of the real-valued input is determined with the same procedure of the n-MUX. Note that, the binarization boundary is not a critical factor dependent on the problem difficulty, as the steady-state GA is designed to independently change the sub-condition c_i based on only its corresponding attribute. This paper uses $\{6, 11, 20, 37\}$-RMUXs to evaluate the scalability of the algorithm.

Real-valued majority-on problem.

The n-bit majority-on problem (n-MOP) is also originally defined with binary inputs [13]. In n-MOP, if the number of “1” exceeds the number of “0”, the correct class is 1; otherwise 0. This paper extends n-MOP to a real-valued classification task; each attribute x_i is also binarized at the common boundary 0.5. The n-MOP and n-RMOP are highly overlapping problems, where LCSs often suffer to improve the performance [13]. This paper uses $\{11, 15\}$-RMOPs.

4.2 Experimental settings

We employ the following experimental paradigm. One iteration involves a set of one training input and one test input of the problem. For the training input, UCS activates the training phase to produce the solution. For the test input, it activates only the test phase in order to evaluate the system classification accuracy as the UCS performance. In addition, we evaluate the population size (i.e., the number of rules in $[P]$) to evaluate the generalization capacity of UCS. The UCS performance and the population size are reported as an average of 30 trials. We test UCS and UCS with our rule-repair algorithm (denoted by “Ours”).

We use the following UCS parameter settings with respect to [20,4]; $\beta = 0.2$, $\delta = 0.1$, $\nu = 10$, $\theta_{GA} = 25$, $\chi = 0.8$, $\theta_{\text{del}} = 20$, $\theta_{\text{sub}} = 20$, $acc_0 = 0.99$, $P_{\#} = 0.8$, $\mu = 0.04, \tau = 0.4$, $s_0 = 1.0$, and $m_0 = 0.1$. The subsumptions are turned on. For $\{6, 11, 20, 37\}$-RMUXs and $\{11, 15\}$-RMOPs, we set N and the maximum iterations to $\{800, 5000, 30000, 30000, 10000, 100000\}$ and $\{50000, 200000, 1000000, 1500000, 200000, 500000\}$, respectively. For our repair algoritm, we set $D = 0.01$.

4.3 Result

Figs. 2 and 3 summarize the performances and the population size. As shown in Fig.2, our rule-repair algorithm successfully boosts the UCS performance on all the problems employed in this paper. Specifically, our rule-repair algorithm improves the performance at early iterations, where we can expect that many inaccurate rules exist in $[!C]$. For instance, it reaches almost the optimal performance after 100,000 iterations on 11-RMUX, but UCS requires 200,000 iterations to reach it. The performances on RMOPs do not reach the optimal performance due to the complexity of the overlapping problem, which is a similar tendency to existing works [13, 19].

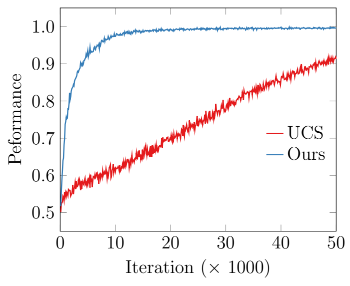
A possible drawback of our rule-repair algorithm is to increase the population size since it enhances a bias to produce specific rules with less rule-generality. As shown in Fig. 3, this insight can be observed in early generations. However, our rule-repair algorithm successfully prevents the increase of the population size over iterations. Besides, it produces a more compact population than that of UCS on 6, 11-RMUXs. Note that the population size reaches the maximum population size N if the cover-delete cycle occurs. This tendency can be observed for both UCSs on RMOPs, and so our rule-repair algorithm itself does not provoke the cover-delete cycle. We suspect that N should be further increased to cover various niches defined in RMOPSs [19]. Thus, we can empirically confirm that our rule-repair algorithm successfully prevents the cover-delete cycle.

Finally, we further give empirical insights to confirm the efficiency of our rule-repair algorithm. Fig. 4 shows the summation of numerosity of rules in $[!C]$ over iterations(i.e., $\sum_{cl \in [!C]} cl.num$), on 11-RMUX and 11-RMOP. Note that the population involves many inaccurate rules if $\sum_{cl \in [!C]} cl.num$ is a large value since inaccurate rules tend to be over-generalized; and so those rules frequently match inputs, resulting in the increase of the $\sum_{cl \in [!C]} cl.num$. From the figure, it is obvious that our rule-repair algorithm contributes to decreasing the summation of numerosity of $[!C]$. This means that our rule-repair algorithm successfully reduces the inaccurate rules by improving their classification accuracy.

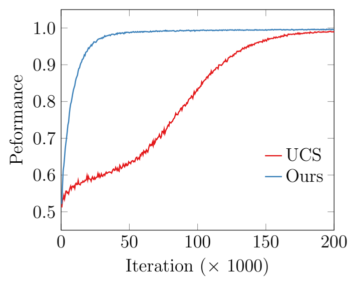
In summary, our rule-repair algorithm successfully repairs the inaccurate rules, and thus it boosts the UCS performance while preventing the increase of the population size as well as the cover-delete cycle.

5 Analysis

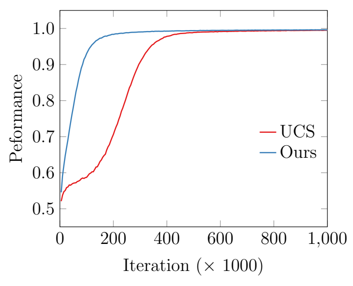
This section presents analytical insights into our minimum rule-repair strategy.



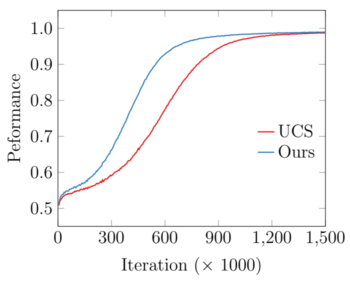
a) 6-RMUX



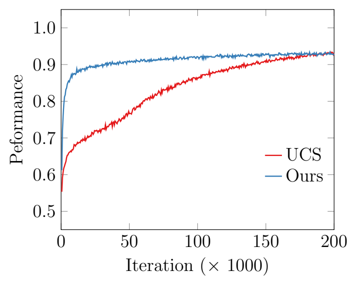
b) 11-RMUX



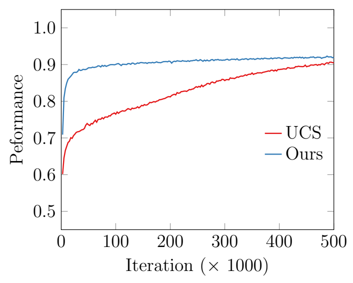
c) 20-RMUX



d) 37-RMUX



e) 11-RMOP



f) 15-RMOP

Fig. 2. The performances.

5.1 Analysis of the number of sub-conditions to be repaired.

One of our strategy to minimize the reduction of the rule-generality is to repair either l_i or u_i for *one* sub-condition c_i. In this subsection, we validate the impact of this strategy. In detail, we here extend our algorithm to repair Ψ sub-conditions at the same time, where Ψ is the number of sub-conditions to be repaired; and a default setting is $\Psi = 1$. More specifically, we insert a "`for`" loop (i.e., `for` $i = 1$ `to` Ψ) between `line 3` and `4` in Algorithm 1; and each dimension index `k` is randomly sampled with no duplicates.

We here compare the performances of UCS with the rule-repair algorithm for $\Psi = \{1, 3, 5\}$. Note that the rule-generality of repaired rules should be rapidly decreasing with the increase of Ψ. Figs. 5 and 6 show the performances and the population size on 11, 20, 37-RMUXs with the same experimental settings (see Subsection 4.2), respectively. As shown in those figures, the performance gradually degrades with the increase of Ψ. This tendency is clearly highlighted with the increase of the problem dimensions $d = \{11, 20, 37\}$. Besides, the population size with $\Psi = 1$ decreases slightly faster than $\Psi = \{3, 5\}$. This indicates that our minimum rule-repair strategy (i.e., $\Psi = 1$) successfully discovers the optimum rules faster than the other settings in this paper. Accordingly, those experimental results empirically confirm our hypothetical insight; the rule-condition should be repaired with as minimum reductions of the rule generality as possible.

5.2 Analysis of the minimum distance D.

Next, we analyze the impact of the hyperparameter D, which controls the minimum distance between a specific input value x_k and $\hat{l}_k$ (or $\hat{u}_k$). As noted in Section 3, the reduction of the rule-generality can be also minimized in terms of D, e.g., $D = 10^{-10}$. Thus, according to our hypothesis (i.e., to minimize the reduction of the rule-generality), D may be an important parameter dependent on the performance.

We here compare the performances with $D = \{0.001, 0.01, 0.1\}$. Fig. 7 shows the performances on 11, 20, 37-RMUXs with the same experimental settings as in Section 4.2. Note that the UCS performance of 37-RMUX is reported as an average of 10 trials due to the increase of the computation time. As shown in this figure, our rule-repair algorithm with $D = 0.1$ very slightly improves the performance; however, compared to the impact of Ψ, no significant impact of D is observed even with the increase of the problem dimensions. This is because that D does not significantly

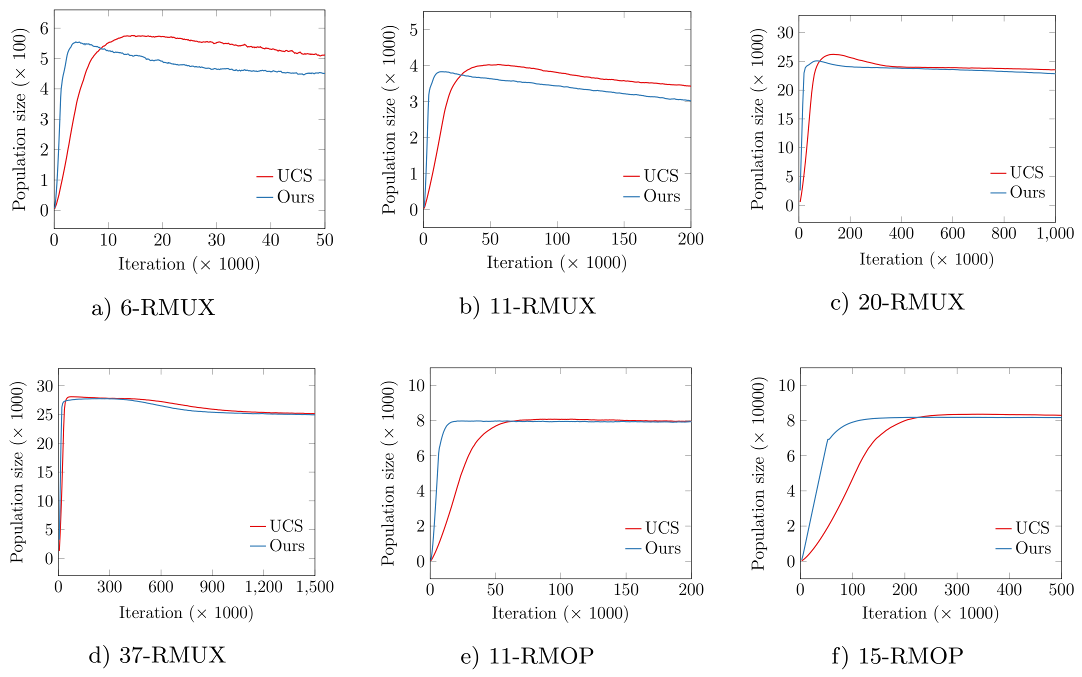


Fig. 3. The population sizes.

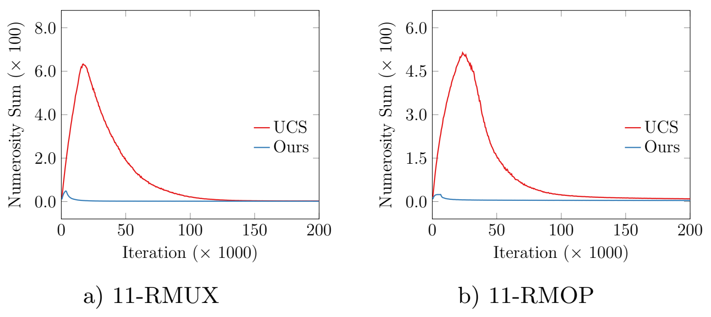
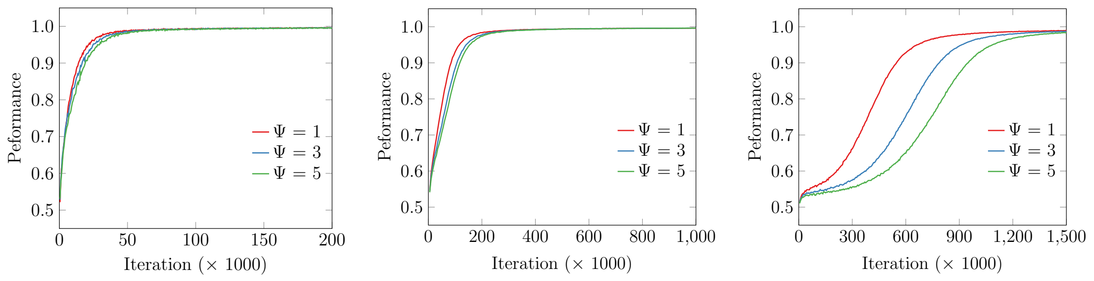


Fig. 4. The summation of numerosity of rules in [!C].

change a matching probability of the rule-condition to an input. For instance, suppose 11-MUX and a rule-condition C with $\{l_i = 0.1, u_i = 0.9\}, \forall i = \{1, 2, \cdots, 11\}$, its matching probability can be $0.0859 (= (0.9 - 0.1)^{11})$ for randomly-sampled inputs under a uniform distribution. Then, consider that we repair l_1 with the first element of input $x_i = 0.4$, the sub-condition c_1 can be rewritten as $\{l_1 = 0.4 + D, u_1 = 0.9\} (i = 1)$. Then, the matching probability of C can be $\{0.0536, 0.0526, 0.0430\}$ for $D = \{0.001, 0.01, 0.1\}$, respectively. Consequently, we suppose that D cannot be an important parameter dependent on the performance unless it is set to an enough small value.

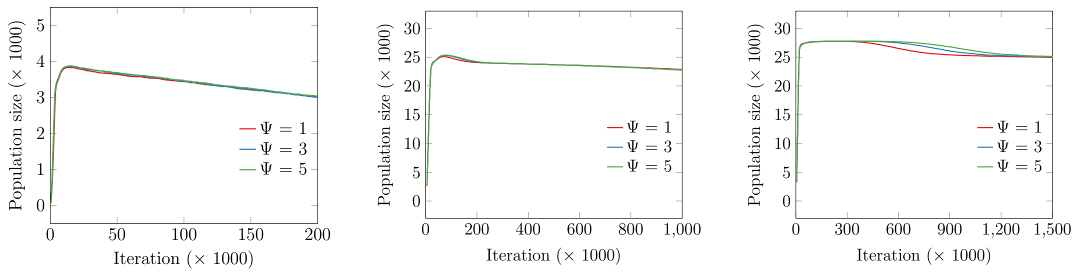
6 Conclusion

This paper proposed a minimum rule-repair algorithm for the UCS classifier system with real-valued inputs on classification problems. Our concept is to repair inaccurate rules with a possible minimum reduction of the rule-generality in order to avoid the problematic cover-delete cycle. Accordingly, we identified the following two principles to achieve this purpose; 1) to repair the rule-condition in order to eliminate one incorrect input from a matching subspace represented by its rule-condition, and 2) to repair either a lower value or an upper value for one dimension x_i. Experimental results confirmed the adequacy of those principles. Consequently, UCS with our rule-repair algorithm successfully boosts the performance while preventing the increase of the population size as well as the cover-delete cycle.



a) 11-RMUX b) 20-RMUX c) 37-RMUX

Fig. 5. The performances of UCS with the rule-repair algorithm with $\Psi = \{1, 3, 5\}$



a) 11-RMUX b) 20-RMUX c) 37-RMUX

Fig. 6. The population of UCS with the rule-repair algorithm with $\Psi = \{1, 3, 5\}$

In future work, we apply our algorithm to real-world classification tasks, where some difficulties, e.g., missing attribute and class imbalance, can be observed. We further analyze the impact of the hyperparameter D (the minimum distance) on non-uniformed distributions of inputs, aiming to reveal its optimal setting up guide.

References

1. Mani Abedini and Michael Kirley. Guided rule discovery in xcs for high-dimensional classification problems. In *Australasian Joint Conference on Artificial Intelligence*, pages 1–10. Springer, 2011.
2. Mani Abedini and Michael Kirley. An enhanced xcs rule discovery module using feature ranking. *International Journal of Machine Learning and Cybernetics*, 4(3):173–187, 2013.
3. Sneha Aenugu and Lee Spector. Lexicase selection in learning classifier systems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 356–364, 2019.
4. Ester Bernadó-Mansilla and Josep M Garrell-Guiu. Accuracy-based learning classifier systems: models, analysis and applications to classification tasks. *Evolutionary computation*, 11(3):209–238, 2003.
5. Keivan Borna, Shokoofeh Hoseini, and Mohammad Ali Mehdi Aghaei. Customer satisfaction prediction with Michigan-style learning classifier system. *SN Applied Sciences*, 1(11):1450, 2019.
6. Larry Bull, Ester Bernadó-Mansilla, and John H. Holmes, editors. *Learning Classifier Systems in Data Mining*, volume 125 of *Studies in Computational Intelligence*. Springer, 2008.
7. Martin V Butz, David E Goldberg, and Pier Luca Lanzi. Computational complexity of the xcs classifier system. In *Foundations of Learning Classifier Systems*, pages 91–125. Springer, 2005.
8. Martin V Butz, David E Goldberg, and Kurian Tharakunnel. Analysis and improvement of fitness exploitation in xcs: Bounding models, tournament selection, and bilateral accuracy. *Evolutionary computation*, 11(3):239–277, 2003.
9. Martin V Butz, Tim Kovacs, Pier Luca Lanzi, and Stewart W Wilson. Toward a theory of generalization and learning in xcs. *IEEE transactions on evolutionary computation*, 8(1):28–46, 2004.
10. Essam Debie and Kamran Shafi. Implications of the curse of dimensionality for supervised learning classifier systems: theoretical and empirical analyses. *Pattern Analysis and Applications*, 22(2):519–536, 2019.
11. Essam Debie, Kamran Shafi, Chris Lokan, and Kathryn Merrick. Reduct based ensemble of learning classifier system for real-valued classification problems. In *2013 IEEE Symposium on Computational Intelligence and Ensemble Learning (CIEL)*, pages 66–73. IEEE, 2013.

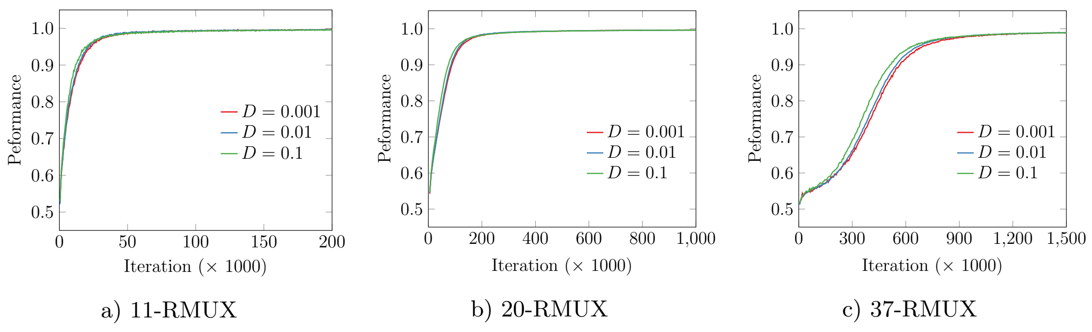


Fig. 7. The performances of UCS with the rule-repair algorithm with $D = \{0.001, 0.01, 0.1\}$

12. Toktam Ebadi, Mengjie Zhang, and Will Browne. Xcs-based versus ucs-based feature pattern classification system. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 839–846, 2012.
13. Muhammad Iqbal, Will N Browne, and Mengjie Zhang. Reusing building blocks of extracted knowledge to solve complex, large-scale boolean problems. *IEEE Transactions on Evolutionary Computation*, 18(4):465–480, 2013.
14. H John. Holland. escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. *Machine learning, an artificial intelligence approach*, 2:593–623, 1986.
15. Tim Kovacs. Evolving Optimal Populations with XCS Classifier Systems. Technical Report CSR-96-17 and CSRP-96-17, School of Computer Science, University of Birmingham, Birmingham, U.K., 1996.
16. Pier Luca Lanzi et al. A study of the generalization capabilities of xcs. In *ICGA*, pages 418–425. Citeseer, 1997.
17. Kazuma Matsumoto, Ryo Takano, Takato Tatsumi, Hiroyuki Sato, Tim Kovacs, and Keiki Takadama. XCSR based on compressed input by deep neural network for high dimensional data. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1418–1425, 2018.
18. Sotiris Moschoyiannis and Vasily Shcherbinin. Fine tuning run parameter values in rule-based machine learning. In *RuleML+ RR (Supplement)*, 2019.
19. Masaya Nakata, Will Browne, Tomoki Hamagami, and Keiki Takadama. Theoretical xcs parameter settings of learning accurate classifiers. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '17, page 473–480, New York, NY, USA, 2017. Association for Computing Machinery.
20. Masaya Nakata and Will N Browne. Learning optimality theory for accuracy-based learning classifier systems. *IEEE Transactions on Evolutionary Computation*, 2020.
21. Masaya Nakata and Kazuhisa Chiba. Design strategy generation for a sounding hybrid rocket via evolutionary rule-based data mining system. In *Intelligent and Evolutionary Systems*, pages 305–318. Springer, 2017.
22. David Pätzel, Anthony Stein, and Masaya Nakata. An overview of lcs research from iwlcs 2019 to 2020. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, GECCO '20, page 1782–1788, New York, NY, USA, 2020. Association for Computing Machinery.
23. M Roozegar, MJ Mahjoob, MJ Esfandyari, and M Shariat Panahi. Xcs-based reinforcement learning algorithm for motion planning of a spherical mobile robot. *Applied Intelligence*, 45(3):736–746, 2016.
24. Masakazu Tadokoro, Satoshi Hasegawa, Takato Tatsumi, Hiroyuki Sato, and Keiki Takadama. Knowledge Extraction from XCSR Based on Dimensionality Reduction and Deep Generative Models. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 1883–1890. IEEE, 2019.
25. Masakazu Tadokoro, Satoshi Hasegawa, Takato Tatsumi, Hiroyuki Sato, and Keiki Takadama. Local covering: Adaptive rule generation method using existing rules for xcs. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2020.
26. Ryan J. Urbanowicz and Jason H. Moore. Learning Classifier Systems: A Complete Introduction, Review, and Roadmap. *J. Artif. Evol. App.*, 2009:1:1–1:25, January 2009.
27. Ryan J Urbanowicz and Jason H Moore. Exstracs 2.0: description and evaluation of a scalable learning classifier system. *Evolutionary intelligence*, 8(2):89–116, 2015.
28. Stewart W Wilson. Classifier fitness based on accuracy. *Evolutionary computation*, 3(2):149–175, 1995.
29. Stewart W Wilson. Get real! xcs with continuous-valued inputs. In *International Workshop on Learning Classifier Systems*, pages 209–219. Springer, 1999.
30. Stewart W Wilson. Mining oblique data with xcs. In *International Workshop on Learning Classifier Systems*, pages 158–174. Springer, 2000.

MultiObjective Optimization Approach under uncertainty for Workforce Routing and Scheduling Problem in Home Health Care

Mariem BELHOR
Univ. Artois, LGI2A Laboratory,
Univ.Manouba, ENSI,SERCOM
Bethune, France
mariem_belhor@ens.univ-artois.fr

Adnen EL-AMRAOUI
Univ. Artois, LGI2A Laboratory
UR 3926, Technoparc FUTURA
Bethune, France
adnen.elamraoui@univ-artois.fr

François DELMOTTE
Univ. Artois, LGI2A Laboratory
UR 3926, Technoparc FUTURA
Bethune, France
francois.delmotte@univ-artois.fr

Abderrazak JEMAI
Univ.Carthage, EPT,SERCOM
Laboratory, INSAT, 1080
Tunis, Tunisia
Abderrazak.Jemai@insat.rnu.tn

Abstract— Home Health Care (HHC) is defined as visiting, assisting, nursing and delivering medical, nonmedical and paramedical services to patients. Transportation costs constitute one of the largest forms of expenditure in the HHC. Decision makers are inclined to optimize their activities in order to increase demand for HHC. They are confronted with multiple objectives to be optimized simultaneously as minimizing their operating costs while maximizing the satisfaction of their patients. A majority of the previous works consider only the deterministic models which ignore the uncertainties and solutions obtained by these deterministic models are usually less robust in case of any possible changes in practical situations. In this paper, we focused our attention on the management of workforce in Home Hemodialysis. Therefore, we propose a new Multi-objective model for Workforce Routing and Scheduling in Home Hemodialysis. The data of our model are deterministic except travel, service and start times are uncertain. Multi-objective optimization under uncertainty has gained considerable attention in recent years due to its practical applications in real-life. However, we apply two new Fuzzy Multi-objective Evolutionary Algorithms (FMOEAs) recently proposed in the literature to solve our model which are the Extended Non-dominated Sorting Genetic Algorithm II (E-NSGA-II) and the Extended Strength Pareto Evolutionary Algorithm (E-SPEA2). These algorithms are applied for the first time in the field of HHC. The obtained results show their effectiveness and suitability to our problem.

Keywords—Home Hemodialysis, Workforce Routing and Scheduling Problem, Multiple Salesman Travelling Problem, Multi-Objective Optimization, Uncertainty, Fuzzy Pareto Dominance, E-NSGAII, E-SPEA2

I. INTRODUCTION

Workforce scheduling appear in many spheres of life like hospitals, industry, public transportation, airlines companies, etc. [8]. Workforce scheduling in HHC has been reported as one of the largest single factors of financial, productivity and quality performance. The aim of HHC company is to provide the care (medical and paramedical services) to patients in their own homes [46] and to accord a formal assessment of their needs. This support is made by the staff like nurses, doctors, maintenance technicians, carriers, etc.

Home Hemodialysis is one of the most pathologies commonly treated in HHC. Home Hemodialysis consists in bringing to the patient's home a blood purification machine. This machine must be reliable. It must be regularly monitored and maintained by a maintenance technician, because we can't admit a risk for the patient who is connected to the machine 7 times a week. Every technician visit daily a number of patients located at different places. The costs generated by the maintenance process are very high.

The main objectives of HHC companies are the minimization of costs and maximization of customer satisfaction. Decision-makers have applied a lot of strategies to optimize the objectives of HHC companies. Most real optimization problems are described using several objectives that must be optimized simultaneously [35]. In this paper, a workforce scheduling and routing problem is formulated as a multi-objective model. Our problem presents several similarities with the well-known Multiple Travelling Salesman Problem (mTSP).

Travelling Salesman Problem (TSP) is Known as an NP-hard problem in combinatorial optimization, The importance of this problem is due to the fact that it is used in many fields such as transportation, logistics, problem of routing and many others scientific and industrial fields [43]. Solving a TSP consists on finding the shortest path to travel through a given number of cities. The traveler has to travel through all the given cities exactly once and return to the same city from where he started. Given, the distances between each pair of cities, what route should the salesman choose in order to minimize the total distance traveled? Instead of distance, other notions such as time, cost, etc., can be considered as well [43] [45].

In optimization problems, especially those arising from real problems, the values of certain parameters may be uncertain. This can come from a lack of information, prediction or estimation erros or a bad analysis of some data [6]. In our model, travel and service times are uncertain due to transport congestion, difficulty to identify the fault, etc. The proposed model aims to optimize two objectives: the first one consists on minimizing the overall service times for all workers while the second consists on minimizing the waiting time for patients. To solve our Multi-objective mTSP model, we apply two modified algorithms based on two well-known MOEAs which are the E-NSGAII and E-SPEA2. They are proposed by [9]. These algorithms are reconfigured to deal with uncertain data in

Multiobjective optimization. We evaluate our model with an illustrative example of Home Hemodialysis.

This work is organized as follows: section 2 summarizes the literature review on the workforce scheduling and routing problems and the multi-objective optimizing solving approaches. Section 3 presents our mathematical formulation for the considered problem. Solutions approaches are described in section 4. Experimental results are discussed and presented in section 5. Section 6 concludes the work.

II. LITERATURE REVIEW

A. *Workforce Scheduling and Routing Problems*

Workforce scheduling represents the assignment of the employees to the defined shifts for a period of time [8]. In the HHC field, scheduling problem consists on designing a set of routes for medical service suppliers to visit a number of patients located at different places and asking for a set of medical tasks [46]. There are many scenarios in which personnel must carry out tasks at different locations hence requiring transportation. Examples of these types of scenarios include nurses visiting patients at home, technicians carrying out repairs at customers' locations and security guards performing rounds at different premises, etc. We refer to these scenarios as workforce scheduling and routing problems as they usually involve the scheduling of personnel combined with some form of routing in order to ensure that employees arrive on time at the locations where tasks need to be performed [38].

Many types of personnel scheduling and routing problems in HHC field have been tackled in the literature. [48] worked on nurse scheduling, their objectives are to minimize the travel cost and maximize patient satisfaction. They defined a linear model inspired from the mTSP model but with some specific constraints. And, they solved it using Cplex. [49] modelled the nursing routing problem with a Mixed Integer Linear Programming (MILP) that aims to maximize the number of visits. [50] proposed a MILP model to optimize travel for HHC staff. To solve their problem, the authors used Cplex solver. [51] focused on the process of cancer products delivery, as part of a treatment for Home Chemotherapy treatment. The drugs delivery can be performed by the nurses or by specialized deliverers. [52] proposed a new model of Vehicle Routing Problem (VRP) with specific constraints for nurses and technicians scheduling on Peritoneal Dialysis (PD). Then, they solved the model using GAMS [1]solver.

We propose in Table 1 a classification of same recent works dealing with the HHC scheduling and routing problem. This classification, is based on the following characteristics:

- Type of the resource to be planned:

Nurse (N), Technician (T), Doctor (D) and Delivery man (L)

- Optimization criteria to be optimized

-Transport, Service Time/ cost (TST): Minimize transport/ service time or cost

-Number of visits (V): Maximize the number of visits

-Number of resources (R): Minimize the number of resources

- Waiting time for patient (W): Minimize the delays due to traffic or service times.

Table 1: Classification of works on HHC scheduling and routing problems

Work	**Resource**				**Optimization criteria**			
	N	**T**	**D**	**L**	**TST**	**V**	**R**	**W**
[48]	x				x		x	
[49]	x				x	x		x
[50]	x		x		x			
[51]	x		x		x			
[52]	x		x	x	x			
[53]	x						x	
[54]				x	x			
[55]	x							x
[56]	x					x		
[58]	x					x		
Our Work		x			x			x

Most of these papers are dealing with nurse and physician scheduling. In our work, we focus our attention on maintenance technicians scheduling in Home Hemodialysis. Two objectives are proposed in this paper:

- Minimize the overall service times for all workers
- Minimize the waiting time for patients.

B. *Multi-objective Optimization Problems*

Multi-objective optimization (MOO) considers more than one objective function to be optimized simultaneously. Solving a Multi-Objective Problem (MOP) implies obtaining a set of best solutions called Pareto optimal set [6]. The concept of Pareto dominance is of fundamental importance to MOO, as it allows to compare two objective vectors in a precise sense [41]. The concept of dominance is applied to multi-objective problems to compare two solution candidates; if one of these solution is dominated by the other one. In particular, the dominance is a method for the classification of the solutions which ensures the selection of the best solution .Having several objective functions, the notion of "optimum" changes, because in MOPs, we are really trying to find good compromises or "trade-offs" rather than a single solution as in global optimization. We will use the most commonly accepted term, Pareto optimum [32]. These notions will be detailed in the section 4.

[1] https://www.gams.com/optimization-solvers/

In recent years, MOP have received considerable attention in the HHC field. [19] proposed two multi-objective models for organizing routes of HHC. [48] have proposed a new multi-objective model to optimize the degree and the level of competence of the patient. [55] considered three objectives which are the minimization of the total working times of all caregivers, the maximization of the quality of service and the minimization of the maximal working time difference among nurses and auxiliary nurses. [56] have proposed a multi-objective approach based on a MILP, it find an effective feasible working plan for each resource on a daily basis, which ensures the patients' and the caregivers' satisfaction while controlling costs and respecting patients' preferences. [57] are interested in overtime costs of the nurses and customer preferences on visit and nurses.

To solve MOP, most of the recent works have used metaheuristics. Table 2 presents the resolution methods used in some recent works on HHC.

Table 2: Resolution methods for MOPs

Work	**Publication Year**	**Resolution Methods**	
		Exact Methods	**Metaheuristics**
[55]	2019		x
[40]	2019		x
[60]	2019		x
[57]	2016		x
[19]	2015		x
[49]	2015		x
[56]	2015	x	
[48]	2013	x	
Our Work			x

C. *Multi-Objective Optimization problems under Uncertainty*

MOO under uncertainty is considered one of the most important areas of research in decision making. Most real-world problems have multiple objectives to optimize simultaneously and which are often contradictory. Most of the works in the literature have focused on deterministic multi-objective problems where the solutions are a set of exact values [6]. The authors of [6] and [9] defined new concepts to deal with multi-objective problems under uncertainty.

In HHC field, few papers have considered uncertainty in their models. [20] Formulated a model for an HHC Routing and Scheduling Problem with taking into account uncertain travel and service times. [37] consider the HHC Routing Problem with Fuzzy Demand, which comes from the logistics practice of the HHC company. [59] propose uncertainty of patient demand over a multiple day time horizon, when scheduling and routing decisions are taken. But very few articles have proposed multi-objective models under uncertainty. For example, [39] proposed a multi-objective model under uncertainty in which, they have optimize the assignment of patients to each worker and choose the best route for all workers in order to perform their tasks. [59] have studied the HHC problem with patient demand is subject to uncertainty. They proposed a metaheuristic which able to efficiently return solutions near to the optimal.

A wide variety of resolution methods have been proposed in the literature to solve MOO under uncertainty. Table 3 presents some related works.

Table 3: Resolution methods for MOP under uncertainty

Work	**Publication Year**	**Resolution Methods**
[20]	2019	Simulated Annealing (SA), Neighborhood and Tabu Search
[9]	2018	A new fuzzy Pareto dominance; A generic fuzzy extension of well-studied MOEAs
[52]	2017	worst case robust optimization
[37]	2017	Fuzzy Chance Constraint Programming, Hybrid Genetic Algorithm
[11]	2016	Gaussian processes and the efficient global optimization EGO method
[44]	2015	Artificial Bee Colony algorithm
[16]	2015	NSGA-II, Monte Carlo method A new framework based on the "Uncertain Pareto front",
[10]	2005	Pareto based multi-objective evolutionary algorithm (PMOEA)
[2]	2014	Fuzzy multi-criteria particle swarm Optimization (FPSO)
[39]	2014	Fuzzy Simulated Evolution Algorithm
[6]	2014	Fuzzy extension of SPEA2 and NSGAII
[26]	2002	Fuzzy Evolutionary Approach
Our work		Extended NSGAII and Extended SPEA2

III. PROBLEM DESCRIPTION

In this section, we deal with the problem of workforce routing and scheduling in home hemodialysis treatment. For each HHC company, there are a number of workers available each day that should be scheduled. Each worker has a maximum workload (for example: 5 working hours, from 8:00 am to 1:00 pm). Every day, a set of patients is chosen in advance according to the daily capacities of the workers and the distances to be covered. We propose a new multi-objective model based on mTSP that aims to minimize the overall service times for all workers and minimize the waiting time for patients. A patient can only be assigned to one technician and one at a time. Travel, service and start times are uncertain data. The rest of the data are deterministic.

A. *Notation*

W : Set of Workers, $w = 1..W$

P : Set of Nodes, $i = 0..P$, $\{0\}$ represents the HHC Company and$\{1..P\}$ represents patients

$\widetilde{S_{iw}}$: Working time performed by Worker w at Patient $i, \forall\ i\ \in P$

C_w : Maximum workload of each Worker $w, \forall\ w\ \in W$

e_i : Earliest start time of service at Patient $i, \forall\ i\ \in P$

l_i : Lastest start time of Service at Patient $i, \forall\ i\ \in P$

$\widetilde{T_{ij}}$: Travel time between the leading Patient i and the following Patient j, if they assigned to the same Worker w,$\forall\, i,j\ \in P,\ i \neq j, \forall\, w\ \in W$

A : Number of patients to be visited by each worker, $A = \frac{P-1}{W}$

B : Number of paths to be visited by each worker, $B = A - 1$

B. Decision Variables

$$X_{iw} = \begin{cases} 1 & \text{if patient } p \text{ is assigned to the Worker } w, \\ & \forall\, p \in P, \forall\, w \in W \\ 0 & \text{Otherwise} \end{cases}$$

$$Y_{ijw} = \begin{cases} 1 & \text{if patients } i \text{ and } j \text{ are assigned to the same} \\ & \text{Worker } w, \forall\, i,j\ \in P,\ i \neq j, \forall\, w \in W \\ 1 & \text{Otherwise} \end{cases}$$

$\widetilde{D_{iw}}$: The start time of service performed by Worker w at the patient $i, \forall\, p\ \in P, \forall\, w\ \in W$

C. Mathematical Formulation:

$$\text{Minimize } \widetilde{Z_1} = \textstyle\sum_{i,j\in P}\sum_{w\epsilon W}(\widetilde{T_{ij}} + \widetilde{S_{iw}}).Y_{ijw} \tag{1}$$

$$\text{Minimize } \widetilde{Z_2} = \textstyle\sum_{j\epsilon P}\sum_{w\epsilon W} \widetilde{D_{jw}}\ X_{jw} \tag{2}$$

Subject to

$$\textstyle\sum_{w\epsilon W} X_{0w} = W \tag{3}$$
$$\textstyle\sum_{w\epsilon W} X_{iw} = 1, \forall\, i \in P \tag{4}$$
$$\textstyle\sum_{i\epsilon P\backslash\{0\}} X_{iw} = A\ , \forall\, w \in W \tag{5}$$
$$\textstyle\sum_{j\epsilon P\backslash\{0\}}\sum_{w\epsilon W} Y_{ojw} = W \tag{6}$$
$$\textstyle\sum_{i\epsilon P\backslash\{0]}\sum_{w\epsilon W} Y_{i0w} = W \tag{7}$$
$$\textstyle\sum_{i,j\in P\backslash\{0\}} Y_{ijw} = B\ , \forall\, w \in W \tag{8}$$
$$\textstyle\sum_{w\in W} Y_{ijw} = 1\ , \forall\, i,j \in P\backslash\{0\} \tag{9}$$
$$\textstyle\sum_{i,j\epsilon P,i\neq j} (\widetilde{T_{ij}} + \widetilde{S_{iw}}).Y_{ijw} \leq C_w, \forall\, w\ \in W \tag{10}$$
$$\textstyle\sum_{j\epsilon P} Y_{jiw} - Y_{ijw} = 0, \forall\, i \in P,\ w \in W \tag{11}$$
$$\widetilde{D_{jw}} \geq (\widetilde{D_{iw}} + \widetilde{T_{ij}} + \widetilde{S_{iw}}).\,\mathrm{Y}_{ijw}, \forall\, i,j \in P,\ w \in W \tag{12}$$
$$e_i \leq \widetilde{D_{iw}} \leq l_i \forall\, i\ \in P, \forall\, w\ \in W \tag{13}$$
$$X_{iw} \in \{0, 1\}, \forall\, i\ \in P, \forall\, w\ \in W \tag{14}$$
$$Y_{ijw} \in \{0, 1\}, \forall\, i,j\ \in P, i \neq j, \forall\, w\ \in W \tag{15}$$
$$\widetilde{D_{iw}} \geq 0, \forall\, i\ \in P, \forall\, w\ \in W \tag{16}$$

The first objective function aims to minimize the overall service times for all workers, which includes travel times and working times. The second objective function attempts to minimize the overall start times for all workers, which tends to minimize the waiting time for all patients. Constraint (3) guarantee that all workers go through the HHC company. Constraint (4) ensure that each patient is assigned to only one worker. Constraint (5) forces each worker to be assigned to a fixed number of patients. Constraint (6) and Constraint (7) guarantee that all workers start and end their works in HHC Company. Constraint (8) forces each worker to visit a fixed number of paths. Constraint (9) ensure that each worker cross a path exactly once. Constraint (10) guarantee that the working time of each worker does not exceed its maximum workload. Constraint (11) allows us to avoid the sub-tours. Constraint (12) calculates the start service time at the following patient if two patients are assigned to the same worker. Constraint (13) forces the worker to arrive at the patient in a determined timing (from the earliest start time and does not exceed the latest start time). Constraint (14) and Constraint (15) indicate that decision variables X_{iw} and Y_{ijw} are binary. Constraint (16) indicates that the start time $\widetilde{D_{iw}}$ must be positive.

IV. SOLUTION APPROACH

In this work, we aim to find a solution on which we can agree that is optimal in some sense. To do this, it can be interesting to approximate a set of interesting solutions. This strategy implies to avoid so-called Pareto dominated solutions.

A. Pareto dominance

The concept of Pareto dominance is of fundamental importance to MOO, as it allows to compare two objective vectors in a precise sense. It can improve in one objective without deteriorating the performance in any other objective [27].

MOO can be solved using conventional technique and Evolutionary based technique. [43]

- Conventional technique:
    - Weighted Sum Technique: this technique converts multiple objectives into single objective using linear combination of objectives
    - Constraint Based Technique: this technique considers only one objective at a time and treats remaining k-1 objectives as constraints.
- Evolutionary based technique: are well suited for solving several complex multi-objective problems with more objectives. They generates a Pareto set at the end of each run.

B. Multi-objective Evolutionary Algorithms (MOEAs):

To solve our MOO model, we choose MOEAs for the several raisons:

- They have the ability to search partially ordered spaces for several alternative trade-offs. They find a wide range of non-dominated solutions close to the true pareto-optimal solutions [36].

- They have proved to be powerful and suitable for MOO because of their population-based search and ability to find multiple optimal solutions with a good spread.[9]

The Pareto optimality approach is employed to identify a set of pareto optimal solutions in order to obtain the best trade-offs between different objectives [16]. The most popular MOEAs are:

▪ Elitist non-dominated sorting Genetic Algorithm (NSGAII): According to [41], this algorithm consists of the following steps:

- Finds the non-dominated solutions in the population and marks these points as the first front.
- Calculates the average distance between members of each front on the front itself. Parents are selected from the population by using binary tournament selection based on the rank and crowding distance.
- An individual is selected if the rank is smaller than the others or if crowding distance is greater than the others.
- Crowding distance is compared only if ranks for both individuals are the same.
- The selected population generates the offspring from the crossover and mutation operators.
- The selection is based on the rank and on the crowding distance on the last front.

▪ Strength Pareto Evolutionary Algorithm (SPEA2) [41]:

- The non-dominated set is separated from the population of candidate solutions.
- Takes into account the number of dominating and dominated solutions in computing the raw fitness of a solution.
- Locate and maintain a front of non-dominated solutions. This is achieved by using evolutionary process to explore the search space.
- A selection process uses a combination of the degree to which a candidate solution is dominated

Table 4: Comparison between NSGAII & SPEA2

	NSGAII	**SPEA2**
Fitness	Pareto dominance	Pareto dominance
Diversity	Crowding-distance	Nearest neighbor
Selection	Binary tournament	Binary tournament
Replacement	Elitist	Generational
Stopping	nb of generations	nb of generations

C. Fuzzy MultiObjective Optimization

Real-world problems are usually affected by uncertainties that should be taken into account in the optimization and which are described by perturbations on decision variables or on parameters [22]. In this context, recent works suggested methods that determine the most robust solutions. The randomness and the fuzziness have been studied in the MOP models and have been applied with random or fuzzy parameters [4]. We focus our attention on fuzzy numbers [1]. They offer a natural way for expressing uncertain values [12]. In this paper, the fuzzy data are represented by triangular fuzzy numbers [1] [3] [7].

The Triangular Fuzzy Number (TFN) is used to represent uncertain information. The authors in [6], [9] and [12] have defined a triangular fuzzy number (*A)* as equation (17). The TFN is defined as a normal fuzzy set which is represented with a triplet [$\underline{a}$, $\hat{a}$,$\overline{a}$], where [$\underline{a}$,$\overline{a}$], is the interval of possible values and $\hat{a}$ denotes its kernel value. Each element x in A has a value within [0,1] which was affected by a Membership Function $\mu_A(x)$.

$$\mu_A(x) = \begin{cases} 1, & x = \hat{a} \\ \frac{x-\underline{a}}{\hat{a}-\underline{a}}, & \underline{a} \le x \le \hat{a} \\ \frac{\overline{a}-x}{\overline{a}-\hat{a}}, & \hat{a} \le x \le \overline{a} \\ 0. & Otherwise \end{cases} \qquad (17)$$

In our case, the input date and the objective functions will be defined by this fuzzy form, as follows:

$$Z = \begin{cases} Z_1 = [\underline{Z_1}, \widehat{Z_1}, \overline{Z_1}] \\ Z_2 = [\underline{Z_2}, \widehat{Z_2}, \overline{Z_2}] \end{cases} \qquad (18)$$

The classical multi-objective methods cannot be applied to solve our multi-objective model with triangular-valued functions. Therefore, we apply two new algorithms proposed by [9]. These algorithms will be describe in the next sub-section.

D. Fuzzy Multi-Objective Evolutionary Algorithm for mTSP (FMOEAmTSP) :

In order to solve our model, we apply two new algorithms from the literature which are proposed by [9]. In this paper, we are the first who apply the E-NSGAII and the E-SPEA2 to the mTSP model and in the HHC area. The structure of these two algorithms is presented in Fig.1. The main contribution of these algorithms consists on replacing the standard Pareto by the fuzzy Pareto dominance.

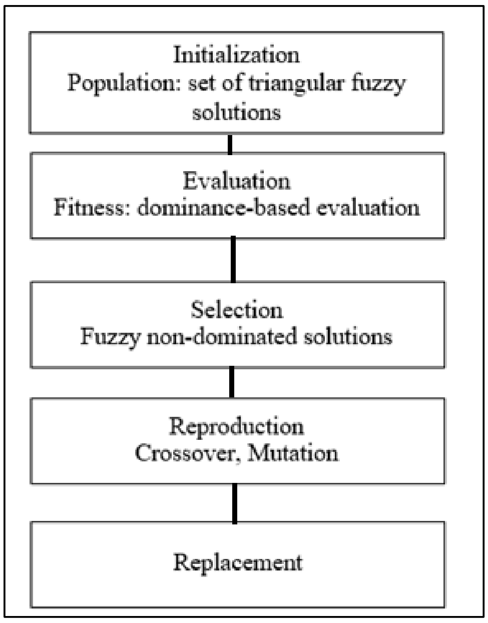

Initialization
Population: set of triangular fuzzy solutions
Evaluation
Fitness: dominance-based evaluation
Selection
Fuzzy non-dominated solutions
Reproduction
Crossover, Mutation
Replacement

Figure 1: Structure of the E-NSGA-II & E-SPEA

We follow the same steps of E-NSGA which are detailed in Algorithm.1:

```
Algorithm E-NSGAII
Input  : Population size $N$
         Maximum number of generations $G$
Output: Pareto set approximation
begin
    1.Initialization. create a random population $P$ of $N$ fuzzy solutions;
    repeat
        2.Fitness Assignment. rank all solutions using fuzzy Pareto dominance;
        3.Environmental Selection. select the non-dominated fuzzy solutions
        based on their expected crowding values and copy them in an external
        population $P'$;
        if size of $P'$ exceeds $N$ then
        |   add the least crowded solutions to $P'$;
        else if size of $P'$ is less than $N$ then
        |   set $P'$ with dominated solutions;
        else
        |   the environmental selection is completed;
        end
        4.Elitism. update $P \cup P'$
    until 5. Stopping condition. Number of generations $> G$ is satisfied
    6.Mating Selection. perform a binary crowded tournament selection to select
    parents from $P'$;
    7.Variation. apply crossover and mutation operators to the mating pool;
    8.Replacement. replace old population by the resulting offspring population.
end
```

Algorithm 1: E-NSGAII [9]

We consider also the same parameters defined by E-SPEA2 that is described in Algorithm 2.

```
Algorithm 2: E-SPEA2
Input  : Population size $N$
         Triangular archive $A$
         Maximum number of generations $G$
Output: Pareto set approximation
begin
    1.Initialization. create a random population $P$ of $N$ fuzzy solutions and
    create an empty triangular archive $A$ of fixed size $M$;
    repeat
        2.Fitness Assignment. rank solutions using fuzzy Pareto dominance;
        3.Environmental Selection. copy all non-dominated solutions from $P$ to
        the triangular archive $A$;
        if size of $A$ exceeds $M$ then
        |   $A$ is pruned by means of a clustering procedure;
        else if size of $A$ is less than $M$ then
        |   fill $A$ with best dominated solutions;
        else
        |   the environmental selection is completed;
        end
        4.Elitism. update $A$;
    until 5. Stopping condition. Number of generations $> G$ is satisfied;
    6.Mating Selection. perform a binary tournament selection with replacement
    on $A$ to fill the mating pool;
    7.Variation. apply crossover and mutation operators to the mating pool;
    8.Replacement. replace old population by the resulting offspring population.
end
```

Algorithm 2: E-SPEA2 [9]

V. EXPERIMENTATIONS

In this section, we simulate our model with an example of the workforce Scheduling and Routing in Home Hemodialysis. We focus our attention on maintenance process for dialysis machines which represents one of the expense factors for the HHC Company. We implement for the first time the E-NSGAII and the E-SPEA2 in Java language using the Integrated Development Environment Apache NetBeans IDE[2], version 8.2. It provides robust editors that display support for the latest Java technologies. We integrated the JMETAL[3] framework that stands for Metaheuristic Algorithms in Java. These algorithms are executed on a 2.50-GHz Intel(R) Core(TM) i5-7200 CPU computer with 8 GB of RAM and running under Windows 10.

In our case, there is no common benchmark available in the literature for stochastic mTSPs, we are basing on a close-up example of reality inspired by data provided by the Biomedicine Agency [4]in the kidney annual report for 2017[5] as shown in Table 5. We consider the following example to validate our proposed model:

[2] NetBeans IDE : https://netbeans.org/
[3] JMETAL : http://jmetal.sourceforge.net/
[4] https://www.agence-biomedecine.fr/
[5] https://www.francerein.org/files/France%20Rein/mediatheque/rapport-rein-2017.pdf

Table 5: Biomedicine Agency data

Patients on Dialysis in France	47985 patients
Patients on Hemodialysis in France	44,978 patients (93.7%)
Patients on HHC in region Haut-de-France	4929 patients
Patients on Hemodialysis HHC in region Haut-de France	40 patients (0.8%)

We take the example of a home hemodialysis company that is located in the region of “Haut-de-France”, France and which has 40 patients and 2 maintenance technicians. Each technician must visit a specific set of patients and each patient must be visited only once a day. These technicians must start their work at 8:00 am from HHC Company and return there at the end of their services. The data used for the test are presented in table 6. HHC is represented by node 0.

Table 6: Configuration table

Parameter	Value
Number of Technicians	W = 2
Number of Patients	P \{0} = 40
Start time from HHC Company	8:00 am
maximum workload C_w	5 Hours = 300 minutes
Number of patients assigned to each patient	5 patients
Number of paths assigned to each patient	4 paths
Approximate working time	20 minutes
e_i	$D_{iw} - 5min$
l_i	$D_{iw} + 5min$

Each maintenance technician visits a specific number of patients. In our case, we have 40 patients and 2 technicians. We consider that these technicians travel to patients over a 4-day horizon. We balance the number of patients to be visited by all technicians between the days on the horizon: Total number of patients / number of days on the horizon. Table 7 shows the approximate travel times between the different nodes.

As we know, that travel times cannot be known in advance given several constraints such as traffic on roads, accidents, strikes, weather and many other factors that can delay or even advance the start time of service. For these reasons, we have defined that travel times are uncertain data. We cannot predict the working time for certain causes (an unexpected breakdown in the machine, the technician delayed to finish the repair, etc.). Our data are defined as Triangular Fuzzy Number:

- $T_{ij} = \left[\underline{T_{ij}}, \widehat{T_{ij}}, \overline{T_{ij}}\right]$
- $S_{iw} = \left[\underline{S_{iw}}, \widehat{S_{iw}}, \overline{S_{iw}}\right]$

The start time of service performed by a technician is calculated as follows:

- $\underline{D_{jw}} = \underline{D_{jw}} \vee \underline{T_{ij}} \vee \underline{S_{iw}}$ (19)
- $\widehat{D_{jw}} = \widehat{D_{0w}} \vee \widehat{T_{ij}} \vee \widehat{S_{iw}}$ (20)
- $\overline{D_{jw}} = \overline{D_{0w}} \vee \overline{T_{ij}} \vee \overline{S_{iw}}$ (21)
- $D_{jw} = [\underline{D_{jw}}, \widehat{D_{jw}} \overline{D_{jw}}]$

Table 7: Approximate Travel Times (min)

i/j	0	1	2	3	4	5	6	7	8	9	10
0	0	40	40	40	40	40	40	40	40	40	40
1	40	0	95	120	65	30	105	35	20	80	50
2	40	95	0	105	50	20	95	20	30	65	35
3	40	30	65	0	35	35	80	30	20	50	20
4	40	20	50	80	0	50	65	20	35	35	30
5	40	35	20	65	30	0	50	35	50	20	35
6	40	50	20	50	20	80	0	50	65	30	20
7	40	65	30	35	20	95	20	0	80	20	50
8	40	80	20	20	50	20	30	80	0	35	65
9	40	95	35	30	65	120	20	95	105	0	20
10	40	105	50	20	80	135	35	105	120	65	0

We present in table 8 the parameters of E-NSGA-II and E-SPEA2.

Table 8: Algorithm parameters

Algorithm	Population size	Generation size	Crossover %	Mutation%
E-NSGA-II	10	10	60	10
E-SPEA2	10	10	25	10

In order to generate new solutions, called offspring, we code individuals with decimal coding. The chromosome representation is showed by Fig.2. We remember that each route must start and finish in the HHC company. Our chromosome presents only the patients without HHC company.

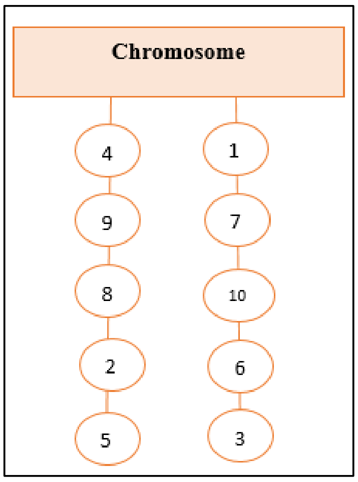


Figure2: Chromosome representation

After simulation, we obtain the results which are presented in Table 9, Table 10 Fig.3 and Fig.4.

Table 9: Obtained results by E-NSGA-II

#	Route	Travel times	Working times	Z_1	Z_2
1	0→3→8→5→2→1→0	[135,160,165]	[75,100,125]	[420,520,620]	[910,960,1010]
	0→4→7→6→10→9→0	[135,160,185]	[75,100,125]		
2	0→1→4→6→9→10→0	[190,215,240]	[75,100,125]	[535,635,735]	[1465,1515,1565]
	0→8→2→5→3→7→0	[195,220,245]	[75,100,125]		
3	0→1→4→6→9→10→0	[190,215,240]	[75,100,125]	[485,585,685]	[1270,1320,1370]
	0→5→2→7→3→8→0	[145,170,195]	[75,100,125]		
4	0→10→3→8→2→1→0	[220,245,270]	[75,100,125]	[505,605,705]	[1235,1285,1335]
	0→6→4→7→9→5→0	[135,160,165]	[75,100,125]		
5	0→3→10→6→9→5→0	[145,170,195]	[75,100,125]	[575,675,775]	[1485,1535,1585]
	0→4→1→2→7→8→0	[280,305,330]	[75,100,125]		
6	0→2→7→10→1→8→0	[145,170,195]	[75,100,125]	[505,605,705]	[1570,1620,1670]
	0→5→3→9→6→4→0	[210,235,260]	[75,100,125]		
7	0→7→2→1→6→4→0	[250,375,400]	[75,100,125]	[698,798,898]	[1948,1998,2048]
	0→4→5→3→9→8→0	[298,323,348]	[75,100,125]		
8	0→10→9→8→7→6→0	[280,305,330]	[75,100,125]	[645,745,845]	[1920,1970,2020]
	0→1→2→3→4→5→0	[215,240,265]	[75,100,125]		
9	0→2→10→6→5→1→0	[195,220,245]	[75,100,125]	[560,660,760]	[1520,1570,1620]
	0→3→9→4→8→7→0	[255,280,305]	[75,100,125]		
10	0→7→1→2→3→4→0	[355,380,405]	[75,100,125]	[765,865,965]	[2010,2060,2110]
	0→8→6→10→5→9→0	[260,285,310]	[75,100,125]		

According to table 9 and Fig.3, the solutions are represented by three values. The best solutions are those which have the best trade-offs between objectives functions values. In our case, the first solution is the best one because it is non-dominated by any other solution and it has the best trade-off between objetive functions values.

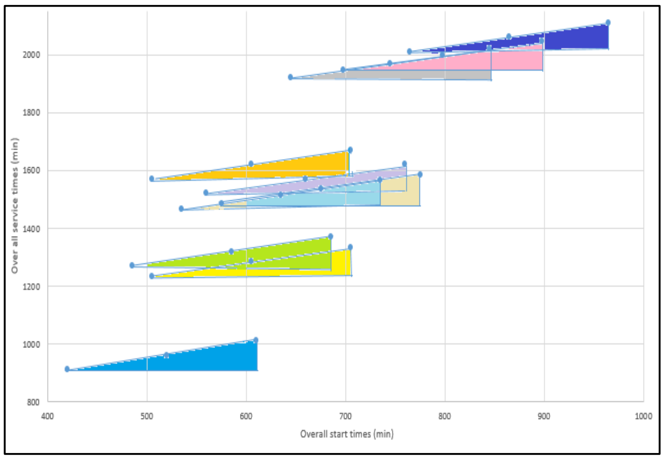


Figure 3: Representation of solutions in the objective space

The obtained results by the algorithm E-SPEA2 are showed in table 10 and Fig.4.

Table 10: Obtained results by E-SPEA2

#	Route	Travel times	Working times	Z_1	Z_2
1	0→3→8→5→2→1→0	[135,160,165]	[75,100,125]	[420,520,620]	[910,960,1010]
	0→4→7→6→10→9→0	[135,160,185]	[75,100,125]		
2	0→7→3→5→2→8→0	[195,220,245]	[75,100,125]	[535,635,735]	[1465,1515,1565]
	0→10→9→6→4→1→0	[190,215,240]	[75,100,125]		
3	0→1→4→6→9→10→0	[190,215,240]	[75,100,125]	[485,585,685]	[1270,1320,1370]
	0→5→2→7→3→8→0	[145,170,195]	[75,100,125]		
4	0→9→10→6→7→4→0	[135,160,165]	[75,100,125]	[420,520,620]	[910,960,1010]
	0→1→2→5→8→3→0	[135,160,185]	[75,100,125]		
5	0→1→2→3→4→5→0	[215,240,265]	[75,100,125]	[645,745,845]	[1920,1970,2020]
	0→6→7→8→9→10→0	[280,305,330]	[75,100,125]		
6	0→8→1→10→7→2→0	[145,170,195]	[75,100,125]	[505,605,705]	[1570,1620,1670]
	0→5→3→9→6→4→0	[210,235,260]	[75,100,125]		
7	0→10→9→8→7→6→0	[280,305,330]	[75,100,125]	[645,745,845]	[1920,1970,2020]
	0→1→2→3→4→5→0	[215,240,265]	[75,100,125]		
8	0→10→3→8→2→1→0	[220,245,270]	[75,100,125]	[505,605,705]	[1235,1285,1335]
	0→6→4→7→9→5→0	[135,160,165]	[75,100,125]		
9	0→5→4→3→2→1→0	[215,240,265]	[75,100,125]	[645,745,845]	[1920,1970,2020]
	0→6→7→8→9→10→0	[280,305,330]	[75,100,125]		
10	0→3→10→6→9→5→0	[145,170,195]	[75,100,125]	[575,675,775]	[1485,1535,1585]
	0→4→1→2→7→8→0	[280,305,330]	[75,100,125]		

According to table 10 and Fig.9, we interpret that there are two non-dominated solutions which have the same fitness value. These routes have the same trade-offs between the first and the second objective functions. Therefore, decision-makers can choose one of these solutions (solution 1 or solution 4).

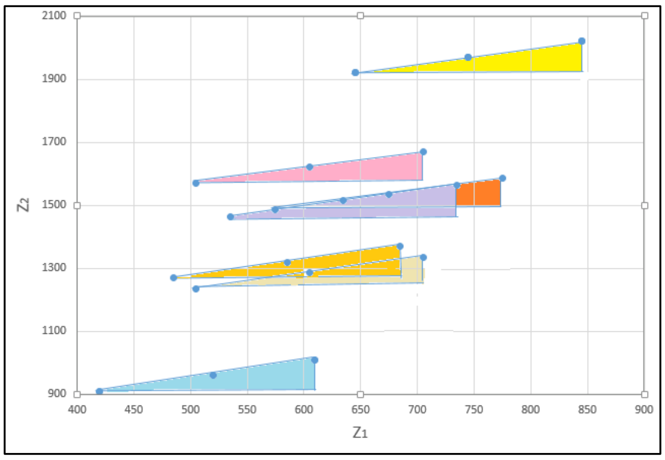


Figure 4: solutions' representations in the objective space

We compare between obtained results by E-NSGA-II and E-SPEA2 in table 11.

Table 11: Comparison of results

Algorithm	# pareto-optimal solution	Solution	Z_1	Z_2
E-NSGA-II	1	0→3→8→5 →2→1→0	[420,520,620]	[910,960,1010]
		0→4→7→6 →10→9→0		
E-SPEA2	2	0→3→8→5 →2→1→0	[420,520,620]	[910,960,1010]
		0→4→7→6 →10→9→0		
		0→9→10→6 →7→4→0	[420,520,620]	[910,960,1010]
		0→1→2→5 →8→3→0		

We observe that the performance of algorithms are similar. E SPEA2 has usually better convergence to the global Pareto front than E-NSGAII but E-NSGAII is the fastest algorithm because it doesn't involve expensive calculations or archive procedure as in E-SPEA2.

VI. CONCLUSION

Home Hemodialysis companies aims to minimize costs and maximize customers' satisfaction. We focused our attention on the maintenance process which represents one of the expense factors for the HHC Company. Therefore, we have proposed a new multi-objective model for the workforce routing and scheduling problem, extended of mTSP. The data of our model are deterministic except travel, service and start times are uncertain. Our model aims to minimize the overall service times for all technicians and minimize the delays due to the traffic and service times which directly influences to the patients' satisfaction. To solve our MOP under uncertainties, we choose to apply two extended algorithms from literature, proposed by [9]. E-NSGAII and E-SPEA2 have been modified in order to adapt them to the MOP containing uncertain data. These two algorithms are applied for the first time to mTSP models and the HHC area. The obtained results have shown the effectiveness of the E-NSGA-II and the E-SPEA2 and their suitability for our problem.

In future works, we intend to extend our model in many ways:

- As in the literature there is no collective benchmark to test stochastic data for mTSP, we plan to design a benchmark with more instances.
- Simulate our model with new approaches basing on other types of HHC workforces such the drug deliverers.
- Add new constraints to our model such as the vehicle capacity constraint. In this case, we must adapt our problem to VRP models.

REFERENCES

[1] L. Zadeh, Fuzzy Sets, Information and Control, 1965.

[2] M.Mutingi,C.Mbohwa,*"A fuzzy-based particle swarm optimisation approach for task assignment in home healthcare"*. South African Journal of Industrial Engineering. 25. 10.7166/25-3-777. 2014.

[3] T. Ross, Fuzzy Logic with Engineering Applications, John Wiley & Sons, 2010

[4] J. Zhou, F. Yang, K. Wang, Multi-objective optimization in uncertain random environments, Fuzzy Optim. Decis. Mak. Springer 13 (4) .2014 397–413

[5] R. I. Bolaños, M. G. Echeverry, and J. W. Escobar, "A multiobjective non-dominated sorting genetic algorithm (NSGA-II) for the multiple traveling salesman problem," *Decis. Sci. Lett.*, vol. 4, no. 4, pp. 559–568, Jul. 2015.

[6] O. Bahri, N. Ben Amor, and T. El-Ghazali, "Optimization algorithms for multi-objective problems with fuzzy data," in *IEEE SSCI 2014 - 2014 IEEE Symposium Series on Computational Intelligence - MCDM 2014: 2014 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making, Proceedings*, 2015, pp. 194–201.

[7] S. Heilpern, "Fuzzy sets and systems Representation and application of fuzzy numbers 1," 1997.

[8] N. Safaei, D. Banjevic, and A. K. S. Jardine, "Multi-objective maintenance workforce scheduling in a steel company," in *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 2009, vol. 42, no. 4 PART 1, pp. 1049–1054.

[9] O. Bahri, E. G. Talbi, and N. Ben Amor, "A generic fuzzy approach for multi-objective optimization under uncertainty," *Swarm Evol. Comput.*, vol. 40, pp. 166–183, Jun. 2018.

[10] P. Limbourg, "Multi-Objective Optimization of Problems with Epistemic Uncertainty."

[11] J. Guerra and M. M. Schoenauer, "Optimisation multi-objectif sous incertitudes de phénomènes de thermique transitoire ED MITT : Mathématiques appliquées Équipe d'accueil ISAE-ONERA MOIS."

[12] O. Bahri, N. Ben Amor, and T. El-Ghazali, "CCIS 443 - New Pareto Approach for Ranking Triangular Fuzzy Numbers."2014

[13] M. A. González, A. Oddi, and R. Rasconi, "Multi-Objective Optimization in a Job Shop with Energy Costs through Hybrid Evolutionary Techniques."ICAPS.2017

[14] I. Saad and M. Benrejeb, "Optimisation multicritère par Pareto-optimalité de problèmes d'ordonnancement en tenant compte du coût de la production."2006

[15] H.Allaoua, "Routage et planification des personnels pour l'hospitalisation à domicile", Thèse de doctorat, Paris .2014

[16] C. Villa, E. Lozinguez, and R. Labayrade, "Multi-objective optimization under uncertain objectives: Application to engineering design problem," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2013, vol. 7811 LNCS, pp. 796–810.

[17] B. Oumayma, B.A. Nahla and E.-G. Talbi, "A possibilistic framework for solving multi-objective problems under uncertainty: Definition of new Pareto optimality", IPDPSW 2013, pp. 405-414, 2013

[18] M. Mlakar, T. Tušar, and B. Filipič, "Comparing Solutions under Uncertainty in Multiobjective Optimization," Mathematical Problems in Engineering, vol. 2014, Article ID 817964, 10 pages, 2014.

[19] G. Fabrice. "Problème de tournées de véhicules avec contraintes de synchronisation dans le cadre de structures de maintien à domicile." 2015

[20] Y. Shi, T. Boudouh, and O. Grunder, "A robust optimization for a home health care routing and scheduling problem with consideration of uncertain travel and service times," *Transp. Res. Part E Logist. Transp. Rev.*, vol. 128, pp. 52–95, Aug. 2019.

[21] J. Dubois-Lacoste, "A study of Pareto and Two-Phase Local Search Algorithms for Biobjective Permutation Flowshop Scheduling."2018

[22] Y. Jin, J. Branke, "Evolutionary optimization in uncertain environments - a survey", IEEE Trans. on Evol. Comput.pages pp. 303-317 .2005

[23] H. Jin, "Workforce planning in manufacturing and healthcare systems," University of Iowa, Iowa City, Iowa, USA, 2016.

[24] Y. Liu, H. Dong, N. Lohse, and S. Petrovic, "A multi-objective genetic algorithm for optimisation of energy consumption and shop floor production performance," *Int. J. Prod. Econ.*, vol. 179, pp. 259–272, Sep. 2016.

[25] M. Song and D. Chen, "An improved knowledge-informed NSGA-II for multi-objective land allocation (MOLA)," *Geo-Spatial Inf. Sci.*, vol. 21, no. 4, pp. 273–287, Oct. 2018.

[26] I. Kacem, S. Hammadi, and P. Borne, "Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic," 2002.

[27] M. T. M. Emmerich and A. H. Deutz, "A tutorial on multiobjective optimization: fundamentals and evolutionary methods," *Nat. Comput.*, vol. 17, no. 3, pp. 585–609, Sep. 2018.

[28] D. Brockhoff GECCO 2018 "tutorial on evolutionary multiobjective optimization". *2018*

[29] K. Deb. "Multi-Objective Optimization Using Evolutionary Algorithms". John Wiley & Sons, Inc., New York, NY, USA.2001

[30] N. Melab and E.-G. Talbi, "D1.1 Tutorial on parallel multi-objective optimisation," 2016.

[31] A. Konak, D. W. Coit, and A. E. Smith, "Multi-objective optimization using genetic algorithms: A tutorial," *Reliab. Eng. Syst. Saf.*, vol. 91, no. 9, pp. 992–1007, Sep. 2006.

[32] C. A. C. Coello, C. San, P. Z. México, and D. F. 07300, "A Short Tutorial on Evolutionary Multiobjective Optimization."2001

[33] E. Zitzler, M. Laumanns, and S. Bleuler, "A Tutorial on Evolutionary Multiobjective Optimization."

[34] O. spanjaard. "Optimisation multi-objectifs : un tutorie"l. 33ème journée JFRO - 23 juin 2015

[35] S. Chelouti and K.Kaidi, "Résolution d'un problème d'optimisation multi-objectif fractionnaire Linéaire flou en nombres entiers".2016

[36] S. Bhargava, "A Note on Evolutionary Algorithms and Its Applications," 2013.

[37] Y. Shi, T. Boudouh, and O. Grunder, "A fuzzy chance-constraint programming model for a home health care routing problem with fuzzy demand," in *ICORES 2017 - Proceedings of the 6th International Conference on Operations Research and Enterprise Systems*, 2017, vol. 2017-January, pp. 369–376.

[38] J. A. Castillo-Salazar, D. Landa-Silva, and R. Qu, "Workforce scheduling and routing problems: literature survey and computational study," *Ann. Oper. Res.*, vol. 239, no. 1, pp. 39–67, Apr. 2016.

[39] M. Mutingi and C. Mbohwa, "Multi-objective homecare worker scheduling: A fuzzy simulated evolution algorithm approach," *IIE Trans. Healthc. Syst. Eng.*, vol. 4, no. 4, pp. 209–216, Oct. 2014.

[40] H. Habibnejad-Ledari, M. Rabbani, and N. Ghorbani-Kutenaie, "Solving a multi-objective model toward home care staff planning considering cross-training and staff preferences by NSGA-II and NRGA," *Sci. Iran.*, vol. 26, no. 5 E, pp. 2919–2935, 2019.

[41] V.popa andrii, E.Chołodowicz and P.Orlowski. "Comparison of SPEA2 and NSGA-II Applied to Automatic Inventory Control System Using Hypervolume Indicator." 2017

[42] T. Lust and J. Teghem, "The Multiobjective Traveling Salesman Problem: A Survey and a New Approach.".2010

[43] K. Jayamoorthi, D. Karunanidy, A. Jayavel, and S. Ramalingam, "ARTICLE A SURVEY ON MULTI-OBJECTIVE TRAVELLING SALSMAN PROBLEM," 2017.

[44] Z. Wang, J. Guo, M. Zheng, and Y. Wang, "Uncertain multiobjective traveling salesman problem," *Eur. J. Oper. Res.*, vol. 241, no. 2, pp. 478–489, Mar. 2015.

[45] I. A. Hameed, "Multi-objective Solution of Traveling Salesman Problem with Time," in *Advances in Intelligent Systems and Computing*, 2020, vol. 921, pp. 121–132.

[46] J. Jemai, M. Chaieb, and K. Mellouli, "The Home Care Scheduling Problem: A modeling and solving issue," in *2013 5th International Conference on Modeling, Simulation and Applied Optimization, ICMSAO 2013*, 2013.

[47] G. Du, X. Liang, and C. Sun, "Scheduling optimization of home health care service considering patients' priorities and timewindows," *Sustain.*, vol. 9, no. 2, 2017.

[48] F. Gayraud, L. Deroussi, N. Grangeon, and S. Norre, "A New Mathematical Formulation for the Home Health Care Problem," *Procedia Technol.*, vol. 9, pp. 1041–1047, 2013.

[49] B. Issaoui, I. Zidi, E. Marcon, and K. Ghedira, "New multi-objective approach for the home care service problem based on scheduling algorithms and variable neighborhood descent," *Electron. Notes Discret. Math.*, vol. 47, pp. 181–188, Feb. 2015.

[50] D. Aiane, A. El Amraoui, and K. Mesghouni, "A New Optimization Approach for a Home Health Care Problem," 6th IESM Conference, October 2015

[51] S, Chahed. " *Modélisation et analyse de l'organisation et du fonctionnement des structures d'hospitalisation à domicile* ".Laboratoire Génie Industriel Ecole Centrale Paris Grande Voie des Vignes 92925 Châtenay-Malabry Cedex.2008.

[52] M. Issabakhsh, S.-M. Hosseini-Motlagh, M.-S. Pishvaee, and M. Saghafi Nia, "A Vehicle Routing Problem for Modeling Home Healthcare: a Case Study," 2018.

[53] M. Shateri, "Resource allocation and Risk Analysis of Dialysis Centres," 2015.

[54] O. Bräysy, P. Nakari, W. Dullaert, and P. Neittaanmäki, "An optimization approach for communal home meal delivery service: A case study," *J. Comput. Appl. Math.*, vol. 232, no. 1, pp. 46–53, Oct. 2009.

[55] J. Decerle, O. Grunder, A. Hajjam El Hassani, and O. Barakat, "A memetic algorithm for multi-objective optimization of the home health care problem," *Swarm Evol. Comput.*, vol. 44, pp. 712–727, Feb. 2019

[56] L. En-Nahli, H. Allaoui, and I. Nouaouri, "A multi-objective modelling to human resource assignment and routing problem for home health care services," in *IFAC-PapersOnLine*, 2015, vol. 28, no. 3, pp. 698–703.

[57] K. Braekers, R. F. Hartl, S. N. Parragh, and F. Tricoire, "A bi-objective home care scheduling problem: Analyzing the trade-off between costs and client inconvenience," *Eur. J. Oper. Res.*, vol. 248, no. 2, pp. 428–443, Jan. 2016.

[58] R. Liu, B. Yuan, and Z. Jiang, "Mathematical model and exact algorithm for the home care worker scheduling and routing problem with lunch break requirements," *Int. J. Prod. Res.*, vol. 55, no. 2, pp. 558–575, Jan. 2017.

[59] P. Cappanera, M. G. Scutellà, F. Nervi, and L. Galli, "Demand uncertainty in robust Home Care optimization," *Omega (United Kingdom)*, vol. 80, pp. 95–110, Oct. 2018.

[60] S. R. Ait Haddadene, N. Labadie, and C. Prodhon, "Bicriteria vehicle routing problem with preferences and timing constraints in home health care services," *Algorithms*, vol. 12, no. 8, 2019.

Automatic generation of metaheuristic algorithms

Sergio Iturra[1], Carlos Contreras-Bolton[2], and Víctor Parada[1]

[1] Departamento de Ingeniería Informática, Universidad de Santiago de Chile, Santiago, Chile
{sergio.iturra, victor.parada}@usach.cl
[2] Departamento de Ingeniería Industrial, Universidad de Concepción, Concepción, Chile
carlos.contreras.b@udec.cl

Abstract. Designing a heuristic algorithm to solve an optimization problem can also be seen as an optimization problem. Such a problem seeks to determine the best algorithm contained in the search space. The objective function corresponds to the computational performance of the algorithm measured in terms of computational time, complexity, number of instructions or number of elementary operations. The automatic design of algorithms has been explored for several combinatorial optimization problems. In this work, we extend this exploration towards the automatic design of metaheuristics to find solutions for the traveling salesman problem. The process is carried out by genetic programming. The resulting algorithms are combinations of well-known metaheuristics and, in some cases, present better computational performance than the existing algorithms for the set of selected test instances.

1 Introduction

There is a family of optimization problems that come from various fields of knowledge and are characterized by the tremendous computational difficulty that arises when trying to determine an optimal solution. Such optimization problems, which we call complex problems here, are considered difficult because currently, an algorithm that can solve all the instances of a problem with computational efficiency is not known [1, 2]. In this field, it is accepted that an algorithm is efficient when it requires many steps that grow polynomially with the input. A typical method for addressing complex optimization problems is through mathematical programming, which considers an objective function that corresponds to the criteria to be optimized and a set of constraints that define the solution space that contains the optimal solution. Integer programming algorithms utilize the enumeration of the solution space, a task that may require very high computational time or memory even when dealing with small examples of the problem [3, 4]. This challenge constitutes one of the main fields of scientific research in the area of combinatorial optimization, and it is a relentless search to improve existing techniques or to find new methods for addressing this situation. The motivation behind this search consists of innumerable practical situations that occur in various areas of knowledge, such as transport [5], health care [6], sports [7], production processes [8], and logistics [9].

One of the most commonly used practical approaches for addressing the family of complex optimization problems considers metaheuristics. A metaheuristic is a method that describes a general procedure to effectively inspect the solution space of an optimization problem and thus determine the best solution inspected [10]. In recent decades, this field has increased considerably because numerous metaheuristics have been generated, and a wide variety of problems have been studied under this approach [11]. Although a metaheuristic does not guarantee the determination of the optimal solution, in practice, they are very effective because they require a low computational time and provide a solution close to the optimal, and in many instances, they return the optimal solution. Such techniques have originated analogizing with different phenomena in nature, such as the species evolution, particle swarms, bee and ant colonies, and pure substance cooling. In general, they can be classified into single-solution search, population-based search, or hybrid metaheuristics [10]. Although their origins are varied, some characteristics are shared by several metaheuristics: a) they carry out the search process by gradually visiting solutions that belong to the problem-space, b) they work on a current solution or a current set of solutions in every step, c) the problem optimization function inherently guides the search process, d) they use exploration and exploitation strategies and e) they partially store the search space.

Recent literature has shown the emergence of a wide variety of hybrid metaheuristics that have better computational performance than the same metaheuristics used individually [12–16].

Such hybrid algorithms arise when considering the best components of metaheuristics and assembling them appropriately for the optimization problem at hand. The variety of works considers hybridizations of metaheuristics, with other metaheuristics, constraint programming, search tree techniques, and mathematical programming. However, the main deficiency that arises in this field is the design step because it is difficult to know in advance the appropriate combination for each optimization problem. It is necessary to identify how many and what components can be integrated to generate the hybrid algorithm that responds with good computational performance for the specific optimization problem. A framework or practical guide that facilitates the design task is not known in the literature. In practice, the authors have found the appropriate combination of components through computational experiments that manually test some possibilities among the many possible combinations. Nor is there a standard method for carrying out experimentation, and although today there is a technological advance that allows a large number of numerical tests, to the best of our knowledge, the automation of this process has not been explored.

Another approach used to address complex optimization problems is the automatic generation of an algorithm (AGA). The AGA automatically assembles the components that potentially compose an algorithm for a given optimization problem [17–19]. This task is possible because determining the best algorithm for an optimization problem is also a master optimization problem. Consequently, the search for the best algorithm for a given problem reduces to solving the master problem by some of the existing methods, which in practice can be any of the current metaheuristics. The elementary components that can be considered are diverse; they can be specific heuristics already existing for the problem, the atomic parts of such heuristics, or exact algorithms of mathematical programming. Genetic programming (GP) is particularly appropriate for this task because it artificially evolves populations of syntactic trees that represent combinations of instructions, such as those that occur in an algorithm [20]. In this way, several algorithmic combinations can be represented with syntax trees and combined by evolutionary computing. This technique has allowed the generation of new algorithms for combinatorial optimization problems [21–24], a fact that suggests that the same technique could automatically produce hybrid metaheuristics. AGA is not only an automatic method for combining thousands of components and exploring the space composed of all hybrid metaheuristics but also determining the appropriate algorithm for each optimization problem, thus providing an experimental standard for this field of knowledge.

In this work, we use AGA to generate single-solution hybrid metaheuristic algorithms for the traveling salesman problem (TSP). The algorithms are constructed through GP by evolving syntactic trees [17]. The components of syntactic trees are functions and terminals, which are instructions typically used to write pseudocode and primary components typically considered in the heuristic, metaheuristic and exact methods. In addition, a set of instances is selected and divided into two groups: the first group is used for the construction of metaheuristics, and the second group is used to evaluate the already constructed hybrid metaheuristics.

In the following section, the literature review is presented. The procedures for generating the metaheuristic algorithms are described in the third section. The computational results of the generated algorithms are presented in the fourth section. The conclusions of the study are presented in the last section.

2 Literature review

Only recently have the first attempts to automatize the design of hybrid metaheuristics appeared. One of them is the novel approach by [25] that proposes a meta-GA to automate the hybridization of metaheuristics. The authors consider the following algorithms: simulated annealing, tabu search, iterated local search and memetic algorithm to solve the TSP. Additionally, [25, 14] efficiently solved the aircraft landing problem and the two-dimensional bin packing problem using the same approach. Their automated designed hybrid metaheuristics showed better performance than individually used metaheuristics and manually created hybrid metaheuristics. Recently, [32] propose a work that uses fuzzy logic and fuzzy systems to create a cooperative scheme for the automatic selection of proper metaheuristic algorithms and control searching process dynamically. Their approach was tested on 0-1 knapsack problem, and the computational experiments showed to be much more effective in searching the solution space. Although, it did not differ from the other algorithms in terms of computing times. However, since these were the first attempts to automatize the process in the literature, some drawbacks have been observed. Such as depending on predefined templates (structures), only selection automatically of sequences of metaheuristics to use, and the resulting algorithms with their analysis are not shown. The first implies that several successful combinations within the search space may not be visited due to fixed structures. The second means that new automatically generated algorithms are only sequences of metaheuristics that are run one after another. Therefore, there is no hybridization among components of a metaheuristic with another metaheuristic completely different. The third means that these new automatically generated algorithms are not available to the scientific community in the field.

From the research line of automatic algorithm configuration [26], attempts are also being made to generate hybrid metaheuristics using tools that are typically used in this area, such as IRACE [27]. Thus, in [28], instead of seeking parameters, they sought metaheuristic components and tuned the parameters, all in a single process. The authors used a predefined framework, with grammars for each metaheuristic to generate hybrid metaheuristics for three combinatorial optimization problems. Other similar approaches were previously carried out by [29], who used tools to design stochastic local searches automatically and non-hybrid metaheuristics as ACO algorithms [30]. Alfaro-Fernández et al. [31] generated hybrid metaheuristics following the previous methodology described for solving hybrid flowshop scheduling problems. Their algorithms are competitive against state-of-the-art algorithms. Recently, Pagnozzi and Stützle [33–35] proposed an automatic design system of stochastic local search for permutation flowshop problem and other two variants that consider additional constraints. This uses a configuration tool to combine algorithmic components following a set of rules defined as a context-free grammar. Their experiments show that the generated algorithms outperform the state-of-the-art. However, these approaches from the configuration of algorithm parameters have some limitations, such as the use of predefined templates and grammars that limit the search space. The authors did not use a specific search method for the new task of searching for potential new combinations, but instead, they used IRACE, which is a specialized method for tuning parameters; thus, they possibly failed to explore possible good combinations. Besides, some approaches are limited to only some metaheuristics, missing the opportunity to mix, for instance, metaheuristics based on a population of solutions with other single-solution-based. The same authors noted that these first approaches correspond to proofs-of-concept [28]. Therefore, there many options for investigating approaches to improve or create new and more appropriate approaches.

3 Procedure to generate metaheuristic

To find a solution for the metaproblem, a set of primary components must be created. These components are designed by means of a set of functions and terminals and give rise to the new algorithms to be produced so they must be typical components of algorithms (functions) as well as tools that allow the construction of a solution for the optimization problem (terminals). From the definition of these elementary components, an initial population of syntax trees can be configured that can evolve into a sequence of later populations using reproduction, crossover, and mutation, which are typical operators in evolutionary computing [36]. The initial population is generated by the ramped half and half mechanism consisting of randomly generating half the population with full trees up to a default depth and the other half, with partially full trees [37]. The fittest syntax trees are randomly selected and reproduced into the new population. Furthermore, two types of mutations are considered: "point mutation" and "shrink mutation". In the first mutation, a syntax tree node is changed by a function that uses the same number of parameters, while in the second, a node is changed by any of the functions that act directly on the container where the current solutions are stored. The crossover between two syntax trees is performed by replacing a node of the first syntax tree by a section of the second. The fitness evaluation requires a set of adaptation instances that must be adequately selected. In our case, due to the generated algorithms' stochastic nature, every instance is evaluated several times. The syntax trees finally produced are decoded as algorithms and must also be evaluated externally so that a second set of control instances is required. The set of examples of each type is divided into two groups: the first is used to automatically construct metaheuristics, and the second is used to evaluate the algorithms already built. This evolutionary process is described for t generations in Fig. 1.

The set of functions contains the basic instructions present in any algorithm. They are defined by means of the parameters P_1 and P_2, which are boolean variables, so a "true" indicates that

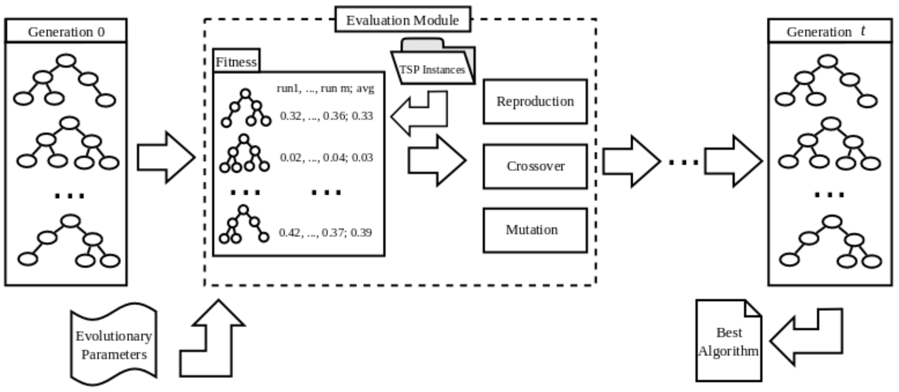


Fig. 1. Process of automatic generation of algorithms.

the parameter performed an action; otherwise, it is "false". Specifically, the main functions are While(P_1, P_2), which runs the parameter P_2, whereas P_1 returns "true", And(P_1, P_2) runs the parameter P_1 and P_2 returning "true" if both returned values are also "true"; and IfThen(P_1, P_2) activates P_2 when running P_1 returns "true".

The set of terminals is divided into two groups. The terminals are based on metaheuristics and terminals that construct a TSP solution. The first terminals contain components of three well-known metaheuristic algorithms: iterated local search (ILS), simulated annealing (SA) and variable neighborhood search (VNS) [10]. These metaheuristics are decomposed in atomic parts, and 12 terminals are obtained, such as terminals based on the computation of the temperature of SA, acceptance criteria of an ILS or SA, and operations in the neighborhood of VNS.

The following terminals designed for the TSP are based on typical heuristics for the problem and elementary operations to be executed on a solution container:

- ShiftCity: randomly shifts a city of the current solution.
- SwapCities: randomly exchanges two cities of the current solution.
- BlockReverse: randomly reverses the order of the cities of the current solution.
- BlockRule: iteratively reverses randomly the order of the cities of the current solution.
- SwapCitiesRule: iteratively exchanges randomly two cities of the current solution.
- 2-Opt: is an adaptation of a local search procedure for this problem known as 2-opt.

Terminals return true if the action for which they are intended is executed and, false otherwise.

The quality of an algorithm is measured by means of the relative error in the objective function for a set of instances used during the evolution. In addition, we consider another measure which is the number of obtained solutions that happen to be optimal (also named hits). Let n_l be the number of instances, z_i the value of the optimal solution of the instance i, and u_i the obtained value of the algorithm for instance i. α and β are values that belong to the interval $[0, 1]$ and are used to arbitrarily handle the weight of each term. Furthermore, let hit_i be equal to 1 if the optimal solution is found. Then, the evaluation function is represented in Equation (1). The minimum value of this function is zero, which means that by seeking its minimization, algorithms that solve problem instances and have a large number of hits are explored, where $\alpha = 0.9$ and $\beta = 0.1$.

$$f = \alpha \frac{1}{n_l} \sum_{i=1}^{n_l} \frac{|u_i - z_i|}{z_i} + \beta \frac{n_l - \sum_{i=1}^{n_l} hit_i}{n_l} \tag{1}$$

The evolutionary process is implemented in the ECJ 25, which is an evolutionary computation library coded in Java [38]. The computational experiment is performed on a Google Virtual Machine instance with 2.0 GHz (8 virtual processors, Intel Skylake) and 7.0 GB of RAM. Genetic parameters involved in the GP are population size, 100; number of generations, 30; probabilities: crossover, 0.85; mutation: 0.10; reproduction: 0.05, and the parameter $k = 15$. The comparison (called testing) of the best-obtained algorithm is performed with 14 TSPLIB instances [39], with up to 101 cities, and in the evolution phase we use three TSPLIB instances, with up to 58 cities.

4 Results

The best algorithm found finds near-optimal solutions for the 14 evaluation instances. The algorithm was obtained after performing three runs with the same parameter values, changing only the seed. Consequently, 3,000 combinations were inspected. In turn, to evaluate the algorithm, ten runs were performed with each instance, and the resulting values are presented in Table 1. The name of the instance is described in the first column, the second column contains the minimum relative error value with respect to the optimal solution, while in the third column, and the average relative error of all runs is presented. In the last column, the average computational time of the algorithm with each instance is reported. In the second column, it is observed that in ten instances, the optimal solution was obtained in at least one of the ten runs. In the ten runs, the optimal solution of the brazil58 instance was obtained. Note that the average computational time was between 0.47 and 3.3 seconds. This result suggests that it is feasible to combine different elementary components of the various metaheuristics and assemble such components appropriately to face the TSP.

Table 1. Performance of generated metaheuristic on 14 instances.

Instances	Min (%)	Avg (%)	Avg time (Sec)
eil51	0.23	1.03	0.47
berlin52	0.00	3.48	0.50
brazil58	0.00	0.00	0.69
st70	0.00	1.10	1.21
eil76	0.00	1.21	1.53
pr76	0.00	0.37	1.48
rat99	0.00	1.73	3.44
kroA100	0.00	0.43	3.42
kroB100	0.00	1.01	3.78
kroC100	0.00	0.44	3.30
kroD100	0.07	1.24	3.34
kroE100	0.00	0.50	3.45
rd100	0.85	1.75	3.32
eil101	0.32	1.21	3.55
average	0.11	1.11	2.39

The structure of the generated algorithm corresponds to a variant of the ILS algorithm. Algorithm 1 has four stages. In the first stage (lines 10-15), the algorithm performs a local search, and in the second stage, it verifies the acceptance or rejection of the solution found by a local search process (lines 16-21). In the third stage, a perturbation is performed (line 22), and in the fourth stage, the algorithm ends with a new local search (line 24-26). The algorithm considers two stop criteria explicitly established in lines 27 and 29. Line 27 establishes that if the first local search does not improve the current solution, the algorithm stops. Line 29 stops the algorithm according to the number of iterations predefined as the instance size. The generated metaheuristic differs from ILS in the order of the instructions. There is also noise in the structure because it incorporates elements of SA that do not contribute to the TSP solution; these are instructions that work as bloating code that may arise when using GP.

The extension of AGA to automatically produce metaheuristics produced similar results to those found when AGA was used to produce specific heuristics for various optimization problems. New metaheuristics were produced, which are combinations of the initially defined components of well-known metaheuristics. In addition, such metaheuristics produce near-optimal solutions for at least a small group of examples. It is clear that the scalability of this result requires more extensive experimentation. Likewise, the generated algorithms must be properly parameterized to obtain a better computational performance, a process that could also be included automatically. An interesting consequence of the result is that the automation process adopted significantly accelerates research in this field. Many algorithmic combinations can be explored with low computational effort; in fact, the reported experiment lasted 158 minutes.

Algorithm 1 Generated algorithm

```
1:  function FUNCTION
2:      if 2-opt() then
3:          for 1 to $k$ do
4:              2-opt();
5:              2-opt();
6:          end for
7:      end if
8:  end function
9:  repeat
10:     $\alpha \leftarrow$ false;
11:     $\beta \leftarrow$ false;
12:     for 1 to $k$ and FUNCTION() = true do
13:         $\alpha \leftarrow$ true;
14:         LinearCooling();
15:     end for
16:     if $\alpha$ = true and 2-opt() then
17:         for 1 to $k$ and LogCooling() do
18:             $\beta \leftarrow$ true;
19:             Deterministic_ChooseIfBetter();
20:         end for
21:     end if
22:     if $\alpha$ = true and $\beta$ = true and SwapCities() then
23:         if 2-opt() then
24:             FUNCTION();
25:         end if
26:     else
27:         break;
28:     end if
29: until $iter = n$
```

5 Conclusions

This work described a process for the automatic design of metaheuristics for TSP. The algorithms were produced by automatic generation algorithms through genetic programming from a set of elementary components. In particular, we used terminals based on the metaheuristic algorithms ILS, SA, and VNS. The resulting algorithms are combinations of existing metaheuristics, and the best algorithm found is a variant of ILS. However, the generated metaheuristic provides good performance in terms of the solution quality in a very short computational time. Future research will focus on improving the performance of the generated metaheuristics and combining the different components of the metaheuristics in a better way. Additional future research will focus on extending the proposed method to other variants of the TSP or even other optimization problems.

Acknowledgment

This research was partially funded by the Complex Engineering Systems Institute (ICM: P-05-004-F, CONICYT: FB016). We also acknowledge projects USA1899–Vridei 061919VP-PAP Universidad de Santiago de Chile, DICYT-USACH 061919PD, and VRID INICIACIÓN 220.097.016-INI, Vicerrectoría de Investigación y Desarrollo (VRID), Universidad de Concepción.

References

1. Korte, B., Vygen, J.: Combinatorial Optimization: Theory and Algorithms. 6th edn. Volume 21 of Algorithms and Combinatorics. Springer Berlin Heidelberg, Berlin, Heidelberg (2018)
2. Papadimitriou, C.H., Steiglitz, K.: Combinatorial optimization: algorithms and complexity. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1982)
3. Conforti, M., Cornuéjols, G., Zambelli, G.: Integer Programming. Volume 271 of Graduate Texts in Mathematics. Springer International Publishing, Cham (2014)

4. Schrijver, A.: Theory of linear and integer programming. John Wiley & Sons, Inc. (1998)
5. Toth, P., Vigo, D.: Vehicle Routing: Problems, Methods, and Applications. 2nd edn. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2014)
6. Kolker, A.: Healthcare Management Engineering: What Does This Fancy Term Really Mean? Springer New York, New York, NY (2012)
7. Wright, M.: Operational Research Applied to Sports. Palgrave Macmillan UK, London (2015)
8. Silver, E.A., Pyke, D.F., Thomas, D.J.: Inventory and Production Management in Supply Chains. 4th edn. Taylor & Francis, Boca Raton (2016)
9. Wang, J.: Management Science, Logistics, and Operations Research. Advances in Logistics, Operations, and Management Science. IGI Global (2014)
10. Talbi, E.G.: Metaheuristics: From Design to Implementation. John Wiley & Sons, Inc., UK (2009)
11. Gendreau, M., Potvin, J.Y., eds.: Handbook of Metaheuristics. Volume 272 of International Series in Operations Research & Management Science. Springer International Publishing, Cham (2019)
12. Blum, C., Puchinger, J., Raidl, G.R., Roli, A.: Hybrid metaheuristics in combinatorial optimization: A survey. Applied Soft Computing **11** (2011) 4135–4151
13. Blum, C., Raidl, G.R.: Hybrid Metaheuristics: Powerful Tools for Optimization. Artificial Intelligence: Foundations, Theory, and Algorithms. Springer International Publishing, Cham (2016)
14. Hassan, A., Pillay, N.: Hybrid metaheuristics: An automated approach. Expert Systems with Applications **130** (2019) 132–144
15. Raidl, G.R., Puchinger, J., Blum, C.: Metaheuristic hybrids. In Gendreau, M., Potvin, J.Y., eds.: Handbook of Metaheuristics. Springer International Publishing, Cham (2019) 385–417
16. Ting, T.O., Yang, X.S., Cheng, S., Huang, K.: Hybrid metaheuristic algorithms: Past, present, and future. In Yang, X.S., ed.: Recent Advances in Swarm Intelligence and Evolutionary Computation. Springer International Publishing, Cham (2015) 71–83
17. Acevedo, N., Rey, C., Contreras-Bolton, C., Parada, V.: Automatic design of specialized algorithms for the binary knapsack problem. Expert Systems with Applications **141** (2020) 112908
18. Contreras-Bolton, C., Gatica, G., Parada, V.: Automatically generated algorithms for the vertex coloring problem. PLoS ONE **8** (2013) e58551
19. Ryser-Welch, P., Miller, J.F., Asta, S.: Generating human-readable algorithms for the travelling salesman problem using hyper-heuristics. In: Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference - GECCO Companion '15, New York, New York, USA, ACM Press (2015) 1067–1074
20. Pétrowski, A., Ben-Hamida, S.: Evolutionary Algorithms. John Wiley & Sons, Inc., Hoboken, NJ, USA (2017)
21. Bertolini, V., Rey, C., Sepúlveda, M., Parada, V.: Novel methods generated by genetic programming for the guillotine-cutting problem. Scientific Programming **2018** (2018) 1–13
22. Contreras-Bolton, C., Rey, C., Ramos-Cossio, S., Rodríguez, C., Gatica, F., Parada, V.: Automatically produced algorithms for the generalized minimum spanning tree problem. Scientific Programming **2016** (2016) 11
23. Loyola, C., Sepúlveda, M., Solar, M., Lopez, P., Parada, V.: Automatic design of algorithms for the traveling salesman problem. Cogent Engineering **3** (2016)
24. Parada, L., Herrera, C., Sepúlveda, M., Parada, V.: Evolution of new algorithms for the binary knapsack problem. Natural Computing **15** (2016) 181–193
25. Hassan, A., Pillay, N.: A meta-genetic algorithm for hybridizing metaheuristics. In Oliveira, E., Gama, J., Vale, Z., Lopes Cardoso, H., eds.: Progress in Artificial Intelligence. Springer, Cham (2017) 369–381
26. Stützle, T., López-Ibáñez, M.: Automated Design of Metaheuristic Algorithms. In M., G., J.Y., P., eds.: Handbook of Metaheuristics. Volume 272. Springer, Cham (2019) 541–579
27. López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., Stützle, T.: The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives **3** (2016) 43–58
28. López-Ibáñez, M., Kessaci, M.E., Stützle, T.: Automatic Design of Hybrid Metaheuristics from Algorithmic Components. Technical report, TR/IRIDIA/2017-012, IRIDIA, Université Libre de Bruxelles, Belgium (2017)
29. Marmion, M.E., Mascia, F., López-Ibáñez, M., Stützle, T.: Automatic design of hybrid stochastic local search algorithms. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Volume 7919 LNCS., Springer, Berlin, Heidelberg (2013) 144–158
30. López-Ibáñez, M., Stutzle, T.: The automatic design of multiobjective ant colony optimization algorithms. IEEE Transactions on Evolutionary Computation **16** (2012) 861–875
31. Alfaro-Fernández, P., Ruiz, R., Pagnozzi, F., Stützle, T.: Automatic Algorithm Design for Hybrid Flowshop Scheduling Problems. European Journal of Operational Research **282** (2020) 835–845
32. Tezel, B.T., Mert, A.: A cooperative system for metaheuristic algorithms. Expert Systems with Applications **165** (2021) 113976

33. Pagnozzi, F., Stützle, T.: Automatic design of hybrid stochastic local search algorithms for permutation flowshop problems. European Journal of Operational Research **276** (2019) 409–421
34. Pagnozzi, F., Stützle, T.: Evaluating the impact of grammar complexity in automatic algorithm design. International Transactions in Operational Research **00** (2020) itor.12902
35. Pagnozzi, F., Stützle, T.: Automatic design of hybrid stochastic local search algorithms for permutation flowshop problems with additional constraints. Operations Research Perspectives **8** (2021) 100180
36. Eiben, A., Smith, J.: Introduction to Evolutionary Computing. 2nd edn. Natural Computing Series. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
37. Poli, R., Langdon, W.B., Mcphee, N.F.: A Field Guide to Genetic Programming. Lulu Enterprises, UK Ltd (2008)
38. Luke, S., Sean: Ecj then and now. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion on – GECCO '17, New York, New York, USA, ACM Press (2017) 1223–1230
39. Reinelt, G.: TSPLIB–A traveling salesman problem library. ORSA Journal on Computing **3** (1991) 376–384

Particle swarm optimisation without panopticon to evaluate private social choice

Vicenç Torra and Edgar Galván

[1] Department of Computing Science, Umeå University, Umeå, Sweden, vtorra@ieee.org
[2] Naturally Inspired Computation Research Group, Department of Computer Science, Maynooth University, Lero, Maynooth, Ireland, edgar.galvan@mu.ie

Abstract. In a recent paper we introduced differentially private random dictatorship as a private mechanism for social choice. Differentially private mechanisms are evaluated in terms of their utility and information loss. In the area of social choice it is not so straightforward to evaluate the utility of a mechanism. It is therefore difficult to evaluate a differentially private social choice mechanism.
In this paper we propose to use a particle swarm optimization-like problem to evaluate our differentially private social choice method. Standard particle swarm optimization (PSO) can be seen in terms of a panopticon structure. That is, a structure in which there is a central entity that knows all of all. In PSO, there is a particle or agent that knows the best position achieved by any of the particles or agents. We propose here PSO without panopticon as a way to avoid an omniscient agent in the PSO system.
Then, we compare different social choice mechanisms for this PSO without panopticon, and we show that differentially private random dictatorship leads to good results.

1 Introduction

In our recent work [12], we studied random dictatorship [2, 4] as voting mechanism that satisfies differential privacy under some conditions, and defined a variation of this method that is differentially private.

In data privacy [11, 5, 13] data protection mechanisms are often evaluated in terms of their utility. Data protection mechanisms based on secure multiparty computation are known to be good with respect to utility as they provide loss-less computation and do not make any perturbation on the output of the function. In contrast, data protection mechanisms that follow differential privacy [3] or k-anonymity [8] cause some information loss to the data or computation.

As a result of this, it is relevant to evaluate the utility of random dictatorship and of its differentially private version.

Nevertheless, the evaluation of the utility of a voting mechanism is an ill-defined problem. Voting mechanisms are usually evaluated in terms of their properties based on individual preferences. Examples of properties include [1, 2, 9] Condorcet conditions, the independence of irrelevant alternatives, etc. These conditions are defined assuming that voters possess ordinal utility functions. That is, voters have an order on the alternatives (e.g., they prefer alternative a_1 to alternative a_2). In contrast, it is not considered a numerical evaluation of each alternative.

It is known that a numerical utility model does not fit well with voting procedures. Observe that for any ordinal utility function (i.e., a_1 is preferred to a_2), there are infinitely many (numerical) utility functions compatible with the ordinal one. Moreover, if we consider (numerical) utility functions for each voter (i.e., a numerical value for each alternative as $u_i(a_1)$, $u_i(a_2)$, … for voter i), the majority rule does not necessarily maximize the total utility. That is, if we define the social good of a selected alternative as the addition of voters' utility for this alternative ($\sum u_i(a)$ for selected alternative a), majority voting does not necessarily lead to the best option. The same applies to other social choice mechanisms.

In this paper we propose a federated learning [7] type of problem using particle swarm optimisation (PSO) [6] to evaluate private social choice mechanisms (as the one introduced in [12]). The goal is to find an optimal (aggregated) position that is the best for a set of agents. The problem is formulated as a particle swarm optimisation (PSO) problem [6] in which there is no omniscient agent with knowledge of the so-far best optimal position for all as it is the case for standard PSO.

I.e., no panopticon, as we say. The best optimal position is obtained through successive voting in line with successive aggregations in federated learning.

The structure of this paper is as follows. In Section 2 we review probabilistic social choice and a differentially private version of it. In Section 3 we review particle swarm optimisation and introduce particle swarm optimisation without panopticon. This later approach is to avoid the system of particles omniscient on the best position of each particle. Section 4 discusses the evaluation of differentially private social choice in terms of particle swarm optimisation. The paper finishes with some conclusions and directions for future work.

2 Probabilistic social choice

Let I be a set of agents and A a set of alternatives. Let the goal be to select the preferred alternative for the set of agents. That is, the alternative that most of the agents prefer.

To formulate this problem we model agents preferences in terms of preference relations on the set of alternatives. That is, the preference relation $\succeq_i$ is defined in terms of subsets of $A \times A$. In our context, we have only access to the best preferred option of an agent $i \in I$ and this is just its vote, an alternative $a \in A$. So, for all a', $a \succeq_i a'$ for this agent $i \in I$.

Plurality voting is to select the alternative that receives the most votes or preferences. In contrast, uniform random dictatorship proceeds as follows.

Method 1 *From [12]. This method selects an agent i in I according to a uniform distribution on I, and then uses $\succeq_i$ to select the most prefered alternative by agent i as outcome. That is, once i is selected from I, the method returns $a \in A$ such that $a \succeq_i a'$ for all $a' \in A$.*

This approach can be equivalently implemented considering all alternatives, their frequency (votes), and then selecting one alternative using a probability distribution proportional to the frequency.

We defined in [12] two differentially private versions of random dictatorship. Their difference was on whether the voting was compulsory or optional. We give below the definition where voting is not compulsory but optional.

Method 2 *From [12], let $A = \{a_1, \ldots, a_m\}$ be the set of alternatives. Let I be the set of agents, and let $\succeq_i$ be the corresponding preference relations for $i \in I$ on the alternatives A. Then, enlarge I with a set of agents $I_0 = \{e_1, \ldots, e_m\}$ such that $\succ_i$ for $i \in I_0$ has as its prefered alternative the ith alternative in A.*

Then, apply uniform random dictatorship on $I \cup I_0$.

This voting procedure satisfies differential privacy for an appropriate parameter ϵ. The following lemma establishes bounds for the ϵ parameter.

Lemma 1. *From [12], differentially private random dictatorship as defined in Method 2 satisfies differential privacy for any*

$$\epsilon \geq \log \frac{2|I \cup I_0|}{|I \cup I_0| + 1}.$$

For a large number of agents, it is easy to see that we can compute a bound for ϵ. That is, $\epsilon > \log(2) = 0.6931$. Naturally, the more alternatives we have, the more agents we need to tend to this limit. Figure 1 represent the bound in Lemma 1 for 4, 8 and 16 alternatives. As we will describe later, we use in our experiments 8 alternatives. The corresponding figure shows the bound for a number of agents between 3 and 200.

When we can ensure that there is at least an agent for each alternative, we have bounds that do not depend on the alternative. Nevertheless, this is not necessarily the case in our scenario. See [12] for details.

3 Particle swarm optimisation based evaluation

As briefly mentioned in the introduction, we define the evaluation scenario in terms of PSO [6]. We have a function $f : \mathbb{R}^n \to \mathbb{R}$ and we are interested in finding its minimum. To do so, we have a set of S agents or particles. Each particle $i \in \{1, \ldots, S\}$ has a position p_i in the n dimensional space, and a velocity v_i.

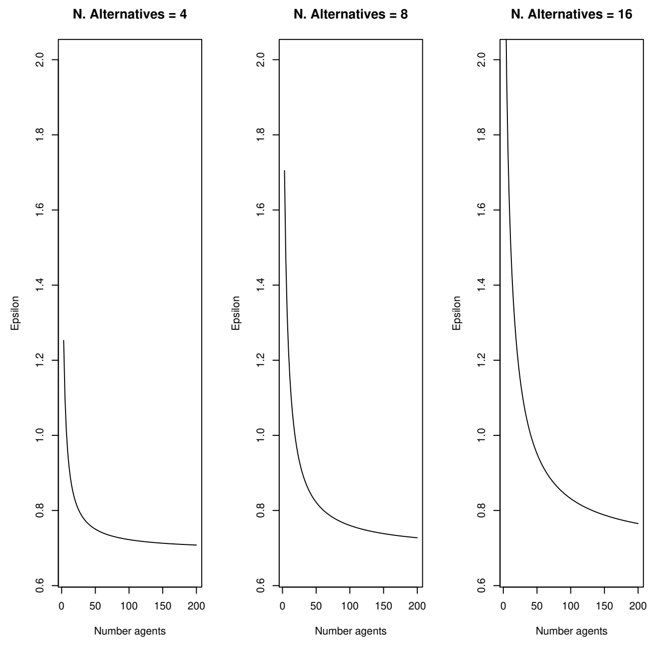


Fig. 1. Bounds for the ϵ parameter (y axis) in differentially private random dictatorship as defined in Method 2 using the results of Lemma 1 (read text). We have considered the case of 4, 8, and 16 alternatives (left, middle, and right graphs) and the number of agents ranging from 3 to 200 (x axis).

3.1 Standard particle swarm optimisation

In a standard particle swarm optimisation solution, each particle records the best position found so far. This is denoted by b_i. In addition, we keep track of the global best position found so far in the whole system. This is denoted by g.

The procedure iteratively computes a new position for each particle until a certain termination criteria is met. In each iteration, the best position is updated when necessary. More precisely, for the ith particle, we compute a new velocity:

$$v_i = \omega v_i + \phi_p r_p (b_i - p_i) + \phi_g r_g (g - p_i) \tag{1}$$

where ω is the inertia weight, ϕ_p and ϕ_g are acceleration coefficients one for the best position of the particle and the other for the best global position; and where r_p and r_g are random vectors following a uniform distribution in [0,1].

Then, we update the position of the ith particle as follows:

$$p_i = p_i + v_i.$$

When $f(p_i) < f(b_i)$ then we update the best position $b_i = p_i$, and if $f(p_i) < f(g)$ then we update the global best position $g = p_i$.

3.2 Particle swarm optimisation without panopticon

In PSO, the position of any agent or particle is public. Our scenario differs from the standard PSO scenario because we consider it private. Therefore, we cannot use the global best position g when computing a new position or velocity for any particle.

Instead, we consider an additional *system* particle that is led by all the particles. This *system* particle has its own position and velocity. We denote them by p_G and v_G, respectively. These position and velocity are public.

The position p_G is analogous to the aggregated model in federated learning. This position is based on agent's positions, and it is computed as an aggregation (using our social choice mechanisms) of previous and current information.

As the best global position g is not available, Equation 1 cannot be used. Thus we compute the velocity of each particle in a slightly different way. As the system particle position is known and any particle can evaluate whether this position is better or not than its own, we update particles velocity taking advantage of this knowledge. Formally,

$$v_i = \begin{cases} \omega v_i + \phi_g r_g (p_G - p_i) & \text{if} f(p_G) < f(p_i) \\ \omega v_i + \phi_p r_p (b_i - p_i) & \text{otherwise.} \end{cases} \tag{2}$$

Equation 2 is similar to Equation 1 but updating does not depend on g, and the updating rule depends on whether the ith particle is in a better position than system's one (i.e., $f(p_G) < f(p_i)$).

Updating of system's position needs to take into account that access to other particles' positions is not permitted. Our proposal is that particles can provide a direction where to lead the system particle. This direction plays the role of the velocity vector, but there are two main differences.

– One is that the direction is a vector but it does not have a magnitude.
– Another one is that not all directions are possible, but only a limited number of them.

We have these constraints because we consider that supplying an arbitrary direction or velocity is not feasible from a privacy perspective: the space of alternatives would be too large to protect (too many possible angles and magnitudes).

The number of directions n_d is a parameter of the system. This parameter is interpreted according to the following example. In this work we only consider functions with two variables, so they are functions of the form $f : \mathbb{R}^2 \to \mathbb{R}$. So, all n_d directions are on the plane.

When $n_d = 4$ it means that particles can vote for four directions and they correspond to the following direction vectors $(1, 0)$, $(0, 1)$, $(-1, 0)$ and $(0, -1)$. In general, each possible direction $a = 0, \ldots, n_d$ corresponds to a different angle, all angles are equally spaced in $[0, 2\pi]$ and they are defined with respect to the $(1, 0)$ vector. At a given time, each particle computes its angle with this vector $(1, 0)$, say α_i, and then vote for the option $\lfloor a/(2\pi) \rfloor$ which is the nearest option to their own preferred angle.

Given a set of particles, from their votes for their preferred angle, we can select an angle using any social choice approach. In particular, we can use plurality voting (i.e., select the most frequent angle), random dictatorship, and differentially private random dictatorship for selecting an angle. This process leads to an angle α_G which can then be used to find a direction vector v_{α_G}. That is, v_{α_G} is the unit vector with angle α_G with the vector $(1, 0)$. Once the vector is known, we update the global position as follows:

$$p_G = p_G + \omega_G v_{\alpha_G},$$

where ω_G is the inertia weight of the global position.

3.3 Analogy with federated learning

Our approach has similarities with the standard procedure in federated learning. Note that agents access p_G. So, we assume that this information is publicly available. This is similar to accessing the average model in federated learning. Then, the information that agents provide in our system, that is, direction, can be seen as the difference between the global model and the local model in federated learning. Our approach is more restrictive than in federated learning, as we are dealing with a context in which agents can only vote for a few options. This has, of course, advantages from a privacy point of view.

4 On the evaluation of differentially private social choice

We have considered different scenarios in order to evaluate differentially private social choice. Different scenarios differ on the function to be minimized, the social choice procedure, and the parameters of the system. We discuss these elements below.

4.1 Functions

In order to evaluate our approach we have used the following functions in $\mathbb{R}^2$, selected from the review work by Sengupta et al. [10] on particle swarm optimisation. For each function we also include the range of the two variables (x_1, x_2).

We have selected functions in $\mathbb{R}^2$ because they provide a simple scenario with only a few voting options and compatible with the example in [12] of a cohort of drones guiding a ground vehicle. Drones vote continuously to guide the vehicle. The selected direction, landmark or position at any time is not so important. It is the overall set of decisions (the rough path) what influences the trajectory of the vehicle.

The functions we consider are the following ones.

– Quadratic function ($x_1, x_2 \in [-100.0, 100.0]$):

$$f_1(x_1, x_2) = x_1^2 + x_2^2$$

– Schwefel's problem 2.22 ($x_1, x_2 \in [-10.0, 10.0]$):

$$f_2(x_1, x_2) = |x_1| + |x_2| + |x_1| \cdot |x_2|$$

– Schwefel's problem 1.2 ($x_1, x_2 \in [-100.0, 100.0]$):

$$f_3(x_1, x_2) = x_1^2 + (x_1 + x_2)^2$$

– Generalized Rosenbrock's function ($x_1, x_2 \in [-2.0, 2.0]$):

$$f_4(x_1, x_2) = 100 * (x_2 - x_1 * x_1)^2 + (x_1 - 1)^2$$

– Generalized Schwefel's problem 2.26 ($x_1, x_2 \in [-500.0, 500.0]$):

$$f_5(x_1, x_2) = -x_1 sin(\sqrt{|x_1|}) - x_2 sin(\sqrt{|x_2|})$$

– Rastrigin's function ($x_1, x_2 \in [-5.12, 5.12]$):

$$f_6(x_1, x_2) = 2 \cdot 10 + x_1^2 - 10cos(2x_1\pi) + x_2^2 - 10cos(2x_2\pi)$$

– Ackley's function ($x_1, x_2 \in [-32.768, 32.768]$):

$$\begin{aligned} f_7(x_1, x_2) = &-20e^{-0.2\sqrt{0.5(x_1^2+x_2^2)}} \\ &-e^{0.5cos(2x_1\pi)+cos(2x_2\pi)} + 20 + e \end{aligned}$$

– Griewank function ($x_1, x_2 \in [-600.0, 600]$):

$$f_8(x_1, x_2) = 1 + (1/4000)(x_1^2 + x_2^2) - cos(x_1) * cos(x_2/\sqrt{2})$$

The optimal solutions for these problems correspond to a function equal to zero, except in the case of f_5 where the best solution corresponds to -12569.5.

4.2 Social choice procedures

The social choice procedures we have considered are:

- plurality voting,
- random dictatorship, and
- differentially private random dictatorship.

Social choice with plurality voting and random dictatorship are used as base line social choice procedures.

In addition, we have also implemented standard PSO. That is, there is an omniscient agent that observes all other agents and stores the best/optimal position found so far. This agent represents the guard in the panopticon.

Then, we consider for each of the three procedures above, two cases according to whether a particle votes or not. They are the following ones:

- A particle always votes;
- A particle only votes when its position is better than the best one, and in this case, decision to vote is based on a probability.

Thus, we have 7 different approaches: (i) PSO, (ii) plurality voting (PV), (iii) random dictatorship (RD), (iv) differentially private random dictatorship (DRD), and (v) plurality voting (bPV), (vi) random dictatorship (bRD), and (vii) differentially private random dictatorship (bDRD) only among those agents that have a position better than the global one.

4.3 Parameters

Our system is defined by the number of particles, the inertia weight ω, and the velocities ϕ_p, ϕ_g and the inertia weight ω_G. In addition, social choice procedures can have additional parameters corresponding to the probabilities related to when to vote. We assume that all particles use the same parameter's values.

We have used different sets of values for the parameters ω, ϕ_p, ϕ_g, and ω_G. They are the following ones.

- ω: 0.005, 0.001, 0.05, 0.1, 0.2, 0.4
- $\phi_p = \phi_g$: 0.01, 0.05, 0.1, 0.2, 0.5, 1.0, 2.0
- ω_G: 0.005, 0.01, 0.05, 0.1, 0.2, 0.4

We have used 50 particles and 1000 iterations to compare the results. That is, 1000 voting processes. A few additional examples that are shown in the figures have been considered with additional iterations (10000 iterations). We have used 8 voting options, corresponding to an angle of $2\pi \cdot i/8$ for $i = 0, \ldots, 7$ from direction $(1, 0)$. We have considered 30 executions, for each of the assignments considered.

4.4 Experiments

We evaluate the utility of differentially private random dictatorship using as test-bed the optimisation problems defined above and as a methodology to solve this problem the particle swarm optimisation without panopticon (as defined in Section 3.2) as well as using the social choice procedures in Section 4.2 with the parameters described in Section 4.3.

Our goal is to see if the differentially private random dictatorship has a comparable behaviour to the ones supposedly better of plurality voting and random dictatorship. To that end, we

- compare the solutions obtained using the three social choice procedures, and
- compare different parametrisations (when particles vote, parameters used).

In addition, we use PSO as the reference value. Nevertheless, as it keeps track of the best solution found so far, we expected PSO to outperform the other methods.

For each set of parameters considered, we have computed the mean of the optimal function found. That is, a mean of the values (*MeanF*) obtained for the 30 different executions. We have also recorded the minimum (*MinF*) obtained in these 30 executions. Table 1 displays optimal values of *MeanF* and *MinF* found for each function and each method: PSO – on the top row, right column; PV, RD, and DRD – middle row, from left to right; bPV, bRD, and $bDRD$ – bottom row, from left to right. Between brackets we display the parameters ω, $\phi_p = \phi_g$ and ω_G used to obtain the optimal solution.

PSO vs. social choice procedures PSO is always better than any other social choice procedure. Except for problem f_5, PSO reaches always the minimum for the 30 executions. That is, except for f_5, both *meanF* and *minF* are always zero. Best PSO solutions are in most of the cases obtained with the parameters $\omega = 0.005$, $\phi_p = \phi_g = 2$, and $\omega_G = 0.005$.

As stated above, this is a natural consequence of PSO being omniscient and keeping track of the best positions found by any agent. Nevertheless, as we show below, the solutions of social choice procedures are also very good and equal in practice for most problems.

Function / *PV* / *bPV*	*RD* / *bRD*	*PSO* / *DRD* / *bDRD*
f_1	*MeanF*	0.0 (0.005 2.0 0.005)
$8.46 \cdot 10^{-6}$	$7.50 \cdot 10^{-6}$	$1.26 \cdot 10^{-5}$ (0.1 2.0 0.005)
$7.63 \cdot 10^{-6}$	$6.93 \cdot 10^{-6}$	$1.45 \cdot 10^{-5}$ (0.4 2.0 0.005)
f_1	*MinF*	0.0 (0.005 2.0 0.005)
$2.20 \cdot 10^{-6}$	$1.58 \cdot 10^{-6}$	$6.97 \cdot 10^{-8}$ (0.05 2.0 0.005)
$9.83 \cdot 10^{-8}$	$1.00 \cdot 10^{-6}$	$8.50 \cdot 10^{-8}$ (0.01 2.0 0.005)
f_2	*MeanF*	0.0 (0.005 2.0 0.005)
0.0033	0.0033	0.0037 (0.01 2.0 0.005)
0.0034	0.0032	0.0041 (0.2 2.0 0.005)
f_2	*MinF*	0.0 (0.005 2.0 0.005)
0.0017	0.0010	$9.70 \cdot 10^{-4}$ (0.05 1.0 0.01)
$4.95 \cdot 10^{-4}$	$6.59 \cdot 10^{-4}$	$2.50 \cdot 10^{-4}$ (0.2 0.5 0.005)
f_3	*MeanF*	0.0 (0.005 2.0 0.005)
$1.00 \cdot 10^{-5}$	$1.10 \cdot 10^{-5}$	$1.68 \cdot 10^{-5}$ (0.4 1.0 0.005)
$9.62 \cdot 10^{-6}$	$9.98 \cdot 10^{-6}$	$2.05 \cdot 10^{-5}$ (0.2 2.0 0.005)
f_3	*MinF*	0.0 (0.005 2.0 0.005)
$8.78 \cdot 10^{-8}$	$5.94 \cdot 10^{-7}$	$7.41 \cdot 10^{-8}$ (0.2 2.0 0.005)
$2.87 \cdot 10^{-7}$	$2.98 \cdot 10^{-7}$	$6.56 \cdot 10^{-7}$ (0.1 1.0 0.005)
f_4	*MeanF*	0.0 (0.2 2.0 0.005)
0.0667	0.0050	0.0070 (0.4 2.0 0.005)
0.0284	0.0027	0.0054 (0.2 2.0 0.005)
f_4	*MinF*	0.0 (0.005 2.0 0.005)
0.0063	$1.57 \cdot 10^{-4}$	$7.67 \cdot 10^{-5}$ (0.4 2.0 0.005)
$9.52 \cdot 10^{-4}$	$4.64 \cdot 10^{-6}$	$4.36 \cdot 10^{-5}$ (0.005 0.2 0.1)
f_5	*MeanF*	-668.00 (0.05 2.0 0.4)
-7.8904	-7.8904	-7.8903 (0.005 2.0 0.05)
-7.8905	-7.8905	-7.8905 (0.005 1.0 0.01)
f_5	*MinF*	-837.96 (0.005 2.0 0.005)
-7.89	-7.89	-7.89 (0.1 2.0 0.05)
-7.89	-7.89	-7.89 (0.01 0.05 0.01)
f_6	*MeanF*	0.0 (0.005 2.0 0.05)
6.96	0.83	1.18 (0.05 1.0 0.05)
6.18	1.02	1.19 (0.1 1.0 0.05)
f_6	*MinF*	0.0 (0.005 2.0 0.005)
$5.40 \cdot 10^{-4}$	$3.93 \cdot 10^{-4}$	$2.91 \cdot 10^{-5}$ (0.1 2.0 0.005)
$2.83 \cdot 10^{-4}$	$2.30 \cdot 10^{-4}$	$1.23 \cdot 10^{-4}$ (0.05 0.5 0.01)
f_7	*MeanF*	$4.44 \cdot 10^{-16}$ (0.005 2.0 0.005)
0.40	0.09	0.14 (0.4 0.1 0.05)
0.25	0.11	0.16 (0.05 0.1 0.05)
f_7	*MinF*	$4.44 \cdot 10^{-16}$ (0.005 2.0 0.005)
0.0043	0.0036	$6.18 \cdot 10^{-4}$ (0.05 2.0 0.005)
0.0011	0.0017	0.0019 (0.005 0.5 0.005)
f_8	*MeanF*	0.0 (0.005 2.0 0.005)
$3.08 \cdot 10^{-6}$	$2.50 \cdot 10^{-6}$	$5.16 \cdot 10^{-6}$ (0.2 2.0 0.005)
$2.56 \cdot 10^{-6}$	$2.64 \cdot 10^{-6}$	$4.30 \cdot 10^{-6}$ (0.005 2.0 0.005)
f_8	*MinF*	0.0 (0.005 2.0 0.005)
$8.05 \cdot 10^{-7}$	$5.11 \cdot 10^{-7}$	$7.41 \cdot 10^{-9}$ (0.2 1.0 0.005)
$2.62 \cdot 10^{-7}$	$5.59 \cdot 10^{-8}$	$1.61 \cdot 10^{-8}$ (0.4 0.05 0.01)

Table 1. Optimal values obtained for the functions $f_1, \ldots, f_8$ using PSO and the three social choice procedures (for both voting strategies: always voting, only voting if better than global optimum). For each function we display *MeanF* (top) and *MinF* (bottom) objective functions achieved. For each pair (function, *MeanF*/*MinF*), we have on the first line: Name of function, value displayed, result using PSO (and parameters ω, $\phi_p = \phi_g$ and ω_G of the optimal result). On the second line we have the results obtained using PV, RD, and DRD and on the third line the results obtained using bPV, bRD, and $bDRD$.

For problems f_1, f_2, f_3, f_4 and f_8 (see Section 4), the optimal values achieved for PSO are 0 and social choice procedures give solutions with values at least less than 0.009, often very close to zero, the global solution. In particular, for f_8 we find a solutions with differentially private random dictatorship (DRD) with an objective function equal to $7.41 \cdot 10^{-9}$. See also solutions for f_1, f_2, f_3, f_4 in Table 1 that are virtually zero. Note that in the table we display both mean values (*MeanF*) and the best solution found (*MinF*).

For problem f_5 (with global minimum of -12569.5), PSO obtains a *meanF* value of -668, and the best of the 30 executions leads to *minF* equal to -837. Social choice solutions (both for *meanF* and *minF*) have an optimal value of around -7. These are the worst results for both PSO and social choice procedures.

For the other problems, f_6, f_7 the best social choice solutions are *meanF* $= 0.83$ and *minF* $= 2.91 \cdot 10^{-5}$, and *meanF*$=0.0979$ and *minF*$=6.18 \cdot 10^{-4}$, respectively.

Social choice procedures and differentially private random dictatorship Among the social choice procedures, for most of the problems the best solutions are either random dictatorship or differentially private random dictatorship. In some cases solutions are better by a factor of 10 or 100 to the one obtained with plurality voting. So, we can state that randomness is not an inconvenience but an advantage.

Only for f_3 the best solutions are obtained using plurality voting. However, in this case, the values achieved by the three social choice procedures are very similar. Observe in Table 1 that the values *meanF* are $9.62 \cdot 10^{-6}$ for bPV and $9.984 \cdot 10^{-6}$ for bRD.

When we compare the case of agents always voting and the case of agents only voting when they have a position better than the global one, we have, as expected, that in most cases, results are better when only those agents with better positions vote.

With respect to parameters of the best solutions, there is more variety here than when using PSO. We can observe in the table that DPD and dDPD has most solutions with $\phi_g = \phi_p = 2.0$ but the best solutions for f_2 are with $\phi_g = \phi_p = 0.5$ and for f_3 are with $\phi_g = \phi_p = 1.0$.

Figure 2 shows the evolution of the system particle when differentially private random dictatorship is used. We can see that except for the problem f_6 the system particle tends to move to the optimal solution. For f_5 the optimal solution found is far from optimal. This evolution is much faster when particles only vote when they know to be in a better position than the system particle. This is illustrated in Figure 3.

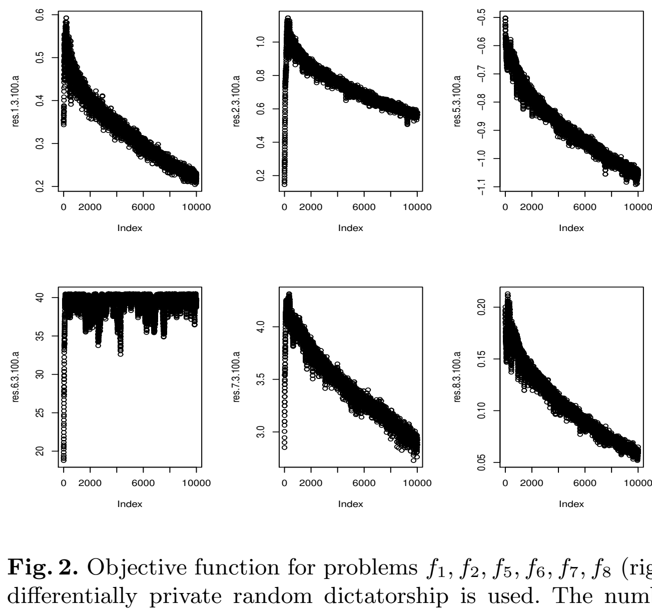


Fig. 2. Objective function for problems $f_1, f_2, f_5, f_6, f_7, f_8$ (right to left, top to bottom) listed above when differentially private random dictatorship is used. The number of alternatives considered is 8 and the number of particles is 100.

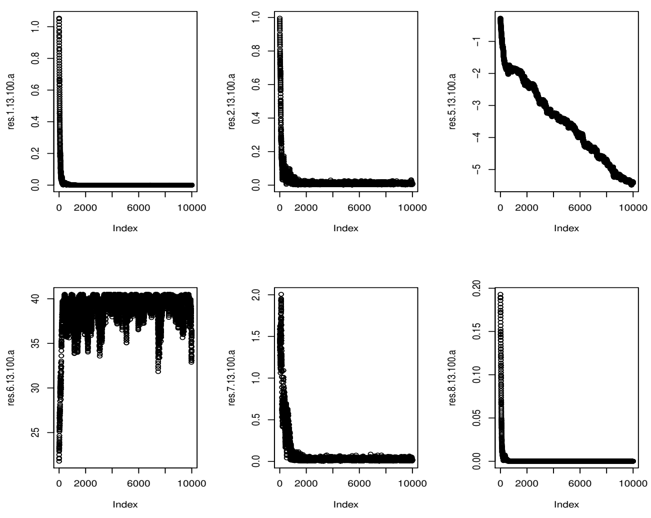


Fig. 3. Objective function for problems $f_1, f_2, f_5, f_6, f_7, f_8$ (right to left, top to bottom) when differentially private random dictatorship is used and when particles only vote if they know to be in a better position. The number of alternatives considered is 8 and the number of particles is 100.

To illustrate that the plurality vote is not always the best alternative, we show the results obtained for Ackley's function f_7. Figure 4 shows the results for the three social choice procedures: plurality rule (left), random dictatorship (middle), and differentially private random dictatorship according to Method 2 (right). Dots correspond to the case of all particles always voting, and lines to the case that only those particles with a better position vote. It can be clearly seen that the plurality rule is not best, and that voting only when a better position is found is clearly better. Recall that the optimal solution for this problem is when the function is exactly zero.

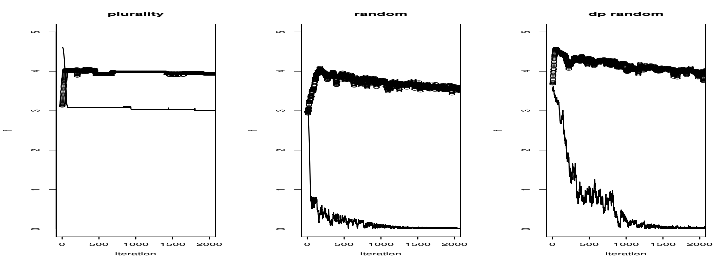


Fig. 4. Objective function for the Ackley's function f_7 when plurality rule (left), random dictatorship (middle), and differentially private random dictatorship (right) are used. Dots correspond to all particles voting and lines to only those particles having a better solution than the system particle voting. The number of alternatives considered is 8. Random dictatorship only voting when solutions are better is the one with the fastest convergence.

Summary The results of our experiments can be summarised stating that

- standard PSO (with panopticon) is the most effective approach considered, but PSO without panopticon is also quite effective and some solutions have no significant difference,
- among social choice procedures implementing our variation of PSO (without panopticon), plurality voting is usually not the best option,
- particles voting only when their solution is better is a better approach than particles voting in all occasions, and, last but not least,

– differentially private random dictatorship can be seen as comparable to random dictatorship.

Therefore, we consider that a qualitative conclusion is that differentially private random dictatorship is a suitable approach to be used in this type of scenario.

5 Conclusions

This paper focuses on the evaluation of differentially private social choice and more particularly on differentially private random dictatorship. It is standard to evaluate data privacy mechanisms in terms of their utility or in terms of the loss they cause. Social choice mechanisms are not so straightforward to evaluate because the preferences or *opinions* of the agents are assumed to be expressed in ordinal terms. This is an important assumption. Our approach permits to evaluate social choice under these assumptions. We have proposed a scenario based on an objective function to optimise by a set of agents, and use a PSO-like procedure for obtaining the best solution through an iterative voting procedure. When numerical evaluations of the preferences exist (through e.g. utility functions), other mechanisms (as aggregation of utility functions) should be used.

We have shown that in our scenario, the results of differentially private random dictatorship are similar to those for random dictatorship, and usually better than those obtained with the plurality voting (i.e., selecting the most preferred option).

As future work we plan to study agents with different privacy requirements (e.g., privacy budgets) and how these different privacy requirements can affect the outcome of the system. Among the privacy options to consider, we have the case that agents want to refrain from voting (opt-out). Our experiment results used 50 particles in two dimensional problems, we will explore the case of larger number of particles and larger dimensions for the problem.

Acknowledgements

Discussions with Guillermo Navarro-Arribas are gratefully acknowledged.

This study was partially funded by the Swedish Research Council (Vetenskapsrådet) (grant number VR 2016-03346), and by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

1. Barberà, S. (1979) Majority and positional voting in a probabilistic framework, Review of Economici Studies 42:2 379-389.
2. Brandt, F. (2017) Rolling the dice: recent results in probabilistic social choice, in U. Endriss (ed.) Trends in computational choice, AI Access 3-26.
3. Dwork, C. (2006) Differential privacy, Proc. ICALP 2006, LNCS 4052, pp. 1-12.
4. Gibbard, A. (1977) Manipulation of schemes that mix voting with chance, Econometrica 45:3 665-681.
5. Hundepool, A., Domingo-Ferrer, J., Franconi, L., Giessing, S., Nordholt, E. S., Spicer, K., de Wolf, P.-P. (2012) Statistical Disclosure Control, Wiley.
6. Kennedy, J., Eberhart, R. C. (1995) Particle swarm optimization, Proc. IEEE Int. Conf. Neural Networks 1942-1948.
7. Li, T., Sahu, A. K., Talwalkar, A., Smith (2019) Federated learning: challenges methods, and future directions, Arxiv.
8. Samarati, P. (2001) Protecting Respondents' Identities in Microdata Release, IEEE Trans. on Knowledge and Data Engineering, 13:6 1010-1027.
9. Sen, A. (2017) Collective choice and social welfare, Penguin Books
10. Sengupta, S., Basak, S., Peters II, R. A. (2018) Particle swarm optimization: a survey of historical and recent developments with hybridization perspectives, Machine learning and knowledge extraction 1 157-191.
11. Torra, V. (2017) Data privacy: Foundations, new developments and the big data challenge, Springer.
12. Torra, V. (2020) Random dictatorship for privacy-preserving social choice, Int. J. of Information Security 19 537-545.
13. Vaidya, J., Clifton, C., Zhu, M. (2006) Privacy Preserving Data Mining, Springer.

A Biased Random-key Genetic Algorithm for the 2-Dimensional Guillotine Cutting Stock Problem with Stack Constraints

M. Guimaraes[1], E. T. Bogue[2], I. A. Carvalho[1], T. F. Noronha[1], A. H. Pereira[1] and S. Urrutia[3]

[1] Departamento de Ciência da Computação,
Universidade Federal de Minas Gerais, Belo Horizonte, Brazil
[2] Departamento de Ciência da Computação,
Universidade Federal de Mato Grosso do Sul, Ponta Porã, Brazil
[3] Faculty of Logistics,
Molde University College, Molde, Norway

1 Introduction

The problem of cutting a large plate of raw material into a specified set of smaller objects is a common industrial challenge in glass [1], paper [2], wood [3], and steel [4] industries, among others. This problem is referred as the 2-dimensional cutting stock problem (2DCSP for short) [5]. It aims at cutting all the smaller objects with the minimum amount of raw material. In this paper, we refer to the large plate of raw material simply as *plate* and to the smaller objects as *items*. Besides, as in [6–9], we assume that they are two-dimensional and have a rectangular shape.

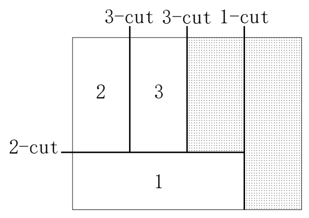


Fig. 1: An example of a cutting pattern.

We focus on the variant of 2DCSP called the 2-dimensional 3-staged cutting stock problem subject to guillotine constraints (2DCSP-3S for short) [6, 7, 9]. In this variant, only *guillotine cuts* are allowed, *i.e.*, cuts that go from one side to the opposite side of the plate and split it into two rectangular pieces. These cuts are divided into *stages*, where each cutting stage consists of a sequence of parallel guillotine cuts. At each stage, the cuts are orthogonal to those of the previous stage, since each piece of plate is rotated by 90° before the next cutting stage begins. We refer to a k-cut as a guillotine cut performed in the k-*th* stage. Besides, we assume that the plate is oriented such that its width is larger than its height, and that the odd staged cuts are vertically oriented, while the even staged cuts are horizontally oriented. An example of a 3-stage cutting pattern used to separate three items from a plate is shown in Figure 1, where the items are numbered from 1 to 3 and the unused pieces of plate are shadowed.

This cutting pattern can be interpreted as a tree, where the root node (at level 0) corresponds to the whole plate, and each node in the k-th level of the tree corresponds to a piece of plate obtained from a k-cut to the piece of plate of its parent node. Therefore, the leaves of this tree corresponds to either items or unused pieces of plate. It is assumed that cuts are performed using a depth first approach in this tree to avoid changing the piece of plate that is in the guillotine. An example of the tree representing the cutting pattern of Figure 1 is given in Figure 2. First, a vertical 1-cut is applied to the root node to detach an unused piece of plate. Next, a horizontal 2-cut is performed to extract the item 1. Then, two successive vertical 3-cuts are executed to obtain items 2 and 3, as well as another unused piece of plate.

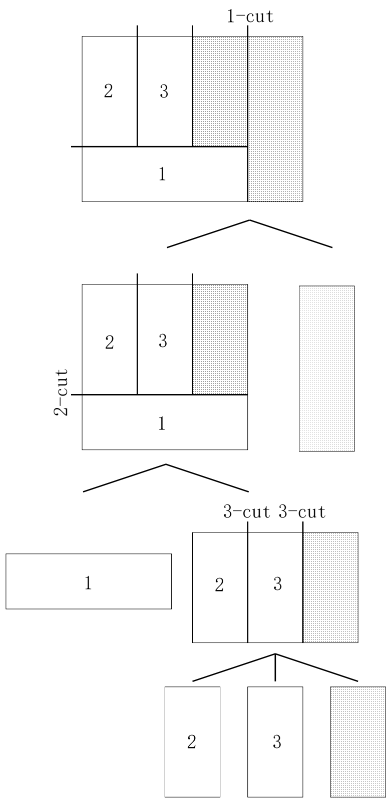


Fig. 2: Representation of the cutting pattern of Figure 1 as a tree.

In the case of 2DCSP-3S, the maximum number of stages is limited to three. However, a single additional 4-cut is allowed if and only if it is used to separate a single item from an unused piece of plate [6, 10–12]. This is known in the literature as *trimming*. Figure 3a shows an example of trimming, where the item 3 is separated from an unused piece of plate by a single 4-cut, while Figure 3b gives an example of an invalid 4-cut used to separate item 3 from item 4.

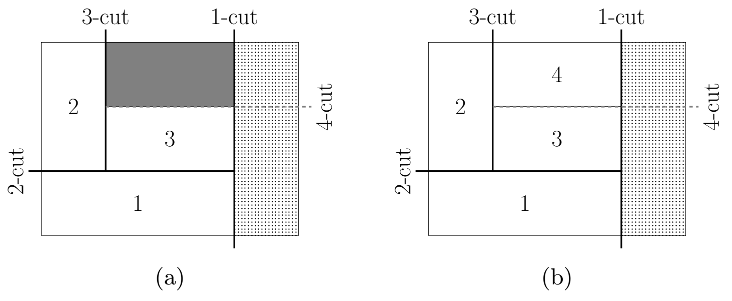


Fig. 3: Example of a *4-cut* cut allowed (a) and not allowed (b).

In this paper, we deal with a variant of 2DCSP-3S that has additional precedence constraints that was recently introduced in [13]. In this case, the items are organized in stacks, where each stack represents a customer request and defines the order in which the items must be cut. That is, if item i precedes item j within a stack, then i must be cut before j. However, there is no precedence constraint between items in different stacks. This constraint comes from applications where items must be stacked and shipped in the exact order that they will be used by the customer, thus avoiding the risk of damaging fragile items (as is the case in the glass industry) or the cost of moving heavy items (as is the case in the steel industry). We refer to this variant of 2DCSP-3S as the 2-dimensional Guillotine Cutting Stock Problem with Stack Constraints (2DCSP-SC). This problem is formally defined below.

Let W and H be the width and the height of the plates, respectively, and I be the set of items to be cut, where each item $i \in I$ has height h_i and width w_i. Besides, let S be the set of stacks that represent customer orders, whereas $b = (\pi_1^s, \pi_2^s, \ldots, \pi_{n_s}^s)$ describes the order in which the items from stack $s \in S$ must be cut, such that $\pi_k^s \in I$ must be cut before $\pi_{k+1}^s \in I$, for all $k = \{1, \ldots, n_s - 1\}$, where n_s is the number of items in s.

A solution to 2DCSP-SC consists of a sequence of cutting patterns P that describes how, and in which order, the plates must be cut. This solution must satisfy all the following constraints: (i) the plate cannot be rotated; (ii) the items can only be rotated by 90°; (iii) all items in I must be cut exactly once; (iv) if $i \in I$ precedes $j \in I$ in a stack, then i must be cut before j; (v) only guillotine cuts are allowed; and (vi) the number of cutting stages is at most three along with the additional 4-cut, as previously described.

The cost $f(P)$ of a solution P to 2DCSP-SC is the amount of raw material used to cut all items. As in [6,14–16], the unused pieces of plate that result from the cutting patterns P are divided into two types. The so called *leftover* is the material to the right of the last 1-cut applied to the last plate. It is assumed that this piece of plate can be reused, and it is not considered in $f(P)$. All the other unused pieces of plate are considered *waste*, as it is assumed that they cannot be reused. For example, in Figure 3a, the unused piece of plate colored in gray is considered waste, while that filled with dots is considered leftover. The objective function $f(P)$ is defined as

$$f(P) = H \cdot W \cdot (|P| - 1) + H \cdot r(P),$$

where $|P|$ denotes the number of plates used, $H \cdot W \cdot (|P| - 1)$ is the total area of the first $|P| - 1$ plates. Furthermore, $r(P)$ gives the position of the last *1-cut* on the last cutting pattern of P. Thus, $H \cdot r(P)$ represents the used area of the last plate.

Let Δ be the set of feasible solutions for 2DCSP-SC. This problem consists of finding a solution $P^* = \text{argmin}_{P \in \Delta} f(P)$, *i.e.*, the cutting patterns that use the least amount of raw material to cut all items in I. When there is only one item per stack, this problem reduces to 2DCSP-3S. Since 2DCSP-3S is NP-Hard [17], 2DCSP-SC is also NP-Hard.

As there is no known technique to design a polynomial-time exact algorithm for NP-Hard problems, this paper focus on heuristic algorithms. However, as far as we can tell, the new precedence constraints introduced in [13] preclude the use of most algorithms in the literature related to 2DCSP-3S [11,14,15], except those in [18], because they were not designed to consider item precedence. Therefore, this paper adapts an Evolutionary Algorithm (EA) described in [18], and also proposes a Biased Random-key Genetic Algorithm (BRKGA) to address the 2DCSP-SC. Computational experiments show that BRKGA outperforms the EA of the literature.

The remainder of this paper is organized as follows. First, related work are discussed in Section 2. Next, a constructive algorithm for 2DCSP-SC, which is used as a decoder for the other heuristics, is proposed in Section 3. Then, the adaptation to 2DCSP-SC of the Evolutionary Algorithm of [18] is detailed in Section 4, and the proposed Biased Random-key Genetic Algorithm is described in Section 5. Finally, computational experiments are reported in Section 6 and concluding remarks are drawn in the last section.

2 Related work

A Sequential Heuristic Procedure (SHP) was proposed in [19]. The first stage of this heuristic selects the height of the cut, the second stage the length of the cut, and the third stage the number of times the generated cut pattern will be used. The authors concluded that the performance of the proposed heuristic is better than heuristics that use fixed measures to define the sizes of the cuts. A variant of 2DCSP-3S in which the plates may contain defects and vary in size was addressed in [20]. The proposed heuristic first sorts the larger sides of the plates in ascending order, and the items are sorted in decreasing order following the same criteria. The algorithm tries to position the widest items on the smallest plates and after all the items are positioned, it checks if any item was placed in any defective area; if so, that item is removed, and the possibility of being added to any of the plates already used is verified. Computational experiments showed that these heuristics obtained better results than those presented in [21].

A heuristic procedure based on Variable Neighborhood Search (VNS) was proposed in [22]. To build an initial solution, three heuristics based on the first-fit approach of [23] were used (3-staged First Fit Decreasing Height with rotations, matching step and Fill Strip), so that the best solution provided by them is selected. Computational experiments concluded that the heuristic proposed in this work provided better solutions than the VNS approach present in [23].

A heuristic that combines a recursive approach and a Beam Search algorithm was proposed in [15]. Unlike branch and bound algorithms, in Beam Search, only elite nodes with high potential are investigated [24]. In this approach, the recursion is used to generate segments of strips, and a Beam Search heuristic is used to obtain the 3-staged cutting patterns considering usable left-over. Computational experiments showed that the heuristic proposed in this work obtained better solutions than those of [6].

A Finite First Fit Heuristic (FFF), an Evolutionary Algorithm (EA), and two strategies based on branch-and-bound have been proposed to solve 2DCSP-3S in [18]. Computational experiments showed that EA obtained better results than the other heuristics. As far as we know, FFF and EA are the best heuristics in the literature that can be adapted to handle the precedence constraint of 2DCSP-SC. Therefore, they are adapted to 2DCSP-SC in sections 3 and 4, respectively.

3 Finite First Fit (FFF) heuristic

In this section, we extend the FFF heuristic of [18] to 2DCSP-SC. This heuristic assumes that all the items are oriented in such a way that their width is greater than or equal to their height, and it does not perform any further rotations. It starts with an empty solution, and, at each iteration, it inserts an item in the solution using the first fit approach described in Algorithm 1.

FFF inserts the items in the order they appear in a permutation Π of the items in I. This permutation is such that if an item i precedes an item j in any stack, i precedes j in Π. This property is necessary to guarantee that there is always a place to insert an item without violating the precedence constraints. That is the case because when an item is inserted in the solution, all preceding items have already been inserted. In the worst case scenario, the next item can be inserted in a new empty plate. The sorting algorithm used to generate permutations that satisfy this property is explained in the next section.

Instead of describing Algorithm 1 using the usual recursive tree representation of a solution, we adopt a novel representation based on what we called a *k-box* representation. A 0-box by definition is a plate. Therefore, it has always height H and width W. Such a box is divided into a sequence of 1-boxes. Therefore, a 1-box has always height H, but can have variable width. It is assumed that the 1-boxes are placed contiguously from the left to the right of their respective 0-box. Therefore, all the area of the 0-box not covered by a 1-box is unused area (waste or leftover). Analogously, a 1-box is divided into a sequence of 2-boxes. Therefore, a 2-box has always the same width of its respective 1-box, but can have variable height. It is assumed that the 2-boxes are placed contiguously from the bottom to the top of their respective 1-box. Therefore, all the area of the respective 1-box not covered by a 2-box is waste. Similarly, a 2-box is divided into a sequence of 3-boxes, each one associated to exactly one item. Therefore, a 3-box has always the same height of its respective 2-box, but its width is exactly the same as that of its corresponding item. It is worth pointing out that, from the width of the 3-box, one can infer if the corresponding item was rotated or not. Besides, it is implicit that if the height of a 3-box is larger than that of the corresponding item, a trimming 4-cut occurs. It is also assumed that the 3-boxes are placed contiguously from the left to the right of their respective 2-boxes. Therefore, all the area of the respective 2-cut not covered by a 3-box is waste. It can be observed that there is a direct correspondence between a k-box and a level k node in the cutting pattern tree. The advantage of the k-box representation is that one does not need to account to the exact position of the nested k-cuts, but only to the size of the corresponding k-boxes, which can be inferred from the width and height of the items in each box.

Algorithm 1 *Finite First-Fit* (FFF)

Input: $\mathcal{I}$ and Π
Output: P

```
 1: $P \leftarrow [\,]$
 2: for each $i$ in $\Pi$ do
 3:     for each $b^0$ in $P$ do
 4:         for each $b^1$ in $b^0$ do
 5:             for each $b^2$ in $b^1$ do
 6:                 if $i$ fits in $b^2$ respecting the precedence constraint then
 7:                     $b^3 \leftarrow i$, and $b^2 \leftarrow b^2 : b^3$
 8:                     continue to the next item in $\Pi$
 9:                 end if
10:             end for
11:             if $[i]$ fits in $b^1$ respecting the precedence constraint then
12:                 $b^2 \leftarrow [i]$, and $b^1 \leftarrow b^1 : b^2$
13:                 continue to the next item in $\Pi$
14:             end if
15:         end for
16:         if $[[i]]$ fits in $b^0$ respecting the precedence constraint then
17:             $b^1 \leftarrow [[i]]$, and $b^0 \leftarrow b^0 : b^1$
18:             continue to the next item in $\Pi$
19:         end if
20:     end for
21:     $b^0 \leftarrow [[[i]]]$, and $P \leftarrow P : b^0$
22: end for
23: return $P$
```

The pseudo-code of FFF is described in Algorithm 1. This heuristic receives an instance $\mathcal{I}$ of 2DCSP-SC and a permutation Π of the items in I, and returns a solution corresponding to a sequence of cutting patterns P that describes how to cut all the items in I without breaking the precedence constraints. In line 1, an empty partial solution P is initialized. At each iteration of the loop of lines 2 to 22, the next item i, according to the permutation Π, is inserted in the solution. Next, at each iteration of the loop of lines 3 to 20, FFF scans every 0-box b^0 in the order they appear in P. Then, at each iteration of the loop of lines 4 to 15, FFF evaluates every 1-box b^1 in the order they appear in b^0. Following, at each iteration of the loop of lines 5 to 10, FFF inspects every 2-box b^2 in the order they appear in b^1. If the item i fits in b^2 (see line 6), a new 3-box with i is initialized and is appended to b^2 in line 7, and the heuristic continues to the next item in Π in line 8. It is worth noting that only the 2-boxes whose items are cut after those that precede i are considered. Otherwise, i could be cut before an item that precedes it. On the other hand, if i does not fit in b^2 but it fits as a new 2-box on top of b^1 without breaking the precedence constraint (see line 11), a new 2-box (containing a single 3-box with i) is appended to b^1 in line 12, and FFF continues to the next item in Π in line 13. Moreover, if the latter is not possible, but i fits as a new 1-box to the right of b^0 without breaking the precedence constraint (see line 16), a new 1-box (containing a single 2-box with i) is appended to b^0 in line 17, and FFF continues to the next item in Π in line 18. Finally, if i does not fit any box of the current solution, a new 0-box (plate) is appended to P in line 21 with a single 1-box containing i. When all the items are inserted in P the solution is returned in line 23. This heuristic is used as a decoder in both the Evolutionary Algorithm of [18] and the Biased Random-key Genetic Algorithm introduced in Sections 4 and 5, respectively.

4 An Evolutionary Algorithm for 2DCSP-SC

In this section, the Evolutionary Algorithm (EA) of [18] is adapted to address 2DCSP-SC. This algorithm represents a solution as an $|I|$-vector, in which each component is a real number (referred to as *key*) in the range $[0, 1]$ associated with an item in I. Each solution is decoded by a decoding heuristic that receives the vector of keys and builds a feasible solution for 2DCSP-SC. Let k_i be the key associated with the item $i \in I$ and w_i be the width of i, the decoding of EA consists of two steps. First, a permutation Π is generated accordingly to the following selection sort algorithm. Let $\rho_i = k_i \cdot w_i$, at each iteration of the sorting algorithm, the item with the largest value of ρ_i, that is on top of a stack in S, is popped from its stack and added to the end of Π. Then, the FFF heuristic is run with the resulting permutation of items. The cost of the solution returned by FFF is used as the fitness of the chromosome.

EA is a steady-state evolutionary algorithm, where each offspring replaces the worst solution in the population. Initial solutions are created at random, and at each iteration two parent solutions are selected randomly. Then, the Order 3 Crossover (OX3) of [25] is applied to these solutions to generate a new offspring. Two mutation operators are used: (*i*) the Reciprocal Exchange (RX), which chooses two items at random and swaps their keys; and (*ii*) the Block Exchange (BX), which swaps the keys of two non-overlapping blocks of consecutive items. The size of these blocks is set to $\lceil 2^R \rceil$, as suggested by [18], where R is a random value in the interval $(0, \lfloor ld\frac{n}{2} \rfloor]$, in order to allow shorter blocks to be chosen more likely. The number of mutations applied to each new offspring solution is chosen as a Poisson-distributed random variable with expected value 2. Every time a mutation is applied to the offspring, either RX or BX is randomly chosen with equal probability.

5 Biased random-key genetic algorithm

Random-key Genetic Algorithms (RKGA) were first introduced by Bean [26] for combinatorial optimization problems for which solutions can be represented as a permutation vector. In this approach, two parents are selected at random from the entire population to implement the crossover operation in the implementation of a RKGA. Parents are allowed to be selected for mating more than once in a given generation.

A Biased Random-key Genetic Algorithm (BRKGA) differs from a RKGA in the way parents are selected for crossover, see Gonçalves and Resende [27] for a review. In a BRKGA, each element is generated combining one element selected at random from the elite solutions in the current population, while the other is a non-elite solution. We say the selection is biased since one parent is always an elite individual and because this elite solution has a higher probability of passing its genes to the offsprings, *i.e.*, to the new generation. A BRKGA provides a better implementation of the essence of Darwin's principle of "survival of the fittest" than the RKGA, since an elite solution has a higher probability of being selected for mating and the offsprings have a higher probability of inheriting the genes of the elite parent.

The BRKGA for 2DCSP-SC evolves a population of chromosomes that consists of $|I|$-vectors of keys, which are decoded exactly as in the Evolutionary Algorithm described in Section 4. We use the parameterized uniform crossover scheme proposed in [28] to combine two parent solutions and produce an offspring. In this scheme, the offspring inherits each of its keys from the best fit of the two parents with probability $\rho > 0.5$ and from the least fit parent with probability $1 - \rho$. BRKGA do not make use of the standard mutation operator, where parts of the chromosomes are changed with a small probability. Instead, the following concept of mutants is used: a fixed number of mutant solutions are introduced in the population in each generation, randomly generated in the same way as in the initial population. Mutants play the same role of the mutation operator in traditional genetic algorithms, diversifying the search and helping the procedure to escape from locally optimal solutions.

The keys associated to each item are randomly generated in the initial population. At each generation, the population is partitioned into two sets: TOP and $REST$. Consequently, the size of the population is $|TOP| + |REST|$. Subset TOP contains the best solutions in the population. Subset $REST$ is formed by two disjoint subsets: MID and BOT, with subset BOT being formed by the worst elements on the current population. As illustrated in Figure 4, the chromosomes in TOP are simply copied to the population of the next generation. The elements in BOT are replaced by newly created mutants that are placed in the new set BOT. The remaining elements of the new population are obtained by crossover with one parent randomly chosen from TOP and the other from $REST$. This distinguishes a biased random-key GA from the random-key genetic algorithm of Bean [26] (where both parents are selected at random from the entire population). Since a parent solution can be chosen for crossover more than once in a given generation, elite solutions have a higher probability of passing their random keys to the next generation. In this way, $|MID| = |REST| - |BOT|$ offspring solutions are created. The algorithm stops when a maximum elapsed time is reached.

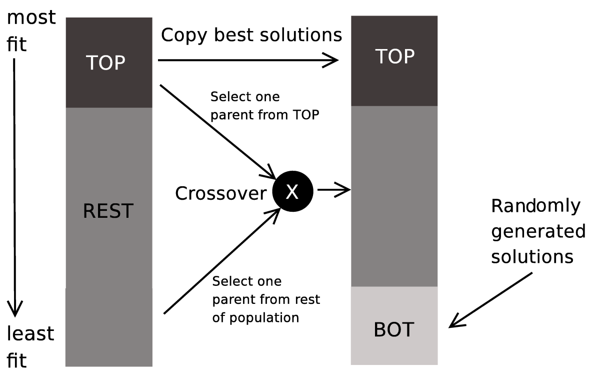

most fit
least fit
TOP
REST
Copy best solutions
Select one parent from TOP
Crossover
X
Select one parent from rest of population
TOP
BOT
Randomly generated solutions

Fig. 4: Population evolution between consecutive generations of a BRKGA.

6 Computational experiments

The computational experiments reported in this section evaluates the performance of the BRKGA and the EA heuristics. These algorithms were implemented in C++ and compiled with GNU *gcc* version 6.3. The population size of both heuristics was set to 1000 solutions, and the stopping criteria was set to 10 minutes of running time. All experiments were performed in a single core of an Intel Xeon machine with 2.00 GHz of clock speed and 16 GB of RAM. As both BRKGA and EA relies on stochastic operators, we ran these heuristics 20 times for each instance using different seeds for the Mersenne Twister pseudo-random number generator [29].

Three sets of instances were used in the experiments, namely *Set A*, *Set B*, and *Set X*. These instances were adapted from realistic ones employed in the ROADEF Challenge 2018, which tackled a similar problem commissioned by Saint-Gobain Glass France, which is one of the world's leading flat glass manufacturers. The original instances resemble scenarios found in Saint-Gobain factories and can be retrieved from the website `http://www.roadef.org/challenge/2018/en/instances.php`. We adapted these instances by ignoring the data regarding specific constraints of Saint-Gobain's guillotines, and using only the data necessary for 2DCSP-SC. In every instance, the plates have $W = 6000$ and $H = 3210$. Table 1 presents the characteristics of each instance set. One can see that the instances greatly vary. The number of stacks range from 1 to 247, while the number of items stretch from 5 to 656. Furthermore, the smallest width of an item is only 345, while the maximum width of an item is equal to 3495. Additionally, the height of the items also vary between 123 and 2010.

Table 1: Characteristics of each instance set

	Set A	Set B	Set X
Number of instances	20	15	15
Min. number of itens	5	68	124
Ave. number of items	101.95	303.87	284.33
Max. number of items	392	656	412
Min. number of stacks	1	2	2
Ave. number of stacks	11.20	32.00	28.53
Max. number of stacks	72	241	247
Min. item width	345	351	353
Ave. item width	1317.35	1410.69	1317.40
Max. item width	3495	2952	2813
Min. item height	137	123	193
Ave. item height	594.82	668.63	600.03
Max. item height	2010	1759	1828

The results for this experiment are displayed in Tables 2 to 4, whereas each table gives the result for a different instance set. The first column of each table displays the instance name, while the second column presents a lower bound (LB) computed as $\sum_{i \in I} w_i h_i$, *i.e.*, the sum of the area of all items in I. The third and fourth columns present the results for the BRKGA. The third column presents the relative optimality gap of the BRKGA's fitness computed against the lower bound displayed in the second column, while the fourth column gives the coefficient of variation (cv) of the algorithm's results. The same information is given for the EA in the fifth and sixth columns, respectively. The last line of each table presents the average relative optimality gap and the average coefficient of variation for each heuristic.

Table 2: Results for the Set A of instances

		BRKGA		EA	
instance	LB	gap (%)	cv (%)	gap (%)	cv (%)
A1	4514704	10.87	0.00	10.87	0.00
A2	77201851	13.32	1.74	13.64	0.92
A3	41796990	19.79	1.10	20.80	1.19
A4	41796990	19.90	0.92	21.06	1.17
A5	56570007	15.55	1.25	17.77	1.16
A6	43254870	14.40	0.82	15.81	0.11
A7	70195170	20.02	1.16	22.25	1.17
A8	138045196	20.53	1.92	22.45	0.80
A9	44879034	21.47	4.63	25.53	0.81
A10	71100239	21.33	1.36	27.08	2.44
A11	64444211	19.18	0.23	21.37	1.43
A12	29180006	20.52	2.58	29.01	0.71
A13	213400977	11.75	0.73	14.20	1.02
A14	226360542	17.98	0.99	20.69	1.03
A15	238633039	16.87	1.33	20.07	0.71
A16	37325677	22.37	0.00	23.59	1.21
A17	19623149	54.83	0.00	54.83	0.00
A18	60282102	22.42	2.00	27.06	2.25
A19	41044876	20.33	1.23	28.23	4.46
A20	14710475	35.25	0.00	35.25	0.00
Average		20.93	1.19	23.58	1.12

One can see from these tables that the relative optimality gap of BRKGA was smaller or equal than that of EA for all evaluated instances. BRKGA obtained an average relative optimality gap of 20.93%, 12.48%, and 13.89% for the sets A, B, and X of instances, respectively, while that of EA was 23.58%, 14.68%, and 16.58%. Therefore, it can be concluded that BRKGA obtained better results than EA when solving the proposed instances. However, one can observe that EA is a more stable method than BRKGA, as its average coefficient of variation was smaller than that of BRKGA for all sets of evaluated instances.

7 Concluding remarks

In this work, we tackled the Two-dimensional Three-staged Cutting Stock Problem with Stack Constraints (2DCSP-SC). We extended the Evolutionary Algorithm (EA) described in [18] to address the 2DCSP-SC. Furthermore, we proposed a Biased Random Key Genetic Algorithm (BRKGA). Both algorithms use the Finite First Fit (FFF) heuristic as a decoder. Computational experiments, performed on three sets of realistic instances, show that BRKGA found solutions with smaller optimally gaps in all but one of the instances tested.

Future works may explore exact methods, such as branch-and-bound algorithms, to improve the lower bounds proposed in this paper. Alternatively, other heuristic methods that do not rely in Genetic Algorithms could be devised for the problem, such as heuristics based on local search.

Table 3: Results for the Set B of instances

		BRKGA		EA	
instance	LB	gap (%)	cv (%)	gap (%)	cv (%)
B1	77110392	10.12	1.00	9.11	0.24
B2	315354085	18.47	1.13	21.04	0.66
B3	349989487	15.36	0.72	16.79	0.62
B4	148205615	15.98	0.23	18.17	0.74
B5	319711555	37.81	0.00	37.81	0.00
B6	192874073	15.69	1.45	18.89	0.87
B7	187746291	11.82	1.41	17.17	1.03
B8	339397811	13.44	0.65	16.68	0.68
B9	293827643	12.06	0.61	14.35	0.82
B10	345904837	14.32	1.23	18.47	0.99
B11	336052870	14.34	0.84	18.81	0.53
B12	259876763	18.19	0.80	22.51	0.54
B13	484072875	17.81	0.94	22.36	0.66
B14	176124110	17.05	1.32	21.33	0.88
B15	432558079	17.07	0.71	20.20	0.91
Average		12.48	0.86	14.68	0.67

References

1. F. Parreño, M. Alonso, and R. Alvarez-Valdes, "Solving a large cutting problem in the glass manufacturing industry," *European Journal of Operational Research*, vol. 287, pp. 378–388, 2020.
2. A. A. Leao, M. M. Furlan, and F. M. Toledo, "Decomposition methods for the lot-sizing and cutting-stock problems in paper industries," *Applied Mathematical Modelling*, vol. 48, pp. 250–268, 2017.
3. J. L. A. P. Galvez, D. Borenstein, and E. da Silveira Farias, "Application of optimization for solving a sawing stock problem with a cant sawing pattern," *Optimization Letters*, vol. 12, no. 8, pp. 1755–1772, 2018.
4. J. Karelahti, "Solving the cutting stock problem in the steel industry," *Helsinki University of Technology*, pp. 1–39, 2002.
5. S. Israni and J. Sanders, "Two-dimensional cutting stock problem research: A review and a new rectangular layout algorithm," *Journal of Manufacturing Systems*, vol. 1, no. 2, pp. 169–182, 1982.
6. R. Andrade, E. G. Birgin, and R. Morabito, "Two-stage two-dimensional guillotine cutting stock problems with usable leftover," *International Transactions in Operational Research*, vol. 23, no. 1-2, pp. 121–145, 2016.
7. M. Aryanezhad, N. F. Hashemi, A. Makui, and H. Javanshir, "A simple approach to the two-dimensional guillotine cutting stock problem," *Journal of Industrial Engineering International*, pp. 8–21, 2012.
8. E. G. Birgin, O. C. Romão, and D. P. Ronconi, "The multiperiod two-dimensional non-guillotine cutting stock problem with usable leftovers," *International Transactions in Operational Research*, vol. 27, no. 3, pp. 1392–1418, 2020.
9. F. Clautiaux, R. Sadykov, F. Vanderbeck, and Q. Viaud, "Pattern-based diving heuristics for a two-dimensional guillotine cutting-stock problem with leftovers," *EURO Journal on Computational Optimization*, vol. 7, no. 3, pp. 265–297, 2019.
10. R. Alvarez-Valdes, R. Martí, J. M. Tamarit, and A. Parajón, "Grasp and path relinking for the two-dimensional two-stage cutting-stock problem," *INFORMS Journal on Computing*, vol. 19, no. 2, pp. 261–272, 2007.
11. M. Hifi and C. Roucairol, "Approximate and exact algorithms for constrained (un) weighted two-dimensional two-staged cutting stock problems," *Journal of combinatorial optimization*, vol. 5, no. 4, pp. 465–494, 2001.
12. H. H. Yanasse and R. Morabito, "Linear models for 1-group two-dimensional guillotine cutting problems," *International Journal of Production Research*, vol. 44, no. 17, pp. 3471–3491, 2006.
13. T. Lydia and V. Quentin, "Challenge roadef - euro 2018 cutting optimization problem description," 2018. [Online]. Available: https://www.roadef.org/challenge/2018/files/Challenge_ROADEF_EURO_SG_Description.pdf
14. Y. Cui, X. Song, Y. Chen, and Y. Cui, "New model and heuristic solution approach for one-dimensional cutting stock problem with usable leftovers," *Journal of the Operational Research Society*, vol. 68, no. 3, pp. 269–280, 2017.

Table 4: Results for the Set X of instances

		BRKGA		EA	
instance	LB	gap (%)	cv (%)	gap (%)	cv (%)
X1	244983403	17.38	0.76	17.93	0.74
X2	147062803	12.08	1.22	17.81	0.81
X3	156830774	15.93	1.16	19.03	1.33
X4	241257058	14.97	1.12	17.78	0.51
X5	78796003	19.99	1.05	22.86	0.59
X6	248147317	15.59	0.68	19.10	0.98
X7	369443070	15.72	0.88	18.85	0.77
X8	134427339	47.35	1.57	47.62	0.00
X9	361189695	15.49	1.06	19.53	1.09
X10	333756718	15.16	0.57	19.14	0.87
X11	223408308	19.19	1.12	23.14	1.26
X12	242393345	16.76	1.44	21.97	1.08
X13	259467828	13.91	0.78	19.41	0.98
X14	158409238	19.33	1.00	22.73	0.85
X15	242691036	19.02	1.71	24.73	1.27
Average		13.89	1.07	16.58	0.87

15. Q. Chen, Y. Chen, Y. Cui, X. Lu, and L. Li, "A heuristic for the 3-staged 2d cutting stock problem with usable leftover," in *2015 International Conference on Electrical, Automation and Mechanical Engineering*. Phuket: Atlantis Press, 2015, pp. 776–779.
16. A. C. Cherri, M. N. Arenales, and H. H. Yanasse, "The one-dimensional cutting stock problem with usable leftover–a heuristic approach," *European Journal of Operational Research*, vol. 196, no. 3, pp. 897–908, 2009.
17. K. K. Lai and J. Chan, "Developing a simulated annealing algorithm for the cutting stock problem," *Computers & Industrial Engineering*, vol. 32, pp. 115–127, 01 1997.
18. J. Puchinger, G. R. Raidl, and G. Koller, "Solving a real-world glass cutting problem," in *Evolutionary Computation in Combinatorial Optimization*, J. Gottlieb and G. R. Raidl, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 165–176.
19. S. Suliman, "A sequential heuristic procedure for the two-dimensional cutting-stock problem," *International Journal of Production Economics*, vol. 99, no. 1-2, pp. 177–185, 2006.
20. M. Jin, P. Ge, and P. Ren, "A new heuristic algorithm for two-dimensional defective stock guillotine cutting stock problem with multiple stock sizes," *Tehnicki Vjesnik*, vol. 22, pp. 1107–1116, 10 2015.
21. V. S. Vassiliadis, "Two-dimensional stock cutting and rectangle packing: binary tree model representation for local search optimization methods," *Journal of food engineering*, vol. 70, no. 3, pp. 257–268, 2005.
22. F. Dusberger and G. R. Raidl, "Solving the 3-staged 2-dimensional cutting stock problem by dynamic programming and variable neighborhood search," *Electronic Notes in Discrete Mathematics*, vol. 47, pp. 133–140, 2015.
23. F. Dusberger and G. R. Raidl, "A variable neighborhood search using very large neighborhood structures for the 3-staged 2-dimensional cutting stock problem," in *Hybrid Metaheuristics*, M. J. Blesa, C. Blum, and S. Voß, Eds. Cham: Springer International Publishing, 2014, pp. 85–99.
24. M. Hifi, R. M'Hallah, and T. Saadi, "Algorithms for the constrained two-staged two-dimensional cutting problem," *INFORMS Journal on Computing*, vol. 20, no. 2, pp. 212–221, 2008.
25. L. Davis, *Handbook of Genetic Algorithms*, ser. VNR Computer Library VNR Computer Library. Van Nostrand Reinhold, 1991. [Online]. Available: https://books.google.com.br/books?id=Kl7vAAAAMAAJ
26. J. C. Bean, "Genetic algorithms and random keys for sequencing and optimization," *ORSA Journal on Computing*, vol. 2, pp. 154–160, 1994.
27. J. F. Gonçalves and M. G. C. Resende, "Biased random-key genetic algorithms for combinatorial optimization," *Journal of Heuristics*, vol. 17, pp. 487–525, 2011.
28. W. Spears and K. deJong, "On the virtues of parameterized uniform crossover," in *Proceedings of the Fourth International Conference on Genetic Algorithms*, R. Belew and L. Booker, Eds. San Mateo: Morgan Kaufman, 1991, pp. 230–236.
29. M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 8, no. 1, pp. 3–30, 1998.

GA and ILS for optimizing the size of NFA models

Frédéric Lardeux[1] and Eric Monfroy[1]

Univ Angers, LERIA, SFR MATHSTIC, F-49000 Angers, France
`firstname.lastname@univ-angers.fr`

Abstract. Grammatical inference consists in learning a formal grammar (as a set of rewrite rules or a finite state machine). We are concerned with learning Nondeterministic Finite Automata (NFA) of a given size from samples of positive and negative words. NFA can naturally be modeled in SAT. The standard model [1] being enormous, we also try a model based on prefixes [2] which generates smaller instances. We also propose a new model based on suffixes and a hybrid model based on prefixes and suffixes. We then focus on optimizing the size of generated SAT instances issued from the hybrid models. We present two techniques to optimize this combination, one based on Iterated Local Search (ILS), the second one based on Genetic Algorithm (GA). Optimizing the combination significantly reduces the SAT instances and their solving time, but at the cost of longer generation time. We, therefore, study the balance between generation time and solving time thanks to some experimental comparisons, and we analyze our various model improvements.

Mots-Clefs. Constraint problem modeling, Grammar inference, SAT, model reformulation, NFA inference.

1 Introduction

Grammatical inference [3] (or grammar induction) is concerned with the study of algorithms for learning automata and grammars from some observations. The goal is thus to construct a representation that accounts for the characteristics of the observed objects. This research area plays a significant role in numerous applications, such as compiler design, bioinformatics, speech recognition, pattern recognition, machine learning, and others.
In this article, we focus on learning a finite automaton from samples of words $S = S^+ \cup S^-$, such that S^+ is a set of positive words that must be accepted by the automaton, and S^- is a set of negative words to be rejected by the automaton. Due to their determinism, deterministic finite automata (DFA) are generally faster than non deterministic automata (NFA). However, NFA are significantly smaller than DFA in terms of the number of states. Moreover, the space complexity of the SAT models representing the problem is generally due to the number of states. Thus, 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.
The problem of inferring NFA has been undertaken with various approaches (see, e.g., [1]). Among them, we can cite ad-hoc algorithms such as *DeLeTe2* [4] that is based on state merging methods, or the technique of [5] that returns a collection of NFA. Some approaches use metaheuristics for computing NFA, such as hill-climbing [6] or genetic algorithm [7].
A convenient and declarative way of representing combinatorial problems is to model them as a Constraint Satisfaction Problem (CSP [8]) (see, e.g., [1] for an INLP model for inferring NFA, or [9] for a SAT (the propositional satisfiability problem [10]) model of the same problem). Parallel solvers have also been used for minimizing the inferred NFA size [11, 2].
Orthogonally to the approaches cited above, we do not seek to improve a solver, but to generate a model of the problem that is easier to solve with a standard SAT solver. Our approach is similar to DFA inference with graph coloring [12], or NFA inference with complex data structures [9]. Modeling thus consists in translating a problem into a CSP made of decision variables and constraints over these variables. As a reference for comparisons, we start with the basic SAT model of [9]. The model, together with a sample of positive and negative words, lead to a SAT instance to be solved by a classic SAT solver that we use as a black box. However, SAT instances are gigantic, e.g., our base model space complexity is in the order of $\mathcal{O}(k^{|\omega_+|})$ variables, and in $\mathcal{O}(|\omega_+|.k^{|\omega_+|})$ clauses, where k is the number of states of the NFA, and ω_+ is the size of the longest positive word of the sample. The second model,

PM, is based on intermediate variables for each prefix [2] which enables to compute only once parts of paths that are shared by several words. We propose a third model, SP, based on intermediate variables for suffixes. Although the two models could seem similar, their order of size is totally different. Indeed, PM is in $\mathcal{O}(k^2)$ while SM is in $\mathcal{O}(k^3)$. We then propose hybrid models consisting in splitting words into a prefix and a suffix. Modeling the beginning of the word is made with PM while the suffix is modeled by SM. The challenge is then to determine where to split words to optimize the size of the generated SAT instances. To this end, we propose two approaches, one based on iterated local search (ILS), the second one on genetic algorithm (GA). Both permit to generate smaller SAT instances, much smaller than with the DM model and even the PM model. However, with GA, the generation time is too long and erases the gain in solving with the Glucose SAT solver [13]. But the hybrid instances optimized with the ILS are smaller, and the generation time added to the solving time is faster than with PM. Compared to [9], which is the closest work on NFA inferring, we always obtain significantly smaller instances and solving time.
This paper is organized as follows. In Section 2 we present the direct model, the prefix model, and we propose the suffix model. We then combine suffix and prefix model to propose the new hybrid models (Section 3). Hybrid models are optimized with iterated local search (Sub-section 3.2), and with genetic algorithm in Sub-section 3.3. We then compare experimentally our models in Section 4 before concluding in Section 5.

2 SAT Models

Given 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 NFA inference problem** consists in building a NFA with k states which validates words of S^+, and rejects words of S^-. Note that the satisfaction problem we consider in this paper can be extended to an optimization problem minimizing k [2].
Let us introduce some notations. Let $A = (Q, \Sigma, q_1, F)$ be a NFA with: $Q = \{q_1, \ldots, q_k\}$ a set of k states, Σ a finite alphabet, q_1 the initial state, and F the set of final states. The empty word is noted λ. We denote by K the set of integers $\{1, \ldots, k\}$.
We consider the following variables:

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

The path $i_1, i_2, \ldots, i_{n+1}$ for $w = w_1 \ldots w_n$ exists if and only if $d = \delta_{w_1,\overrightarrow{q_{i_1}q_{i_2}}} \wedge \ldots \wedge \delta_{w_n,\overrightarrow{q_{i_n}q_{i_{n+1}}}}$ is true. We say that the conjunction d is a c_path, and $D_{w,\overrightarrow{q_iq_j}}$ is the set of all c_paths for the word w between states q_i and q_j.

2.1 Direct Model

This simple model has been presented in [9]. It is based on 3 sets of equations:

1. If the empty word is in S^+ or S^-, we can fix whether the first state is final or not:
$$\text{if } \lambda \in S^+, \qquad f_1 \tag{1}$$
$$\text{if } \lambda \in S^-, \qquad \neg f_1 \tag{2}$$
2. For each word $w \in S^+$, there is at least a path from q_1 to a final state q_j:
$$\bigvee_{j \in K} \bigvee_{d \in D_{w,\overrightarrow{q_1q_j}}} \big(d \wedge f_j\big) \tag{3}$$
With the Tseitin transformations [14], we create one auxiliary variable for each combination of a word w, a state $j \in K$, and a c_path $d \in D_{w,\overrightarrow{q_1q_j}}$: $aux_{w,j,d} \leftrightarrow d \wedge f_j$. Hence, we obtain a formula in CNF for each w:
$$\bigwedge_{j \in K} \bigwedge_{d \in D_{w,\overrightarrow{q_1q_j}}} [(\neg aux_{w,j,d} \vee (d \wedge f_j))] \tag{4}$$
$$\bigwedge_{j \in K} \bigwedge_{d \in D_{w,\overrightarrow{q_1q_j}}} (aux_{w,j,d} \vee \neg d \vee \neg f_j) \tag{5}$$
$$\bigvee_{j \in K} \bigvee_{d \in D_{w,\overrightarrow{q_1q_j}}} aux_{w,j,d} \tag{6}$$

number of cl.	arity	Constraints
$\lvert S^+\rvert.(\lvert\omega_+\rvert+1).k^{\lvert\omega_+\rvert}$	2	(4)
$\lvert S^+\rvert.k^{\lvert\omega_+\rvert}$	$\lvert\omega_+\rvert+2$	(4)
$\lvert S^+\rvert$	$k^{\lvert\omega_+\rvert}$	(4)
$\lvert S^-\rvert.k^{\lvert\omega_-\rvert}$	$\lvert\omega_-\rvert+1$	(7)

Table 1. Clauses for DM_k

number of var	reason
k	final states F
$n.k^2$	transitions δ
$\lvert S^+\rvert.k.^{\lvert\omega_+\rvert}$	Constraints (3)

Table 2. Variables for DM_k

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

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

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

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

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

Size of the models *(see [9] for details)* Consider ω_+ and ω_-, the longest word of S^+ and S^- respectively. Table 1 presents the number of clauses (Column 1) and their arities (Column 2), which are an upper bound of a given constraint group (last column) for the model SM_k. Table 2 presents the upper bound of the number of Boolean variables that are required and why the are required. We can see on Tables 1 and 2 that the space complexity of the DM_k is huge ($\mathcal{O}(\lvert S^+\rvert.k.^{\lvert\omega_+\rvert})$ variables, and $\mathcal{O}(\lvert S^+\rvert.(\lvert\omega_+\rvert+1).k^{\lvert\omega_+\rvert})$ clauses) and with large clauses (up to arity of $\lvert\omega_+\rvert+2$), and that only small instances for a small number of states will be tractable. It is thus obvious that it is important to improve the model DM_k.

2.2 Prefix Model [2]

Let $Pref(w)$ be the set of all the non-empty prefixes of the word w and, by extension, $Pref(W) = \cup_{w\in W} Pref(w)$ the set of prefixes of the words of the set W. For each $w \in Pref(S)$, we add a Boolean variable $p_{w,\overrightarrow{q_1 q_i}}$ which determines whether there is or not a c_path for w from state q_1 to q_i. Note that these variables can be seen as labels of the Prefix Tree Acceptor (PTA) for S [3]. The problem can be modeled with the following constraints:

1. For all prefix $w = a$ with $w \in Pref(S)$, and $a \in \Sigma$, there is a c_path of size 1 for w:

$$\bigvee_{i\in K}\delta_{a,\overrightarrow{q_1 q_i}} \leftrightarrow p_{a,\overrightarrow{q_1 q_i}} \tag{8}$$

With the Tseitin transformations, we can derive a CNF formula. It is also possible to directly encode $\delta_{a,\overrightarrow{q_1 q_i}}$ and $p_{a,\overrightarrow{q_1 q_i}}$ as the same variable. Thus, no clause is required.

2. For all words $w \in S^+ - \{\lambda\}$:

$$\bigvee_{i\in K} p_{w,\overrightarrow{q_1 q_i}} \wedge f_i \tag{9}$$

With the Tseitin transformations [14], we create one auxiliary variable for each combination of $p_{w,\overrightarrow{q_1 q_i}}$ and the status (final or not) of the state q_i: $aux_{w,i} \leftrightarrow p_{w,\overrightarrow{q_1 q_i}} \wedge f_i$. Hence, for each w, we obtain a formula in CNF:

$$\bigwedge_{i\in K}((\neg aux_{w,i} \vee p_{w,\overrightarrow{q_1 q_i}}) \wedge (\neg aux_{w,i} \vee f_i)) \tag{10}$$

$$\bigwedge_{i\in K}(aux_{w,i} \vee \neg p_{w,\overrightarrow{q_1 q_i}} \vee \neg f_i) \tag{11}$$

$$\bigvee_{i\in K} aux_{w,i} \tag{12}$$

3. For all words $w \in S^- - \{\lambda\}$, we obtain the following CNF constraint:

$$\bigwedge_{i\in K}(\neg p_{w,\overrightarrow{q_1 q_i}} \vee \neg f_i) \tag{13}$$

number of cl.	arity	Constraints
$2.k.\|S^+\|$	2	(10)
$k.\|S^+\|$	3	(11)
k	$k+1$	(12)
$k.\|S^-\|$	2	(13)
$\pi.k^2$	2	(15)
$\pi.k^2$	2	(16)
$\pi.k^2$	3	(17)
$\pi.k$	$k+1$	(18)
$\pi.k^2$	2	(19)

Table 3. Clauses for PM_k

number of var	reason
k	final states F
$n.k^2$	transitions δ
$\|S^+\|.k$	Constraints (9)
$\pi.k^2$	Constraints (14)

Table 4. Variables for PM_k

4. For all prefix $w = va$, $w \in Pref(S)$, $v \in Pref(S)$ and $a \in \Sigma$:

$$\bigwedge_{i\in K} (p_{w,\overrightarrow{q_1q_i}} \leftrightarrow (\bigvee_{j\in K} p_{v,\overrightarrow{q_1q_j}} \wedge \delta_{a,\overrightarrow{q_jq_i}})) \tag{14}$$

Applying the Tseitin transformations, we create one auxiliary variable for each combination of existence of a c_path from q_1 to q_i ($p_{v,\overrightarrow{q_1q_j}}$) and the transition $\delta_{a,\overrightarrow{q_jq_i}}$: $aux_{v,a,j,i} \leftrightarrow p_{v,\overrightarrow{q_1q_j}} \wedge \delta_{a,\overrightarrow{q_jq_i}}$. Then, (14) becomes:

$$\bigwedge_{i\in K} (p_{w,\overrightarrow{q_1q_i}} \leftrightarrow (\bigvee_{j\in K} aux_{v,a,j,i}))$$

For each $w \in Pref(S)$, we obtain constraints in CNF:

$$\bigwedge_{(i,j)\in K^2} (\neg aux_{v,a,j,i} \vee p_{w,\overrightarrow{q_1q_i}}) \tag{15}$$

$$\bigwedge_{(i,j)\in K^2} (\neg aux_{v,a,j,i} \vee \delta_{a,\overrightarrow{q_jq_i}}) \tag{16}$$

$$\bigwedge_{(i,j)\in K^2} (aux_{v,a,j,i} \vee \neg p_{w,\overrightarrow{q_1q_i}} \vee \neg\delta_{a,\overrightarrow{q_jq_i}}) \tag{17}$$

$$\bigwedge_{i\in K} (\neg p_{w,\overrightarrow{q_1q_i}} \vee (\bigvee_{j\in K} aux_{v,a,j,i})) \tag{18}$$

$$\bigwedge_{(i,j)\in K^2} (p_{w,\overrightarrow{q_1q_i}} \vee \neg aux_{v,a,j,i})) \tag{19}$$

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

$$PM_k = \bigwedge_{w\in S^+} \Big((10) \wedge \ldots \wedge (12)\Big) \wedge \bigwedge_{w\in S^-} (13) \wedge \bigwedge_{w\in Pref(S)} (15) \wedge \ldots \wedge (19)$$

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

Size of the models Consider ω_+, the longest word of S^+, ω_-, the longest word of S^-, $\sigma = \Sigma_{w\in S}|w|$, and π, the number of prefix obtained by $Pref(S)$ with a size larger than 1 ($\pi = |\{x|x \in Pref(S), |x| > 1\}|$), then:

$$max(|\omega_+|, |\omega_-|) \leq \pi \leq \sigma \leq |S^+|.|\omega_+| + |S^-|.|\omega_-|$$

The space complexity of the PM_k model is thus in $\mathcal{O}(\sigma.k^2)$ variables, and in $\mathcal{O}(\sigma.k^2)$ binary and ternary clauses, and $\mathcal{O}(\sigma.k)$ $(k+1)$-ary clauses.

2.3 Suffix Model

We now propose a suffix model (SM_k), based on $Suf(S)$, the set of all the non-empty suffixes of all the words in S. The main difference is that the construction starts from every state and terminates in state q_1. For each $w \in Suf(S)$, we add a Boolean variable $p_{w,\overrightarrow{q_iq_j}}$ which determines whether there is or not a c_path for w from state q_i to q_j. To model the problem, Constraints (10), (11), (12), and (13) remain unchanged and creation of the corresponding auxiliary variables $aux_{w,i}$ as well.

For each suffix $w = a$ with $w \in Suf(S)$, and $a \in \Sigma$, there is a c_path of size 1 for w:

$$\bigvee_{(i,j)\in K^2} \delta_{a,\overrightarrow{q_iq_j}} \leftrightarrow p_{a,\overrightarrow{q_iq_j}} \tag{20}$$

We can directly encode $\delta_{a,\overrightarrow{q_iq_j}}$ and $p_{a,\overrightarrow{q_iq_j}}$ as the same variable. Thus, no clause is required.

For all suffix $w = av$, $w \in Suf(S)$, $v \in Suf(S)$ and $a \in \Sigma$:

$$\bigwedge_{(i,j)\in K^2} (p_{w,\overrightarrow{q_iq_j}} \leftrightarrow (\bigvee_{k\in K} \delta_{a,\overrightarrow{q_iq_k}} \wedge p_{v,\overrightarrow{q_kq_j}})) \tag{21}$$

We create one auxiliary variable for each combination of existence of a c_path from q_k to q_j ($p_{v,\overrightarrow{q_kq_j}}$) and the transition $\delta_{a,\overrightarrow{q_iq_k}}$: $aux_{v,a,i,k,j} \leftrightarrow \delta_{a,\overrightarrow{q_iq_k}} \wedge p_{v,\overrightarrow{q_kq_j}}$
For each $w = av$, we obtain the following constraints (CNF formulas):

$$\bigwedge_{(i,j,k)\in K^3} (\neg aux_{v,a,i,k,j} \vee p_{w,\overrightarrow{q_kq_j}}) \tag{22}$$

$$\bigwedge_{(i,j,k)\in K^3} (\neg aux_{v,a,i,k,j} \vee \delta_{a,\overrightarrow{q_iq_k}}) \tag{23}$$

$$\bigwedge_{(i,j,k)\in K^3} (aux_{v,a,i,k,j} \vee \neg p_{w,\overrightarrow{q_kq_j}} \vee \neg\delta_{a,\overrightarrow{q_iq_k}}) \tag{24}$$

$$\bigwedge_{(i,j)\in K^2} (\neg p_{w,\overrightarrow{q_iq_j}} \vee (\bigvee_{k\in K} aux_{v,a,i,k,j})) \tag{25}$$

$$\bigwedge_{(i,j,k)\in K^3} (p_{w,\overrightarrow{q_iq_j}} \vee \neg aux_{v,a,i,k,j})) \tag{26}$$

Note that some clauses are not worth being generated. Indeed, it is useless to generate paths starting in states different from the initial state q_1, except when the w is in S, and w is also the suffix of another word from S. Removing these constraints does not change the complexity of the model. This can easily be done at generation time, or we can leave it to the solver, which will detect it and remove the useless constraints.
Thus, the constraint prefix model PM_k for building a NFA of size k is:

$$SM_k = \bigwedge_{w\in S^+} \Big((10) \wedge \ldots \wedge (12)\Big) \wedge \bigwedge_{w\in S^-} (13) \wedge \bigwedge_{w\in Pref(S)\setminus S} (22) \wedge \ldots \wedge (26)$$

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

Size of the models Consider ω_+, ω_-, σ, and π as defined in the prefix model. Table 5 presents the number of clauses (first column) and their arities (Column 2) which are an upper bound of a given constraint group (last column) for the model SM_k. Table 6 presents the upper bound of the number of Boolean variables that are required, and the reason of their requirements. To simplify, the space complexity of SM_k is thus in $\mathcal{O}(\sigma.k^3)$ variables, and in $\mathcal{O}(\sigma.k^3)$ binary and ternary clauses, and $\mathcal{O}(\sigma.k^2)$ $(k+1)$-ary clauses.

3 Hybrid Models

We now propose a family of models based on both the notion of prefix and the notion of suffix. The idea is, in fact, to take advantage of the construction of a prefix p and a suffix s of a word w such that $w = p.s$ to pool both prefixes and suffixes. The goal is to reduce the size of generated SAT instances. The process is the following:

1. For each word w_i of S, we split w_i into p_i and s_i such that $w = p_i.s_i$. We thus obtain two sets, $S_p = \{p_i \mid \exists i, w_i \in S \text{ and } w_i = p_i.s_i\}$ and $S_s = \{s_i \mid \exists i, w_i \in S \text{ and } w_i = p_i.s_i\}$.
2. We then consider S_p as a sample, i.e., a set of words. For each w of S_p, we generate Constraints (15) to (19).
3. We consider S_s in turn to generate Constraints (22) to (26) for each $w \in S_s$.

number of cl.	arity	Constraints
$2.k.\|S^+\|$	2	(10)
$k.\|S^+\|$	3	(11)
k	$k+1$	(12)
$k.\|S^-\|$	2	(13)
$\pi.k^3$	2	(22)
$\pi.k^3$	2	(23)
$\pi.k^3$	3	(24)
$\pi.k^2$	$k+1$	(25)
$\pi.k^3$	2	(26)

Table 5. Clauses for SM_k

number of var	reason
k	final states F
$n.k^2$	transitions δ
$\|S^+\|.k$	Constraints (9)
$\pi.k^3$	Constraints (21)

Table 6. Variables for SM_k

4. Then, for each $w_i = p_i.s_i$, clauses corresponding to p_i must be linked to clauses of s_i.
   – if $w_i = p_i.s_i \in S^-$, the constraints are similar to the ones of (13) including the connection of p_i and s_i:

$$\bigwedge_{(j,k)\in K^2} (\neg p_{p_i,\overrightarrow{q_1q_j}} \vee \neg p_{s_i,\overrightarrow{q_jq_k}} \vee \neg f_i) \tag{27}$$

   – if $w_i = p_i.s_i \in S^+$, the constraints are similar to (9):

$$\bigvee_{(j,k)\in K^2} p_{p_i,\overrightarrow{q_1q_j}} \wedge p_{s_i,\overrightarrow{q_jq_k}} \wedge f_k \tag{28}$$

   We transform (28) using auxiliary variables $aux_{w_i,j,k} \leftrightarrow p_{w,\overrightarrow{q_1q_j}} \wedge p_{w,\overrightarrow{q_jq_k}} \wedge f_i$ to obtain the following CNF constraints:

$$\bigwedge_{(j,k)\in K^2} ((\neg aux_{w_i,j,k} \vee p_{w,\overrightarrow{q_1q_j}}) \wedge (\neg aux_{w_i,j,k} \vee p_{w,\overrightarrow{q_jq_k}}) \wedge (\neg aux_{w_i,j,k} \vee f_k)) \tag{29}$$

$$\bigwedge_{(j,k)\in K^2} (aux_{w_i,j,k} \vee \neg p_{w,\overrightarrow{q_1q_j}} \vee p_{w,\overrightarrow{q_jq_k}} \vee \neg f_k) \tag{30}$$

$$\bigvee_{(j,k)\in K^2} aux_{w_i,j,k} \tag{31}$$

Thus, the hybrid model HM_k for building a NFA of size k is:

$$HM_k = \bigwedge_{w\in S^+} \Big((29)\wedge\ldots\wedge(31)\Big)\wedge \bigwedge_{w\in S^-}(27)\wedge \bigwedge_{p_i\in Pref(S_p)}(22)\wedge\ldots\wedge(26) \bigwedge_{s_i\in Suf(S_s)}(15)\wedge\ldots\wedge(19)$$

and it is possibly completed by (1) or (2) if $\lambda \in S^+$ or $\lambda \in S^-$.
We do not detail it here, but in the worst case, the complexity of the model is the same as SM_k. It is obvious that the split of each word into a prefix and a suffix will determine the size of the instance. The next sub-sections are dedicated to the computation of this separation $w_i = p_i.s_i$ to minimize the size of the generated hybrid instances with the HM_k model.

3.1 Search Space and Evaluation Function For Metaheuristics

The search space $\mathcal{X}$ of this problem corresponds to all the hybrid models: for each word w of S, we have to determine a n such that $w = p.s$ with $|p| = n$ and $|s| = |w| - n$. The size of the search space is thus: $|\mathcal{X}| = \Pi_{w\in S}|w| + 1$.
Even though we are aware that smaller instances are not necessarily easier to solve, we choose to define the first evaluation function as the number of generated SAT variables. However, this number cannot be computed a priori: first, the instance has to be generated, before counting the variables. This function being too costly, we propose an alternative evaluation function for approximating the number of variables. This fitness function is based on the number of prefixes in $Pref(S_p)$ and suffixes in $Suf(S_s)$. Since the complexity of SM_k is in $\mathcal{O}(k^3)$ whereas the complexity of PM_k is in $\mathcal{O}(k^2)$, suffixes are penalized by a coefficient corresponding to the number of states.

$$fitness(S_p, S_s) = |Pref(S_p)| + k.|Suf(S_s)|$$

Empirically, we observe that the results of this $fitness$ function are proportional to the actual number of generated SAT variables. This approximation of the number of variables will thus be the fitness function in our ILS and GA algorithms.

3.2 Iterated Local Search Hybrid Model HM_ILS_k

We propose an Iterated Local Search (ILS) [15] for optimizing our hybrid model. Classically, a best improvement or a first improvement neighborhood is used in ILS to select the next move. In our case, a first improvement provides very poor results. Moreover, it is clearly impossible to evaluate all the neighbors at each step due to the computing cost. We thus decide to randomly choose a word in S with a roulette wheel selection based on the word weights. Each word w has a weight corresponding for 75% to a characteristic of S, and 25% to the length of the word:

$$\text{weight}_w = 75\%/|S| + 25\% * |w|/(\sum_{w_i \in S} |w_i|)$$

The search starts generating a random couple of prefixes and suffixes sets (S_p,S_s), i.e., for each word w of S an integer is selected for splitting w into a prefix p and a suffix s such that $w = p.s$. Hence, at each iteration, the best couple (p, s) is found for the selected word w. This process is iterated until a maximum number of iterations is reached.
In our ILS, it is not necessary to introduce noise with random walks or restarts because our process of selection of word naturally ensures diversification.

Algorithm 1: Iterated Local Search

```
Input: set of words $S$, maximum number of iterations $max\_iter$
    maximum of consecutive iterations allowed without improvement $max\_iter\_without\_improv$
Output: set of prefixes $S_p^*$, set of suffixes $S_s^*$
 1: Couple of prefixes and suffixes sets ($S_p$,$S_s$) is randomly generated
 2: ($S_p^*$,$S_s^*$) = ($S_p$,$S_s$)
 3: repeat
 4:    Choose a word $w$ in $S$ with a roulette wheel selection
 5:    ($S_p$,$S_s$) is updated by the best couple of the sub-search space corresponding only to a modification
       of the prefix and the suffix of word $w$
 6:    if $fitness(S_p, S_s) < fitness(S_p^*, S_s^*)$ then
 7:       ($S_p^*$,$S_s^*$) = ($S_p$,$S_s$)
 8:    end if
 9: until maximum number of iterations $max\_iter$ is reached or ($S_p^*$,$S_s^*$) is not improved since
    $max\_iter\_without\_improv$ iterations
10: return ($S_p^*$, $S_s^*$)
```

3.3 Genetic Algorithm Hybrid Model HM_GA_k

We propose a classical genetic algorithm (GA) based on the search space and fitness function presented in Section 3.1. A population of individuals, represented by a couple of prefixes and suffixes sets, is improved generation after generation. Each generation keeps a portion of individuals as parents and creates children by crossing the selected parents. Crossover operator used in our GA is the well-known uniform crossover. For each word, children inherit the prefix and the suffix of one of their parents randomly chosen. Since the population size is the same during all the search, we have a steady-state GA. A mutation process is applied over all individuals with a probability p_{mut}. For each word w, each prefix and suffix are randomly mutated by generating an integer n between 0 and $|w|$ splitting w into a new prefix of size n and a new suffix $|w| - n$. The search stops when the maximum number of generations is reached or when no improvement is observed in the population during $max\_gen\_without\_improv$ generations.

Algorithm 2: Genetic Algorithm

```
Input: set of words $S$, population size $s_{\mathcal{P}}$, mutation probability $p_{mut}$,
    maximum number of generations $max\_gen$,
    portion of population conserve in the next generation $p_{parents}$,
    maximum of consecutive generations allowed without improvement $max\_gen\_without\_improv$
Output: set of prefixes $S_p^*$, set of suffixes $S_s^*$
 1: Population $\mathcal{P}$ of couples of prefixes and suffixes sets ($S_p$,$S_s$) is randomly generated
 2: ($S_p^*$,$S_s^*$) = Argmin$_{fitness}(\mathcal{P})$
 3: repeat
 4:   Select as parents set $Par$ a portion $p_{parents}$ of $\mathcal{P}$
 5:   Generate $(1 - p_{parents}).s_{\mathcal{P}}$ children by uniform crossover over parents in a set $Children$
 6:   $\mathcal{P} = Par \cup Children$
 7:   Mutate for each individual of $\mathcal{P}$ the prefix/suffix for each words of $S$ with a probability $p_{mut}$
 8:   Update the population
 9:   Update ($S_p^*$,$S_s^*$) if necessary
10: until maximum number of generations $max\_gen$ is reached or ($S_p^*$,$S_s^*$) is not improved since
    $max\_gen\_without\_improv$ generations
11: return $S_p^*$ and $S_s^*$
```

4 Experimental results

To test our new models, we work on the training set of the StaMinA Competition (see http://stamina.chefbe.net). We use 11 of the instances selected in [2][1] with a sparsity $s \in \{12.5\%, 25\%, 50\%, 100\%\}$ and an alphabet size $|\Sigma| \in \{2, 5, 10\}$. We try to generate SAT instances for NFA sizes (k) near to the threshold of the existence or not of an NFA.

4.1 Experimental Protocol

All our algorithms are implemented in Python using specific libraries such as Pysat. The experiments were carried out on a computing cluster with Intel-E5-2695 CPUs, and a limit of 10 GB of memory was fixed. Running times were limited to 10 minutes, including generation of the model and solving time. We used the Glucose [13] SAT solver with the default options. For stochastic methods (ILS and GA), 30 runs are realized to exploit the results statistically. Parameters used for our hybrid models are:

ILS		AG	
max_iter	10 000	$s_{\mathcal{P}}$	100
$max\_iter\_without\_improv$	100	max_gen	3000
		$max\_gen\_without\_improv$	100
		p_{mut}	0.05
		$p_{parents}$	0.03

4.2 Results

Our experiments are reported in Table 7. The first column ($Instance$) corresponds to the official name of the instance, and the second one (k) to the number of states of the expected NFA. Then, we have in sequence the model name ($Model$), the number of SAT variables ($Var.$), the number of clauses ($Cl.$), and the instance generation time (t_M). The right part of the table corresponds to the solving part with the satisfiability of the generated instance (SAT), the decisions number ($Dec.$), and the solving time (t_S) with Glucose. Finally, the last column (t_T) corresponds to the total time (modeling time + solving time). Results for hybrid models based on ILS (HM_ILS_k) and GA (HM_GA_k) correspond to average values over 30 runs. We have decided to only provide the average since the standard deviation values are very small.
The last lines of the table correspond to the cumulative values for each column and each model. When an instance is not solved (time-out), the maximum value needed for solving

[1] We kept the "official" name used in [2].

the other model instances is considered. For the instance generation time (t_M), a credit of 600 seconds is applied when generation did not succeed before the time-out.

We can clearly confirm that the direct model is not usable in practice, and that instances cannot be generated in less than $600s$. The prefix model allows the fastest generation when it terminates before the time out (on these benchmarks, it did not succeed once and was thus penalize for cumulative values). It also provides instances that are solved quite fast. As expected, the instances optimized with GA are the smallest ones. However, the generation is too costly: the gain in solving time is not sufficient to compensate the long generation time. In total, in terms of solving+generation time, GA based model is close to prefix model. As planned with its space complexity (in $\mathcal{O}(k^3)$), suffix based instances are huge and long to solve. However, we were surprised for 2 benchmarks (ww-10-40 and ww-10-50) for which the generated instances are relatively big (5 times the size of the GA optimized instances), but their solving is the fastest. We still cannot explain what made these instances easy to solve, and we are still investigating their structure. The better balance is given with the ILS model: instances are relatively small, the generation time is fast, and the solving time as well. This is thus the best option of this work.
It is very difficult to compare our results with the results of [2]. First of all, in [2], they try to minimize k, the number of states. Moreover, they use parallel algorithms. Finally, they do not detail the results for each instance and each k, except for st-2-30 and st-5-50. For the first one, with $k = 9$ we are much faster. But for the second one, with $k = 5$ we are slower.

5 Conclusion

In this paper, we have proposed to use some metaheuristics algorithms, namely ILS and GA, to improve the size of SAT models for the NFA inferring problem. Our hybrid model, optimized with GA gives, on average, the smallest SAT instances. Solving these instances is also faster than with the direct or prefix models. However, generation of the optimized instances with GA is really too long and is not balanced out with the gain in solving time; it is at the level of the prefix model w.r.t. total CPU time. The ILS model generates optimized instances a bit larger than with GA and a bit smaller than with prefixes. Moreover, the solving time is the best of our experiments, and the generation time added to the solving time makes of the HM_ILS_k our better model.
In the future, we plan to speed up GA to make it more competitive. We also plan to consider more complex fitness functions, not only based on the number of SAT variables but also on the length of clauses. We also plan a model portfolio approach for larger samples.

References

1. Wieczorek, W.: Grammatical Inference - Algorithms, Routines and Applications. Volume 673 of Studies in Computational Intelligence. Springer (2017)
2. Jastrzab, T., Czech, Z.J., Wieczorek, W.: Parallel algorithms for minimal nondeterministic finite automata inference. Fundam. Informaticae **178** (2021) 203–227
3. de la Higuera, C.: Grammatical Inference: Learning Automata and Grammars. Cambridge University Press (2010)
4. Denis, F., Lemay, A., Terlutte, A.: Learning regular languages using rfsas. Theor. Comput. Sci. **313** (2004) 267–294
5. Vázquez de Parga, M., García, P., Ruiz, J.: A family of algorithms for non deterministic regular languages inference. In: Proc. of CIAA 2006. Volume 4094 of LNCS., Springer (2006) 265–274
6. Tomita, M.: Dynamic construction of finite-state automata from examples using hill-climbing. Proc. of the Annual Conference of the Cognitive Science Society (1982) 105–108
7. Dupont, P.: Regular grammatical inference from positive and negative samples by genetic search: the GIG method. In: Proc. of ICGI 94. Volume 862 of LNCS., Springer (1994) 236–245
8. Rossi, F., van Beek, P., Walsh, T., eds.: Handbook of Constraint Programming. 1st edn. Elsevier Science (2006)
9. Lardeux, F., Monfroy, E.: Improved SAT models for NFA learning. In: Proc. of OLA 2021. Communications in Computer and Information Science, Springer (2021) In press.
10. Garey, M.R., Johnson, D.S.: Computers and Intractability, A Guide to the Theory of NP-Completeness. W.H. Freeman & Company, San Francisco (1979)

Table 7. Comparison on 11 instances between the models DM_k, PM_k, SM_k, HM_ILS_k, and HM_GA_k.

Instance	k	Model	Var.	Cl.	t_M	SAT	Dec.	t_S	t_T
		DM_k	190 564	1 817 771	13.46	True	973 213	88.62	102.09
		PM_k	1 276	4 250	1.27	True	3 471	0.10	**1.37**
st-2-10	4	SM_k	5 196	17 578	1.30	True	4 332	0.21	1.51
		HM_ILS_k	1 188	4 179	4.14	True	2 503	**0.05**	4.18
		HM_GA_k	**1 107**	3 884	14.45	True	2 368	**0.05**	14.50
		DM_k	-	-	-	-	-	-	-
		PM_k	4 860	17 150	1,34	False	1 625 706	241,24	242,59
st-2-20	6	SM_k	-	-	-	-	-	-	-
		HM_ILS_k	5 688	21 073	5,39	False	662 354	98,35	**103,74**
		HM_GA_k	**4 735**	17 611	34,65	False	708 356	**94,95**	129,61
		DM_k	-	-	-	-	-	-	-
		PM_k	-	-	-	-	-	-	-
st-2-30	9	SM_k	-	-	-	-	-	-	-
		HM_ILS_k	20 637	78 852	7.55	True	1 998 574	**228.53**	**236.07**
		HM_GA_k	**16 335**	62 832	66.94	True	4 079 686	527.44	594.38
		DM_k	-	-	-	-	-	-	-
		PM_k	4 024	13 464	1.49	True	2 641	**0.08**	**1.57**
st-5-20	4	SM_k	14 964	50 660	1.68	True	23 540	3.23	4.91
		HM_ILS_k	3 608	12 514	7.83	True	14 584	0.72	8.56
		HM_GA_k	**3 522**	12 180	47.84	True	18 344	0.94	48.78
		DM_k	-	-	-	-	-	-	-
		PM_k	5 364	18 054	1.43	True	177 711	21.57	**23.00**
st-5-30	4	SM_k	21 084	71 502	1.87	True	362 318	128.02	129.89
		HM_ILS_k	4 837	16 955	9.90	True	156 631	**19.90**	29.81
		HM_GA_k	**4 705**	16 478	119.42	True	171 062	21.67	141.09
		DM_k	-	-	-	-	-	-	-
		PM_k	6 284	21 216	1.52	False	7 110	0.55	**2.07**
st-5-40	4	SM_k	23 604	80 104	1.55	False	15 708	1.74	3.29
		HM_ILS_k	5 745	20 290	10.29	False	6 206	**0.34**	10.62
		HM_GA_k	**5 548**	19 517	150.50	False	6 204	0.35	150.85
		DM_k	-	-	-	-	-	-	-
		PM_k	11 150	38 745	1.59	False	1 943 735	562.80	564.39
st-5-50	5	SM_k	-	-	-	-	-	-	-
		HM_ILS_k	11 085	40 258	10.80	False	911 280	**238.10**	**248.90**
		HM_GA_k	**10 040**	36 350	279.87	False	1 093 093	287.46	567.33
		DM_k	-	-	-	-	-	-	-
		PM_k	14 200	49 455	1.52	False	1 245 538	383.37	384.89
st-5-60	5	SM_k	-	-	-	-	-	-	-
		HM_ILS_k	13 920	50 568	13.47	False	800 920	**231.82**	**245.29**
		HM_GA_k	**13 180**	47 755	313.30	False	950 601	270.97	584.26
		DM_k	15 012	112 039	2.07	True	69 219	1.52	3.59
		PM_k	3 624	11 900	1.38	True	977	0.03	1.41
ww-10-40	4	SM_k	13 844	46 648	1.25	True	4 173	**0.02**	**1.28**
		HM_ILS_k	2 896	10 342	5.94	True	3 897	0.06	6.00
		HM_GA_k	**2 761**	9 839	75.57	True	2 842	0.04	75.60
		DM_k	80 548	694 641	5.61	True	483 153	103.52	109.14
		PM_k	5 364	17 850	1.28	True	167 390	20.29	21.57
ww-10-50	4	SM_k	20 844	70 482	1.49	True	74 482	11.58	**13.07**
		HM_ILS_k	4 633	16 514	7.71	True	73 534	5.46	13.17
		HM_GA_k	**4 517**	15 940	123.21	True	52 894	**3.38**	126.59

	DM_k	397 451	3 014 842	4 221,15	-	10 821 816	2 041,50	6 262,65
	PM_k	76 783	270 936	612,82	-	9 253 965	1 757,47	2 370,29
Cumulative values	SM_k	151 211	525 099	2 409,15	-	9 379 218	1 879,65	4 268,80
	HM_ILS_k	74 237	271 543	**83,03**	-	4 630 483	**823,33**	**906,36**
	HM_GA_k	**66 450**	242 386	1 225,75	-	7 085 451	1 207,23	2 432,98

11. Jastrzab, T.: Two parallelization schemes for the induction of nondeterministic finite automata on pcs. In: Proc. of PPAM 2017. Volume 10777 of LNCS., Springer (2017) 279–289
12. Heule, M., Verwer, S.: Software model synthesis using satisfiability solvers. Empirical Software Engineering **18** (2013) 825–856
13. Audemard, G., Simon, L.: Predicting learnt clauses quality in modern SAT solvers. In: Proc. of IJCAI 2009. (2009) 399–404
14. Tseitin, G.S. In: On the Complexity of Derivation in Propositional Calculus. Springer Berlin Heidelberg, Berlin, Heidelberg (1983) 466–483
15. Stützle, T., Ruiz, R. In: Iterated Local Search. Springer International Publishing, Cham (2018) 579–605

Spiking Neural Networks and Audio Classification

Farah Medjahed[1], Phillipe Devienne[2] and Abou El Hassan Benyamina[1]

(medjahed.farah@edu.univ-oran1.dz, Philippe.Devienne@univ-lille.fr, benyamina.hassen@univ-oran1.dz)

1. Computer Science Department, Oran1 University, Ahmed Ben Bella, Oran, Algeria

2. Univ. Lille, CNRS, Centrale Lille, UMR 9189 – CRIStAL, F-59000 Lille, France

Keyworks : Automatic Sound Classification, Spiking Neural Networks, Natural computing.

I Introduction

For many years, artificial neural networks have attracted the attention of researchers, they are used in various fields: image processing, signal processing, handwriting recognition, facial recognition. Neural networks are widely used in sound recognition which has become a hot topic of great interest. In this article, we will present in the first part the birth of artificial neural networks passing from the biological aspect to the mathematical aspect and we will mention the different types of neural networks and learning rules that exist. In the second part we will elaborate the audio processing, by presenting the different techniques of audio representation and the main methods used in audio classification. Finally, we will conclude by citing the current work which is in progress which is related to audio classification using Spiking Neural Networks.

II Spiking neural network

Different types of neural network exist we can cite: MLP (Multi-Layer Perception) which consists of an input layer, an output layer and at least one intermediate layer called the hidden layer where each neuron of a layer is connected with all the neurons of the next layer (except the neurons of the output layer). CNN (Convolutional Neural Network), mainly used in image processing, consists also of a succession of layers: an input layer, an output layer and hidden layers made up of numerous convolutional layers, Pooling layers, ReLU (Rectified Linear Unit) correction layers and layers fully connected. Unlike Multilayer Perceptron and Convolutional Neural Network process data of fixed size and go through a fixed number of layers and computational steps and give outputs of fixed sizes, the Recurrent Neural Network (RNN) [1] manipulates variable-sized entries. A traditional recurrent neural network consists of an input layer, an output layer, and a recurring layer. We have also SNN (Spiking Neural Network) which is the type of neural network that mimics the brain the most because it applies the actual functioning of the neuron. In a biological neuron, a pulse is generated when the sum of the changes in the potential of the presynaptic membrane exceeds the threshold.

III Audio Classification

An audio signal is a signal that contains information in the audible frequency range [2]. Audio classification is divided into two parts the front end and the back end [3]. The front end is the part that treats the input to extract features from the audio and the back end is responsible for the classification and the prediction of the output. The audio signal contains information which identifies the sound. Audio representation is responsible for the extraction of these information or features which represents the audio signal. Many feature extraction techniques are used we can find: MFCC (Mel-Frequency Cepstral Coefficient) [4] which is the most used feature extraction technique for audio processing tasks. MFCC consists of a succession of operations. ZCR [5] of an audio frame is the rate at which the signal's sign changes during the frame. In other words, it's the number of times the signal's value changes from positive to negative and back, divided by the frame's length. LPC is used to compress audio. It is the most widely used method in speech recognition. The basic idea of ~~linear prediction is the use of a linear combination of the past time-domain samples to predict the~~ current time- domain sample [6]. We have also The Mel spectrogram [7] which is the representation of the audio in the form of time and frequency. For the audio classification we have GMM (Gaussian Mixture Models) [8] a probabilistic model. The basis for using GMM in audio classification is that the distribution of feature vectors extracted from a class can be modeled by a mixture of Gaussian density. SVM (Support Vector Machines) [9] transform data into a high-dimensional space, this converts the classification problem into a simpler one which can use linear discriminant functions. HMM (Hidden Markov Models) are widely used classification models in speech recognition [10]. HMM is a finite set of states, each of which has a probability distribution associated with it. A collection of probabilities known as transition probabilities governs transitions between states. According to the corresponding probability distribution, an outcome or observation can be generated in a specific state. Only the outcome is known and the underlying state sequence is obscured [4]. Several works have used the Convolutional Neural network in the context of sound classification. CNN is commonly used in image classification and it has improved the performance of the classification in this domain. Dealing with audio input, lead to transform a sound classification problem into an image classification problem. The idea is to use CNN for the classification by transforming the audio into spectrograms and to use these later as an input of a Convolutional Neural Network.

IV Work in progress

The natural world is analog and yet most of the sensors, with which we observe and monitor the real-world data, use discrete quantities. To avoid as much as possible approximation and heavy latency due to digital conversion, our challenge is to copy biological systems and neurological processes. The first step was to design a bioinspired analog cochlea at Lille that we are interfacing with embedded AI algorithms (implemented on low power neuroprocessors) to mimics the cochlea and the audio cortex.

We opt for the use of Spiking Neuron Network in the sound detection for its advantages. SNN are well adapted to processing spatio-temporal event-based data from neuromorphic sensors, which are power efficient. SNN are highly computationally and energy-efficient model it can be exploited in a neuromorphic hardware device. Trying to add something new in this domain, we are working on sound detection using spiking neural networks, the audio data will be converted into impulses using the artificial cochlea designed at Lille then this later will be used as inputs into a spiking neural network for sound detection. the following figure summarize our work.

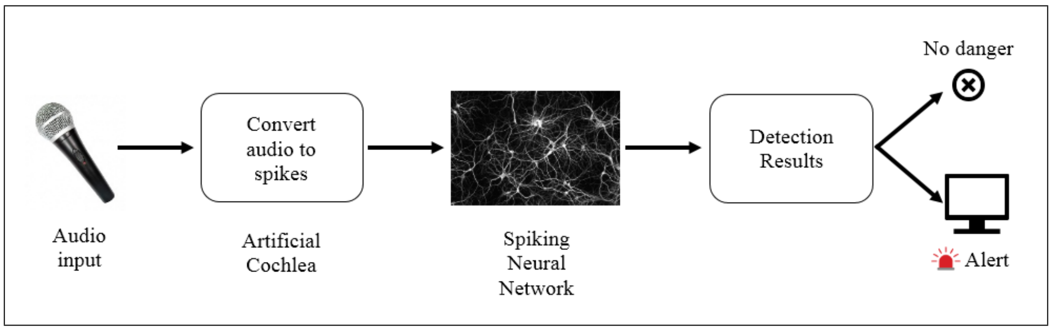


Figure 1: Proposed scheme of sound detection

Our domain of application is sound detection agricultural field related to WATERMED project. The objective of WATERMED 4.0 is to develop and to apply an integrated decision support system based on the Internet of Things, for managing the whole water cycle in agriculture, monitoring water resources (conventional and non-conventional) and water demands including the measure of economic, energy, social and governance factors that influence the water use efficiency in Mediterranean agricultural production areas. The objective of this application is to detect troops of wild boars in order to intercept them or divert them from agricultural fields in time as application results, if there are any wild boars that are detected, an alert is raised and from the strength of the sound, the wild boars can be located.

Training SNNs requires many labeled data that are expensive to obtain in real-world applications, as ~~traditional artificial neural networks (ANNs). In order to address this issue, transfer learning~~ has been proposed and widely used in traditional ANNs, and only very recently to SNNs. We have developed CNN audio neural networks on Nengo which can run SNNs too on different hardware devices (Spinnaker, Loihi, ...). This experimentation is in progress, the first results are encouraging.

V Conclusion

Everyone knows the paradox to build both small and intelligent sensor since the most active research in AI is based on deep convolutional neural networks which are very time-consuming (up to several weeks with GPU servers) and yield enormous energy consumption, despite most recent innovations in parallel digital architectures. However, the brain and biological systems in general, can perform high-performance calculations with much higher efficiency than our most powerful computers, and they do it very quickly and with very low energy consumption.

In parallel, recent works have been published which revolve around audio classification, this subject which has become an interesting subject which pushes researchers to improve classification techniques for much better performance.

In this short paper, we attempted to give an overview of neural networks and the most used classification techniques and give an idea of our work in progress related to audio classification using spiking neural networks. We try to imitate mammal hearing through neuromorphic implementations to design ultra-low-power acoustic and autonomous sensors that could detect very specific events of interest (for instance wild boars) and produce alerts with some associated spatial info. This work is in part supported by the WATERMED 4.0.

References

[1] Alexander Jaffe. Long short-term memory recurrent neural networks for classification of acute hypotensive episodes. 2017.

[2] Christian S. Jensen, Richard T. Snodgrass (auth.), LING LIU, and M. TAMER ÖZSU (eds.). *Encyclopedia of Database Systems*. Springer US, 2009.

[3] Jordi Pons and Xavier Serra. Randomly weighted cnns for (music) audio classification. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 336–340, 2019.

[4] K.S. Rao and Manjunath K.E. Speech recognition using articulatory and excitation source features. *SpringerBriefs in Speech Technology*, 2019.

[5] Theodoros Giannakopoulos and Aggelos Pikrakis. *Introduction to Audio Analysis: A MATLAB Approach*. Academic Press, Inc., USA, 1st edition, 2014.

[6] Li Deng and Douglas O Shaughnessy. *Speech Processing A Dynamic and Optimization Oriented Approach*. CRC Press, 2003.

[7] Mushtaq, Zohaib, Su, and Shun-Feng. Efficient classification of environmental sounds through multiple features aggregation and data enhancement techniques for spectrogram images. *Symmetry*, 12(11), 2020.

[8] P. Dhanalakshmi, Sengottayan Palanivel, and Vivekanandan Ramalingam. Classification of audio signals using aann and gmm. *Applied Soft Computing*, 11:716–723, 01 2011.

[9] Andrey Temko and Climent Nadeu. Classification of acoustic events using svm-based clustering schemes. *Pattern Recogn.*, 39(4):682–694, April 2006.

[10] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

A method for estimating preferences in binary choices

T.Lando[1]

1. University of Bergamo and VSB-TU Ostrava, tommaso.lando@unibg.it

Keywords : decision making, stochastic ordering.

Abstract

Stochastic dominance relations define preorders between random variables by representing general classes of decision makers in terms of preferences. Typically, inference for a given stochastic dominance relation between a pair of random variables, X and Y, is performed in the classic sense, that is, by assuming that the true probability distributions of X and Y are unknown. Differently, here we assume that the distributions of X and Y are known but the preferences of a given decision a maker are uncertain. Based on the recent results of [1], we propose a generalized family of stochastic orders where uncertainty about preferences is modelled by distortion functions, representing random (and latent) subjective revision of probabilities, as perceived by individuals. In particular, the preference of X over Y is defined by:

$$\int_{-\infty}^{x} H \circ F_X(t)dt \leq \int_{-\infty}^{x} H \circ F_Y(t)dt, \forall x \in R,$$

where H is a *random distortion function*, representing risk attitudes, and F_X, F_Y are the cumulative distribution functions of X and Y, respectively. This model generalizes other well known stochastic dominance relations, such as the first and-second-order stochastic dominance. Now, if we denote by Π the probability measure over the space of all possible distortion functions, then, given the distributions F_X, F_Y, by estimating Π we may attempt to estimate the probability of “X is preferred to Y”, as well as an expectation of the preference relation between the two alternatives. Such an estimation is performed by providing questionnaires and surveys on hypothetical gambles, based on the method of the *certainty equivalent*. Responses to these questionnaires furnish information about the risk attitudes of the decision maker. To facilitate the computation, we move to a parametric setting, in which the distortion function may be parametrized by a risk-aversion and a risk-attraction parameters. In this framework, we develop an algorithm which enables the approximation of the probability law Π, which models the behavior of the distortion function, providing an estimated solution for the decision problem of interest. Some illustrations will show how to apply the proposed approach.

References

[1] Lando, T., & Bertoli-Barsotti, L. (2020). Distorted stochastic dominance: a generalized family of stochastic orders. Journal of Mathematical Economics, 90, 132-139.

Deep Learning and Multi-objective Optimization: What is the intersection?

O. Bahri[1] and L. Amodeo[1]

1. LIST3N Lab., University of Technology of Troyes, 10000 Troyes Cedex France.

oumayma.bahri@utt.fr lionel.amodeo@utt.fr

Keywords: Machine Learning, Deep Learning, Architecture Networks, Hyper-Parameters, Multi-objective Optimisation, Evolutionary Algorithms.

1 Introduction

Over many years, Machine Learning (ML) and Optimization (OPT) fields have evolved rapidly but independently from each other. On the one hand, ML research has made many theoretical insights that found practical applications in all areas of science. Deep Learning is a branch of ML, which has recently grown at a remarkable performance and become very popular in this field. In fact, Deep Networks (DN) have been increasingly used in very complex learning task and applied in many application domains such as image processing, pattern recognition, medical diagnosis, etc. On the other hand, OPT, as one of the core components of ML, has attracted much attention of researchers and practitioners. A lot of successful methods and algorithms have been proposed to resolve more complex (or NP-hard) optimization problems.

Today, with the exponential increase in the volume and complexity of data, the interaction of two domains is facing more and more challenges. In particular, we seek to examine the intersection between deep learning and combinatorial optimization.

Several Deep Networks (DNs) have been proposed in the literature such as Recurrent Neural Network (RNN), Deep Belief Network (DBN), Deep Auto Encoder (DAE), and Convolutional Neural Network (CNN). Despite the very interesting performance of each of these DNs, they involve optimization in many rather complicated contexts. For instance, their architecture design is still so far a major challenge. Most of these architectures have been manually designed by experts from well-known companies such as Google. Unfortunately, as a DN architecture has a high number of hyper-parameters, an important issue is how to optimize such parameters?

In this context, researches from the ML area proposed designing this architecture as a Reinforcement Learning (RL) Process. While researchers from the OPT community suggested that better architectures (with well-tuned hyper-parameters) could be found by automated optimization method such as greedy induction methods.

During the last few years, some researchers propose to consider this design task as an optimization problem and then to solve it using an Evolutionary Algorithm EA. For example, for the most difficult DNs like neural network, multiple processing layers need to be performed with a suitable level of accuracy. In this case, EAs have demonstrated their performance and ability to approximate the global optimal and so to escape locally-optimal architectures.

Yet, almost all the existing research works in this direction have mainly focused on the case of single-level optimization problem. To the best of our knowledge, only very few researchers have addressed and treated the architecture design as a bi-level optimization problem where: (1) the upper level minimizes the network complexity and (2) the lower level maximizes the classification accuracy.

Motivated by the originality of such observations, our main idea behind is to further analyze the state-of-the-art regarding this special topic. This opens the door to many interesting questions or issues: Why and when an optimization task is necessary for solving hard deep networks?
How to efficiently achieve optimal DN architecture by a multi-objective optimizer without any need for other pre-training techniques? How to examine the applicability of multi-objective aspects on real-life learning problems?

References

[1] E-G. Talbi (2020). Machine Learning into Metaheuristics: A survey and taxonomy of data-driven metaheuristics . ACM Comput. Surv., hal-02745295.

[2] Louati, H., Bechikh, S., Louati, A., Hung, C. C., & Said, L. B. (2021). Deep convolutional neural network architecture design as a bi-level optimization problem. *Neurocomputing*, *439*, 44-62.

[3] H. Al-Sahaf, Y. Bi, Q. Chen, Q., et al. (2019). A survey on evolutionary machine learning. Journal of the Royal Society of New Zealand, 49(2), 205-228.

[4] S. Sra, S. Nowozin and S. J. Wright (2012). Optimization for machine learning. Mit Press.

[5] K. P. Bennett and E. Parrado-Hernandez (2006). The interplay of optimization and machine learning research. The Journal of Machine Learning Research, 7, 1265-1281.

Energy-efficient buffer allocation : multi-objective study

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. Designing efficient manufacturing systems requires the optimal sizing of the storage spaces (buffer areas). Buffers absorb system disruptions, however, great storage areas result in high work-in-process inventory and investment costs. Therefore, the Buffer Allocation Problem (BAP) remains one of the most researched problems in manufacturing systems design. Nevertheless, the focus has always been put on productivity optimization whereas reaching sustainability requires a special focus on energy efficiency. Hence, we suggest studying the trade-off between throughput and energy consumption when designing efficient storage spaces. A novel energy-efficient form of the BAP for throughput maximization and energy consumption minimization is presented and solved using a multi-objective approach.

Keywords : Buffer allocation problem, Production lines, Multi-objective optimization, Energy efficiency, Epsilon-constraint, NSGA-II

1 Introduction

The buffer allocation problem (BAP) is an NP-hard combinatorial optimization problem [6] that is extensively studied since decades. In this problem, the aim is to find the optimal allocation of storage space among buffer areas in a production line to optimize certain objectives. Analyzing the literature reveals that the objectives to optimize are in most cases the total buffer space allocated to be minimized under a desired production rate (the primal problem [5]) or the equivalent throughput of the system to be maximized under a total limited buffer space (the dual problem [5]), either in a single or a multi-objective optimization approach. Nevertheless, with the current context of ecological awareness, limited energy sources, and increasing energy costs, including the energetic dimension in the BAP becomes crucial. A novel variant of the BAP called the energy-efficient buffer allocation problem (EE-BAP) was presented in [2]. The EE-BAP studies both throughput and energy consumption optimization under a total buffer capacity constraint. Considering the trade-off between both performances, the EE-BAP demonstrates a great potential of energy economics with low throughput deterioration. Nevertheless, a multi-objective study of the EE-BAP could potentially be more effective considering both performances for optimization. Thus, in this paper, the focus is made on the multi-objective study of the EE-BAP investigating optimal/near optimal buffer configurations that respect the total buffer space allowed and provide maximum throughput with minimum energy consumption simultaneously for production systems.

2 Problem formulation

The system considered is a serial production line composed of K unreliable workstations and $K-1$ intermediate buffer areas with finite capacities to be determined. Each machine M_i $\forall i \in \{1..K\}$ is characterized by exponentially distributed failure rate λ_i, repair rate μ_i, and processing rate ω_i. Machines are also characterized by a failure state energy consumption $E_{down,i}$, an idle state energy consumption $E_{no-load,i}$, and an operating energy consumption composed of a constant part $E_{load,i}$ and a variable part $e_{op,i}$ per part processed. ρ_i and E_i are respectively the production rate and the energy consumption of each machine M_i. Finally, ψ is the equivalent throughput of the line and E its total energy consumption. Unlimited supply before the first machine, unlimited storage capacity after the last machine, and operation dependent failure are assumed. Moreover, no setup time is considered and transitions times between machines and buffers are assumed equal to zero. In this study, buffer energy consumption is neglected.

The EE-BAP is studied through a multi-objective framework. Energy consumption and throughput are optimized simultaneously under total buffer space constraint (equation 1).

Find Pareto optimal set of $N = (N_1, N_2, ..., N_{K-1})$ so as to :

$$\begin{cases} maximize \;\; f_1(N) = \psi \\ minimize \;\; f_2(N) = E \\ \text{s.t.} \\ \sum_{j=1}^{K-1} N_j \leq N_{total} & j \in 1...K-1, \\ N_j \in \mathbb{N}^* & j \in 1...K-1. \end{cases} \tag{1}$$

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 [7]). This method has demonstrated a high accuracy with extensively reduced computational time when compared to other methods from the literature, such as the decomposition and aggregation methods. The energy consumption of the production line is evaluated using Markov chain formulations to assess energy consumption per machine state. Further details for throughput and energy consumption evaluation can be found in [1].

4 Multi-objective approach and numerical experiments

The classical epsilon-constraint method (ECM) and the evolutionary multi-objective algorithm non-dominated sorting genetic algorithm NSGA-II are adapted and implemented to solve the EE-BAP described in equation 1. For estimating the Pareto front with ECM, the two versions of the problem are solved: ECM(v1) maximizing the throughput with the energy consumption as a constraint, and ECM(v2) minimizing the energy consumption with the throughput as a constraint. We used a varying number of ϵ_ψ and ϵ_E values. These parameters were chosen equally spaced within an interval formed from the minimum and maximum values of the corresponding single-objective optimization problems. The step is calculated as a function of the feasible domain limits in order to maintain a fixed number of Mixed Integer Non Linear Programming (MINLP) problems to solve for fair comparison of the methods. The NSGA-II is implemented on Python to solve the EE-BAP. It is integrated with Lingo solver 18.0 for the evaluation of the objective functions. Several tests have been first performed in order to set efficiently the different parameters of the algorithm. Final values (table 1) are determined as a compromise between the quality of the final solutions and the convergence time needed for fair comparison of the resolution approaches. The initial population is generated at random according to the number of defined chromosomes. The sorting follows the method used by [3] using also the crowding distance. The tournament method is used for selection, where the confrontation of two individuals of the initial population is made to keep the individuals with the best Pareto front. If the two confronted members are of the same front, the decision is made according to the crowding distance criteria. Moreover, Simulated binary crossover (SBX) with the self-adapted procedure developed in [4] and polynomial mutation are used.

Table 1: NSGA-II parameters

Population size	Number of generations	Crossover probability	Mutation probability
20	50	0.7	$p_m = 1/n$ (where n is the number of decision variables).

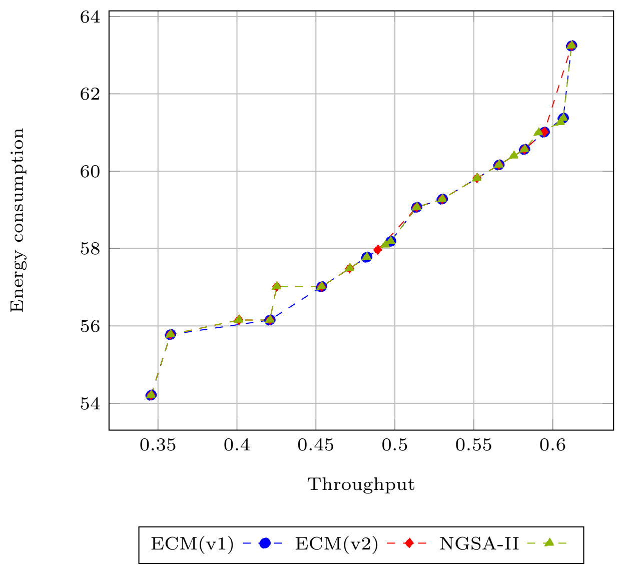


Fig. 1: Instance 1

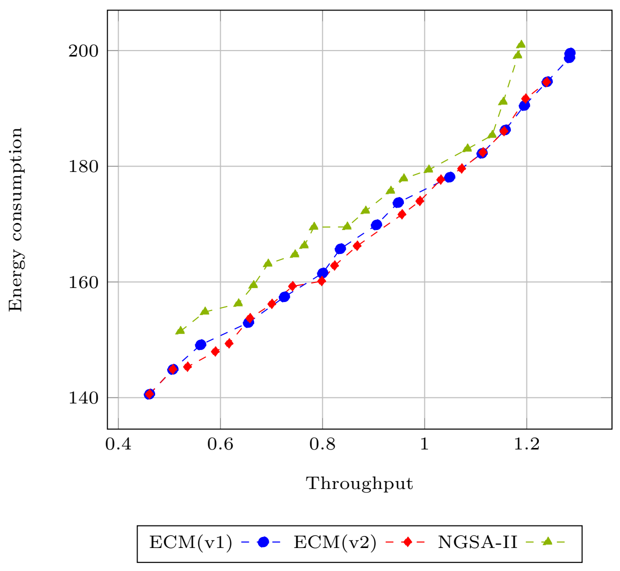


Fig. 2: Instance 2

Numerical experiments are conducted on literature instances. Figures 1 and 2 show the Pareto fronts obtained by ECM and NSGA-II for the test instances given in table 4. Energy parameters are also given in table 5.

Results of the implemented methods are compared using performance indicators. Table 2 gives the overall non-dominated vector generation (ONVG), spacing (SP), hole relative size (HRS), and hyper-volume (HV) metrics of the generated Pareto fronts for each instance. Computational time is also given in table 2. In addition, table 3 presents c-metric (coverage) values.

Table 2: Performance metrics

			ECM(v1)					ECM(v2)					NSGA-II				
Instance	K	N_{total}	ONVG	SP	HRS	HV	CPU(s)	ONVG	SP	HRS	HV	CPU(s)	ONVG	SP	HRS	HV	CPU(s)
1	4	10	13	0.515	2.692	0.212	14	16	0.610	3.933	0.211	8	20	0.493	4.121	0.212	55
2	10	310	16	1.138	1.206	0.258	5720	19	1.073	1.981	0.254	1236	20	1.261	3.022	0.225	2271

Table 3: Coverage metric

	Instance 1			Instance 2		
	ECM(v1)	ECM(v2)	NSGA-II	ECM(v1)	ECM(v2)	NSGA-II
ECM(v1)	0	0.438	0.4	0	0.105	0.833
ECM(v2)	0.769	0	0.55	0.186	0	0.944
NSGA-II	0.692	0.563	0	0	0	0

5 Discussion and conclusions

For cardinality, although the ONVG is not a pertinent measure on its own, it can be noticed from the figures and table 2 that ECM and NSGA-II generate a good number of solutions in the Pareto front approximation. As of the distribution and spread metrics, both methods obtain close values with the ECM outperforming the NSGA-II for the HRS and HV metrics. For the sets coverage comparison, from table 3, we can see that for the Pareto front generated in the case of instance 2, 83% to 94% of the solutions obtained by NSGA-II are dominated by at least one solution of the ECM. In fact, the NSGA-II performs very well for the small sized instance (instance 1) but, when the problem complexity increases, it requires greater computational time to generate a Pareto front at least as good as the ECM. This is mainly due to the efficiency of the non-linear programming method of the EE-BAP. Indeed, using the mathematical solver to solve the EE-BAP for each execution of the ECM takes at most few minutes each. Whereas, the NSGA-II requires various evaluations (population size $\times$ number of generations) using the solver which results in greater computational time.

Future work focuses on the multi-objective study of the energy-efficient BAP considering other approaches and combining other design options such as equipment selection or system configuration that could bring interesting insights. Moreover, this novel BAP allows to explore a new field of research and tackle more interesting areas in manufacturing systems design.

References

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

[2] Y. Alaouchiche, Y. Ouazene, and F. Yalaoui. Energy-efficient buffer allocation problem in unreliable production lines. *The International Journal of Advanced Manufacturing Technology*, pages 1–15, 2021.

[3] 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.

[4] K. Deb, K. Sindhya, and T. Okabe. Self-adaptive simulated binary crossover for real-parameter optimization. In *Proceedings of the 9th annual conference on genetic and evolutionary computation*, pages 1187–1194, 2007.

[5] S. B. Gershwin and J. E. Schor. Efficient algorithms for buffer space allocation. *Annals of Operations research*, 93(1-4):117–144, 2000.

[6] J. MacGregor Smith and F. R. Cruz. The buffer allocation problem for general finite buffer queueing networks. *IIE Transactions*, 37(4):343–365, 2005.

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

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

[9] H. Papadopoulos and M. Vidalis. A heuristic algorithm for the buffer allocation in unreliable unbalanced production lines. *Computers & Industrial Engineering*, 41(3):261–277, 2001.

[10] I. Sabuncuoglu, E. Erel, and Y. Gocgun. Analysis of serial production lines: characterisation study and a new heuristic procedure for optimal buffer allocation. *International Journal of Production Research*, 44(13):2499–2523, 2006.

[11] D. Seong, S. Y. Chang, and Y. Hong. Heuristic algorithms for buffer allocation in a production line with unreliable machines. *International Journal of Production Research*, 33(7):1989–2005, 1995.

Table 4: Parameters of production lines reported in [11, 5, 9, 10, 8, 2]

Instance	K	$\sum_{j=1}^{K-1} N_j$	$(\lambda_i, \mu_i, \omega_i)$
1	4	10	(0.07,0.17,3.7); (0.11,0.37,1.5); (0.49,0.78,1.1); (0.19,0.50,3.0)
3	10	310	(0.20,0.70,2.5); (0.10,0.60,1.5); (0.30,0.80,2.8); (0.20,0.80,3.6); (0.10,0.70,2.1); (0.10,0.60,1.9); (0.30,0.80,2.7);(0.20,0.50,3.0); (0.30,0.60,2.0); (0.10,0.70,2.1)

Table 5: Energy parameters [2]

State Energy	E_{down}	E_{noload}	E_{cload}	e_{op}(*per part*)
Energy Units	1	10	10	8

A Study on the Influence of Runtime Uncertainty in the Optimization of Software Programs

José M. Aragón-Jurado[1], Juan Carlos De la Torre[1], El-Ghazali Talbi[2], and Bernabé Dorronsoro[1]

[1] Department of Computer Science Engineering, Engineering School, University of Cadiz, Spain
{juan.detorre, bernabe.dorornsoro}@uca.es
[2] University of Lille, CRIStAL UMR CNRS 9189, Inria-Lille Nord Europe, France
el-ghazali.talbi@univ-lille.fr

1 Introduction

Every software program (SW) needs to be compiled in order to run on a given hardware platform (HW). Compilers are the main tool to carry out this task. In the process, compiler infrastructures also aim at optimizing the program execution time. For that, a generic combination of code transformations is used to optimize the performance of a SW executed on any HW architecture. However, the specific properties of both the SW and the HW architecture are not taken into account. Therefore, current compilers do not allow fully and efficiently exploiting the features of the underlying HW, and this purpose is extremely difficult, especially when new architectures with well differentiated HW elements are continuously being released.

LLVM [6] is one of the most important compiler infrastructures nowadays. It makes use of an intermediate representation of code, called LLVM IR, and it offers a wide range of code transformations on this representation, called passes. In order to take advantage of the properties of any architecture, a specific combination of such passes is necessary. The different compiler optimization options LLVM offers (-O1, -O2, -O3, etc.) are generic combinations of passes that usually perform well on different architectures.

Finding a specific sequence of passes that optimizes the execution time of a given SW on a HW platform is a difficult task. As it was shown in [3], the impact of applying any pass on the SW performance is really low, being negligible in most cases. Nevertheless, when it is applied in combination with other passes the SW performance can be significantly improved. Furthermore, the order in which the sequence of passes is applied to the SW is of outmost importance, an adding multiple repetitions of the same pass can also have a positive impact.

Additionally, measuring SW execution time is not a trivial task. The reason is that there are different elements such as operating system programs and services that can influence on the SW performance. These inaccuracies in SW execution time cause that different values can be measured when replicating the experiment.

In this work, we compare and evaluate several different methods to deal with the uncertainty in SW execution time when optimizing it. This optimization process is made with a genetic algorithm and targets finding a sequence of passes that minimize the SW execution time. We target in this work a low power computing architecture, motivated by the fact that it is specially significant to optimize SW for such low computing capacity device. It is of major importance correctly dealing with the uncertainty in the runtime measures because it is used to guide the search, and small measured improvements/decays can be caused by the uncertainty, and they might wrongly guide the algorithm in the optimization process.

2 Dealing with Uncertainty in SW Run Time

Generally speaking, uncertainty can be defined as the noise that causes the data deviate from its original values. When we are dealing with large amounts of data, uncertainty appears frequently. In the context of an optimization process, uncertainty can affect the decision variables, the fitness function, or the environment conditions [4, 5].

Our problem is about finding the best sequence of LLVM passes which minimize the runtime of a SW when executed on a specific HW architecture. All LLVM passes are represented with a unique integer number, and a problem solution is an array of integers, representing the passes that should be applied and their order.

In order to evaluate the fitness of a solution, we first compile the program using the sequence of passes, and then the program execution time is measured. The uncertainty present in the measurements makes fitness values assignment difficult. Furthermore, noise probability distribution is unknown, making it difficult to deal with the uncertainty it causes.

In this work, four methods taken from the literature are compared and evaluate to deal with the uncertainty present in the fitness value of a solution:

- **Single run**. The optimized program is executed once, measuring its execution time.
- **Median**. Five independent time measurements are taken, computing the median.
- **Interval**. Five independent time measurements are taken, and based on them, a confidence interval is made using the bootstrap method. The order relation presented in [2] is used to establish the quality of the individuals.
- **Worst-case**. Five independent time measurements are taken, taking the worst one as fitness.

To solve our problem we make use of a cellular Genetic Algorithm [1] to find the best combination of passes to optimize the execution time of a benchmark SW, composed by sequentially running all programs in Polybench [7] benchmark. For the experiments, we used a Raspberry Pi 3 B+ device with Broadcom BCM2837B0 Cortex-A53 64-bit and 1.4 GHz processor as the target architecture.

3 Results and Discussion

Two independent executions of the cGA have been done for each of the four methods considered to evaluate the fitness of individuals, and we selected the best obtained solution per method. In order to evaluate the accuracy computed by each proposed fitness function, we plot in Figure 1 the fitness of the selected solution for each method (as a point), together with the boxplot of the results of running the same optimized program for $1,000$ independent times. This way, we can compare the fitness value computed by the cGA with the true fitness of the solution. A correct fitness estimation would need to be placed inside the box, next to the median.

We can observe in the figure how most of the studied methods offer too optimistic estimations of the fitness value. The single run method, which does not deal with the possible uncertainties is clearly the worst one, offering a fitness value that is close to the lowest measure in the $1,000$ runs experiment. The median and worst-case methods offer better estimations, but still far better than the lower quartile. On the other hand, the interval-based method performs an accurate estimation of the true performance of the optimized SW program, locating the highest and lowest value of the interval inside the box, with the median of the box centered between them.

Probably, performing a higher number of executions of the SW version to evaluate might help improving the accuracy of metrics like the ones based on the median or on the worst-case. However, the fitness evaluation is a costly process for the considered problem, and the time needed to evaluate a solution is proportional to the number of runs performed of the proposed solution. Therefore, finding accurate methods to evaluate solutions in this uncertain environment with the lowest possible number of runs of the program is of utmost importance.

We now compare of the quality of the results found by the GA using each of the four fitness functions. Figure 2 shows the measures after executing $1,000$ times the best solution found by the GA using each method. Boxplot notches represent significant differences among their respective methods when they are not overlapped. Therefore, it can be seen that differences between all methods are significant. We can see that, despite the worst-case method is not the most accurate in the fitness estimation, it can find the SW version that shows the lowest execution time. This can be motivated by the low number of independent runs of the optimization algorithm performed. It should be noted that the execution time of the benchmark without any optimization is around 1.9 seconds, and it goes down to 1.5 seconds when using the optimization flag -O3. Our GA finds a better optimization sequence than the one applied by -O3 (around 17% faster), and improves the original program without any optimization by 33%.

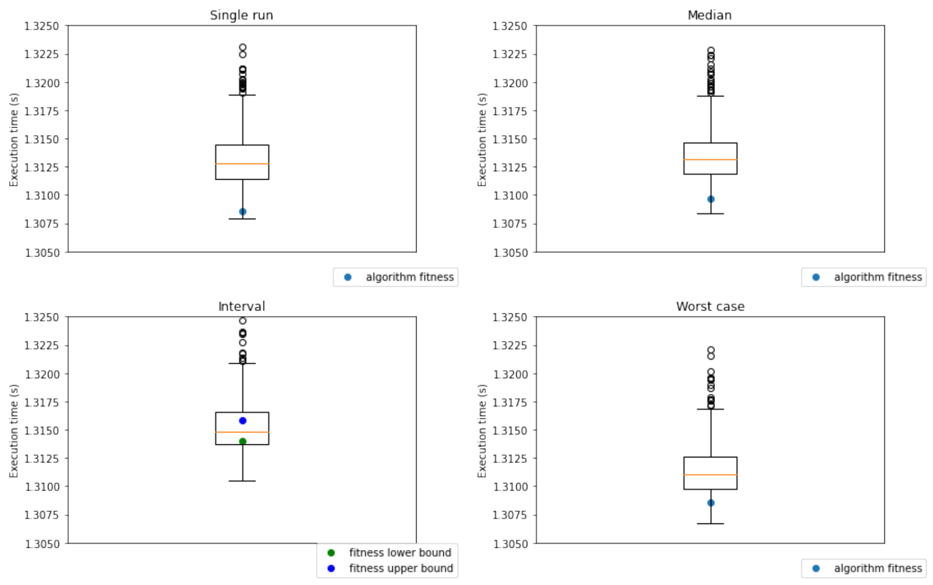


Fig. 1. Accuracy of the fitness value of the best obtained solutions for every studied method, compared with 1, 000 independent runs of the same SW version.

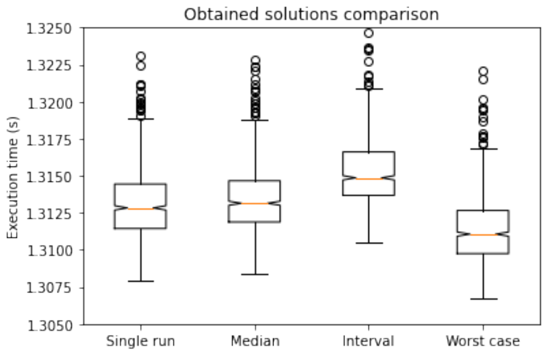


Fig. 2. Comparison of the performance of the best SW versions found with the four studied methods.

4 Conclusions and Future Work

In this work, we focus on the optimization of SW programs using a cellular genetic algorithm to minimize its execution time in a low power computing architecutre. In particular, we study and compare four different metrics to deal with the uncertainty present in the execution time measures. The problem is modeled as a combinatorial optimization problem.

The obtained results evidence the uncertainty present in the considered problem and the need to treat it in order to achieve reliable results when computing the fitness value of a solution. Even some of methods designed for uncertain environments, as it is the case of those based on the median or on the worst-case out of five independent runs, fail to provide good estimations of the true performance of the SW program. Only the method based on the interval, which performs an oversampling technique, is able to accurately estimate it, among the four methods studied in this work. Interestingly, the worst-case method is the one that found the most optimized SW version.

As future work, we consider expanding the study with the optimization of other SW benchmarks as well as the evaluation of different HW architectures. It is also interesting to consider the definition of new methods to deal with uncertainty more accurately and with a lower computational cost.

Acknowledgments

This work was supported by projects iSUN – RTI2018-100754-B-I00 (MCIU/AEI/ERDF, EU), OPTIMALE – FEDER-UCA18-108393 (ERDF, EU), and GENIUS – P18-2399 (Junta de Andalucía, ERDF, EU).

References

1. E. Alba and B. Dorronsoro. *Cellular Genetic Algorithms*, volume 42 of *Operations Research/Computer Science Interfaces.* Springer-Verlag Heidelberg, 2008.
2. A. K. Bhunia and S. S. Samanta. A study of interval metric and its application in multi-objective optimization with interval objectives. *Computers & Industrial Engineering*, 75:169 – 178, 2014.
3. J. C. de la Torre, P. Ruiz, B. Dorronsoro, and P. L. Galindo. Analyzing the influence of LLVM code optimization passes on software performance. In J. Medina, M. Ojeda-Aciego, J. L. Verdegay, I. Perfilieva, B. Bouchon-Meunier, and R. R. Yager, editors, *Information Processing and Management of Uncertainty in Knowledge-Based Systems. Applications*, pages 272–283, Cham, 2018. Springer International Publishing.
4. Z. He, G. G. Yen, and J. Lv. Evolutionary multi-objective optimization with robustness enhancement. *IEEE Transactions on Evolutionary Computation*, 24:494 – 507, 2019.
5. Y. Jin and J. Branke. Evolutionary optimization in uncertain environments — a survey. *IEEE Transactions on Evolutionary Computation*, 9:303 – 317, 2005.
6. LLVM. The llvm compiler infrastructure. https://llvm.org/.
7. L.-N. Pouchet. Polybench/c – homepage of louis-noël pouchet. http://web.cse.ohio-state.edu/ pouchet.2/software/polybench/.

Enhancing the Performance of Software Programs for Devices with Limited Computing Capacity

Carlos Benito-Jareño[1], Juan Carlos De la Torre[1], Pascal Bouvry[2], and Bernabé Dorronsoro[1]

[1] Department of Computer Science Engineering, Engineering School, University of Cadiz, Spain
{juan.detorre, bernabe.dorornsoro}@uca.es
[2] Department of Computer Science, Faculty of Science, Technology and Medicine, Université du Luxembourg
pascal.bouvry@uni.lu

1 Introduction

Mobile computing devices are offering higher and higher computing capacity in the last few years, thanks to the quick advances in technology. However, this increase in processing power comes at the expense of a significant increase in energy consumption. This is, precisely, a major issue in battery dependent devices, because it supposes a serious limitation for their autonomy.

Hardware (HW) designers are fully aware of this problem, and they often include features in their releases with the aim of minimizing the power consumption of the device [3, 6]. However, in order to fully take profit of these features, software (SW) needs to be designed with these specific features in mind, since it is what ultimately determines the behavior of HW. Designing such SW is not easy: it requires deep knowledge on the platform where it will be executed, and continuous modifications would be needed to adapt it to new HW releases. These facts make this approach impractical [4].

Compilers arise as a feasible alternative to designing SW that makes an efficient use of the HW architecture. In the process of generating the executable file, the compiler can make use of those instructions that take profit of the HW specific features. In addition, compilers are also able to modify the code to achieve a better performance. However, they apply some generic sequences of transformations on the code that usually leads to acceptable gains in performance in any architecture. For this reason, more advanced compilers are desirable, able to modify the SW source code to make the best use of the HW platform where it executes.

This work is a preliminary study in which we tackle the automatic optimization of SW to enhance the use it makes of the underlying architecture, reducing this way both its runtime and the energy consumption of the device when executing it. The problem is modelled as a combinatorial optimization problem and, because of its high computational demand (evaluating one solution takes a few tens of seconds), we propose a novel multi-objective cellular genetic algorithm [1, 7] with a micro population to solve it. This is, to the best of our knowledge, the first time that this problem is defined and solved as a multi-objective combinatorial optimization problem.

2 Modelling and solving the problem

The SW optimization problem we consider in this work is defined as a combinatorial optimization problem, whose objective is to find a sequence of source code transformations that modifies the SW with the goal of minimizing both its runtime and the energy consumption of the device when executing it. Therefore, finding a good solution to this problem means obtaining a modified version of the SW code so that it makes the best possible use of the considered HW architecture.

We will make use of the well known LLVM compiler infrastructure [5] to model the problem. LLVM offers a large number of source code transformations, called passes, which can be used to optimize the code. Therefore, the problem is to find a sequence of passes so that the runtime and consumption of the SW are minimized. There are several features of these passes that make the problem really hard to solve. First, it is difficult to know what will be the impact of applying a given pass to a SW code, as it depends on both the HW and on the code itself. Second, passes have by themselves a really low impact on SW performance (generally negligible), but a significant one when applied in combination with other pass(es) [2]. Third, applying a given pass several times has an impact on the resulting code. And fourth, the order in which passes are applied in the code is also important.

In order to evaluate the fitness of an individual, we need to execute the optimized program to measure both its runtime and its energy consumption. Depending on the HW and the program in hand, this evaluation can take quite some time, and if we want some degree of validity, considering the uncertainty tied to these types of measurements, we should perform several evaluations and take some statistics from the data of these repetitions.

Consequently, we have a strong limitation on the number of evaluations we can make so that the genetic algorithm (GA) takes a reasonable amount of time to execute. Therefore, in order to allow the GA performing enough generations (or epochs) to evolve without implying a large number of fitness function evaluations, we use a highly reduced population size. This raises the problem of premature convergence converge, consequently limiting the search capability of the algorithm.

We propose in this work a novel multi-objective micro-GA, that we call *MicroMOCell*. It is an adaptation of the well-known MOCell algorithm [7] in which the population size is dramatically reduced and the feedback of solutions from the archive is avoided in order to mitigate premature convergence. In addition, we implement a population restart mechanism to deal with the premature convergence, which is measured according to the similarity of the individuals on every epoch (according to their Hamming distances).

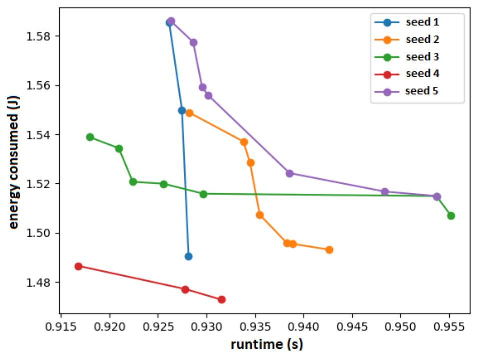
3 Results and discussion

We consider in this work the optimization of the execution time and consumption of a benchmark SW on a Nexus 5 smartphone. The SW to optimize is composed by sequentially running all programs in Polybench [8] benchmark. To measure the energy consumption of the phone when executing the SW, we use a commercial RIDEN RD6006 power supply unit, configured to provide 4.35 V to the mobile phone, the same voltage its battery offers when fully charged.

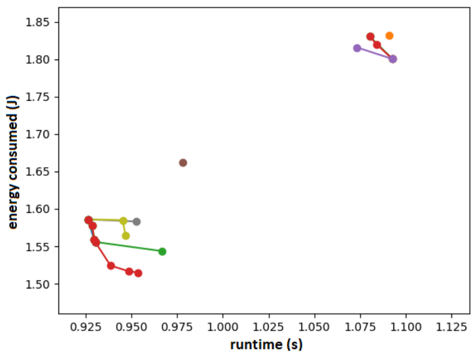
We performed five independent runs of MicroMOCell to optimize our benchmark SW. It is configured with 15 individuals in the population, and $10,000$ evaluations are allowed. We use two-points recombination crossover and the mutation operator changes a pass by another random one. The rest of the configuration is as in MOCell. The Pareto front obtained in every run is shown in Figure 1a. One interesting observation in the figure is the presence of both horizontal and vertical shaped Pareto fronts. These fronts demonstrate the multi-objective nature of the problem: the horizontal Pareto front (the red one) contains solutions with similar values of energy consumption and different runtime values, while the vertical one (in blue color) offers solutions with similar runtime but different consumption. The solutions obtained using seed number 4 (the red one) dominate all solutions found in the other four experiments. The obtained results improve the original SW program by up to 78% in runtime and 64% in consumption, and the best optimized version LLVM can find (obtained with option -O1) by 25% and 20% in time and consumption, respectively.

Figure 1c shows all the generations where there was a contribution of at least one new solution to the archive (i.e., the Pareto front at that stage of the search), for every independent run. It can be seen how the Pareto front is mainly built in the first 250 epochs, and after that point the algorithm makes some sparse contributions to it. After these results, and given that the executions are so slow, it can seem reasonable performing around 300 epochs to get accurate solutions in a reasonable amount of time. However, Figure 1b shows the different Pareto fronts found during the search for some sample run (with seed 5), and we can see that the last contribution to the Pareto front is really important, offering four new high quality results with lower consumption, and this Pareto front update is made after a long period without updating it, later than epoch 600.

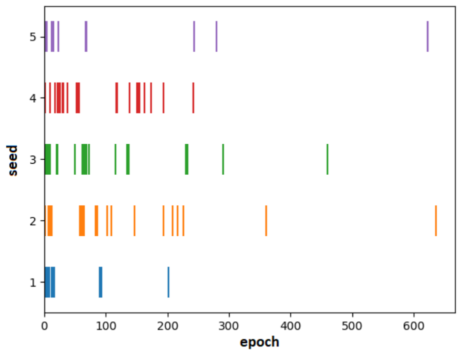
Finally, we are interested in analyzing when the algorithm detects premature convergence to activate the population re-initialization. Figure 1d shows the epochs in which this occurs, for the five studied independent runs. As it can be seen, the population is regularly re-initialized until the end of the run. In average, this happens every $8,7$ epochs, being the minimum and maximum periods without detecting premature convergence of 3 and 34 epochs, respectively.



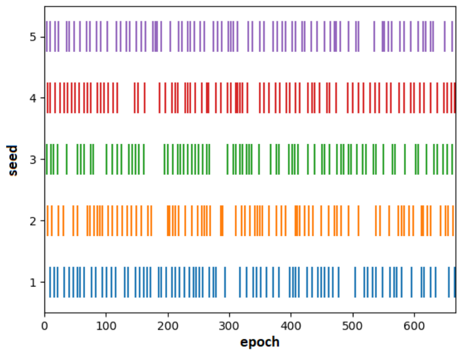
(a) Pareto fronts obtained in the five independent runs performed.



(b) Evolution of the Pareto front during the search for run with seed 5.



(c) Epochs of each MicroMOCell execution improving the Pareto front.



(d) Generations where premature convergence is detected, for the five MicroMOCell executions performed.

Fig. 1: Analysis of the performance of MicroMOCell.

4 Conclusions and future work

We made in this paper a first approach towards the automatic optimization of SW programs on mobile phones. The problem is modelled as a combinatorial optimization one and solved with a novel multi-objective micro cellular genetic algorithm with a population restart mechanism. The objectives were minimizing the SW execution time, as well as the energy consumption of the device when executing it.

The best version found of the SW improves the original one by up to 78% in runtime and 64% in energy consumption. These are highly promising results that deserve further investigations in this research line. In particular, we propose as future work: extending our experiments; designing new multi-objective algorithms, able to deal with the premature convergence; considering other SW programs to optimize; and characterizing the uncertainty in the time and consumption metrics, so as to use this information to take it into account in the optimization process.

Acknowledgments

This work was supported by projects iSUN – RTI2018-100754-B-I00 (MCIU/AEI/ERDF, EU), OPTIMALE – FEDER-UCA18-108393 (ERDF, EU), and GENIUS – P18-2399 (Junta de Andalucía, ERDF, EU).

References

1. E. Alba and B. Dorronsoro. *Cellular Genetic Algorithms*, volume 42 of *Operations Research/Computer Science Interfaces*. Springer-Verlag Heidelberg, 2008.
2. J. C. de la Torre, P. Ruiz, B. Dorronsoro, and P. L. Galindo. Analyzing the influence of LLVM code optimization passes on software performance. In J. Medina, M. Ojeda-Aciego, J. L. Verdegay, I. Perfilieva, B. Bouchon-Meunier, and R. R. Yager, editors, *Information Processing and Management of Uncertainty in Knowledge-Based Systems. Applications*, pages 272–283. Springer International Publishing, 2018.
3. T. Johann, M. Dick, S. Naumann, and E. Kern. How to measure energy-efficiency of software: Metrics and measurement results. In *Proceedings of the First International Workshop on Green and Sustainable Software*, GREENS '12, page 51–54. IEEE Press, 2012.
4. P. Lago, R. Kazman, N. Meyer, M. Morisio, H. A. Müller, and F. Paulisch. Exploring initial challenges for green software engineering: Summary of the first greens workshop, at icse 2012. *SIGSOFT Softw. Eng. Notes*, 38(1):31–33, Jan. 2013.
5. LLVM. The llvm compiler infrastructure. https://llvm.org/.
6. M. Morisio, N. Meyer, H. A. Müller, P. Lago, and G. Scanniello. 4¡sup¿th¡/sup¿ international workshop on green and sustainable software (greens 2015). In *Proceedings of the 37th International Conference on Software Engineering - Volume 2*, ICSE '15, page 981–982. IEEE Press, 2015.
7. A. Nebro, J. Durillo, F. Luna, B. Dorronsoro, and E. Alba. Mocell: A cellular genetic algorithm for multiobjective optimization. *Int. J. Intell. Syst.*, 24:726–746, 07 2009.
8. L.-N. Pouchet. Polybench/c – homepage of louis-noël pouchet. http://web.cse.ohio-state.edu/ pouchet.2/software/polybench/.

ClusterTune : Tuning Metaheuristics Using Clustering for Scheduling Problems

M. Athmani[1] T. Arbaoui[1] F. Yalaou[1]

Université de Technologie de Troyes
med_lamine.athmani@utt.fr
taha.arbaoui@utt.fr
farouk.yalaoui@utt.fr

Abstract. The initial setting of metaheuristics' parameters heavily impacts the quality of the obtained solutions. Thus, the process of finding the best configuration of parameters becomes crucial. In this paper, we propose an instance-specific clustering-based tuning approach called ClusterTune. This approach is a tuning method that uses clustering to parameterize a metaheuristic algorithm used to solve the parallel machine scheduling problem with machine- and sequence-dependent setup times (denoted Rm — Sijk — Cmax). When compared to the untuned metaheuristic algorithm, results show that the metaheuristic algorithm tuned using ClusterTune yields better results in more than 70% of the instances. Moreover, ClusterTune is capable of providing the best parameters' configuration in several instances with 10-30 times less effort. These results also show the importance of using instance characterization in order to tune algorithms.

Keywords : Parameter tuning, Clustering, Metaheuristics, Scheduling, Instance characterization.

1 Introduction

In order to solve combinatorial optimization problems, many methods and algorithms are used, whether they are heuristics, metaheuristics or hyperheuristics, these methods involve several choices and decisions that impact the method's performance. Among these decisions, we find the initial setting of parameters, known as the parameter tuning problem. This problem has been widely studied in the literature. (Eiben and smith [4]) survey tuning methods and classify them into 4 categories: sampling methods, model based methods, meta-evolutionary algorithms and screening methods. The goal of these models is to reduce the number of executions during the tuning phase in order to determine the best configurations. In this work we introduce a novel tuning framework based on a clustering method called ClusterTune. This framework uses the instances' characterization in order to return the best parameters instance-per-instance in contrast to most existing benchmark-wide tuners. This approach is tested to parameterize a genetic algorithm used to solve the parallel machine scheduling problem with machine- and sequence-dependent setup times (denoted $R_m|S_{ijk}|C_{max}$). This problem, chosen because of its complexity, is also widely studied in the literature [7] [3] because of its multiple domain of applications (Vallada and Ruiz [8]). In this problem, we have a set of N tasks that can't be preempted and that have to be processed by one of the M unrelated continuously available parallel machines. Setup times that depend not only on the machine but also the sequence of the tasks are also considered. The proposed framework is tested on many benchmarks [6] [8] available on the literature and shows that the approach tends to adapt well to the different instances' characteristics.

2 Proposed framework

As already mentioned, most of the existing methods are generalist tuners that aim to find configurations that work best for all the benchmarks [1] [2] [5] while disregarding the characteristics of the instances inside that benchmark. On the other hand, instance-specific tuners are computationally expensive and require many executions for each instance. In order to overcome this, we propose our framework called ClusterTune, an instance-specific tuning method based on clustering, and which is a compromise between dataset-based and instance-based tuning. This framework includes four phases. First, we tune our method with each instance using a dedicated computationally expensive tuning method to get a dataset of (Instance, best configuration). Second, we group similar instances together in clusters according to their characteristics. Third, then label each cluster with a list of configurations that work best for the instances inside that cluster. In the last online phase, and given a new instance, we first determine the cluster it belongs to using our trained clustering model and propose the label of that cluster (list of configurations) to be used on the optimization method for this specific instance. The architecture is as follows:

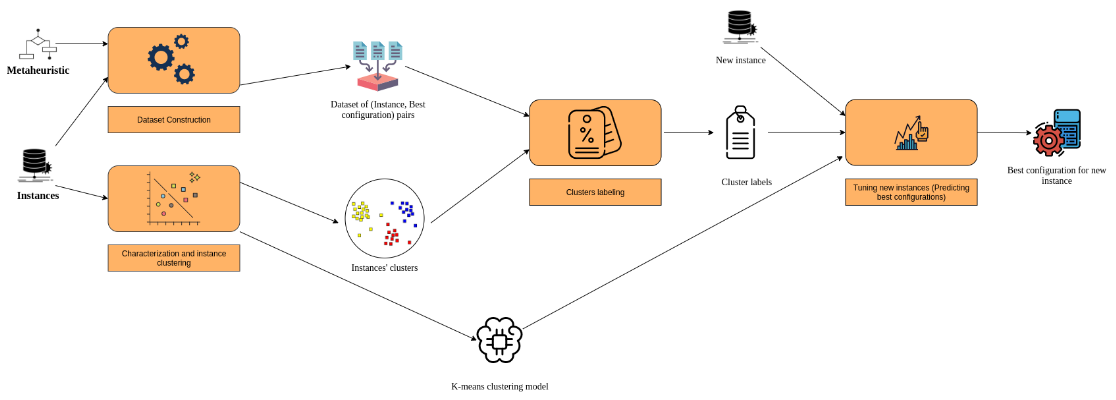

Metaheuristic
Dataset Construction
Instances
Characterization and instance clustering
Dataset of (Instance, Best configuration) pairs
Instances' clusters
K-means clustering model
Clusters labeling
New instance
Cluster labels
Tuning new instances (Predicting best configurations)
Best configuration for new instance

Fig. 1: ClusterTune architecture

The first phase consists of "dataset construction", the goal is to determine the best configuration for each instance of our existing benchmark. We can use a computationally expensive tuning method in this phase since it will only be executed once in order to construct the dataset. The second phase consists of the characterization and clustering of instances, the goal is to choose a characterization that best separates the instances and then cluster the instances using it. This characterization is generally based on a group of problem-dependent features and requires domain knowledge. Next, we used a clustering algorithm in order to group the instances into clusters according to their characteristics, the goal is to group similar instances together and distinguish the different ones. In the third phase, we use the dataset from phase one in order to label each cluster with the most occurring parameter configurations of instances inside that cluster. In the fourth phase, which is the online phase that is used to tune new instances, we first extract the same features that characterize our instances. Next, we use the trained clustering to place the new instance in one of the existing clusters. Finally the label of this cluster is proposed as the list of promising configurations for this instance, by proposing a list of configurations, we present several candidates instead of just one and it's up to the user to determine how many candidates he wants to test in order to get better performances.

3 Numerical experiments

We present the results of applying ClusterTune to tune the genetic algorithm used to solve the problem of unrelated parallel machines with sequence and machine-dependent setup time for makespan minimization $R_m|S_{ijk}|C_{max}$. We propose 3 characterizations C1, C2, C3 to characterize the instances of our problem, we also test with different numbers of configurations that label the clusters ranging in 3, 5, 10 configurations per cluster (T3, T5, T10). The tests are done on the two existing literature benchmarks of Alsalem and Vallada, we generated two extra benchmarks to test the approach in called Set1 and Set2. Results of Table 1 shows a metric based on the Unified configuration set by(Vallada and Ruiz 2011)Vallada and Ruiz [2011], (pop size = 80, p c = 0. 5, pm = 0. 5, p ls = 1). This metric represents the percentage of the number of times that the algorithm tuned by clustering is more efficient (strictly superior)than the genetic algorithm without tuning effort (untuned).

		Set 1	Alsalem			Set 2	Vallada
			Balanced	Proc	Setup		
T_3	C1	81.78	56.85	57.22	54.17	79.89	59.79
	C2	81.78	56.30	57.04	51.58	79.07	60.83
	C3	81.78	56.30	57.04	51.58	79.41	55.10
T_5	C1	87.08	64.07	62.59	61.78	86.37	68.44
	C2	87.08	62.41	64.07	58.26	86.07	69.17
	C3	87.08	62.41	64.07	58.26	86.52	63.23
T_{10}	C1	91.54	69.44	68.70	66.60	91.74	76.15
	C2	91.54	67.41	70.0	64.56	92.30	76.88
	C3	91.54	67.41	70.0	64.56	91.74	69.48

Table 1: All Clustering vs Base

4 Conclusion and perspectives

In this paper we presented a novel instance specific clustering-based tuning framework to solve the parameter tuning problem for metaheuristics, the results show promising results when applied to tune a genetic algorithm used to solve the problem of unrelated parallel machines with sequence and machine-dependent setup time for makespan minimization Rm — Sijk — Cmax in less computational time than classical tuning approaches. Future work includes using this approach to tune other optimization algorithms applied on different optimization problems, other machine learning techniques should also be explored in order to solve the parameter tuning problem.

References

[1] Thomas Bartz-Beielstein, Konstantinos E Parsopoulos, Michael N Vrahatis, et al. "Analysis of particle swarm optimization using computational statistics". In: *Proceedings of the international conference of numerical analysis and applied mathematics (ICNAAM 2004)*. 2004, pp. 34–37.

[2] Mauro Birattari and Janusz Kacprzyk. *Tuning metaheuristics: a machine learning perspective.* Vol. 197. Springer, 2009.

[3] Rodney Oliveira Marinho Diana et al. "An immune-inspired algorithm for an unrelated parallel machines' scheduling problem with sequence and machine dependent setup-times for makespan minimisation". In: *Neurocomputing* 163 (2015), pp. 94–105.

[4] Agoston E Eiben and Selmar K Smit. "Parameter tuning for configuring and analyzing evolutionary algorithms". In: *Swarm and Evolutionary Computation* 1.1 (2011), pp. 19–31.

[5] Volker Nannen and Agoston E Eiben. "Efficient relevance estimation and value calibration of evolutionary algorithm parameters". In: *2007 IEEE congress on evolutionary computation.* IEEE. 2007, pp. 103–110.

[6] G. Rabadi. "Scheduling research virtual center." In: (2008). URL: `http://schedulingresearch.com.`.

[7] A Al-Salem. "Scheduling to minimize makespan on unrelated parallel machines with sequence dependent setup times". In: *Engineering Journal of the University of Qatar* 17.1 (2004), pp. 177–187.

[8] Eva Vallada and Rubén Ruiz. "A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times". In: *European Journal of Operational Research* 211.3 (2011), pp. 612–622.

Hidden genes genetic algorithm applied to variable-sized design space optimal layout problems.

J. Gamot[1,2], M. Balesdent[1], A. Tremolet[1], R. Wuilbercq[1], N. Melab[2], E.-G. Talbi[2]

[1] ONERA/DTIS, Université Paris-Saclay, Palaiseau
juliette.gamot@onera.fr, mathieu.balesdent@onera.fr, arnault.tremolet@onera.fr, romain.wuilbercq@onera.fr
[2] INRIA, Université de Lille, Campus Scientifique, Avenue Henri Poincaré, 59655 Villeneuve d'Ascq
nouredine.melab@inria.fr, el-ghazali.talbi@inria.fr

1 Introduction

The layout design of a system consists in placing components (various instruments and equipment) into an available space. This key aspect has a critical impact on the overall system performance. The layout optimization involves optimizing one or several objectives while satisfying multiple constraints (both geometrical and functional). This type of problems belongs to the class of NP-hard problems due to its mathematical and engineering complexity, and is therefore not solvable by classical optimization methods [1]. During the past 20 years, various methods have been developed in order to solve layout optimization problems. Mostly, meta-heuristic algorithms (e.g. evolutionary algorithms, swarm intelligence) have been used [2], sometimes hybridized with other techniques [3], but also multi-agent methods or others [4].

In most of real-world engineering design problems, many architecture choices are to be made and are often represented as discrete variables in addition to other continuous design variables. Furthermore, in various applications, some of these architecture choices have an impact on the problem definition (number of variables to be optimized, number of constraints). For instance, a possible choice would be either to place in the container two small tanks of fuel or a large one. The number of components is consequently not the same in both cases and hence, the number of design variables will vary from one sub-problem to another. Allowing the number of components to vary during the optimization may provide a better solution at the end. This type of problems is referred to in the literature as both mixed-variable problems and variable-sized design space (VSDS) problems. Most of the aforementioned algorithms are designed to solve fixed-sized design space problems even though some of them can address the mixed-variables problems.

VSDS layout optimization problems are, to the best of our knowledge, not covered in the literature. Previous research on VSDS problems using Genetic Algorithm (GA) introduced new crossover operators to be applied to populations of variable-length chromosomes, as well as the concept of hidden genes [5]. The latter will be the focus of the present paper. In the next section, the problem to solve will be mathematically described. Then, the hidden genes GA will be presented. Lastly, some preliminary numerical results will be presented.

2 Problem Statement

2.1 Formulation of mixed-variable and VSDS optimisation problem

In order to propose a general mathematical formulation of the mixed-variable and VSDS problem, the design variables are introduced and split into 3 categories [6] :

- Continuous variables : $\mathbf{x}$. They refer to real numbers, each defined in a given interval.
- Discrete variables : $\mathbf{z}$. They are non-relaxable variables, defined among a finite set of possibilities.
- Dimensional variables : $\mathbf{w}$. They are also non-relaxable variables defined among a finite set of possibilities. According to their values, the number and type of continuous and discrete variables that are optimized in the problem may vary.

Thus, $\mathbf{x}$ is a $n_x(\mathbf{w})$-dimensional vector, $\mathbf{z}$ is a $n_z(\mathbf{w})$-dimensional vector and $\mathbf{w}$ is a n_w-dimensional vector. Consequently, the generic mono-objective VSDS problem can be mathematically formulated as follows:

$$\begin{cases} \text{Minimize} f(\mathbf{x},\mathbf{z},\mathbf{w}) \\ \text{where : } \mathbf{x} \in F_x(\mathbf{w}) \subseteq \mathbb{R}^{n_x(\mathbf{w})}, \mathbf{z} \in F_z(\mathbf{w}) \subseteq \mathbb{Z}^{n_z(\mathbf{w})}, \mathbf{w} \in F_w \\ \text{subject to :} \\ \mathbf{h}(\mathbf{x},\mathbf{z},\mathbf{w}) = 0 \\ \mathbf{g}(\mathbf{x},\mathbf{z},\mathbf{w}) \leq 0 \end{cases}$$

Where f is the objective function, $\mathbf{h}$ are the equality constraints defined by a $n_h(\mathbf{w})$-dimensional vector and $\mathbf{g}$ are the inequality constraints defined by a $n_g(\mathbf{w})$-dimensional vector.

2.2 Example of application: layout optimization of a satellite module

For illustration purposes, the layout optimization of satellite module is derived. This problem is a representative example of layout optimization problems and is involved in papers [1–3] and is extended in this work to consider variable-sized problems. In order to solve the layout optimisation of a satellite module we will focus on the simplified model of the international commercial communication satellite module (INTELSAT-III).

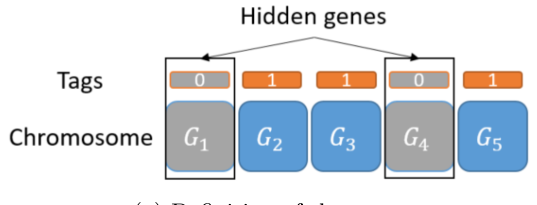
The aim is to find the optimized layout that minimizes the inertia of the whole system composed with N cuboid and cylindrical components on two bearing plates (or four surfaces). Depending on the studies [1, 2], N varies from a few dozens to a hundred. The continuous design variables $\mathbf{x}$ are the position and the orientation of each component on a given surface. The discrete design variables $\mathbf{z}$ are the identifier of the surface on which to place the components. The dimensional variables $\mathbf{w}$ are actually a set of known (or user-defined) subdivisions for each of the components. Some components can be decomposed into several sub-components and the choice of a subdivision varies the number of components along with the number of design variables and constraints. The constraints can either be geometrical or functional. Geometrical constraints encompass: no overlapping between components in pairs and a component and the container, the center of gravity must be placed accurately, some components can be fixed. Functional constraints take into account the existence of a radiation threshold not to be exceeded in the whole module due to radiative emission of some components (e.g. electromagnetism, temperature).

3 Hidden genes genetic algorithm

The proposed algorithm adapted from [5] in order to tackle the above-mentioned problem, is a GA enhanced by the hidden genes method. It is based on a classical GA in which the formulation of the chromosomes have been modified in order to address VSDS problems.

Hidden genes methodology was introduced in [5]. All the chromosomes of a population include the same number of genes (which represent the design variables) and have consequently the same length. However, not all the present genes are expressed during the functions evaluations. Indeed, to each gene of each chromosome of the population is attached a tag taking the value 0 or 1. If the value of the tag is 0, then the corresponding gene will not participate in the fitness evaluation and reciprocally. This mechanism is shown in Figure 1a. The tags are a direct translation of the dimensional variables: their role is to choose which variable will be expressed or not during the fitness evaluation as well as the corresponding constraints to activate.

As the chromosomes are enhanced by tags, the operators of the GA must be compatible with the tags. All the genes, even the hidden ones take part in the operators. The chosen selection and crossover operators are respectively the tournament strategy and a 2-dimensional crossover operator. This last one is introduced in [5] and is illustrated in Figure 1b. The chromosomes and tags undergo an independent crossover operator: for the chromosome it is the Simulated Binary Crossover (SBX) operator and for the tags a n-points crossover operator. For the mutation operator, again the mechanism differs according to the chromosomes or tags. For the chromosomes, a Polynomial Mutation (PM) is applied and for the tags a flip operator with a different probability than the chromosome mutation is applied. Finally, the replacement of the population consists in taking the best individuals among the parents and the offspring populations, and the constraints are treated by constraint-dominance.



(a) Definition of the tags.

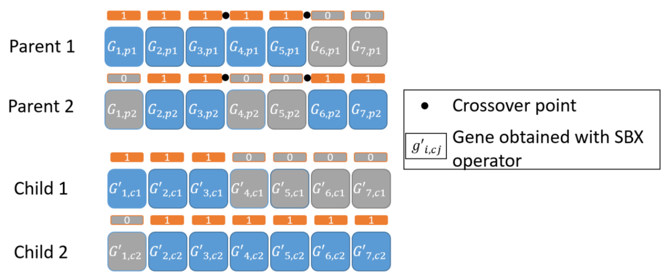
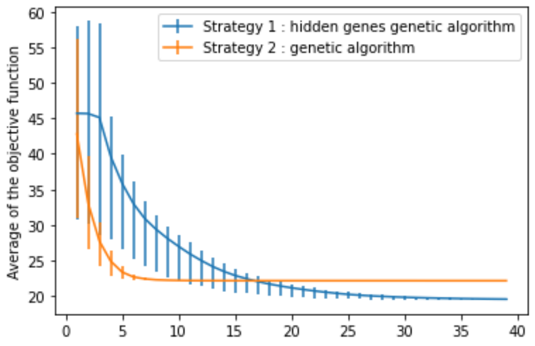

Parent 1
Parent 2
Child 1
Child 2
Crossover point
$g'_{i,cj}$ Gene obtained with SBX operator
$G_{1,p1}$ $G_{2,p1}$ $G_{3,p1}$ $G_{4,p1}$ $G_{5,p1}$ $G_{6,p1}$ $G_{7,p1}$
$G_{1,p2}$ $G_{2,p2}$ $G_{3,p2}$ $G_{4,p2}$ $G_{5,p2}$ $G_{6,p2}$ $G_{7,p2}$
$G'_{1,c1}$ $G'_{2,c1}$ $G'_{3,c1}$ $G'_{4,c1}$ $G'_{5,c1}$ $G'_{6,c1}$ $G'_{7,c1}$
$G'_{1,c2}$ $G'_{2,c2}$ $G'_{3,c2}$ $G'_{4,c2}$ $G'_{5,c2}$ $G'_{6,c2}$ $G'_{7,c2}$
(b) Crossover operator with tags.

Fig. 1: Illustration of the tag mechanism.

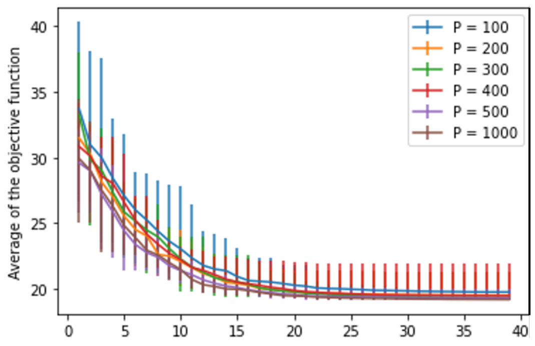
4 Experimental Results

Before applying the algorithm to the layout optimization, it has been tested on different benchmarks introduced in [6]. The algorithm was implemented in the Python programming language and ran on a PC with 2.7 GHz CPU and 8.0 GB RAM.

The algorithm is applied to the Goldstein problem reported in [6]. It is an analytical problem with 5 continuous variables, 4 discrete variables and 2 dimensional variables. Different strategies to implement the chromosomes have been tested : two strategies with the hidden genes method described above compared to one strategy with a classical GA. A convergence curve of the best individual of the populations for the best strategy with the hidden genes and the GA over 10 runs is shown Figure 2a. The error bars represent the minimum and the maximum of the values of the function for the best individual among the 10 runs.



(a) Convergence curve of the best individual for two strategies and 400 individuals over 10 runs.



(b) Convergence curves of the best individual for the strategy 1 and different population sizes P.

Fig. 2: Convergence curves obtained.

The results show a better convergence to the optimum for the hidden genes algorithm as well as a better robustness to initial population. A parametric analysis on the hyperparameters of the GA has been also conducted and the figure shows the convergence curve of the best individual of the strategy with hidden genes for different population sizes.

5 Conclusion and perspective : the layout optimization of the satellite module

A hidden genes GA has been presented in order to tackle VSDS layout optimization problems. It has been successfully applied to different test cases. The algorithm is currently being applied to the layout optimization of a satellite module and results will be available soon. The Figure 3 shows different layouts for the same module and generic components but with different values of one dimensional variable w and thus different chosen subdivisions.

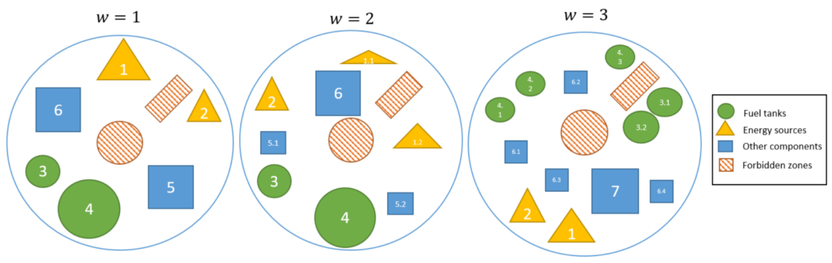


Fig. 3: Different layouts on a bearing plate of a satellite module according to different dimensional variables.

Research about the operators are also conducted in order to improve the accuracy and the success rate of the algorithm. Depending on the results, some research about possible hybrid derivations of the method with multi-agents theory are being considered.

References

1. Y-S. Wang et al. *Cooperative co-evolutionary scatter search for satellite module layout design.* Eng Comput 26(7):761–785, 2009.
2. Z.-G. Sun et al. *Optimal layout design of a satellite module.* Eng Opt, 35 (2003), pp. 513-529.
3. Li et al. *A hybrid multi-mechanism optimization approach for the payload packing design of a satellite module.* Appl. Soft Comput., 45 (2016), pp. 11-26, 2016.
4. S.P. Singh et al. *A review of different approaches to the facility layout problems.* Int J Adv Manuf Technol 30(5):425–433, 2006.
5. O. Abdelkhalik et al. *Hidden genes genetic algorithms for systems architecture optimization.* Proceedings of the Genetic and Evolutionary Computation Conference 2016, 2016, 629–636.
6. J.Pelamatti. *Mixed-Variable Bayesian Optimization, application to aerospace system design.* PhD thesis, Université de Lille, 2020.

Recovery-to-Efficiency: A New Robustness Concept for Multi-objective Optimization under Uncertainty

El-Ghazali Talbi[a,b], Raca Todosijević[c,d]

[a]*INRIA Lille-Nord Europe, 40 Avenue Halley 59650 Villeneuve d'Ascq, France*
[b]*Univ. Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France*
[c]*Université Polytechnique Hauts-de-France, LAMIH, CNRS, UMR 8201, F-59313 Valenciennes, France*
[d]*INSA Hauts-de-France, F-59313 Valenciennes, France*

Abstract

This paper presents a new robustness concept for uncertain multi-objective optimization problems named *recovery-to-efficiency*. An extensive experimental analysis is performed to disclose differences among robust sets obtained using different concepts. For testing purposes, instances from the bi-objective knapsack problem are considered.

1. Introduction

A multi-objective optimization problem in its general form can be formalized as follows:

$$\begin{array}{c} \min\ f(x) \\ \text{subject to } x \in X \end{array} \tag{P}$$

where X is the *feasible set in decision space*, and $f = (f_1, f_2, \ldots, f_n) : X \to \mathbb{R}^n$ is a function vector to be "minimized" subject to some constraints. A solution $x \in X$ is dominated by a solution $x' \in X$, denoted as $x \prec x'$, if $f_i(x') \leqslant f_i(x)$ for all $i \in \{1, \ldots, n\}$, with at least one strict inequality. A solution $x \in X$ is *efficient* if there does not exist any other solution $x' \in X$ such that $x \prec x'$. The set of all efficient solutions is the *efficient set*. One of the most challenging issues in multi-objective optimization is to identify the efficient set, or a good approximation of it for large-size instances.

The above problem is usually referred to as a *deterministic* multi-objective optimization problem, because all parameters are assumed to be known with certainty. In order to take into account parameters' uncertainties, it is desirable to consider the following parameterized family of problems:

$$\begin{array}{c} \min\ f(x, \xi) \\ \text{subject to } x \in X \end{array} \tag{$P(\xi)$}$$

where $f(\cdot, \xi) = (f_1(\cdot, \xi), f_2(\cdot, \xi), \ldots, f_n(\cdot, \xi)) : X \to \mathbb{R}^n$. The (unknown) parameter ξ actually represents a scenario that may occur. It is assumed that ξ varies within a given uncertainty set $\mathcal{U} \subseteq \mathbb{R}^n$. Hence, the uncertain optimization problem corresponding to P is denoted as $(P(\xi) : \xi \in \mathcal{U})$.

The theory of robustness dealing with multi-objective optimization is rather poor, although uncertainty appears in many multi-objective problems. The research attempts in this line mainly aim at bringing the concepts originally proposed for single-objective robust optimization into the multi-objective case (see e.g., (Bokrantz and Fredriksson, 2017; Botte and Schöbel, 2019; Ehrgott et al., 2014; Ide and Schöbel, 2016; Raith et al., 2018).

In this paper, we attempt to further reduce the gap between the fields of robust and multi-objective optimization by extending a robustness concept from single-objective optimization to multi-objective optimization. More precisely, we propose and investigate the so-called *recovery-to-efficiency* robustness concept, which is an extension and generalization of recovery-to-optimality robustness from the single-objective setting Goerigk and Schöbel (2014).

Email addresses: el-ghazali.talbi@univ-lille1.fr (El-Ghazali Talbi), racatodosijevic@gmail.com (Raca Todosijević)

2. Recovery-to-Efficiency

In this section, we introduce a new robustness approach for uncertain multi-objective optimization problems, strictly related to the less conservative approach from single-objective robust optimization, namely recovery-to-optimality (Goerigk and Schöbel, 2014). In order to define this new concept we need a *distance function* $d : X \times X \to \mathbb{R}^n$, which represents the *recovery cost* to transform one solution into another one. It is not required that such function satisfies any property of a norm or a metric, but rather reflects accurately the required effort to modify one solution into another one. As a consequence, such function is typically specific to the problem at hand. For instance, for knapsack problems, it may be the Hamming distance between two solutions.

For each scenario ξ, solving problem $P(\xi)$ means identifying the set of efficient solutions. Let $x(\xi)$ denote the efficient set of problem $P(\xi)$. In order to find $x(\xi)$, many techniques have been proposed in the literature. Among them, the widely-used scalarizing technique consists in choosing vectors $\lambda = (\lambda_1, \lambda_2, \dots \lambda_n) \in \mathbb{R}^n, \sum_{i=1}^n \lambda_i = 1, \lambda_i > 0$, functional F and solving a series of sub-problems

$$(P(\xi,\lambda)) \ \min F(f(x,\xi),\lambda),\ x \in X.$$

Assuming that, for each given scenario ξ, we can compute the set of efficient solutions $x(\xi)$, we seek for a set x from which each set $x(\xi)$, $\xi \in \mathcal{U}$, may be easily accessed, analogously to the concept presented by Goerigk and Schöbel (2014). The notion of "easily accessible" here refers to finding the solution set x from which the decision maker can choose one solution, and after a given scenario ξ becomes known with certainty, he/she can easily transform the chosen solution into an efficient solution from set $x(\xi)$. Then denote the solution set x as the set of robust recoverable solutions. As already pointed out, a similar concept already exists in single-objective optimization (Goerigk and Schöbel, 2014), where the authors seek for a solution x that can easily be transformed into an optimal solution for each considered scenario. The multi-objective concept that we present here actually generalizes the one from Goerigk and Schöbel (2014) for any number of objectives.

Let us now assume that each set $x(\xi)$ is generated by means of a scalarizing approach, such that $x(\xi) = \{x(\xi,\lambda) : \lambda \in \Lambda\}$, where $x(\xi,\lambda)$ refers to the solution of problem $P(\xi,\lambda)$ and $\Lambda = \{\lambda \in \mathbb{R}^n \mid \sum_{i=1}^n \lambda_i = 1, \lambda_i > 0\}$. Then, one approach to generate the robust set x of solutions is to generate point $x(\lambda)$ for each $\lambda \in \Lambda$, which is in some sense close to the points $x(\xi,\lambda)$, $\xi \in \mathcal{U}$. If the closeness is measured as the largest distance, the problem of finding $x(\lambda)$ turns to solving the following optimization problem:

$$\begin{aligned} \min \ & \sup_{\xi \in \mathcal{U}} d(x(\lambda), x(\xi,\lambda)) \\ \text{subject to } & x(\xi,\lambda) \in x(\xi),\ \xi \in \mathcal{U} \\ & x(\lambda) \in X \end{aligned} \qquad \text{(Rec-Eff center}(\lambda))$$

If the objective in (Rec-Eff center(λ)), which aims at minimizing the maximum distance, is replaced by an objective aiming at minimizing the sum of distances (i.e., $\min \ \sum_{\xi \in \mathcal{U}} d(x(\lambda), x(\xi,\lambda))$, we refer to the resulting problem as (Rec-Eff median(λ)). It is important to notice that, once the decision maker has chosen a solution $x(\lambda) \in x$ from the provided robust set, he/she will automatically know how to react if a certain scenario $\xi \in \mathcal{U}$ is realized. The proper reaction would be to change the solution $x(\lambda)$ by the solution $x(\lambda,\xi)$.

However, in the case that the problem $P(\xi,\lambda)$ has more than one optimal solution, the better option would be to determine simultaneously $x(\lambda)$ and $x(\xi,\lambda)$. Let us denote the value of a solution $x(\xi,\lambda)$ with respect to the problem $P(\xi,\lambda)$ as $g(x(\xi,\lambda))$, and the optimal value of $P(\xi,\lambda)$ as $g_{opt}(\xi,\lambda)$. We may define a new set of problems as follows:

$$\begin{aligned} \min \ & \sup_{\xi \in \mathcal{U}} d(x(\lambda), x(\xi,\lambda)) \\ \text{subject to } & g(x(\xi,\lambda)) \leq g_{opt}(\xi,\lambda),\ \xi \in \mathcal{U} \\ & x(\lambda), x(\xi,\lambda) \in X,\ \xi \in \mathcal{U} \end{aligned} \qquad \text{(Rec-Eff center\_opt}(\lambda))$$

which sought to determine, simultaneously, the robust solution $x(\lambda)$ and the efficient solution $x(\xi,\lambda)$ for each scenario. In the case that each problem $P(\xi,\lambda)$ has a unique solution, problems (Rec-Eff center(λ)) and (Rec-Eff center_opt(λ)) are equivalent. Otherwise, problem (Rec-Eff center_opt(λ)) aims at producing the set of $x(\lambda)$ with the cheapest recovery-to-efficiency costs.

3. Experimental Analysis

In this section, we are interested in comparing the recovery-to-efficiency robust sets obtained for problems (Rec-Eff center), (Rec-Eff median, (Rec-Eff center_opt), and (Rec-Eff median_opt). For testing purposes, we consider the following bi-objective knapsack problem:

$$\max(\sum_{j=1}^{n} c_j^1 x_j, \sum_{j=1}^{n} c_j^2 x_j)\ x_j \in x \in X \tag{1}$$

where

$$X = \{x | \sum_{j=1}^{\ell} w_j x_j \leq W, x \in \{0,1\}^{\ell}\} \tag{2}$$

We consider 60 instances of the bi-objective knapsack problem with different sizes. The number of items in the instances varies from 50 to 500. For each setting, 10 different instances are independently generated. The costs of each item c_j^1 and c_j^2 as well as the weights w_j are generated as random integer numbers in $[\![10, 100]\!]$. We assume that the objective coefficients c_j^1 and c_j^2 are subject to uncertainty, and thus 10 different realizations of these values are randomly generated (each realization corresponds to one scenario). The capacity of the knapsack is set as $W = \lceil \sum_{j=1}^{\ell} w_j/2 \rceil$ in all scenarios. In all settings, the set Λ is defined as $\Lambda = \{(0.00 + 0.01 \cdot k, 1 - 0.01 \cdot k) \mid k \in \{0, 1, \ldots, 100\}\}$. As a solver we use the CPLEX 12.9. MIP solver which is able to optimally solve single objective knapsack instances in short time (less than 60 seconds). The distance between solution pairs is taken as the Hamming distance between binary strings. To generate efficient solutions, we use weighted sum and Chebyshev scalarizing techniques. In the later case, we use the reference points h_1 and h_2, defined as $h_i = \max_{\xi \in \mathcal{U}}\{\max c^i(\xi)_j x_j : \sum_{j=1}^{\ell} w_j x_j \leq W, x \in \{0,1\}^{\ell}\}$, $i = 1, 2$, where $\mathcal{U}$ represents set of 10 different scenarios, and $c^i(\xi)_j$, $i = \{1, 2\}$, $j = \{1, 2, \ldots, \ell\}$ realization of objective function values in the scenario $\xi \in \mathcal{U}$.

4. Concluding Remarks

In this paper, we presented a new robustness concept for uncertain multi-objective optimization problems. The so-obtained recovery-to-efficiency approach is inspired by existing robustness concepts from uncertain single-objective optimization called recovery-to-optimality (Goerigk and Schöbel, 2014). An extensive experimental analysis allowed us to disclose differences among robust sets obtained using different concepts. First, as expected, the robust sets do not follow the shape of the Pareto font. Second, the weighted sum scalarizing technique has less impact on the difference of objective function values of concepts (Rec-Eff center_opt) and (Rec-Eff center), than Chebyshev. The same holds considering concepts (Rec-Eff median_opt) and (Rec-Eff median).

References

Bokrantz, R., Fredriksson, A., 2017. Necessary and sufficient conditions for pareto efficiency in robust multiobjective optimization. European Journal of Operational Research 262 (2), 682 – 692.

Botte, M., Schöbel, A., 2019. Dominance for multi-objective robust optimization concepts. European Journal of Operational Research 273 (2), 430–440.

Ehrgott, M., Ide, J., Schöbel, A., 2014. Minmax robustness for multi-objective optimization problems. European Journal of Operational Research 239 (1), 17–31.

Goerigk, M., Schöbel, A., 2014. Recovery-to-optimality: A new two-stage approach to robustness with an application to aperiodic timetabling. Computers & Operations Research 52, 1–15.

Ide, J., Schöbel, A., 2016. Robustness for uncertain multi-objective optimization: a survey and analysis of different concepts. OR Spectrum 38 (1), 235–271.

Raith, A., Schmidt, M., Schbel, A., Thom, L., 2018. Multi-objective minmax robust combinatorial optimization with cardinality-constrained uncertainty. EuropeanJournal of Operational Research 267 (2), 628 – 642.

A new strategy for collective energy self-consumption in the eco-industrial park : Mathematical modeling and economic evaluation

Hamza GRIBISS, MohammadMohsen AGHELINEJAD and Farouk YALAOUI

UTT, LIST3N, France
`(hamza.gribiss, mohsen.aghelinejad, farouk.yalaoui)@utt.fr`

Abstract. Renewable energies are increasingly used around the world to replace fossil energy resources such as gas, coal, and oil sources in order to reduce greenhouse gases. Eco-industrial parks promote the use and sharing of renewable energy sources between factories in a collective self-consumption framework. This article presents a new strategy of photovoltaic self-consumption in an eco-industrial park, that combines collective and individual self-consumption. This strategy has been compared with the classical configuration of self-consumption, in which factories do not share a common photovoltaic installation. Two mathematical models have been proposed and solved for these two configurations, the results show that the new strategy is more efficient than the classical configuration of individual self-consumption.

Mots-Clefs. Eco-industrial park, Renewable energy, Collective self-consumption, Mathematical modeling.

1 Introduction

Energy production is mainly based on fossil energy resources such as gas, coal, and oil sources. According to the UN (United Nations Organization), the use of these resources results in global warming of 1.5 °C due to the emission of greenhouse gases [1]. Among the targets set out in the European Union's (EU) climate and energy framework for 2030, is to reduce greenhouse gases emissions and to increase the share of renewable energy [2].

Energy self-consumption is an important option, which drives to increase renewable energy sources as a result of high energy prices and the emission of greenhouse gases. There are two types of self-consumption :

– The individual self-consumption, that is part of the total energy production consumed by the system. It refers to the process by which a producer consumes its energy production [3].
– The collective self-consumption, that is the case where several consumers share the same energy production. For instance, an industrial park contains several factories that share a photovoltaic production [4]. There is another form of collective self-consumption, particularly in eco-industrial parks. Factories exchange the surplus of energy between themselves, in terms of energy symbioses.

Eco-industrial parks (EIPs) are characterized as a set of factories located in the same geographic area, with the goal of fostering cooperation and resource sharing [5]. EIPs aim to efficiently exchange natural resources, reduce overall environmental impact, and increase economic benefits to participants [6].

This article presents a study, which combines individual and collective self-consumption in an eco-industrial park. In order to minimize energy costs and greenhouse gas emissions. The rest of this article is organized as follows. Section 2 presents the industrial symbioses involving renewable energy. Section 3 presents the problem description and section 4 introduces the mathematical model. Section 5 provides the data generation for testing the model. Section 6 includes a discussion of the results. Section 7 draws conclusions with some directions for future research.

2 Related work

The main objective of eco-industrial parks is to facilitate industrial symbiosis between a set of production units that can generate exchanges of waste, materials and energy. Industrial symbiosis has been defined by Chertow et al [7] as a collective commitment including physical exchanges of materials, energy and products between factories that have geographical proximity. In this context, Butturi et al [8] proposed a multi-objective optimization to evaluate energy symbiosis, that includes the integration of renewable energy sources within an eco-industrial park and considering both economic and environmental issues. They discuss three scenarios that provide individual company and park managers with relevant information, which support the decision-making regarding the economic sustainability and environmental impacts of energy symbiosis. Jiang et al [9] presented a genetic algorithm to solve a multi-objective optimization, that propose an exchange the electricity between four parks in absence of grid power. Their aim was to minimize a power interruption, storage system cost, and customer dissatisfaction. Heendeniya [10] proposed an agent-based model to exchange energy between prosumers, which have their own PV power generation and a battery storage. Each agent tries individually and collectively to maximize self-consumption of renewable energy.

The economic feasibility of self-consumption in eco-industrial parks and remote areas has been studied in several papers. Among these works, Contreras et al [11] present a cooperative planning framework that integrates long-term planning and short-term operation of an energy collective composed of consumers sharing a photovoltaic and storage system. Their objective was to determine the optimal size of the PV plus storage system, that reduces total costs. Long-term planning gives each consumer an idea of how much they can save, which allows them to decide to join the collective or not. Pedrero et al [12] presented an economic evaluation of shared self-consumption of PV installations between three halls with the addition of the option to sell surplus energy. They prove that the economic feasibility depends largely on the compensation for the electricity fed into the grid.

A study on the integration of renewable energy in eco-industrial parks in the literature has been treated by Butturi et al [13]. The result of this study shows that a few articles have considered the integration of renewable energy. Among the works that have been published after this research [13], we find Jiang et al [9], which discuss in their paper the exchange of renewable energy between four parks with the integration of a storage system. Butturi et al [8] treat the case of renewable energy exchange between factories. In another study, Pedredro et al [12] deal with the case of collective self-consumption with the option of selling the surplus energy between 3 factories that share a photovoltaic installation.

The result of this state of the art shows that there is a lack of articles that deal with the combination of individual and collective self-consumption within an eco-industrial park, as well as few papers have considered in the same study the storage option and the option of selling the surplus energy.

In this paper, a new strategy of energy self-consumption in eco-industrial park is introduced. It allows the merge of both individual and collective self-consumption with the integration of storage systems and the addition of the option of selling surplus energy.

3 Problem definition

In this section, the eco-industrial park strategy and the classic individual self-consumption configuration are presented. The strategy of the eco-industrial park is as follows:

– Each factory has its own photovoltaic installation that provides energy to satisfy the demands. Excess energy is either stored in the factory's battery or sold to the grid.
– In case the factory's self-production is not sufficient to guarantee its energy requirements, the factory relies on the park's photovoltaic installation or the park's battery.
– The park's photovoltaic installation and the park's battery provide a percentage of their energy to each factory. This percentage depends on the investment cost of each factory for the creation of the park. The surplus energy from the park's photovoltaic installation is either stored in the park's battery or sold to the grid.

– The appeal to the main grid is made in case of an emergency where the factory's own production, its battery, the park's production and the park's battery are not enough to guarantee the energy demands needs in the factories.

The Figure 1 represents the schema of this strategy for a case of three factories.

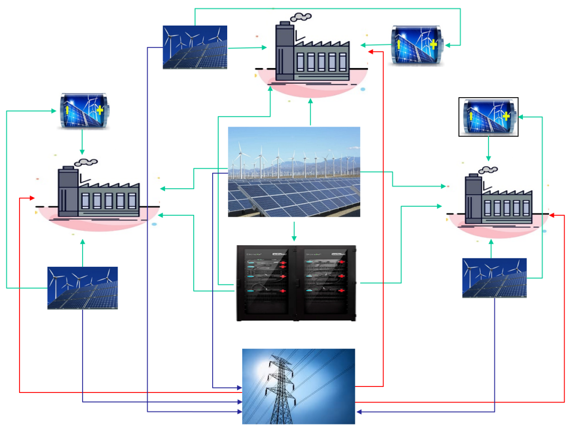


Fig. 1. Schema of strategy for a case of three factories.

The objective is to compare this strategy with the classical configuration (individual self-consumption) which is already studied in several articles in the literature. This configuration is very present in residential buildings that use photovoltaic panels with the integration of batteries [3]. As an example to this study, Braun et al [14] used a lithium-ion battery to increase self-consumed photovoltaic energy with the addition of the option of selling the surplus energy.

The structure of this configuration in our study is as follows :

– Each factory has its own photovoltaic installation that provides energy to satisfy the demands. Excess energy is either stored in the factory's battery or sold to the grid.
– In case the factory's self-production is not sufficient to guarantee its energy requirements, the factory draws energy from the grid.

The Figure 2 represents the schema of classical individual self-consumption for a case of three factories.

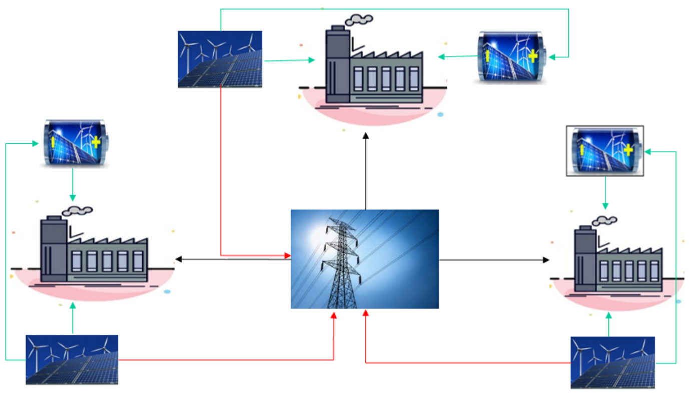


Fig. 2. Schema of individual self-consumption for a case of three factories.

4 The Mathematical Models

In this section, two mathematical models are addressed to present the eco-industrial park's strategy and the classical configuration for self-consumption which are presented in the previous section.

4.1 Parameters and decision variables

Sets :

$j = \{1, \dots, J\}$: set of factories in the eco-industrial park
$t = \{1, \dots, H\}$: set of the time period (in hours)
$i = \{1, \dots, I\}$: set of PV installation in the park

Parameters :

$D_{j,t}[KWh]$: Energy demand of factory j at period t
$Q^p_{i,t}[KWh]$: Amount of renewable energy available in the park at period t
$Q^u_{j,t}[KWh]$: Amount of renewable energy available in the factory j at period t
$P^a_t[€/KWh]$: Price of energy from the grid at period t
$P^v_t[€/KWh]$: Price of energy sold to the grid at period t
$P^u_t[€/KWh]$: Price of energy drawn from the factory's production at period t
$P^p_t[€/KWh]$: Price of energy drawn from the park's production at period t
$P^{dbu}_t[€/KWh]$: Price of energy drawn from the factory's battery at period t
$P^{dbp}_t[€/KWh]$: Price of energy drawn from the park's battery at period t
$SOC^{max}_j[kWh]$: Maximum state of charge of the factory's battery j
$SOC^{min}_j[kWh]$: Minimum state of charge of the factory's battery j
$SOCP^{max}[kWh]$: Maximum state of charge of the park's battery
$SOCP^{min}[kWh]$: Minimum state of charge of the park's battery
η^{char} : Losses due to the battery's charging
η^{dech}: Losses due to the battery's discharge
$CIU_j[€]$: Investment costs related to the factory j for its own photovoltaic energy and battery over the H horizon
$CIP_j[€]$: Investment costs related to the factory j for the photovoltaic production and battery of the park over the H horizon
$CIP[€]$: Total investment costs of the park over the horizon H
$Pr^u_j[\%]$: The contribution rate of the factory j to the construction of the park

Decision variables :

$E^u_{j,t}[KWh]$: Amount of energy used by the factory j that is drawn from its production at period t
$E^p_{i,j,t}[KWh]$: Amount of energy produced by the park that is drawn by factory j at period t
$B^{cu}_{j,t}[KWh]$: Amount of energy produced by the factory j stored in its battery at period t
$B^{du}_{j,t}[KWh]$: Amount of energy used by the factory j that is drawn from its battery at period t
$B^{cp}_t[KWh]$: Amount of energy produced by the park stored in its battery at period t
$B^{dp}_{j,t}[KWh]$: Amount of energy from the park's battery that is drawn by the factory j at period t
$R^a_{j,t}[KWh]$: Amount of energy drawn from the grid by the factory j at period t
$R^{vu}_{j,t}[KWh]$: Amount of energy produced by the factory j that is sold to the grid at period t
$R^{vp}_{t,i}[KWh]$: Amount of energy produced by a photovoltaic installation e and sold to the grid at period t
$R^{vp}_t[KWh]$: Amount of energy produced by the park that is sold to the grid in period t
SOC_{jt}: State of charge of the factory's battery j at period t
$SOCP_t$: State of charge of the park's battery j at period t
c^p_t: = 1 if the park's battery is charging at period t. 0 otherwise
d^p_t : = 1 if the park's battery is discharging at period t. 0 otherwise
$c^u_{j,t}$: = 1 if the factory's battery j is charging at period t. 0 otherwise
$d^u_{j,t}$: = 1 if the factory's battery j is discharging at period t. 0 otherwise

In the following, the model of the classical self-consumption configuration is presented as model 1 and the model of the park strategy is presented as model 2.

4.2 Objective function of model 1

The objective function aims to minimize the energy cost of the factories. It is calculated by subtracting the following two blocks:

– The first is the summation of the variable costs of energy which are: the costs of purchasing energy from the grid, the costs of using a photovoltaic installation and batteries in the factories.
– The second is the sale of the surplus energy to the grid by the factories.

$$\min Z_1 = \sum_{j=1}^{J} \sum_{t=1}^{H} (P_t^a \times R_{j,t}^a + P_t^u \times E_{j,t}^u + P_t^{dbu} \times B_{j,t}^{du} - P_t^v \times R_{j,t}^{vu}) \tag{1}$$

4.3 Constraints of model 1

Constraint (2) ensures that the total demand of each factory j is satisfied by the energy source available in factory j at period t.

$$E_{j,t}^u + B_{j,t}^{du} + R_{j,t}^a = D_{j,t} \quad \forall j, t \tag{2}$$

Constraint (3) ensures that the sum of the quantities of energy drawn by each factory j and stored in battery j and sold to the grid must be equal to the quantity of energy available in the factory's photovoltaic installation j at period t.

$$E_{j,t}^u + B_{j,t}^{cu} + R_{j,t}^{vu} = Q_{j,t}^u \quad \forall j, t \tag{3}$$

The constraints (4) and (5) and (6) represent the state of charge initial and final and at period t respectively of the battery of the factory j.

$$SOC_{j,0} = SOC_j^{min} \tag{4}$$

$$SOC_{j,T} = SOC_j^{min} \tag{5}$$

$$SOC_{j,t} = SOC_{j,(t-1)} + \eta^{char} \times B_{j,t}^{cu} - 1/\eta^{dech} \times B_{j,t}^{du} \quad \forall j, t \tag{6}$$

Constraint (7) ensures that the factory's battery j is protected against accelerated aging.

$$SOC_j^{max} \leq SOC_{j,t} \leq SOC_j^{min} \quad \forall j, t \tag{7}$$

Additionally, the amount of charging and discharging of the batteries of the factories must also meet the upper and lower bound constraints. Constraints (8) and (9) refer to the maximum to be charged in the battery of the factory j at period t and constraints (10) and (11) represent the maximum to be discharged in the battery of the factory j at period t.

$$B_{j,t}^{cu} \leq SOC_j^{max} \times c_{j,t}^u \quad \forall j, t \tag{8}$$

$$B_{j,t}^{cu} \geq c_{j,t}^u \quad \forall j, t \tag{9}$$

$$B_{j,t}^{du} \leq SOC_j^{max} \times d_{j,t}^u \quad \forall j, t \tag{10}$$

$$B_{j,t}^{du} \geq SOC_j^{max} \times d_{j,t}^u \quad \forall j, t \tag{11}$$

Constraint (12) ensures the choice between charging or discharging of the battery of the factory j at period t.

$$c_{j,t}^u + d_{j,t}^u \leq 1 \quad \forall j, t \tag{12}$$

4.4 Objective function of model 2

The aim of the objective function in model 2 is the same as in model 1, it minimizes the energy cost of the factories in the park. Equation (13) represents this objective function.

$$\begin{aligned} \min Z_2 = \sum_{j=1}^{J} \sum_{t=1}^{H} (&P_t^a \times R_{j,t}^a + +P_t^u \times E_{j,t}^u + P_t^{dbu} \times B_{j,t}^{du} + P_t^p \times \sum_{i=1}^{I} (E_{i,j,t}^p) \\ &+ P_t^{dbp} \times B_{j,t}^{dp} - P_t^v \times (R_{j,t}^{vu} + R_t^{vp})) \end{aligned} \tag{13}$$

4.5 Constraints of model 2

Constraints (3-12) of model 1 are the same used in model 2. In addition, the following constraints have been used in model 2.

Constraint (14) ensures that the total demand of each factory j is satisfied by the energy sources available in the park and factory j at period t.

$$E^{u}_{j,t} + B^{du}_{j,t} + \sum_{i=1}^{I}(E^{p}_{i,j,t}) + B^{dp}_{j,t} + R^{a}_{j,t} = D_{j,t} \quad \forall j,t \tag{14}$$

Constraint (15) ensures that the sum of the quantities of energy demands of all factories satisfied by the park's production and the energy stored in the park's battery and the energy sold to the grid by the park must be equal to the quantity of energy available in the park's production at period t.

$$\sum_{j=1}^{J}(E^{p}_{i,j,t}) + B^{cp}_{t} + R^{vp}_{t,i} = Q^{p}_{i,t} \quad \forall t,i \tag{15}$$

Constraint (16) represents the total energy sold to the grid by the production park.

$$\sum_{i=1}^{I}(R^{vp}_{t,i}) = R^{vp}_{t} \quad \forall t \tag{16}$$

Constraint (17) represents the percentage of energy to be drawn from the park's production and the battery of the park by each factory j over the horizon H.

$$\sum_{t=1}^{H}(B^{dp}_{j,t} + \sum_{i=1}^{I}(E^{p}_{i,j,t})) = Pr^{u}_{j} \times \sum_{t=1}^{H}\sum_{i=1}^{I}(Q^{p}_{i,t}) \quad \forall j \tag{17}$$

The constraints (18) and (19) and (20) represent the state of charge initial and final and at period t respectively of the park's battery.

$$SOCP_0 = SOCP^{min} \tag{18}$$

$$SOCP_T = SOCP^{min} \tag{19}$$

$$\begin{aligned} SOCP_t = SOCP_{(t-1)} &+ \eta^{char} \times B^{cp}_{t} \\ &- 1/\eta^{dech} \times \sum_{j=1}^{J}(B^{dp}_{j,t}) \quad \forall t \end{aligned} \tag{20}$$

Constraint (21) ensures that the park's battery is protected against accelerated aging.

$$SOCP^{max} \leq SOCP_t \leq SOCP^{min} \quad \forall t \tag{21}$$

Additionally, the amount of charging and discharging of the park's battery must also meet the constraints of the upper and lower limits. Constraints (22) and (23) refer to the maximum to be charged in the park's battery at period t and Constraints (24) and (25) represent the maximum to be discharged in the park's battery at period t.

$$B^{cp}_{t} \leq SOCP^{max} \times c^{p}_{t} \quad \forall t \tag{22}$$

$$B^{cp}_{t} \geq c^{p}_{t} \quad \forall t \tag{23}$$

$$\sum_{j=1}^{J}(B^{dp}_{j,t}) \leq SOCP^{max} \times d^{p}_{t} \quad \forall t \tag{24}$$

$$\sum_{j=1}^{J}(B^{dp}_{j,t}) \geq SOCP^{max} \times d^{p}_{t} \quad \forall t \tag{25}$$

Constraint (26) ensures the choice between charging or discharging of the park's battery at period t.

$$c^{p}_{t} + d^{p}_{t} \leq 1 \quad \forall t \tag{26}$$

5 Data generation

This section presents how the model data was generated, such as photovoltaic installation, and battery size, and investment costs.

5.1 Photovoltaic installation in factories

In this study, the maximum size of photovoltaic production will be installed on the roofs of factories is considered. As a hypothesis, the available surface on the roofs of these factories taken into account to install photovoltaic panels is $S^{min} \leq S \leq S^{max}$ where $S^{min} = 800m^2$ and $S^{max} = 1200m^2$ Based on [15] a 1.9 m^2 monocrystalline solar panel can produce 365 W, so the maximum size of the photovoltaic installation that can be placed in a surface S is $\alpha = \frac{S \times 365}{1,9}$

To calculate the amount of energy $Q^u_{j,t}$ produced with a PV installation α, in each period t during the horizon H, the European Commission's Photovoltaic Geographic Information System (PVGIS) [16] is used. The PVGIS-SARAH radiation database is chosen, it can offer PV load profiles with a resolution of one hour between 2005 and 2016, which in turn were used to generate an average annual PV load profile for each factory roof.

To summarize, for each factory j of surface S_j, it is possible to install a PV production size α_j that gives an amount of energy $Q^u_{j,t}$ at each period t.

5.2 Photovoltaic installation in the park

The size of the PV installation in the park is chosen with the following method:

- Calculating the difference between the total demand and the total energy produced by all factories in one year, which is defined by $R = \sum_{t=1}^{8760} \sum_{j=1}^{J} D_{j,t} - \sum_{t=1}^{8760} \sum_{j=1}^{J} Q^u_{j,t}$
- Using PVGIS, the determination of the size of the PV installation that produces a percentage k of R, i.e : $\sum_{t=1}^{8760} \sum_{i=1}^{I} Q^p_{i,t} = k \times R$. This size is used to calculate the amount of energy produced in the park over 4 horizons (1 month, 1 season, 2 seasons and 4 seasons).

In this study, the cases where k is 20% 40%, 60%, and 80% are compared.

5.3 Battery size in the factories and in the park

In a study of PV self-consumption by Luthander et al [3], they report that PV self-consumption can be increased by 13-24% by using a battery capacity of 0.5-1 kWh for each KW of PV power installed. In this case study, a 1 kWh lithium battery is installed for every 1 kW of PV power installed.

5.4 Investment cost of the factories

For each factory, there are two types of investment costs :

- Fixed investment costs for its own photovoltaic energy and battery over the H horizon.
- Fixed investment costs for the photovoltaic production and battery of the park over the H horizon.

To calculate the investment costs in the factories and the park, the data of Pedrero et al is used [12], represented in tables 1 and 2.

Table 1. PV installations reference costs

PV Power	Reference Cost [€/W]
$\leq 10kW$	1.5
$10kW - 100kW$	0.9
$100kW - 1MW$	0.75

To calculate the contribution cost for each factory, table 3 is relied upon.

Table 2. Economic parameters for the calculation of investment costs

Parameter	Value
PV modules service life	25 (year)
Inverter service life	15 (year)
Inverter cost	$0.2€/W$
Maintenance cost	$0.02€/(W \times year)$

Table 3. Percentage of contribution for each factory

	Total demand	Percentage of contribution
Usine j	$\sum_{t=1}^{H} D_{j,t}$	$Pr_j^u = \frac{\sum_{t=1}^{H} D_{j,t}}{\sum_{t=1}^{H}\sum_{j=1}^{J} D_{j,t}}$

5.5 Different energy costs

The variation in electricity prices over the optimisation horizon can affect the total energy cost. In this paper, time-of-use (TOU) pricing is put to use to balance electricity supply and demand. The purchase price of the electricity grid is the most expensive and the price of the energy from the factory is the cheapest. The prices are classified in this order :

$$P_t^u \leq P_t^{dbu} \leq P_t^p \leq P_t^{dbp} \leq P_t^a \quad \forall t$$

6 Numerical study

In this section, illustrative examples are considered to validate and evaluate the presented models, which are solved by CPLEX on an Intel Core i5 with 2.7 GHz and 8 GB RAM.

To compare the results between the proposed new strategy and the classical configuration of individual self-consumption, the ratio between the investment and the price paid by all factories at the end of the horizon H is calculated.

$X = \frac{CI1}{G1}$ and $Y = \frac{CI2}{G2}$ represent this ratio in the case of individual self-consumption and the strategy respectively

with :

- $CI1$: Total investment cost of the factories over the horizon H for the case of individual self-consumption.
- $G1$: Total cost paid by the factories on the horizon H for the case of the individual self-consumption.
- $CI2$: Total investment cost of the factories over the horizon H for the case of the strategy.
- $G2$: Total cost paid by the factories on the horizon H for the case of the strategy.

In the rest of this paper, four different cases of the eco-industrial park are used. The first one contains 3 factories, the second 6 factories, the third 9 factories and the fourth 15 factories. Each case is tested over 4 horizons (1 month, 1 season, 2 seasons and 4 seasons) which are presented in hours. The values represented in the following tables are the average value of the gap between X and Y over the 4 horizons.

The gap can be calculated by $\frac{Y-X}{X} \times 100$. For example, the value 235.4 presented in table 4 represents this gap in the case of 3 factories and $k = 40\%$. It was calculated using the previous gap formula with $X = 0,1184$ and $Y = 0,39712$. The more this gap is greater, the considered strategy is more performed.

Variation of the size of the photovoltaic installation in the park

Table 4 represents the values of the gap for each case of the eco-industrial park by varying the percentage k of the photovoltaic installation of the park As a result, it can be concluded that :

- In each instance, the use of the proposed strategy model is better than the individual self-consumption model.
- Despite the increase in the number of factories, there is a small decrease of the gap, which gives the possibility to add several factories in the same park without having the problem of decreasing the gap.

Table 4. Variation of the size of the photovoltaic installation in the park

	3 factories (%)	6 factories (%)	9 factories (%)	15 factories (%)	average (%)
$k = 20\%$	105,50	105,51	102,60	102,59	104,05
$k = 40\%$	235,40	235,44	228,65	228,62	232,03
$k = 60\%$	398,19	386,38	386,16	386,08	389,20
$k = 80\%$	608,16	603,25	588,69	588,58	597,17
average	336,81	332,64	326,52	326,47	

– By increasing the energy of the park by 20% the gap increases between 69,15% and 123,12%.

Variation in the selling price of surplus energy

Due to low feed-in tariffs [10] especially for large installations, three categories of energy selling price are considered such as : $P_t^v = 1/10 \times P_t^a$, $P_t^v = 1/5 \times P_t^a$ and $P_t^v = P_t^a$ where P_t^v :Price of energy sold to the grid at period t and P_t^a : Price of energy from the grid at period t.

Table 5 represents the gap's values for each case of the eco-industrial park by varying the selling price of the surplus energy.

Table 5. Variation in the selling price of surplus energy

	3 factories (%)	6 factories (%)	9 factories (%)	15 factories (%)	average
$P_t^v = 1/10 \times P_t^a$	231	231,04	224,28	224,25	227,64
$P_t^v = 1/5 \times P_t^a$	235,40	235,44	228,65	228,62	232,02
$P_t^v = P_t^a$	298,99	299,00	291,13	266,79	288,97
average	255,13	255,16	248,13	239,89	

It is concluded that :

– The proposed strategy is more efficient than the individual self-consumption in each case.
– The increase in the selling price of surplus energy increases the gap between the proposed strategy and the classical configuration of individual self-consumption.

The proposed strategy gives better results than the classical configuration of individual self-consumption even in the months when there is little photovoltaic production such as January.

7 Conclusion

This paper develops two mathematical models, the first one represents a new strategy of self-consumption in eco-industrial parks and the second one defines the classical configuration of individual self-consumption. In both models, the option of storage and sale of surplus energy have been addressed. This study represents a step forward in the under-investigated field regarding the integration of renewable energy in eco-industrial park [13]. The two models were tested and compared, the results show that with this new strategy the factories can reduce the price of electricity compared to the classical configuration of individual self-consumption.

According to [17] the necessary condition to justify the creation of an eco-industrial park, is to prove that the benefits achieved by working with a collective strategy of factories are superior to the benefits achieved by working as a single factory. This study shows that investing collectively among the factories is more efficient than investing alone.

With this strategy, several factories can be in the same park because the results show that despite the increase in the number of factories, there is a small decrease in the gap and the results obtained by this new strategy are still more efficient.

References

1. Nations unies, https://www.un.org/fr/, accessed on : 2021-01-12.
2. European unions, https://europa.eu/, accessed on : 2021-02-24.

3. Rasmus Luthander, Joakim Widén, Daniel Nilsson, and Jenny Palm. Photovoltaic self-consumption in buildings: A review. *Applied energy*, 142:80–94, 2015.
4. Daniele Menniti, Nicola Sorrentino, Anna Pinnarelli, Stefano Mendicino, Pasquale Vizza, and Gaetano Polizzi. A blockchain based incentive mechanism for increasing collective self-consumption in a nonsumer community. In *2020 17th International Conference on the European Energy Market (EEM)*, pages 1–6. IEEE, 2020.
5. Mathilde Le Tellier, Lamia Berrah, Benoit Stutz, Jean-François Audy, and Simon Barnabé. Towards sustainable business parks: A literature review and a systemic model. *Journal of cleaner production*, 216:129–138, 2019.
6. Zhe Liu, Michelle Adams, Raymond P Cote, Yong Geng, Jingzheng Ren, Qinghua Chen, Weili Liu, and Xuesong Zhu. Co-benefits accounting for the implementation of eco-industrial development strategies in the scale of industrial park based on emergy analysis. *Renewable and Sustainable Energy Reviews*, 81:1522–1529, 2018.
7. Marian R Chertow. Industrial symbiosis: literature and taxonomy. *Annual review of energy and the environment*, 25(1):313–337, 2000.
8. Maria Angela Butturi, Miguel A Sellitto, Francesco Lolli, Elia Balugani, and Alessandro Neri. A model for renewable energy symbiosis networks in eco-industrial parks. *IFAC-PapersOnLine*, 53(2):13137–13142, 2020.
9. Youhua Jiang, Jinwan Yang, Chunji Wang, and Yilong Cao. Electricity optimal scheduling strategy considering multiple parks shared energy in the absence of grid power supply. *International Transactions on Electrical Energy Systems*, 30(12):e12634, 2020.
10. Charitha Buddhika Heendeniya. Agent-based modeling of a rule-based community energy sharing concept. In *E3S Web of Conferences*, volume 239. EDP Sciences, 2021.
11. Jesus E Contreras-Ocaña, Arshpreet Singh, Yvon Bésanger, and Frédéric Wurtz. Integrated planning of a solar/storage collective. *IEEE Transactions on Smart Grid*, 12(1):215–226, 2020.
12. Juan Pedrero, Patxi Hernández, and Álvaro Martínez. Economic evaluation of pv installations for self-consumption in industrial parks. *Energies*, 14(3):728, 2021.
13. MA Butturi, F Lolli, MA Sellitto, E Balugani, R Gamberini, and B Rimini. Renewable energy in eco-industrial parks and urban-industrial symbiosis: A literature review and a conceptual synthesis. *Applied Energy*, 255:113825, 2019.
14. Martin Braun, Kathrin Büdenbender, Dirk Magnor, and Andreas Jossen. Photovoltaic self-consumption in germany: using lithium-ion storage to increase self-consumed photovoltaic energy. 2009.
15. Pv surface, https://fr.electrical-installation.org/, accessed on : 2021-03-01.
16. Pvgis, https://re.jrc.ec.europa.eu/, accessed on : 2021-03-15.
17. Marianne Boix, Ludovic Montastruc, Catherine Azzaro-Pantel, and Serge Domenech. Optimization methods applied to the design of eco-industrial parks: a literature review. *Journal of Cleaner Production*, 87:303–317, 2015.

Genetic Algorithm for Neural Network Architecture optimization

A. Makki[1], E-G. Talbi[2], P. Bouvry[3], and G. Danoy[4]

[1] Université du Luxembourg, Luxembourg
ayman.makki@uni.lu
[2] Université de Lille, Lille
el-ghazali.talbi@univ-lille.fr
[3] Université du Luxembourg, Luxembourg
pascal.bouvry@uni.lu
[4] Université du Luxembourg, Luxembourg
gregoire.danoy@uni.lu

1 Introduction

The discovery of GPU-efficient convolutional neural networks architectures with AlexNet [12] in 2012 has opened a new cross-disciplinary field for research with a special interest in the design of architectures for efficient training of neural networks. Around the same period, several publications offered a variety of architectures that tried to outperform state-of-the-art setups on classical benchmark datasets (Dan Ciresan Net [4] in 2011, VGG [17] in 2014, GoogleLeNet in 2015 [20]). However, despite yielding impressive results on classical benchmark, those architectures have been requiring intensive handmade fine-tuning which is time consuming and appears as groping inside the search space of architectures. Therefore, there has been a growing interest in the design of automated techniques to fasten and improve the research of architectures for deep neural networks.

In this paper, we would like to introduce the use of GA on a particular factorized hierarchical chained search space originally designed for the MnasNet neural network [22] in 2019. Since the original publication, there has been several improvements proposed both on the search space [23], the scaling to deeper networks [24] or even the inner optimizer routine [6] that led to current state-of-the-art performance (99.70% on CIFAR-10 and 96.08% on CIFAR-100) and competing performances on ImageNet (88.61%). Since we did not yet launch a fully mature experiment, our results are still below state-of-the-art but remain promising considering the small amount of time required to obtain those results. Eventually, we present the various promising tracks we would like to pursue in order to enhance those preliminary results.

2 Related work

A detailed and comprehensive review of Neural Architecture Search techniques has been proposed in [21]. It exhibits the numerous and various approaches to solve this heavily complex optimization problems and highlights the fact that it does not seem to be a particular setup that distinguishes itself by its performances. While focusing mostly on the search space design, it does not precisely distinguish all the techniques that have been experimented towards the research of an optimal architecture.

There has been many solutions proposed : Random Search (RS), Reinforcement Learning (RL) [1], Particle Swarm Optimization (PSO) [10], Bayesian Optimization (BO) [13], Genetic Algorithm (GA) [19] [14] [3]. All proved to be able to find good candidates for training and various scoring based on their accuracies on the test set have been introduced to discriminate them.

We would like to mostly focus and introduce the use of GA in this paper while taking advantage of an already designed search space described in [22]. To do so, we carefully implemented suitable operators that we will describe while precising all required experimental parameters to allow reproducibility of ours results.

3 Problem Formulation

3.1 Design of the search space

Since the rise of deep convolutional neural networks, there has been several attempts to enhance the classical convolution operations in order to improve its interpolation performance, reduce its memory footprint or decrease its computing time. We will first present the main building blocks of ours search space before introducing the global factorized hierarchical structure of the architectures which can be generated in this search space.

Building blocks of the search space. We decided to include two main building blocks in our search space following [22] : Depthwise separable convolution [11] and Inverted Residual with linear bottleneck [16]. Being designed for an efficient design of neural networks architecture on mobiles, they both present the advantage of reducing the latency of the network and the memory footprint. We described precisely all the technical details of the implementation of those blocks in the Figure 1. We also added the possibility for those blocks to be enhanced by an optional Squeeze & Excitation module which is also described in Figure 1. Eventually, all those blocks include a skip connection to ensure a better gradient propagation towards the earliest blocks of the architecture.

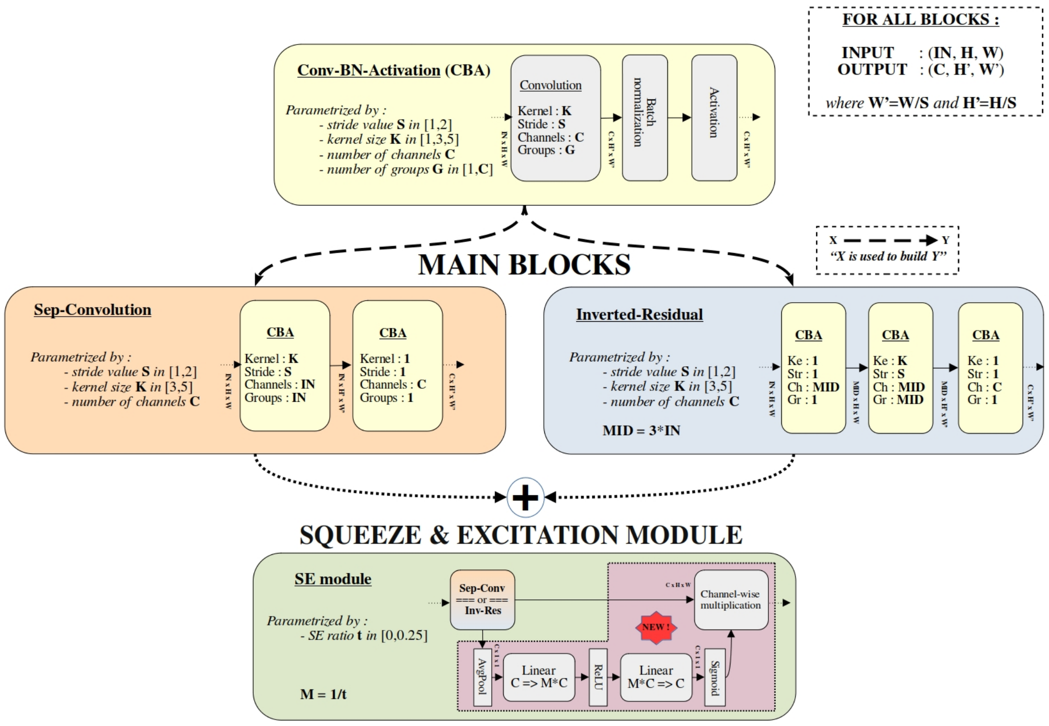


Fig. 1: A complete description of the Depthwise Separated Convolution and the Inverted Residual convolution with linear bottleneck. (Top) A description of a Convolution-Batch Normalisation-Activation module (CBA) to be used to build the main blocks. (Center) A comprehensive description of the main blocks used in the search space. (Bottom) A schematic representation of the SE module and how it is added to one of the main blocks.

A block-based factorized hierarchical search space. As a factorized hierarchical search space, we split an architecture representation into three different stages : the global architecture level, the block level and the layer level. Each of them is being designed by a sequence of the next stage : the global architecture is a sequence of blocks, each of them being composed of several layers. Each block has its own set of parameters which shape the structure of its layers. An architecture is therefore designed by a sequence of sets of 6 parameters which specify all the characteristics of each block. We also added a fixed classical convolution with stride 2 and 32 channels before the first block to ensure the network being able to extract useful features at the earliest stage.

A schematic description of this search space is given in Figure 2. It progressively zooms on each stage of an architecture and presents the data flow shape at each stage. One should note that the first layer of each block is the only one modifying the data shape either by striding the convolution or by a variation in the number of channels. All following layers are simple duplicate of the first one with stride 1 and same number of channels.

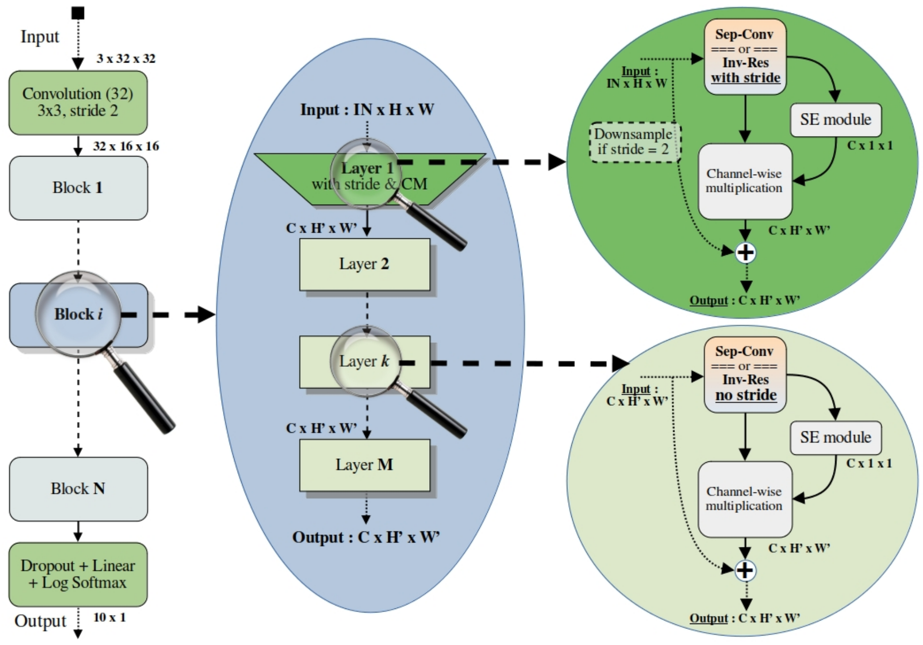


Fig. 2: A schematic representation of the search space. (Left) represents the global architecture level. (Middle) describes the representation of a single block as a sequence of layers where we set apart the first layer enforcing channel augmentation and stride. (Right) describes a single layer with its main computational module, its SE module as well as its shortcut connection (which might be a downsampling module if required).

As specified previously, a block is being configured by 6 different parameters and each block is being parametrized independently. This means that they are no global parameters that influence the whole network in this search space (such as a dropout rate for instance). This seems particularly suited for local search over this search space since this enables easy implementation of a neighborhood for an architecture : each parameter only controls a small subset of the global configuration of the network and influences a single block. We described all the parameters for a single block in Table 1. In order to introduce the mutations of the Genetic Algorithm, we distinguished two different sections in the parameters : the primordial parameters which have a strong influence on the design of the architecture, and which must be mutated to ensure a change in a given architecture, and the secondary ones which have a weaker influence on the architecture design.

Parameter Name	Possible values	Comments
Primordial parameters		
Operation Type	$[0, 1]$	Choice between Sep. Convolution and Inv. Residual
Kernel Size	$[3, 5]$	The size of the convolution kernel
Stride	$[1, 2]$	The stride of the first layer convolution operation
Channel multiplier	$[0.5, 1, 1.5, 2, 3]$	Number of channels multiplier relative to previous block number of layers
Secondary parameters		
Number of layers	$[1, 2, 3]$	Repeats of the layer within the block
SE ratio	$[0, 0.25]$	Usage or not of a SE module

Table 1: A description of the parametrization of a block within the search space. Regarding the channel multiplier : a value of 2 for block k would indicate that its output would have twice more channels than its input from block $k-1$.

3.2 Genetic search algorithm

Genetic Algorithms are a population based meta-heuristic classified as an evolutionary method. It optimizes a given black-box function by simulating several generations of a population of candidates and uses various operators to select, reproduce and mutate individuals. In order to implement a GA in our search space, we must define our evolutionary operators (crossover and mutation) as an adaptation to the numerical representation we defined.

Crossover operator. To select the best individuals at each generation, we simply randomly picked 5% of the population and select the best individual as much time as the actual population size. Then, we defined the crossover as a simple One Point Crossover at the block level : given two selected parents, we switch a single block whose index has been sampled from a uniform distribution over all blocks. This is described in Figure 3.

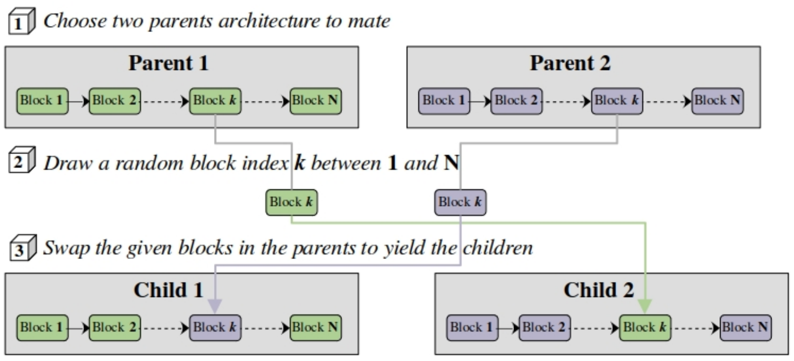

1 Choose two parents architecture to mate
Parent 1
Block 1
Block 2
Block k
Block N
Parent 2
2 Draw a random block index k between 1 and N
3 Swap the given blocks in the parents to yield the children
Child 1
Child 2

Fig. 3: A schematic representation of the crossover operator in the GA. To generate children from the parents, a single block is being exchanged between the parents.

Mutation operator. The mutation operator is defined to potentially modify each block on one or several of its parameters with a predefined neighborhood for each parameter. It select the blocks to be modified with a Bernoulli distribution with probability equal to the inverse of the number of blocks and randomly select the parameter to be modified within each block. Then it chooses a value in the neighborhood of this parameter value to apply the mutation on the candidate. This has been illustrated in Figure 4.

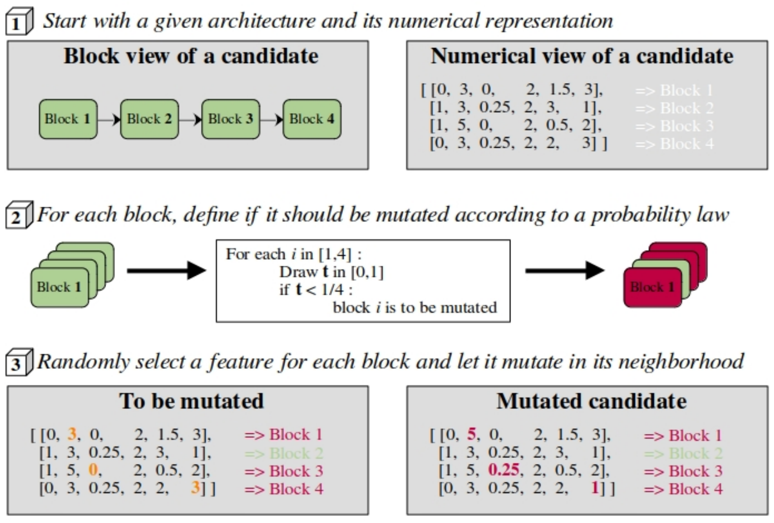

1 Start with a given architecture and its numerical representation
Block view of a candidate
Block 1
Block 2
Block 3
Block 4
Numerical view of a candidate
[[0, 3, 0, 2, 1.5, 3], => Block 1
[1, 3, 0.25, 2, 3, 1], => Block 2
[1, 5, 0, 2, 0.5, 2], => Block 3
[0, 3, 0.25, 2, 2, 3]] => Block 4
2 For each block, define if it should be mutated according to a probability law
Block 1
For each i in [1,4] :
Draw t in [0,1]
if t < 1/4 :
block i is to be mutated
Block 1
3 Randomly select a feature for each block and let it mutate in its neighborhood
To be mutated
[[0, 3, 0, 2, 1.5, 3], => Block 1
[1, 3, 0.25, 2, 3, 1], => Block 2
[1, 5, 0, 2, 0.5, 2], => Block 3
[0, 3, 0.25, 2, 2, 3]] => Block 4
Mutated candidate
[[0, 5, 0, 2, 1.5, 3], => Block 1
[1, 3, 0.25, 2, 3, 1], => Block 2
[1, 5, 0.25, 2, 0.5, 2], => Block 3
[0, 3, 0.25, 2, 2, 1]] => Block 4

Fig. 4: A schematic representation of the mutation operator in the GA. For each block of a given candidate, we randomly draw the blocks to be mutated - for each of them, we select a random feature to be mutated in a predefined neighborhood.

3.3 Multi-objective optimization

We might have chosen either the loss value or the accuracy on the test set as an objective function for the GA. However, the search space can be extremely large depending on the number of blocks and there might be plenty of architectures with the same accuracy on the test set. This requires to add a more complex objective function that takes into account several characteristics of each candidate to ensure a more precise discrimination and selection between them.

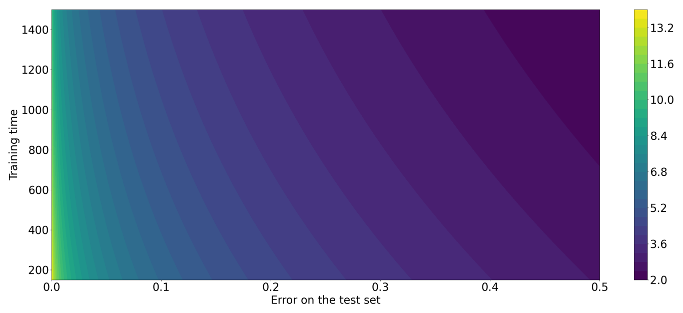


Fig. 5: The bi-objective search space for the GA objective function. The x axis represents the error on the test set and the y axis represents the training time for a given architecture.

To do so, we have implemented a multi-objective function that takes into account both the error on the test set as well as the training time on the training set. It has been defined as :

$$f(e,t) = -\log(10^{-3} + e) * (1 + \exp\left(-\frac{t - T_-}{T_+ - T_-}\right))$$

where T_- and T_+ are the minimum and maximum training time predefined depending on the number of blocks allowed in the search space. There is a 2D representation of the objective space illustrated in Figure 5. It has been designed to allow an easy trade-off between training time and accuracy with the following rule : an increase in accuracy is much more valuable than an decrease in training time. This induces that the GA process should be able to pay a heavy training time in order to gain accuracy on the test set while always preferring architectures which are faster to train for similar accuracies.

4 Experimental setup

In the following paragraphs, we would like to precisely defined the experimental setup we used to obtain the following results. We will clearly distinguish the setup used in the search process than the one for the final training of best architectures. We also give a detailed view of the hardware used. From the software point of view, we have used the nightly version of PyTorch [15] to have access to the newly implemented AutoAugment module.

Search method. We have been using the DEAP framework [7] to develop and adapt GA to our optimization problem. We have been using GA with the crossover and mutation operator defined in the previous section. We have used a particular module called Hall of Fame which allow to save on the long term the best individuals to reintroduce them in the population over time. We made several experiments with either randomly initializing the first generation or selecting it among the best candidates over a lots of randomly drawn architectures inside the search space. In the final experiments, we allowed a population size of either 400 or 2000 individuals while randomly sampling the first generation to prevent bias from previous researches and to correctly quantify the amount of time required for the GA process.

In order to reduce the training time while allowing candidates to train enough to be discriminated from each other, we used only 50% of the training set during the GA process while using the whole test set. We also used Stochastic Depth [9] with $p_{min} = 0.5$ to allow faster training. We did not use any data augmentation technique and allow each architecture to train for 10 epochs. We used the Stochastic Gradient Descent with Nesterov Momentum as the optimizer together with Exponential Moving Average [25] and with a learning rate of 0.05, a batch size of 64, a momentum of 0.9 and weight decay of 10^{-4}.

Final training of best architectures. At the end of the GA optimization, we have selected the architectures with the best score for a full training. The full training has been made over 500 epochs over the whole training set with a starting learning rate of 0.05 that is decreased of 3% every 2 epochs starting from epoch 50. We used the Sharpness-Aware Minimization [6] together with the SGD with Nesterov Momentum parametrized as specified during the GA process. We also used the stochastic depth during training similarly to the search process.

To ensure a correct regularization during the full training and prevent overfitting, we implemented :

- ShakeDrop modules [26] over all layers with probability 0.5
- Data augmentation over the training set only using AutoAugment [5]
- We also added the CutMix technique [27] in the data augmentation policy with probability 0.25

Hardware. We have been using various hardware components during our experiments. Table 2 details the technical specifications of the hardware using during both the search process and the final training. The number of the configuration used will be displayed in the results section.

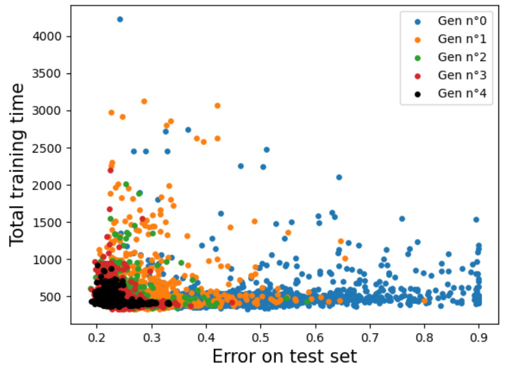
Num.	Nb nodes	CPU	Nb cores	RAM	GPU	Total GPUs
1	12	2 x POWER8 NVL 1.0	10/CPU	128 GiB	4 x Nvidia Tesla P100	48
2	2	2 x Intel Xeon E5-2698 v4	20/CPU	512 GiB	8 x Nvidia Tesla V100	16
3	2	2 x AMD EPYC 7452	32/CPU	128 GiB	2 x Nvidia A100	4
4	8	2 x Intel Xeon Gold 6126	12/CPU	192 GiB	2 x Nvidia Tesla P100/V100	16
Total	**24**	**58 CPUs**	**640 cores**	**4352 GiB**		**84**

Table 2: A comprehensive description of the hardware used during the experiments.

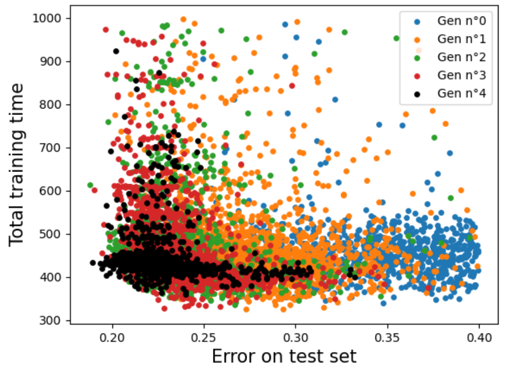
5 Results

5.1 Optimization inside the search space

In a first time, we will represent the solutions of the optimization process within the search space. Our GA method has been tested multiple times on both CIFAR10 and CIFAR100 yielding most of the time similar results. We would like to illustrate the effectiveness of our method to optimize both objectives at the same time. In order to do so, we have represented each generation by a different color and plot all individuals on the bi-objective space. This has been done on CIFAR10 in Figure 6 and on CIFAR100 in Figure 7.



(a) Overview of all generations

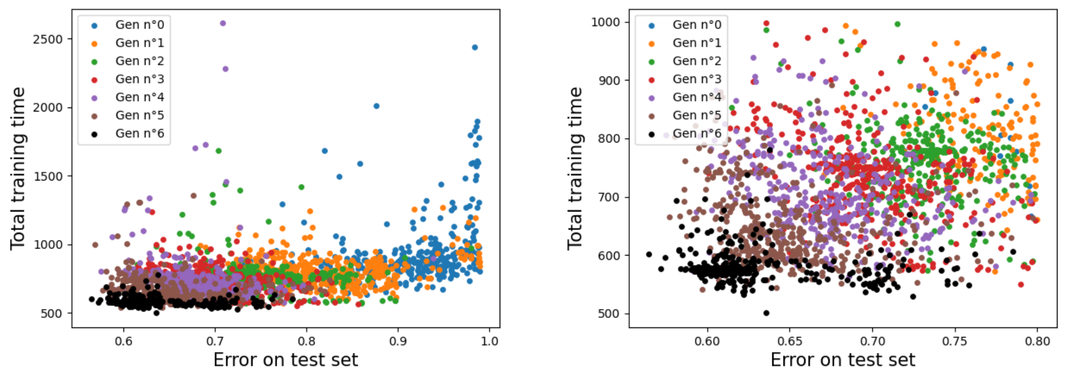


(b) Zoom on latest generations

Fig. 6: Results of GA search with bi-objective score function on the CIFAR10 dataset. The search space has been configured with 8 cells and each generation contains 2000 individuals. The x axis represents the error on the test set and the y axis represents the training time. Each color represents a different generation.

For the CIFAR10 experiment, the search space was set to 8 cells and we allowed an important population size : 2000. The idea behind such as massive population size was to try to ensure that the random initialization would sample suitable architecture to allow efficient breeding of the candidates and improved offspring. On the other hand, the CIFAR100 experiment was tested with a population size of only 400 since the architecture length was of 10 cells.

As expected, each generation seems to improve its population on both objectives and for both datasets. Let us recall that the initial population is totally randomized such that the GA requires to be able to reconstruct good architectures from random sampled ones. In order to confirm and quantify the improvement of the objectives over time, we have computed basic statistics on both experiments and objectives. Those are represented in Table 3.



(a) Overview of all generations (b) Zoom on latest generations

Fig. 7: Results of GA search with bi-objective score function on the CIFAR100 dataset. The search space has been configured with 10 cells and each generation contains 400 individuals. The x axis represents the error on the test set and the y axis represents the training time. Each color represents a different generation.

The statistics clearly highlights that all the objective are almost continuously improving for both experiments. This seems to indicate that the prolongation of each experiment might be leading to better results.

Dataset	Gen 0		Gen 1		Gen 2		Gen 3		Gen 4		Gen 5		Gen 6	
Accuracy	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
CIFAR10	**54.84**	13.69	**71.82**	5.26	**76.34**	2.69	**76.57**	2.11	**77.40**	2.07				
CIFAR100	**8.74**	5.58	**18.10**	6.49	**25.67**	4.88	**29.84**	4.10	**32.65**	4.18	**35.09**	4.37	**37.12**	4.10
Time	Mean		Mean		Mean		Mean		Mean		Mean		Mean	
CIFAR10	**500**		**513**		**475**		**463**		**446**					
CIFAR100	**939**		**807**		**768**		**738**		**732**		**660**		**580**	

Table 3: Statistics on time and accuracy over all generations. It clearly highlights the capacity of the GA to optimize both objectives at the same time.

5.2 Fine-tuning

In a second time, we need to introduce the results of the final training experiment with the experimental setup specified in the previous section. We should first state that the architectures trained in this section aren't the results of the experiments illustrated previously : we took the best candidates we had over all experiments. Beside, there has been many attempt to optimize the final training optimizing process but we believe that it did not yet reach its full capacity.

The final results for CIFAR10 are presented in Table 4 where we describe, for each paper, the test set error and the number of parameters. We should specify that a fair comparison should also take into account the required training time, since it is a key point of our optimization process but this data was not available for the presented papers. As we can see in the table, with only a few generations, our GA search process has been able to find two architectures that rank at the third place among quoted results, one of them having a very low number of parameters (2.35 M).

On the other hand,the results for CIFAR100 displayed in Table 5 shows a less good ranking of our method (4th) but still highlights its capacity to find architectures with a small number of

Name	Test Set Error	Nb of Parameters	Search Method
NSGA-NET [14]	2.02	4 M	GA
FDA [3]	4.94	11.18 M	GA & FDA
MetaQNN [1]	6.92	5.1 M	RL
HMCNAS [3]	9.41	5.1 M	Markov Chain & BO
CGP-CNN (ResSet) [18]	5.01	2.01 M	CGP
MOPSO/D-Net [10]	5.88	8.1 M	PSO
CNN-GA [19]	3.22	2.9 M	GA
This paper (lowest error)	**4.44**	**10.99 M**	GA (6 cells)
This paper (lowest nb params)	**4.74**	**2.35 M**	GA (6 cells)

Table 4: Results of the final training on CIFAR10

parameters. This might highly be induced by the design of the search space that originally intended to target efficient neural networks for mobile phones.

Name	Test Set Error	Nb of Parameters	Search Method
NSGA-NET [14]	20.74	3.3 M	GA
FDA [3]	21.11	12.5 M	GA & FDA
MetaQNN [1]	27.14	11.18 M	RL
CGP-CNN (ResSet) [18]	25.1	3.43 M	CGP
CNN-GA [19]	20.53	4.1 M	GA
This paper	**22.29**	**4.51 M**	GA (8 cells)

Table 5: Results of the final training on CIFAR100

6 Conclusion and discussion

We presented the application of GA methods to the factorized hierarchical search space issued from [22]. It showed a great capacity to find promising architectures while allowing a good flexibility within bi-objective optimization. There are plenty of aspects of this work that might be enhanced to allow a more efficient search.

First, we wish to implement other research methods over the same search space (BO & Local Search). Then, we could think about improving the search space by adding a zero module to allow variable architecture size and new modules as those being defined in [24]. Eventually, one could also think about scaling up good candidates using different rules to create deeper architectures from good candidates from the optimization process.

Finally, it might be an interesting path to decompose the optimization process into different stages : optimization of data flow, optimization of computational modules and optimization of secondary parameters. Indeed, the data flow of an architecture (resolution of the image at each step and number of channels) seem to be one of the key point of an architecture efficiency in training and a major aspect of prevention of overfitting. Besides, this would reduce the complexity of the optimization process by breaking it into smaller and easier optimization problems.

7 Acknowledgments

A particular thanks must be dedicated to the whole team of Grid5000 [2] for making available all the setups described in Table 2.

References

1. Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *CoRR*, abs/1611.02167, 2016.
2. 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.
3. Renato J. Cintra, Stefan Duffner, Christophe Garcia, and André Leite. Low-complexity Approximate Convolutional Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, December 2018.
4. Dan Ciresan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella, and JÃ¼rgen Schmidhuber. Flexible, high performance convolutional neural networks for image classification. pages 1237–1242, 07 2011.
5. Ekin Dogus Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data. *CoRR*, abs/1805.09501, 2018.
6. Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *CoRR*, abs/2010.01412, 2020.
7. Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, jul 2012.
8. Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *CoRR*, abs/1709.01507, 2017.
9. Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. *CoRR*, abs/1603.09382, 2016.
10. Jing Jiang, Fei Han, Qinghua Ling, Jie Wang, Tiange Li, and Henry Han. Efficient network architecture search via multiobjective particle swarm optimization based on decomposition. *Neural Networks*, 123:305–316, 2020.
11. Lukasz Kaiser, Aidan N. Gomez, and François Chollet. Depthwise separable convolutions for neural machine translation. *CoRR*, abs/1706.03059, 2017.
12. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.
13. Vasco Lopes and Luís A. Alexandre. HMCNAS: neural architecture search using hidden markov chains and bayesian optimization. *CoRR*, abs/2007.16149, 2020.
14. Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh D. Dhebar, Kalyanmoy Deb, Erik D. Goodman, and Wolfgang Banzhaf. NSGA-NET: A multi-objective genetic algorithm for neural architecture search. *CoRR*, abs/1810.03522, 2018.
15. Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.
16. Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018.
17. Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.
18. Masanori Suganuma, Masayuki Kobayashi, Shinichi Shirakawa, and Tomoharu Nagao. Evolution of deep convolutional neural networks using cartesian genetic programming. *Evolutionary Computation*, 28(1):141–163, 2020.
19. Yanan Sun, Bing Xue, Mengjie Zhang, and Gary G. Yen. Automatically designing CNN architectures using genetic algorithm for image classification. *CoRR*, abs/1808.03818, 2018.
20. Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.
21. El-Ghazali Talbi. Automated design of deep neural networks: A survey and unified taxonomy. *ACM Comput. Surv.*, 54(2), March 2021.
22. Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile. *CoRR*, abs/1807.11626, 2018.

23. Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019.
24. Mingxing Tan and Quoc V. Le. Efficientnetv2: Smaller models and faster training. *CoRR*, abs/2104.00298, 2021.
25. Xu Tian, Jun Zhang, Zejun Ma, Yi He, and Juan Wei. Exponential moving average model in parallel speech recognition training. *CoRR*, abs/1703.01024, 2017.
26. Yoshihiro Yamada, Masakazu Iwamura, and Koichi Kise. Shakedrop regularization. *CoRR*, abs/1802.02375, 2018.
27. Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. *CoRR*, abs/1905.04899, 2019.

Optimization of deep neural network: Application to the forecast of energy consumption[*]

T. Firmin[1], E-G. Talbi[2], S. Claudel[3], and J-Y. Lucas[3]

[1] Centre de Recherche en Informatique, Signal et Automatique de Lille
thomas.firmin@inria.fr
[2] University of Lille INRIA, France.
el-ghazali.talbi@univ-lille.fr
[3] EDF R&D, Paris, France

1 Introduction

Electricity is not storable, therefore Electricité De France (EDF) must adjusts its energy production to meet the demand and not creating accident due to under or overproduction. Data are complex, the energy consumption is noisy and affected by features like meteorological or calendar data. For that reason the company uses a Deep Neural Network (DNN) as a regression model to forecast the consumption.
Such networks raise several questions about their optimization. DNNs are associated to many hyperparameters forming a mixed search space. So, finding the best combination is a NP-hard optimization problem.

First of all, we have to define a valid search space. This step is crucial, optimizing in a poorly defined search space will follow to worthless results. Implementation of metaheuristics and optimization algorithms such as, Simulated Annealing (SA), Evolutionary Algorithm (EA), Bayesian Optimization (BO) [4], Chaotic Optimization (CO)[5], Fractal Decomposition Algorithm (FDA)[6], Ensemble Methods and Online Aggregation, has permitted to find multiple and significant optimized solutions and meta-models, compared to the initial model.
Other works applied to the same domain, using CNN-LSTM or DHN networks also performed a hyperparameters optimization using Sequential Model-based Optimization [1], genetic algorithm [2], or even manually [3]. But in this paper we will focus on the optimization aspect of such networks.

One major constraint associated to our problem is the parallelization of those metaheuristics.[7–9] To speed-up the process, algorithms and the loss function have been computed using multi-nodes, multi-GPUs and multi-CPUs strategies on Grid5000, a large-scale and flexible testbed for HPC, Big Data and AI.

2 Description of a complex problem

2.1 A regression problem

The goal of this project is to optimize a DNN composed of multiple convolutional, pooling and dense layers. The company uses this model to forecast the daily electrical consumption in France. The outcome variable is the consumption in Megawatt. It is a time series, with a multi-level seasonality (yearly, weekly and daily).

- **Meteorological data**: temperature, wind, cloudiness...
- **Calendar data**: date, public holiday, timestamp...

Data are mixed, there are qualitative features such as the temperature or categorical data as week days. The model makes daily predictions half hour by half our. To evaluate the quality of a model, we will use the **Root Mean Squared Error (RMSE)** which gives a measure of how good predictions are.

$$RMSE(\hat{\theta}) = \sqrt{E((\hat{\theta} - \theta)^2)}$$

[*] This project was funded by the PGMO project in partnership with EDF

Where $\hat{\theta}$ is the estimator, the predicted consumption, and θ is the estimated parameter, the observed consumption.

2.2 Challenges and strategies for the optimization

The goal here is to perform a Hyperparameter Optimization (HPO) on the model. We will not deal with the optimization of the network architecture, known as Neural Architecture Search (NAS), nor AutoDNN which combines HPO and NAS. We divided data from 2014 to 2019 into a training and validation data set, 80% and 20% respectively. Here, the optimization consists in finding the best combination of hyperparameters which minimizes the RMSE computed with predictions made by the DNN after the learning phase. Such models are associated to numerous hyperparameters, and those are forming a mixed search space. Here are the 18 selected:

Table 1: Selected hyperparameters and bounds

Name	**Type**	**Bounds**	**Recurrence**
Number of filters	Discrete	[1,100]	2
Kernel size	Discrete	[1,96]	2
Number of neurons	Discrete	[2,200]	3
Activation	Categorical	relu, sigmoid, softmax, softsign, tanh, selu, elu, linear, swish	6
Learning rate	Continuous	[0.0001,0.5]	1
Optimizer	Categorical	sgd, rmsprop, adam, adadelta, adagrad, adamax, nadam, ftrl	1
Pooling size	Discrete	[1,96]	2
Batch size	Discrete	[10,110]	1

According to previous descriptions, we can notice two main challenges for the optimization:

– Expensive, black-box and noisy loss function
– Large scale and mixed search space

Thanks to EDF and their model, we know one combination of hyperparameters and one model which gives satisfactory predictions. This solution has been determined manually. Therefore the goal of the project is to find a methodology to automate this search. As a reference point, the initial **RMSE** is equal to **858**. The best optimized solution has **814** and the best meta-model **721**.

– **Step 1, local search:**
The first objective was to determine if the model was easily perfectible or not. That is why we used SA starting from EDF initial solution to explore its neighborhood.
– **Step 2, exploration and exploitation:**
Then we use a combination of a EA and SA. We took advantage of the exploration feature of the EA to find promising areas of the search space where to apply an exploitation strategy.
– **Step 3, advanced optimization:**
However, EA and SA require a lot of evaluations to be efficient. So, by reducing significantly the number of required loss function call, BO is a solution to our problem. Then we introduced 2 recent strategies: CO which uses a chaotic dynamic to efficiently explore and exploit the search space, and FDA which uses fractals to divide the search space into smaller promising sub-spaces.
– **Step 4, Ensemble methods:**
Thanks to the previous strategies, we had several 'good models'. We noticed that some of them were more or less experts on specific data. That is why, we developed Bagging and Adaboost. Finally the Online Aggregation method was the best methodology to combine experts on a time series for regression problem.

2.3 Software and hardware specifications

We used Python 3.9, Tensorflow 2.5, DEAP 1.3.1 for EA, GPflow 2.1 and GPy for BO, Opera for the Online Aggregation, MPI and CUDA 11.1 for the distributed computing part and GPUs computing
We performed experiments on Grid5000, using a maximum of 16 GPUs NVIDIA Tesla V100 and 128 CPUs Intel Xeon Gold 5220, during multiple sessions of 14 to 62 hours last.

3 Algorithms descriptions and paralleled versions

During prior experiments we noticed over-fitting, therefore we decided to implement a recent method called Sharpness Aware Minimization (SAM).[10]

Simulated Annealing on EDF initial solution: SA allows escaping from local optima using an acceptance probability. As a reminder the goal of using SA here, is to determine whether or not it is easily improvable. Otherwise, at least we can obtain data about the neighborhood landscape.

Evolutionary Algorithm and Simulated Annealing: To tackle the exploration phase of the search space, we decided to implement a classical *steady-state* EA. Furthermore, we combined EA and SA by selecting the best solution found by EA and applying an intensification phase with SA.

Advanced Optimization: With BO, the goal is to interpolate an expensive objective function with a surrogate model, here, it is a Gaussian Process (GP) with a Radial Basis Function kernel (RBF). Regarding the acquisition function, we used Expected improvement (EI) and Lower Confidence Bound (LCB), to optimize the acquisition function we applied EA and CO.

To perform better exploration and exploitation phases, we used CO on the HPO problem. Chaos allows to quickly and violently move over the search space, unlike the exploitation phase where chaos is used to waggle points around an initial solution.
Instead of using EA for finding promising areas of the search space. We used FDA [6]. It uses hyper-spheres, a tree search algorithm (Beam Search), an exploration (CGS [5]) and an exploitation (ILS [6]) strategies, to decompose the search-space into smaller sub-spaces in a fractal way.

Ensemble Methods and Online Aggregation: Bagging has the advantage to reduce predictions variance, and being easily parallelizable. For regression problem, the aggregation is made by averaging predictions of all models. Then, we used Adaboost to accentuate the expertise of models selected by previous optimization methods. To do predictions with the meta-model, we can use a weighted median [11], a simple linear combination [12], etc. Weights associated to models are computed according to the quality of predictions on data patterns.

Finally, setting unchangeable weights to combine models isn't the best way to perform the aggregation on a time series. Indeed, associating models to data patterns involves the loss of temporal aspect of the validation data. Instead, we can consider validation data-set as a time series. It allows to build an adaptive meta-model according to previous errors of predictions. This is the idea of online aggregation, which is more suitable for this industrial problem.[13, 14]

Table 2: Experiments summary

Step	**Algorithm**	**Best RMSE**	**Improvement**
Initial		858	0%
Exploitation	SA	858	0%
Exploration and exploitation	EA	1211	0%
	EA + SA	1046	0%
Advanced optimization	BO	1099	0%
	BO + SAM	855	0.3%
	CO + SAM	955	0%
	FDA + SAM	1052	0%
	FDA + SAM + SA	814	5%
Ensemble methods	Bagging	936	0%
	Adaboost	836	2.6%
	Online aggregation	721	19%

4 Conclusion

The objective of this project was to find a method to automate HPO problem for the forecasting of energy consumption in France. We had access, thanks to EDF, to a good initial solution. The first step consists in trying to optimize and explore the neighborhood of this initial model. Results showed that EDF solution was a local optimum, hard to optimize. Therefore, the second step consists in exploring the search space to find better area to apply an exploitation procedure. To do so we combined EA and SA, but once again results were good but insufficient. However our models were over-fitting, to deal with this problem we applied SAM method. Then, to reduce the number of necessary loss function call, we implemented BO with CO as the acquisition function optimizer. We also used FDA to efficiently find new promising area of the search space, and combined with SA, we found the best solution. Finally we noticed that some models were more or less experts on certain data. We decided to implement classical ensemble methods, such as bagging and Adaboost. However, by using those methods we lost the temporal characteristic of our data. So, a solution was to use Online Aggregation for time series regression, and thanks to this adaptive method we found the best meta-model.

References

1. Rahman, Aowabin & Srikumar, Vivek & Smith, Amanda. (2018). Predicting electricity consumption for commercial and residential buildings using deep recurrent neural networks. Applied Energy. 212. 372-385. 10.1016/j.apenergy.2017.12.051.
2. Ahmad MW, Mouraud A, Rezgui Y, Mourshed M. Deep Highway Networks and Tree-Based Ensemble for Predicting Short-Term Building Energy Consumption. Energies. 2018; 11(12):3408. https://doi.org/10.3390/en11123408
3. Kim, Tae-Young & Cho, Sung-Bae. (2019). Predicting Residential Energy Consumption using CNN-LSTM Neural Networks. Energy. 182. 10.1016/j.energy.2019.05.230.
4. El-Ghazali Talbi. Optimization of deep neural networks: a survey and unified taxonomy. 2020. hal-02570804v2
5. Nassime Aslimani, El-Ghazali Talbi, Rachid Ellaia. Tornado: An Autonomous Chaotic Algorithm for-Large Scale Global Optimization. 2020. hal-02499326v2
6. Amir Nakib, Léo Souquet, El-Ghazali Talbi. Parallel fractal decomposition based algorithm for big continuous optimization problems. Journal of Parallel and Distributed Computing, Elsevier, 2019, 133, pp.297-306.
7. Talbi, El-Ghazali. (2015). Parallel Evolutionary Combinatorial Optimization. 1107-1125. 10.1007/978-3-662-43505-2_55.
8. Jialei Wang, Scott C. Clark, Eric Liu, & Peter I. Frazier. (2019). Parallel Bayesian Global Optimization of Expensive Functions.
9. Clément Chevalier, David Ginsbourger. Fast Computation of the Multi-points Expected Improvementwith Applications in Batch Selection. 2012. hal-00732512v2
10. Pierre Foret, Ariel Kleiner, Hossein Mobahi, & Behnam Neyshabur. (2021). Sharpness-Aware Minimization for Efficiently Improving Generalization.

11. Drucker, "Improving Regressors using Boosting Techniques", 1997.
12. Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. An Introduction to Statistical Learning : with Applications in R. New York :Springer, 2013.
13. Cesa-Bianchi, N., & Lugosi, G. (2006). Prediction, Learning, and Games. Cambridge: Cambridge University Press. doi:10.1017/CBO9780511546921
14. Opera: Online Prediction by ExpeRt Aggregation. Pierre Gaillard, Yannig Goude. R package, 2016. https://cran.r-project.org/web/packages/opera/opera.pdf

A review of the Electric School Bus Routing Problem

H. Bideq[1], K. Ouaddi[2], R. Ellaia[3] and A. Gorge[4]

1. LERMA – EMI, Rabat Morocco
hajar.bideq@gmail.com
2. UM5 – Rabat Morocco
khaoula.ouaddi@gmail.com
3. LERMA – EMI, Rabat Morocco
rachid.ellaia@gmail.com
4. UM6P – ABS, Rabat Morocco
Agnes.GORGE@um6p.ma

Abstract - *Electric vehicles are lately receiving a lot of researchers' attention, especially in the transportation field. It reflects the growing consciousness of the substantial harm that traditional vehicles are causing. In this present paper, we concentrate our focus on the Electric School Bus Routing Problem (ESBRP) since it is one of the real-world applications that rely on a fleet of fuel vehicles.*

Keywords: School Bus Routing Problem (SBRP), Electric School Bus Routing Problem (ESBRP), Electric Vehicle Routing Problem (EVRP), Electric Vehicles (EV)

1 Introduction

Carbon emissions, an unbalanced ecosystem, increasing respiratory diseases, and reliance on oil are a few of much alarming earth's distress' aspects. They are symptoms of the excessive use of traditional diesel vehicles. On the one hand, improving air quality is a must, and on the other hand, transport services are the backbone of our economic growth. Trying to choose between those two is like being between a rock and a hard place. Fortunately, electric vehicles tackle most of the issues caused by classic petroleum engines and give us an environment-friendly alternative **[1]**.

In this context, several companies that assure pickup and delivery systems, such as DHL and FedEx, are gradually opting for an electric fleet. This kind of service concerns an EVRP variant called EVRP with Pickup and Delivery Problem (EVRPPD). EVRPPD involves goods transportation and people's transportation, which makes it rich in possible real-world applications. This paper focuses on only one specific application: The ESBRP. It is an SBRP extension. For readers interested in knowing more about EVRP and its variants, here are some detailed literature reviews **[2] [3] [4]**.

The plan of this paper is as follows: there is a brief review of the SBRP in section 2. Section 3 elucidates the two ESBRP works in the literature, with a mention of some articles that are closely related to it. The last one is for some conclusions and perspectives.

2 School Bus Routing (SBRP)

In this section, we briefly define the SBRP and related works to introduce the ESBRP later on.

The SBRP is one of the VRP with Pickup and Delivery's real-world applications for students' transportation **[5]**. It was first introduced in 1969 by Newton and Thomas **[6]** and has drawn many researchers' attention since then. The growing interest in the SBRP during the last decades is due to the expansion of students and schools, which makes the manually generated routes expensive and heavy **[7]**. We can count more than one hundred publications that consider SBRP from an operations research angle **[8]**, but even with such a number, we can still find that each version is unique. This variety is due to, on the one hand, the fact that SBRP is problem-related **[9]**, and on the other hand, the possibility of applying it in different contexts like employees' transport **[10]**. The United States **[11][12][13]** and China **[9][14]** are on the top list of the countries where the SBRP is most studied and applied in real-world cases.

Given several buses and groups of students and schools, the SBRP aims to find the best routing to pick up all children from the closest bus stop to their homes and deliver them to their respective schools. It consists of five sub-problems: data preparation, bus stop selection, bus route generation, school bell time adjustment, and route scheduling **[15]**. Most of the publications consider only one sub-problem, which is mainly the bus route generation. **Table 1** presents a non-exhaustive classification of some SBRP's papers, based on the most frequent features/characteristics found in the literature. For readers who are interested in knowing more about SBRP, we recommend **[5][8][15]**.

Reference	**Sub-problems**	**Objectives**	**Fleet type**	**Type of loads**
[11] Bertsimas et al (2019)	• Bus stop selection. • Bus route generation. • Bus route scheduling. • School bell adjustment.	• Total cost • Number of stops. • Walking distance for students.	Heterogeneous fleet	Mixed loads not allowed
[12] Park et al (2012)	Bus Route generation.	Number of vehicles.	Homogeneous fleet	Mixed loads are allowed
[14] Chen et al (2015)	• Bus route generation • Bus route scheduling	• Total cost. • Total travel distance. • Number of vehicles.	Heterogeneous fleet	Mixed loads are not allowed
[13] Kamali et al (2013)	• Bus stop selection • Bus route generation	Total travel distance.	Homogeneous fleet	Mixed loads are not allowed.

Table 1: Classification based on the sub-problems studied, the objectives considered, the type of the vehicles fleet and the allowance or not of mixed loads.

3 Electric School Bus Routing (ESBRP)

In this section, we define ESBRP and exhibit the two related works.

As shown in **table 1**, SBRP can have different objectives: *minimizing the cost, minimizing the number of vehicles, etc*. However, all of the publications mentioned above miss a crucial aspect which is the environment's sustainability. That is the purpose of ESBRP.

ESBRP is an SBRP's variant that uses electric buses instead of the usual heavy vehicles and considers the particularities of having such a fleet. **Figure 1** illustrates a solution for a single-school ESBRP example. The choice of electric buses is justified since they are clean-energy vehicles that have the lowest emissions **[16]**.

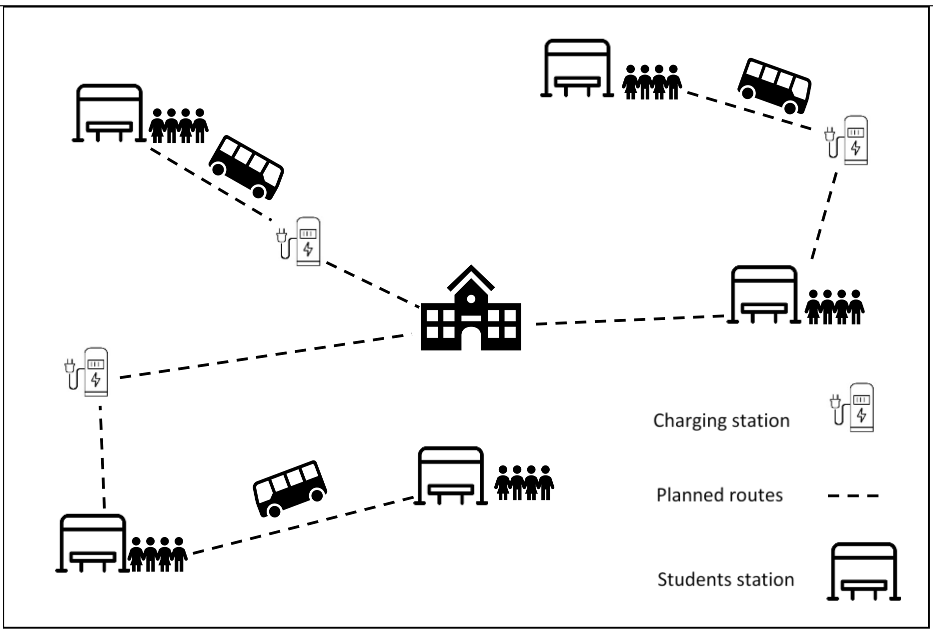


Figure 1 : An example of an ESBRP solution.

However, while SBRP is widely studied, especially in the last two years, only two publications **[17][18]** consider ESBRP. These articles are real-world cases from Mexico and Turkey. We also discuss **[19][20][21]** that are related to ESBRP in the coming sections.

Related works:

[18] works on a real road network of a Turkish university campus. The authors formulate a mathematical model that handles the route generation sub-problem using electric buses or internal combustion engine vehicles. Their model is multi-objective and aims to: minimize the costs, minimize CO2 emissions and minimize the time of charging.

However, finding the best optimal set of routes is not the purpose of their work. They make a comparative study between electric vehicle use and internal combustion engine vehicle use. **[18]** base their comparison on the cost. They define it as an addition of three essential elements:

- The charge of the vehicles used.
- The cost of implementing a charging station.
- The environmental impact of the fleet (it is equal to zero for electric buses).

[18] formulate a MIP that generates optimal routes and defines where to place charging nodes. This latter is related to an interesting contribution of this article which is the consideration of intermediate nodes. The same authors of **[18]** expand this topic in **[22]**.

They conduct experiments on different small-sized sets of real-world data of 43 nodes in total. Results show a gain of 40% when preferring electric vehicles over internal combustion engine vehicles. According to **[18]**, the difference cost-wise occurs even though the distance travelled by the two types of buses is the same. Nevertheless, every rose has its thorns. The electric SBRP is 3 hours slower than the ICEV SBRP due to the recharging time needed.

Unlike **[18]**, **[17]** uses a metaheuristic to solve the SBRP. **[17]** applies **[23]**'s classic SBRP model while considering the specificities of a homogeneous electric fleet of buses. They formulate a mathematical model for ESBRP that generates the optimal routes and minimizes the cost and trip times of the students. Their model takes into account various restrictions that make it more complex:

- Electric buses should not exceed their maximum capacity.
- Students have to be on time at school.
- The electric buses get charged once they are in the depot.
- The energy of the electric buses should never be zero.

[17] tests their model using CPLEX on a set used by Diaz-Parra in **[24]** but while considering an electric fleet. They also apply their model to real-world data from the Autonomous University of Morelos State (AUMS), located in Cuernavaca, Mexico. They obtained it from conducting surveys of the students of the university. **[17]** adopts another solution approach which is the genetic algorithm with a tournament selection and a K-point crossover. They claim that using electric vehicles is more economical than using fuel buses.

Some reflections:

The SBRP literature lacks studies on ESBRP compared with other variants of EVRP, while it is becoming crucial to trade traditional heavy buses for environment-friendly vehicles such as hybrid or electric buses. This scarceness in publications might be because of the relative newness of the EVRP. Otherwise, it could be that most schools' administrations or transport societies are not yet ready to make the transition, which is intelligible due to the complexity of the task. **[20]** propose a model that can assist organizations in the early replacement of the 100% diesel fleet. **[19]** tackle the main limitations of using electric buses, which are charging time and battery life **[4]** and propose two beneficial routing methods. As a process, the electrification target is not easy to attend but, the comparative results shown in **[21]** can be an incentive for many schools and organizations to consider a transition.

4 Conclusion and perspectives

We presented a brief review of the ESBRP, which is still not extensively studied. During our work, we only found two articles directly related to ESBRP, while there are more than one hundred publications on SBRP. This absence of literature reflects the current lack of interest in environmental damages caused by traditional diesel buses. But at the same time, it is normal since making the electrical transition is not evident for most organizations.

While reading about EVRP and its endless environmental benefits, one of our research perspectives is to expand works on ESBRP, especially in other areas like employees' transportation.

References

[1] Desaulniers, G., Errico, F., Irnich, S., & Schneider, M. (2016). Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research*, *64*(6), 1388-1405.

[2] Juan, A. A., Mendez, C. A., Faulin, J., De Armas, J., & Grasman, S. E. (2016). Electric vehicles in logistics and transportation: A survey on emerging environmental, strategic, and operational challenges. *Energies*, *9*(2), 86.

[3] Qin, H., Su, X., Ren, T., & Luo, Z. (2021). A review on the electric vehicle routing problems: Variants and algorithms. *Frontiers of Engineering Management*, 1-20.

[4] Schneider, M., Stenger, A., & Goeke, D. (2014). The electric vehicle-routing problem with time windows and recharging stations. *Transportation science*, *48*(4), 500-520.

[5] Toth, P., & Vigo, D. (Eds.). (2014). *Vehicle routing: problems, methods, and applications*. Society for Industrial and Applied Mathematics, 193-212.

[6] Newton, R. M., & Thomas, W. H. (1969). Design of school bus routes by computer. *Socio-Economic Planning Sciences*, *3*(1), 75-85.

[7] Bodin, L. D., & Berman, L. (1979). Routing and scheduling of school buses by computer. *Transportation Science*, *13*(2), 113-129.

[8] Ellegood, W. A., Solomon, S., North, J., & Campbell, J. F. (2020). School bus routing problem: Contemporary trends and research directions. *Omega*, *95*, 102056.

[9] Li, L. Y., & Fu, Z. (2002). The school bus routing problem: a case study. *Journal of the Operational Research Society*, *53*(5), 552-558.

[10] Leksakul, K., Smutkupt, U., Jintawiwat, R., & Phongmoo, S. (2017). Heuristic approach for solving employee bus routes in a large-scale industrial factory. *Advanced Engineering Informatics*, *32*, 176-187.

[11] Bertsimas, D., Delarue, A., & Martin, S. (2019). From school buses to start times: Driving policy with optimization.

[12] Kim, B. I., Kim, S., & Park, J. (2012). A school bus scheduling problem. *European Journal of Operational Research*, *218*(2), 577-585.

[13] Kamali, B., Mason, S. J., & Pohl, E. A. (2013). An analysis of special needs student busing. *Journal of Public Transportation*, *16*(1), 2.

[14] Chen, X., Kong, Y., Dang, L., Hou, Y., & Ye, X. (2015). Exact and metaheuristic approaches for a bi-objective school bus scheduling problem. *PloS one*, *10*(7), e0132600.

[15] Park, J., & Kim, B. I. (2010). The school bus routing problem: A review. *European Journal of operational research*, *202*(2), 311-319.

[16] Li, L., Lo, H. K., Xiao, F., & Cen, X. (2018). Mixed bus fleet management strategy for minimizing overall and emissions external costs. *Transportation Research Part D: Transport and Environment*, *60*, 104-118.

[17] Díaz-Parra, O., Ruiz-Vanoye, J. A., Fuentes-Penna, A., Zezzatti, A. O., Cruz-Hernández, E., Estrada-García, S., & Alvarado-Pérez, A. (2020). Editorial for Volume 11 Number 2: Electric School Bus Routing Problem for Smart Cities. *International Journal of Combinatorial Optimization Problems and Informatics*, *11*(2), 1.

[18] Hulagu, S., & Çelikoglu, H. B. (2019, June). A multiple objective formulation of an electric vehicle routing problem for shuttle bus fleet at a university campus. In *2019 6th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)* (pp. 1-5). IEEE.

[19] Li, L., Lo, H. K., Xiao, F., & Cen, X. (2018). Mixed bus fleet management strategy for minimizing overall and emissions external costs. *Transportation Research Part D: Transport and Environment*, *60*, 104-118.

[20] Pelletier, S., Jabali, O., Mendoza, J. E., & Laporte, G. (2019). The electric bus fleet transition problem. *Transportation Research Part C: Emerging Technologies*, *109*, 174-193.

[21] Noel, L., & McCormack, R. (2014). A cost benefit analysis of a V2G-capable electric school bus compared to a traditional diesel school bus. *Applied Energy*, *126*, 246-255.

[22] Hulagu, S., & Celikoglu, H. B. (2020). An Electric Vehicle Routing Problem With Intermediate Nodes for Shuttle Fleets. *IEEE Transactions on Intelligent Transportation Systems*.

[23] Gavish, B., & Shlifer, E. (1979). An approach for solving a class of transportation scheduling problems. *European Journal of Operational Research*, *3*(2), 122-134.

[24] Díaz-Parra, O., Ruiz-Vanoye, J. A., & Zavala-Díaz, J. C. (2011). School bus routing problem library-SBRPLIB. *International Journal of Combinatorial Optimization Problems and Informatics*, *2*(1), 23-26.

Author Index

A. Carvalho Iago, 149–158
Aghelinejad Mohammadmohsen, 197–206
Ait El Cadi Abdessamad, 86–89
Akhmedova Shakhnaz, 76–85
Alaouchiche Yasmine, 175–179
Almeida Guimarães Marcos Vinícius, 149–158
Amodeo Lionel, 173, 174
Aragon-Jurado Jose Miguel, 180–183
Arbaoui Taha, 188–190
Arnao Molina Máximo, 103–110
Athmani Mohamed Elamine, 188–190

Bahri Oumayma, 173, 174
Balesdent Mathieu, 191–193
Becerra-Rozas Marcelo, 103–110
Belhor Mariem, 121–130
Bellenguez Odile, 57–65
Benito-Jareño Carlos, 184–187
Benyamina Abou El Hassane, 169–171
Bideq Hajar, 223–227
Boufflet Jean-Paul, 47–56
Bouvry Pascal, 184–187, 207–217
Bouzidi Halima, 86–89

Castillo Mauricio, 103–110
Cisternas-Caneo Felipe, 103–110
Claudel Sandra, 218–222
Contreras-Bolton Carlos, 131–138
Crawford Broderick, 103–110
Czimmermann Peter, 39–46

Danoy Gregoire, 207–217
De La Torre Juan Carlos, 180–187
De Prado Javier, 24–33
Delmotte François, 121–130
Devienne Philippe, 169–171
Dorronsoro Bernabe, 180–187

El Ghazali Talbi, 207–217
El-Amraoui Adnen, 121–130
Ellaia Rachid, 223–227

Firmin Thomas, 218–222

Galvan Edgar, 139–148
Gamot Juliette, 191–193
Gmys Jan, 90–92
Gorge Agnès, 223–227
Gribiss Hamza, 197–206

H. Pereira Armando, 149–158
Hamasaki Koki, 111–120

Iturra Sergio, 131–138

Janáček Jaroslav, 9–23
Jatschka Thomas, 93–102
Jemai Abderrazak, 121–130

Khoukhi Saâdia, 2–8
Kvet Marek, 9–23

Lando Tommaso, 172
Lardeux Frédéric, 159–168
Larroche François, 57–65
Le Badezet Benoît, 57–65
Lemus-Romani José, 103–110
Lopez-Garcia Alvaro, 66–75
Lucas Jean-Yves, 218–222
López-López Aurelio, 66–75

Makki Ayman, 207–217
Massa Franck, 34–38
Massonnet Guillaume, 57–65
Medjahed Farah, 169–171
Melab Nouredine, 90–92
Monfroy Eric, 159–168
Moscatelli Sandro, 24–33
Moukrim Aziz, 47–56

Nakata Masaya, 111–120
Niar Smail, 86–89
Noronha Thiago, 149–158
Nouredine Melab, 191–193

Ouaddi Khaoula, 223–227
Ouarnoughi Hamza, 86–89
Ouazene Yassine, 175–179
Ouberkouk Mohamed Amine, 47–56

Parada Victor, 131–138

Piñeyro Pedro, 24–33

Raidl Günther, 93–102
Rodemann Tobias, 93–102
Rodríguez-Gómez Gustavo, 66–75
Rubio José-Miguel, 103–110

Sadet Jérémy, 34–38
Semenkin Eugene, 76–85
Soto Ricardo, 103–110
Stanovov Vladimir, 76–85

T. Bogue Eduardo, 149–158
Talbi El-Ghazali, 34–38, 86–92, 180–183, 191–196, 218–222
Tansini Libertad, 24–33
Tapia Diego, 103–110
Tison Thierry, 34–38
Todosijevic Raca, 194–196
Torra Vicenc, 139–148
Tremolet Arnault, 191–193
Trujillo Embry Andrés, 103–110
Turpin Isabelle, 34–38

Urrutia Sebastián, 149–158

Viera Omar, 24–33

Wuilbercq Romain, 191–193

Yalaoui Farouk, 175–179, 188–190, 197–206