Proceedings OLA'2022 International Conference on Optimization and Learning

ola 2022

18-20 July 2022, Syracusa, Italia

Organizers

UNIVERSITÀ degli STUDI di CATANIA

Université de Lille

Table of contents

Maximum Information Coverage and Monitoring Path Planning With Unmanned Surface Vehicles Using Deep Reinforcement Learning

Samuel Yanes Luis[1⋆], Daniel Gutiérrez Reina[2], and Sergio Toral[3]

University of Seville, Sevilla, Spain {syanes[1],dgutierrezreina[2],storal[3]}@us.es

Abstract. Manual monitoring large water reservoirs is a complex and high-cost task that requires many human resources. By using Autonomous Surface Vehicles, informative missions for modeling and supervising can be performed efficiently. Given a model of the uncertainty of the measurements, the minimization of entropy is proven to be a suitable criterion to find a surrogate model of the contamination map, also with complete coverage pathplanning. This work uses Proximal Policy Optimization, a Deep Reinforcement Learning algorithm, to find a suitable policy that solves this maximum information coverage path planning, whereas the obstacles are avoided. The results show that the proposed framework outperforms other methods in the literature by 32% in entropy minimization and by 63% in model accuracy.

Keywords: Deep Reinforcement Learning · Informative Path Planning · Autonomous Surface Vehicles.

1 Introduction

More than 80% of human activities wastewater is discharged into rivers and seas without prior treatment, making the task of monitoring hydrological resources essential for the sustainability of the planet and developing populations [1]. Manual monitoring of water quality in very large lakes and rivers is a costly task, especially when the environment is polluted and can pose a risk to field operators. Additionally, manual monitoring is inefficient, as manned vessels are heavier and tend to use fossil fuels. On the other hand, the deployment of static sensor networks has certain disadvantages. The measurement points are fixed and cannot change their trajectory depending on the information needs of biologists and authorities.

This work proposes the use of autonomous surface vehicles (ASVs) for dynamic monitoring of biologically at risk scenarios, such as Lake Ypacaraí, where spills and uncontrolled eutrophication have caused a dangerous bloom of blue-green algae colonies. With these electric vehicles equipped with high-quality

⋆ Participation financed by the Ministry of Universities under the FPU-2020 grant of Samuel Yanes Luis and by the Regional Govt. of Andalusia under PAIDI 2020 funds - P18-TP-1520.

sensor modules to measure turbidity, ammonium, dissolved oxygen, etc., multi-objective monitoring missions can be carried out (see Fig. 1). The use of these vehicles, however, requires the development of an intelligent routing module capable of dealing with the environmental monitoring requirements, which are: planning obstacle-free paths, sampling the entire objective surface with minimization of the redundancy, and a surrogate model that truly represents the sampled variables. The informativeness criterion, from the perspective of Information Theory, can be designed using entropy as an indicator of the model uncertainty of the lake variables. Let there be a surrogate model $\hat{f}(X)$, representing the value of a pollution variable in the navigable domain $X \in \mathbb{R}^2$, the reduction of the entropy $H(X|f)$ by incorporating new information implies a decrease of the information clutter, that is, the certainty about the obtained model. The objective of an information planner will then be to maximize the information gain ΔI from an instant t to $t+1$.

Finding the set of samples that reduce the entropy of the process while avoiding the nonnavigable areas makes this problem, called the Maximum Information Coverage Path Planning Problem (MICPP), nonpolynomial hard. Due to the unmanageable dimension of the possibilities in tracing a route within the lake that meets the requirements, it is necessary to use Artificial Intelligence (AI) or Metaheuristic Optimization techniques to find solutions that, in most cases, are suboptimal. In this work, we propose the use of deep reinforcement learning (DRL) algorithms to find solutions to the problem. With an a priori model of the covariance of the variables (kernel) and the use of a robust optimization algorithm of a deep policy (PPO), a framework is proposed that makes it possible to find good solutions to the MICPP problem in a reasonable amount of time. Furthermore, this framework allows one to obtain routes that meet a second objective: detecting possible pollution peaks that are difficult to model. As the treatment of the information is independent of any physical model and only bases its optimality on the entropy decrease, it can be applied in a wide range of cases: gas leakage detection, electromagnetic indoors characterization, patrolling and surveillance, etc.

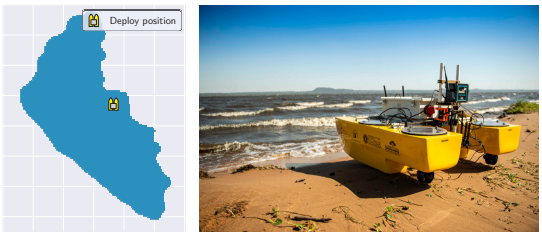


Fig. 1. ASV prototype (right) developed for monitoring the Lake Ypacaraí. The ASV is equipped with a high performance sensor module that is able to measure pH, dissolved oxygen, turbidity, and ammonia concentration. Every mission starts from the deploy zone marked in the map (left).

The main contributions of this work are:

- A Deep Reinforcement Framework to solve the Maximal Distributed Information Coverage problem and global path planning in the Ypacaraí Lake.
- An analysis of the stability, generalization, and performance of the proposed DRL approach and other well-known planning heuristics.

This article is organized as follows. In Section II, an overview of recent advances in patrolling and monitoring using autonomous vehicles is presented. In Section III, the problem is presented formally with its assumptions, and the DRL framework is explained. In Section IV, the simulations are described with an analysis of the results. Finally, the conclusions and future lines are described in Section V.

2 State of the art

The use of Unmanned Surface Vehicles for monitoring aquatic environments is becoming increasingly common thanks to the development of robotics technology and battery autonomy. In [2], the use of inexpensive surface vehicle swarms is proposed for sea border patrolling and environmental monitoring. In [3], an aquatic robot is also used to perform an autonomous bathymetry study in oceanographic contexts. These works use sensor modules and network connectivity to perform their tasks and avoid obstacles.

If we focus on the task of monitoring hydrological resources, we can separate the contributions of the literature into three topics: modeling [4, 5], patrolling [6, 7], and coverage [8, 9]. In the first branch, the ultimate goal is to find a scalar field that represents the environmental variables of water (turbidity, pH, etc.). In works such as [4], the use of Bayesian optimization algorithms together with Gaussian processes is proposed for obtaining faithful models with few samples. In this sense, our proposal includes the use of Gaussian processes as a way to obtain a model of the environment. Other contributions such as [5] explicitly work on multi-objective optimization, while in our proposal the coverage tries to find a path independent of the underlying variables. In the second application, ASVs are typically used to solve the homogeneous [6, 7] and heterogeneous patrolling problem. In the first case, the aim is to find periodic routes that minimize the average visit time of each area, while in the second case, this time is weighted according to a previously specified importance map. These approaches have in common that the map is modeled as a discrete, metric, undirected graph, the resolution of which severely affects the dimension of the problem. In this new framework, on the other hand, the resolution of the map does not affect the complexity of the problem, since the state and action space are continuous. In the third application, vehicles can be used to cover, given a path length, the maximum possible area [8]. This work proposes to maximize the area covered by the vehicle by using a Genetic Algorithm that penalizes passing through already visited areas. In [9], a similar approach is used to adapt to the detection of cyanobacterial blooms in the same Lake Ypacaraí. Both approaches to the full

coverage problem have as a counterpart that the vehicle is limited to perform straight trajectories from shore to shore of the lake. In addition, coverage is considered binary (covered or uncovered). Both limitations are overcome in this work when it is specified, in the first case, that the action space can take any direction at each instant and, in the second case, that the coverage level is measured with a smooth function in the circular surrounding of every sample (radial kernel), since the underlying real scalar field can be considered smooth as well.

Regarding the use of DRL for autonomous vehicle monitoring, there are many examples in the literature that implement deep policy optimization. In the case of [10], a Monte Carlo optimization is used to minimize entropy in the task of monitoring crop fields. The main differences of our proposal with this one are i) the action space used in this work is continuous, which increases the complexity of the solutions, and ii) in [10] terrain constraints are not taken into account. In this work, we consider that the agent must adapt to a real morphology with non-navigable areas, which means not only finding informative routes, but also avoiding obstacles.

3 Methodology

3.1 Sequential decision problem

The MICPP simultaneously addresses two learning problems: i) the vehicle must find navigable routes as long as possible to maximize the sampled area, and ii) these routes must optimally minimize the information entropy given a model of uncertainty of environmental variables. This is a sequential decision problem, since the vehicle must choose, at each instant, which next point p^{t+1} to move to achieve both objectives. This sequential process is defined as a Markov Decision Process (MDP) in which the scenario with state s_t, the agent performs an action a_t according to a policy $\pi(s_t)$ that maps s to a which generates a reward r_t according to a reward function $R(s_t, a_t)$. Within this framework, the ASV must learn to decide at every instant the next point of movement that maximizes the episodic discounted reward. The optimal policy $\pi^*(s_t)$ is:

$$\pi^* = \max_{\pi} \sum_{t=0}^{T} R^{\pi}(s_t, a_t) \tag{1}$$

In this mathematical context, the ASV takes a sample of the water variables, updates its surrogate model and uncertainty, and decides which point to move to until a complete path is completed. In this work, we model the uncertainty of the process using a decreasing radial function (RBF kernel). This kernel function maps the correlation of two samples (x, x') depending on the Eulerian distance d and a scaling parameter l. The further the physical points, the lower the correlation between the samples.

$$K(x, x') = exp\left(-\frac{d(x, x')^2}{2l}\right) \tag{2}$$

This a priori model of the correlation is selected because it fits the only hypothesis that the underlying ground truth is smooth and the data behave like a multivariate Gaussian distribution (MGD). In this way, it is possible to obtain the correlation between a set of observation points X_{obs} by evaluating pairwise every physical point on the map:

$$\Sigma[X] = K(x, x') \quad \forall x, x' \in X \tag{3}$$

Hence $\Sigma[X_{obs}]$, constitutes the covariance matrix of the observation points. When a new measurement is incorporated into the measured set $X[meas]$, the conditioned correlation matrix noted as $\Sigma[X_{obs}|X_{meas}]$ can be calculated as:

$$\Sigma[X_{obs}|X_{meas}] = \Sigma[X_{obs}] - \Sigma[X_{obs}, X_{meas}] \times \Sigma[X_{meas}]^{-1} \times \Sigma[X_{meas}, X_{obs}] \tag{4}$$

This conditioned correlation indicates the uncertainty of the map given an equally distributed set of observation points and the new measuring points as in Fig. 2.

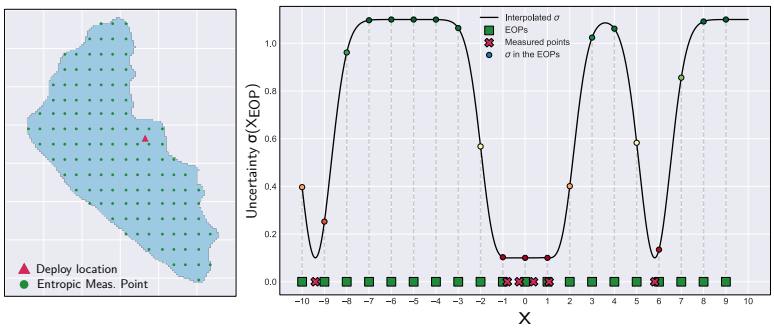


Fig. 2. Process of conditioning. The green squares represent the observation point for the entropy measurement. Sampling (red crosses) diminishes the uncertainty in those points according to Eq. (4), as the conditioned covariance between X_{obs} and X_{meas} decreases.

Then, with the MGD hypothesis, the conditioned entropy $H[X_{obs}|X_{meas}]$ and the information gain are defined as

$$H(\Sigma[X_{obs}|X_{meas}]) = \frac{1}{2}\log(\Sigma[X_{obs}|X_{meas}]) + \frac{D}{2}(1 + \log(2\pi)) \tag{5}$$

$$\Delta I(X^t|a) = H(X^t) - H(X^{t+1}|a) \tag{6}$$

with D as the dimension of $\Sigma[X_{obs}|X_{meas}]$ [11].

Once the path is completed and different samples are obtained at each visited point, this information can be used to obtain a model of contamination of the environment. Since the model is obtained at the end of the collection of maximally distributed points, the regression method is independent of decision-making.

With respect to the action space of the agent A, it is defined to be continuous $A \in [-1, 1]$. An action a_t represents the heading angle $\psi \in [-\pi, \pi]$ of the straight trajectory between the next point and the current one, if possible, with a constant distance between them of d_{meas} (see Fig. 3). The ASV is restricted to a distance budget of 40 km due to the capacity of the battery at a constant speed of 2 m/s. At every reached point, a measurement is taken, and the model is updated. The path is considered complete when the maximum distance $D_{max} = 40kms$ is reached.

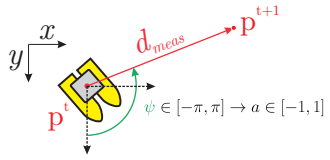


Fig. 3. Movement of the drone. The action space is contained in $[-1, 1]$.

The reward function $R(s, a)$ should sequentially represent the objective to be achieved. It is logical to impose that the reward function should be directly the gain of information $\Delta I(s, a)$. To further introduce the objective that the ASV should have collision-free trajectories, a penalty $c = -1$ is imposed when the next point p^{t+1} chosen by the policy cannot be reached from the current position. To avoid large changes in the reward range, the information gain is bounded between -1 and 1. Thus, the reward function would be:

$$R(s_t, a_t) = \begin{cases} \min(max(\Delta I(s_t, a_t), -1), 1) & \text{if valid.} \\ c & \text{otherwise} \end{cases} \tag{7}$$

3.2 DRL Framework

To learn a policy, the Proximal Policy Optimization (PPO) algorithm is chosen, which is an on-policy reinforcement learning algorithm. In PPO, a deep policy $\pi(s|\theta)$ is updated using the stochastic gradient descent (SGD) approach over a loss function. This loss evaluates the advantage of each action in each state as a function of its reward and weights the gradient step depending on the difference between the policy prior to the optimization step and the new optimized policy. By limiting the distance between them, either by saturating the step by ϵ or penalizing the Kullback-Leibler (KL) distance between them, it is possible to robustly optimize the behavior without incurring instability so easily.

$$L(s,a,\theta_k,\theta) = \min\left[\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)} A^{\pi_{\theta_k}}(s,a)\, \text{clip}\left(\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)}, 1-\epsilon, 1+\epsilon\right) A^{\pi_{\theta_k}}(s,a)\right] \tag{8}$$

In this paper, the double constraint on the policy update has been imposed to make training more robust: both $KL(\pi_{\theta_k}, \pi_\theta)$ and the fraction in (7) are penalized. In the PPO, a two-headed neural network is trained. The first head corresponds to the value function $V(s)$ used in the advantage value $A^{\pi_{\theta_k}}(s,a)$, trained with the accumulated discounted reward R after the n steps of every mission. The other head directly returns the action bounded by the action limit $[-1, 1]$.

The state s_t for the PPO is defined to contain all information available in the problem. In this way, s_t is as a 3-channel image of $H \times W$ pixels. These channels correspond to the following. I) A binary representation of the navigation map $N_{map} \in \mathbb{R}$. II) The standard deviation σ of each physical point on the map of the RBF kernel, evaluated in the form of Eq. 2 and represented with the image shape. III) The discretized path that the ASV has completed so far, represented by pixels with values in [0,1], where 1 corresponds to the actual position and 0 for the starting position. The latter channel merges temporal dependencies to overcome the Markovian assumption of the reward: it must only depend on the current state and the current action.

Regarding the deep policy, the Convolutional Neural Network (CNN) is implemented for estimating the features of the state and processing the next action. The CNN backbone is composed of four 2D convolutional layers of [128, 64, 16, 16] filters, respectively. After the convolutional structure, a dense neural network (DNN) transforms the features into an action $\pi(s)$ and a state-value $V(s)$. The DNN is composed of 3 lineal layers of 256 neurons and 3 layers of 64 neurons in every separate head for the action and state-value. The activation function is the Leaky Rectifier Linear Unit (Leaky-ReLu) to avoid the dying-ReLu effect of low-value gradients. See Fig. 4 for the complete architecture.

4 Simulations and Results

This section discusses the hyperparameterization of the PPO algorithm, the usefulness of the reward function in terms of stability, and finally, the comparison of the results with other algorithms proposed in the literature for similar problems. To compare the model accuracy, a Shekel function [1] with randomly positioned and random size peaks is implemented to represent the scalar contamination field f to cover (see Fig. 5). The simulations were executed using an AMD Ryzen9 3900 (3.8 GHz) with a Nvidia RTX 2080 Super-8GB GPU and 16 GB of RAM.

[1] https://deap.readthedocs.io/en/master/code/benchmarks/shekel.py

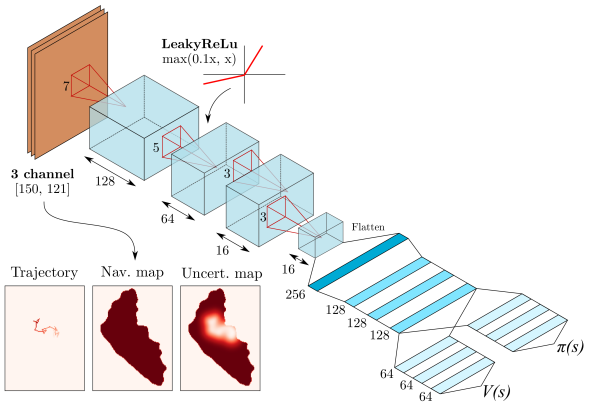


Fig. 4. CNN proposed for the deep policy. It is composed by a convolutional backbone of 4 blocks and a two-head dense block. The activation function is Leaky-ReLu.

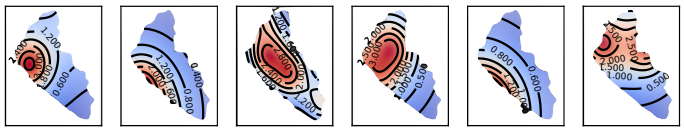


Fig. 5. Six random ground truths representing different scalar variables that the ASV must cover and model in the informative mission. The generator function is the Shekel function with a random number of peaks with random sizes uniformly distributed across the map.

4.1 Evaluation Metrics

For the analysis of the results, different metrics have been used to describe the utility of the learned policy.

- **AER**: The Accumulated Episodic Reward. It represents the decrease in total entropy over the mission time.
- $\boldsymbol{A}^{nr}$: Represents the effective area in (km^2) covered by the ASV. A zone x is considered covered if the uncertainty $\sigma(x)$ is lower than 0.05.
- $\boldsymbol{MSE}$: Mean square error between the surrogate model used and the variables ground truth.
- $\boldsymbol{\xi}$: Peak detection rate. In the presence of random peaks k, that is, local maxima of the ground truth of contamination, the average rate of detected peaks $\xi = \mathbb{E}[k_{detected}/k]$. A local maximum is considered detected when the uncertainty in its location is lower than 0.05.

4.2 Learning results

For the simulations, the hyperparameters in Table 1 were used. The PPO algorithm is less sensible to the selection of hyperparameters, and the following values were selected by trial. Every execution is equivalent to 1×10^6 steps. The learning rate was linearly annealed from 1×10^-4 to 1×10^-5 to enhance the stability of the learning and avoid catastrophic forgetting. In regards to the termination condition of the episode, two different approaches have been tested: apply the penalization and end the episode when colliding, or apply the penalization and let the episode continue from the same previous state.

Hyperparameter	**Value**
Clip value (ϵ)	0.2
Learning Rate (α)	$1 \times 10^{-4} \rightarrow 1 \times 10^{-5}$
max. KL permitted	5
Discount factor (γ)	0.95
Batch size ($\mathcal{B}$)	64
Entropy loss regularization	0.01
Surrogate loss horizon (n_steps)	512
Collision penalty (c)	-1
Kernel lengthscale (l)	10

Table 1. Hyperparameters involved in the PPO algorithm.

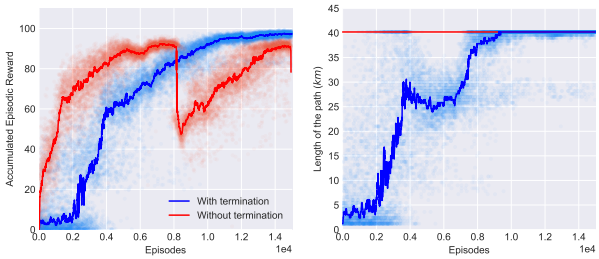


Fig. 6. Learning curves.

In Fig. 6 is shown the learning curves of the optimization process. The optimization shows a robust convergence to a high-reward solution. It is observed that the results of the first variant of the algorithm are more robust to the typical learning instabilities of DRL. The continuous penalty in the case of failure to terminate the episode degenerates the policy to the point of unlearning. The two main tasks can be considered learned: the length of the episode grows monotonically on average throughout the process until the maximum path length of 40 km is reached. This indicates that the deep agent assimilates the terrain constraints. On the other hand, the task of entropy minimization is effectively performed,

since the trajectories have an increasingly informative character as the training proceeds.

It is important to note that the policy resulting from optimization with PPO $\pi(s;\theta)$ is a stochastic policy whose output is a Gaussian distribution $\mathcal{N}(\mu,\sigma)$. As training progresses, $pi(s;\theta)$ becomes more deterministic with a smaller standard deviation. This results in the fact that, despite having learned to synthesize collision-free trajectories, it happens that sampling an action may produce one. To verify the best behavior learned, the performance evaluation must force the action with the highest probability $a_t = median\,[\pi(s,\theta)]$.

4.3 Metric comparison with other methods

To compare the results of the proposed method, five different heuristics from the literature have been implemented: i) a lawn mower (LM) trajectory, ii) GA optimization, iii) a collision-free random search, iv) greedy policy with respect to $\sigma(X)$, and v) greedy policy with respect to the expected improvement of a Gaussian process like in [4]. Ten different scenarios not seen in DRL optimization have served as a validation pool. In relation to MSE metrics, two regression methods have been used with the samples taken: the first is a Gaussian process as in [4], and the second method is a Support Vector Regressor (SVR). Both regression models use an RBF kernel with the same parameters as the one proposed to compute the uncertainty model.

	LM		**GA**		**σ-greedy**		**EI-greedy**		**Random**		**DRL**	
Metric	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
AER	83.10	0	72.75	0	70.31	9.17	57.28	9.19	53.61	10.36	**98.75**	7.08
A^{nr}	62.13	0	70.09	0	57.44	8.03	39.74	6.15	46.56	9.94	**94.91**	8.74
ξ	0.54	0.23	0.52	0.28	0.48	0.29	0.34	0.15	0.42	0.18	**0.72**	0.20
MSE_{gp}	0.08	0.10	0.11	0.13	0.05	0.04	0.18	0.13	0.20	0.22	**0.04**	0.09
MSE_{svr}	0.25	0.12	0.35	0.23	0.30	0.11	0.54	0.67	0.55	0.23	**0.07**	0.03

Table 2. Statistical results of different approaches for the coverage and informative coverage in the validation scenarios-

In Table 2, the comparative results are presented. It can be seen that the proposed algorithm is able to realize much more spatially distributed trajectories for entropy minimization. The entropy minimization factor reflected in the AER and the effective coverage area is significantly higher in the proposed algorithm (32% higher on average with the other approaches). It is logical to think that these two metrics are closely related since decreasing entropy leads to visiting uncovered areas and vice versa. In 7, the statistical results of the execution of the

different approaches in the validation set are depicted. The proposed approach not only obtains better results but also faster within a mission time objective. This is related to the detection rate of singular events xi. Deep policy, as a direct consequence of generating a highly distributed path with low redundancy, achieves 33% more detections than the best-positioned algorithm (LM). The LM algorithm generates a very intensive path, which achieves good homogeneous coverage, but too intensive due to monitoring redundancy.

In the MSE using different regression methods, we obtain very good values with the PPO algorithm, with an improvement of up to 67% using a GPR. This metric is greatly affected by information redundancy: visiting already covered areas does not provide extra information, so the proposed algorithm, through entropy reduction, manages to clearly beat any other informative trajectory for a wide range of possible ground truths. The fact that this is the case whether using a GP or an SVR also indicates that entropy is a good criterion for point selection in scalar-field regression.

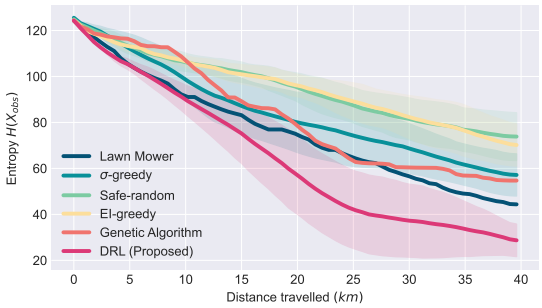


Fig. 7. Process entropy $H(X_{obs})$ over a 40 km monitoring mission.

5 Conclusions

In this paper, a new framework based on Deep Reinforcement Learning has been proposed for minimizing entropy in environmental scenarios using autonomous vehicles. Training by the PPO algorithm using convolutional networks and a graphical formulation of the state results in informative paths with high information utility. The proposal is capable of beating other algorithms and heuristics in maximizing information gain, which leads to improvements in other related metrics more or less directly related to entropy: area covered, location of singular events, and decreasing error in an arbitrary regression model. The latter is of interest because the algorithm does not depend on a particular type of model to work, but rather the improvement follows as a consequence of the information criterion. Furthermore, this approach raises the possibility of training on scenarios with arbitrary boundary conditions and simultaneously solving the task of

obstacle avoidance and informational patrolling. It is proposed that, in future lines of work, a temporal factor can be included in the coverage for the entropy to be reduced on the time axis also for nonstationary environments. Furthermore, a next step is the application of different importance criteria, the decrease of entropy: a non-homogeneous informative coverage that emphasizes areas of high contamination.

References

1. U. N. Organization, "United Nations Organization. Objective 6", 2021. [Online]. Available: www.un.org/sustainabledevelopment/es/water-and-sanitation/. [Accessed 12 November 2021].
2. F. J. Velez et al., "Wireless Sensor and Networking Technologies for Swarms of Aquatic Surface Drones," 2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall), 2015, pp. 1-2, doi: 10.1109/VTCFall.2015.7391193.
3. H. Ferreira et al., "Autonomous bathymetry for risk assessment with ROAZ robotic surface vehicle," OCEANS 2009-EUROPE, 2009, pp. 1-6, doi: 10.1109/OCEANSE.2009.5278235.
4. Peralta Samaniego, F., Gutierrez-Reina, D., Toral Marin, S. L., Arzamendia, M., and Gregor, D. O. (2021). A Bayesian Optimization Approach for Water Resources Monitoring through an Autonomous Surface Vehicle: The Ypacarai Lake Case Study. IEEE Access, 9, 9163–9179. https://doi.org/10.1109/ACCESS.2021.3050934
5. Samaniego, F. P., Reina, D. G., Marin, S. L. T., Arzamendia, M., and Gregor, D. O. (2021). A Bayesian Optimization Approach for Multi-Function Estimation for Environmental Monitoring Using an Autonomous Surface Vehicle: Ypacarai Lake Case Study. Electronics (Switzerland), 10(963), 9163–9179. https://doi.org/10.1109/ACCESS.2021.3050934
6. Yanes Luis, S., Reina, D. G., and Marin, S. L. T. (2020). A Deep Reinforcement Learning Approach for the Patrolling Problem of Water Resources Through Autonomous Surface Vehicles: The Ypacarai Lake Case. IEEE Access, 8, 204076–204093. https://doi.org/10.1109/access.2020.3036938
7. Yanes Luis, S., Reina, D. G., and Marin, S. L. T. (2021). A Multiagent Deep Reinforcement Learning Approach for Path Planning in Autonomous Surface Vehicles: The Ypacaraí Lake Patrolling Case. IEEE Access, 9, 17084–17099. https://doi.org/10.1109/access.2021.3053348
8. Arzamendia, M., Gregor, D., Reina, D. G., and Toral, S. L. (2019). An evolutionary approach to constrained path planning of an autonomous surface vehicle for maximizing the covered area of Ypacarai Lake. Soft Computing, 23(5), 1723–1734. https://doi.org/10.1007/s00500-017-2895-x
9. Arzamendia, M., Gutierrez, D., Toral, and others (2019). Intelligent Online Learning Strategy for an Autonomous Surface Vehicle in Lake Environments Using Evolutionary Computation, in IEEE Intelligent Transportation Systems Magazine, vol. 11, no. 4, pp. 110-125. doi: 10.1109/MITS.2019.2939109.
10. Rückin, J., Jin, L., and Popović, M. (2021). Adaptive Informative Path Planning Using Deep Reinforcement Learning for UAV-based Active Sensing. 1–7. Retrieved from http://arxiv.org/abs/2109.13570
11. Rasmussen, C. E., and Williams, C.K.I..Gaussian Processes for Machine Learning. The Mit Press, 2006, pp. 202-210 ISBN026218253X

Deep learning based method for automatic toll collection. Application to the french case

Amir Nakib[1][/0000−0001−9620−9324]

Université Paris Est Créteil, Laboratoire LISSI, Vitry sur Seine, France
nakib@u-pec.fr

Abstract. In this work, a real world problem of the vehicle type classification for Automatic Toll Collection (ATC) is considered. This problem is very challenging because any loss of accuracy even of the order of 1% quickly turns into a significant economic loss. To deal with such problem, many companies currently use optical sensors (OS) and human observers to correct the classification errors. Herein, a novel vehicle classification method is proposed. It consists in regularizing the problem using one camera to obtain vehicle class probabilities using a set of Convolutional Neural Networks (CNN), then, uses the Gradient Boosting based classifier to fuse the continuous class probabilities with the discrete class labels obtained from OS. The method is evaluated on a real world dataset collected from the toll collection points of the VINCI Autoroutes French network. Results show that it performs significantly better than the existing ATC system and, hence will vastly reduce the workload of human operators.

Keywords: Deep learning · computer vision · Intelligent transportation.

1 Introduction

Automatic toll collection (ATC) is a practical use case in computer vision, and machine lerarning. Although the ATC systems are already deployed in many countries [18], however, human efforts are yet very much necessary to manually correct the misclassifications because of economic consequence of any loss of the classification system.

Recently, the progress in high performance computing, such as cloud computing with graphics processing units (GPU), coupled with the advances of machine learning algorithms, such as Convolutional Neural Networks (CNN) [14], accelerates the development of a number of computer vision based solutions for AVR [3,17,20,8,27,1,31,33,5,29,32,10].

This work aims to improve an existing ATC system that uses OS to classify vehicles for french classification case.

The current system is based on an optical sensor (OS) that makes misclassifications due several reasons: sensor noise, measurements proximity of inter-class vehicles, and additionally attached items with vehicles.

While an obvious solution is to update the OS itself, an alternative is to exploit additional sources of information. Existing ATC setup (see Sect. 2) captures images for the human operators to manually correct the misclassifications. This gives an opportunity to exploit the images and to develop a computer vision based vehicle type classifier using recent approach based on CNN [7]. Furthermore, this classifier has to be integrated with the OS to further enhance the overall performance. The above points motivate to develop a novel system which exploits information from both OS and camera by combining results from the OS, and image classifiers.

To do so, a challenging dataset is collected, where vehicles are categorized based on their physical measures that decreases the inter-class and increases the intra-class variations. Besides, it includes a large variations of the captured images with different conditions, *e.g.,* illumination, occlusion, poses, localization, multi-vehicle presence, etc., for instance see Fig. 1. This enhances the visual intra-class variations (*e.g.,* see Fig. 2), and makes the vehicle classification a significantly challenging problem.

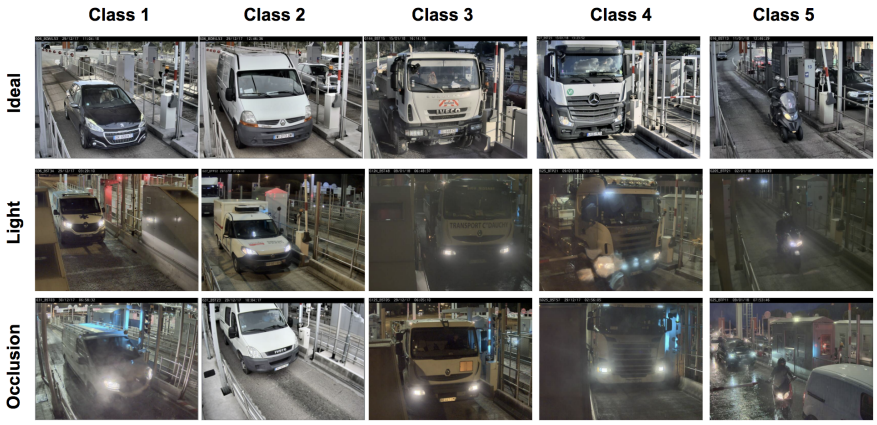


Fig. 1: Examples from different classes at different conditions.

CNN based [7] methods become the *de facto standard* for image-based object recognition [23]. It has been vastly adopted by the recent image-based[1] vehicle classifiers [1,31,33,11,5,29,32,10]. Most of them applies the vehicle *detection followed by classification* approach, which has several drawbacks: (a) dependency on the detectors' performance; (b) hard to determine the true class when multiple vehicles are present, and (c) increase of the computation time. Considering these, this research uses the *detection-free and holistic-scene* based approach for the *CNN based* classifiers.

[1] Vehicle classification has been performed by different sources of information, where *image* is one of the important source (see Sect. **??** for others).

Fig. 2: Intra-class (*4*) variations due to pose and perspectives.

The proposed *CNN based* classifiers achieves significant improvement as a stand-alone classifier. However it has some limitations, e.g., a frontal view image often causes difficulties to distinguish classes 3 and 4, which can be well classified with the OS. This indicates that a robust method for the ATC problem can be developed by efficiently combining the image based classifiers with the OS decisions. Therefore, a novel vehicle type recognition method is proposed, which performs the ensemble approach at two stages/layers:

1. ***1st layer:*** combines the outputs from two different types of classifiers: (a) OS based and (b) CNN [14] based. It provides a concatenated vector (of decision and uncertainty) as an input to the next ensemble layer.
2. ***2nd layer:*** combines the classifiers decisions obtained by training on different weighted sets of the data (output from *feature layer*). The Gradient Boosting [6,4] (GB) method is applied to perform this task.

The proposed method is evaluated on the collected dataset and compared with the existing system. Results indicate that it significantly outperforms the existing system with a large (99.03% compared to 52.77%) margin, and hence alleviates the necessity to employ the vast amount of human efforts. Besides, comparison with a set of CNN based methods shows that it performs better than the *state-of-the-art* approaches.

The contributions of this research can be summarized as follows: (a) introduce a challenging and practical *vehicle classification* use case in the context of ATC; (b) propose a novel vehicle classification method; (c) propose a modification of VGG-16 CNN architecture which significantly ($\approx$ 9 times) reduces its complexity; (d) achieved very high accuracy and significantly outperform the existing system (e) provide interesting visual explanations about a CNN classifier from both *vision* (using GradCam [24]) and *learning* (using t-SNE [19]) perspectives and (f) provide an in-depth analysis to discover the remaining challenges and explore the future works.

This work extends our recent work [26] by incorporating: (a) additional study of the related work; (b) newer contributions on the method by changing from

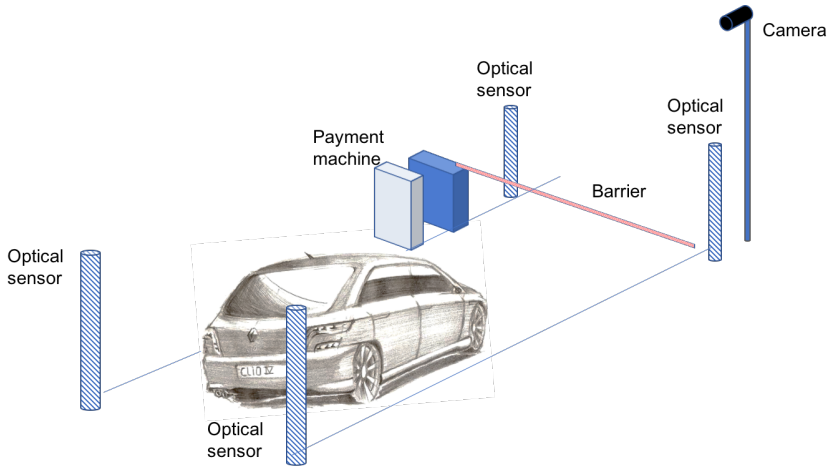


Fig. 3: Existing toll collection set-up.

CNN+OS to *CNN1+CNN2+OS*; (c) extensive experiments and evaluation with possible alternatives methods from the *state-of-the-art* and (d) enhanced discussions with different visualization strategies.

2 Problem formulation

This paper considers the ATC problem within an existing pay tolls setup installed throughout the motorways owned by the *VINCI Autoroutes* company. This problem requires classifying the vehicles into five distinct classes based on certain specifications and physical measurements, such as height, weights and number of axles. The amount of toll payment is determined based on the *detected* class[2] type. Fig. 1(a) illustrates examples from different classes, which are primarily distinguished as follows:

- Class 1: light vehicles; height less than 2 meters.
- Class 2: intermediate vehicles; height between 2 and 3 meters.
- Class 3: heavy vehicles; height over 3 meters and have 2 axles.
- Class 4: very heavy vehicles; height over 3 meters and have more than 2 axles.
- Class 5: motorbikes, side cars and trikes.

The pay tolls are equipped with several OS, cameras and inductive loops, see Fig. 3 for an illustration. Existing system uses the decisions from these OS to classify the vehicle type. The camera is used to capture image/video, which is

[2]More details of the class types specification are available at *https://www.vinci-autoroutes.com/fr/classes-vehicules*.

later used by the human operators to verify the misclassifications. The inductive loops are used for vehicle detection on the toll in order to trigger the photo capture.

The OS is an apparatus disposed at the entrance (called ***pre-OS***) and the exit (called ***post-OS***) of the toll collection lanes and used to measure the height and the number of axles of the vehicles. Usually it provides the class label as the values: *1,2,3,4 and 5*. However, occasionally it provides *0* or *9* to indicate a missing or inconsistent detection.

The vehicle dataset is collected from the cameras located at the pay tolls. It consists of total 73,638 images: 44,437 of class 1, 8073 of class 2, 11,466 of class 3, 3,262 of class 4 and 6,400 of class 5. Therefore, the samples for different classes are distributed non-uniformly, where class 1 has much larger number of samples compared to others. Fig. 1 illustrates several examples of the images from the dataset.

3 Proposed Solution

Our method exploits the existing setup (see Fig. 3) and take the data from both OS and the camera. While the OS directly provides the class label decision, the camera provides the color image. It adopts the CNN [14] based classification strategy to determine the vehicle type from the input color images. Next, it fuses two different categories of classifiers output: (a) the continuous class probabilities from the CNN classifier and (b) the discrete class labels obtained from two (pre and post) OS. The fusion is accomplished using the Gradient Boosting [6] classifier to obtain the vehicle class.

The basic architectural ideas of a CNN [14] consist of the *Convolution* and *Pooling* operations.

The proposed method exploits two different CNN models based architectures. Then, the Gradient Boosting [6] which is a popular heterogeneous data classification method constructs a single strong predictor by iteratively combining the weaker predictors. This combination is achieved by a greedy procedure, where the gradient descent is applied in the function space.

Let $(\Gamma_i^t)^{t=1,2} = \left(\gamma_{ij}^t\right)_{j=1,2,3,4,5}^{t=1,2}$ be the continuous class probabilities obtained from the CNN-t models, $S_i = \left(s_i^{pre}, s_i^{post}\right)$ be the discrete decision labels obtained from the pre-OS and post-OS and y_i denote the true class label. Now, let $\mathbf{x}_i = \left(\Gamma_i^1, \Gamma_i^2, S_i\right) = \left(\gamma_{i1}^1, \ldots \gamma_{i5}^1, \gamma_{i1}^2, \ldots \gamma_{i5}^2, s_i^{post}, s_i^{post}\right)$ be the concatenated feature vector obtained by combining Γ_i^t and S_i. Therefore, $\mathbf{x}_i$ represents the outcome of the ensemble (here by concatanation) applied at the *1st layer* of the proposed method. Next, the GB method is used to accomplish the desired ensemble task at the *2nd layer*. The goal of GB method is to find an approximation of a function $F(\mathbf{x})$ which minimizes the multi-class classification loss $\mathcal{L}_{Multiclass}\left(y_i, F\left(\mathbf{x_i}\right)\right)$ as:

$$\mathcal{L}_{Multiclass} = \frac{\sum_{i=1}^{N} \omega_i log\left(\frac{e^{a_{iy_i}}}{\sum_{j=0}^{M-1} e^{a_{ij}}}\right)}{\sum_{i=1}^{N} \omega_i} \tag{1}$$

where ω_i is the weight, a_i is the value of the target function for the i^{th} sample. The proposed method uses the CatBoost algorithm [4] to perform classification with the GB method. It is chosen because of its efficiency to fuse multiple categories of data, particularly the discrete categorical data.

4 Results and Discussion

First, the CNN models are trained to obtain the class probabilities for each image. Then, the GB method is trained, and subsequently used to get the final class label. The training dataset is selected by randomly splitting the collected dataset into train/validation/test sets with 70%/15%/15% proportion respectively. This split provides 51.5K samples for the training-set and 11K samples for both test and validation sets. Distributions of the samples-per-class on all sets have similarity with the distribution of the entire dataset.

The images from the training set are used to optimize the CNN models parameters. The CNN models are initialized with the parameters learned for the ImageNet [23] object classification by the same model. The weighted/balanced Softmax loss [15,7] is applied as the optimization objective to address the *class imbalance distribution*. These weights correspond to the inverse of class volume. The *L2* regularization is applied on the CNN weights. The learning rate is set to 0.001 and the mini-batch size is set to 100. Data augmentation is applied by horizontally flipping the images. The Stochastic Gradient Descent (SGD) [14] method is used for optimization, which is chosen empirically.

The training data for the GB method is obtained by concatenating the outputs of the CNN Softmax layer and the one-hot encoded values from OS. The CatBoost [4] classifier is used with the depth set to 6, learning rate set to 0.03 and the maximum number of iterations set to 500.

This section evaluates the proposed approach on the test set and compare it with the competitive methods. The classification accuracy is used for the comparison and the precision measure is used for an in-depth class-wise analysis. Additionally, the running time is measured to evaluate time complexity.

This research considers a novel vehicle type recognition use case for ATC. Unfortunately, no existing benchmark is available, except the OS, which are components of proposed method. Therefore, in order to perform a competitive evaluation several alternative methods (which could be applied for this task) are considered as follows:

- **Car type Classification by Huttunen *et. al.* [11] (CCH)**: used the AlexNet [13] for detection-free and holistic scene based car classification.
- **Object of interest classification (*OIC*)**: existing VMMR [3,?,?,?,?,?,?,?,?,?,?,?] methods mostly follow this approach. In order to compare with this type of methods, a competitive method is developed as follows: (a) crop the object of interest from the database images using the RetinaNet[3] [15] object detector, pre-trained on the COCO dataset [16]; (b) apply an empirical rule to

[3]Other object detectors, such as Faster R-CNN [22] and YOLOv2 [21] were explored and RetinaNet [15] is chosen based on its performance.

choose the vehicle of interest in case of multiple detection and (c) train a CNN model on the cropped images. Sect. 4.1 provides additional details.
– **Single components of the proposed method**: PRE-OS, POST-OS, CNN-1 and CNN-2.

Table 1: Comparison among the competitive methods using the accuracy (in %) and computation time (in milliseconds).

Method	**Acc (%)**	**Time (ms)**
CCH	94.77	7
OIC	94.84	387
Fusion of classifiers (ours)	**99.03**	63
CNN-1	95.71	30
CNN-2	95.36	44
PRE-OS	0.10	NA
POST-OS	52.77	NA

The test set accuracies of these methods are reported in Table 1, which shows that the proposed method provides the best result (99.03%). Moreover, it not only outperforms the existing deployed solution (52.77%) significantly, but also provides reasonably better results than the alternative CNN-based solutions, *i.e.* CCH (94.77%) and OIC (94.84%). Indeed, the large performance gap ($\approx$ 4%) with the stand-alone CNN-based methods reveals the effectiveness of the proposed ensemble approach. Note that the components of the ensemble are chosen empirically by exploring numerous CNN combinations (see Sect. 4.1 for futher details). The individual CNN components of the proposed method performs better than the competitive methods. The comparison with the CCH method justifies the choice of the CNN models. Moreover, the comparison with the OIC method justifies the choice of the *detection-free* classification approach.

Besides accuracy, these methods are compared with the computation time, that is measured on the NVIDIA T4 GPU machine with 16 GB of GPU-memory. The right-most column of Table 1 indicates that the proposed appoach is executed within a reasonable computation time and hence is well acceptable for the given ATC task. The OIC approach is the most expensive. Comparison of CCH, CNN-1 and CNN-2 shows that the computation time is related to the complexity of the CNN models. Note that, the proposed method runs two CNNs in parallel, which reduces the time ($\approx$ 10ms).

Next, the chosen ensemble method, *i.e.* CatBoost [4] is briefly evaluated by comparing with several commonly used classifiers, such as RF, SVM, MLP and XGBoost[25]. Note that, the ensemble based on scores averaging is not applicable for the heterogenious outputs obtained from the CNNs and OS. The test-set accuracies reported in the Table 2 show that the CatBoost method provides the best result.

Table 2: Comparison of the different classification methods to fuse the outputs from the CNNs and OS.

Method	**RF**	**SVM**	**MLP**	**XGBoost**	**CatBoost**
Acc (%)	98.82	98.90	98.95	98.97	**99.03**

Next, the accuracy and precision for each vehicle category is studied to analyze the in-depth characteristics of the proposed method and its components. Table 3 presents the per-class accuracy, from which the observations are: (a) the proposed method provides the best accuracy for classes 1, 2, 3 and 5 and (b) it is 1.63% less accurate than the post-OS for class 4. The performance of pre-OS is very poor[4]. The post-OS performs best on class 4, well on classes 2 and 5 and poor on class 3 and 1. Indeed, its performance on class 3 and 4 indicates that it is biased towards the class 4. The accuracies of the CNN models show that CNN-1 is better for classes 1,2,3 and 5 and CNN-2 (Inception) is better for class 4. Therefore, the classification strengths of each model for different classes justify their integration within the proposed method.

Table 4 provides the per-class precision measures, which show that the proposed method gives best precision for all classes. Indeed, from the business perspective this measure is the most suitable metric to evaluate the trustworthiness of a method. Therefore, the high precisions indicate that the predictions from the proposed method are highly reliable.

Table 3: ***Accuracy*** (%) analysis for each vehicle category.

	1	***2***	***3***	***4***	***5***	***Total***
Proposed	**99.92**	**97.11**	**97.62**	94.91	**99.90**	**99.03**
PRE-OS	0.03	0.25	0.29	0.00	0.10	0.10
POST-OS	45.27	86.07	29.21	**96.54**	82.47	52.77
CNN-1	98.21	88.62	90.65	81.47	99.59	95.36
CNN-2	98.92	90.11	92.74	69.45	99.27	95.71

Next, in Table 5 the confusion matrix of the proposed method is analyzed to gain further insights on the classification performance. The observations are:

- For class 1 it produces 0.08% error, which are misclassified as class 2 (0.03%) and class 3 (0.05%).
- For class 2 it produces 2.9% error, which are misclassified as class 1 (1.15%), 3 (1.65%) and class 4 (0.08%).

[4]This is due of the source (customer-care centers) of the dataset where the majority of the images correspond to low confidence pre-OS data.

Table 4: ***Precision*** analysis for each vehicle category.

	1	***2***	***3***	***4***	***5***
Proposed	**99.66**	**98.66**	**97.22**	**95.88**	**100.00**
PRE-OS	0.10	0.05	0.74	0.00	0.08
POST-OS	95.45	17.90	77.27	92.40	95.10
CNN-1	98.97	86.83	90.08	78.59	99.90
CNN-2	98.65	88.43	89.37	87.66	99.69

Table 5: Confusion matrix computed from the classification results of the proposed method.

True	**Predicted class**						
class	***1***	***2***	***3***	***4***	***5***	***All***	***Recall***
1	6661	2	3	0	0	6666	99.92
2	14	1178	20	1	0	1213	97.11
3	8	14	1681	19	0	1722	97.62
4	0	0	25	466	0	491	94.91
5	1	0	0	0	963	964	99.90
All	6684	1194	1729	486	963	11056	
Prec.	99.66	98.66	97.22	95.88	100.00		

- For class 3 it produces 2.4% error, which are misclassified as class 1 (0.46%), class 2 (0.81%) and class 4 (1.10%).
- For class 4 it produces 5.1% error, which are misclassified as class 3.
- For class 5 it produces 0.1% error, which are misclassified as class 1.

The results in Table 1 show that the proposed ensemble approach provides significant (3.32%) improvement compared to its best individual classifier. This encourages to identify the important features to better understand the contribution of the fusion components. Besides, the PRE-OS = 2 and POST-OS = 4 have been identified as the most important OS features. These provides sufficient evidence to realize that the OS inputs are crucial to identify certain vehicle properties (*e.g.* the number of axles) which are difficult to infer from the images.

There are certain scenarios in which the OS are known to perform relatively poorly for particular classes:

- Vehicles from class 1 may be misclassified as class 2 if it has something on its roof (*e.g.* lights or luggage carrier).
- Vehicles from classes 2 and 5 are often confused and require an operator to correct the misclassifications.
- Vehicles from class 1 with a trailer attached to them should be classified as class 2 by classification rules, but are usually classified as class 1 by the OS.

On the other hand, the image could be inadequate for correct classification. For example, the *same* truck can be classified as class 3 or 4 based on the presence of a trailer behind it, which can be invisible in the image. Likewise, the lack of visibility (due to occlusion or frontal view image only) of the the number of axles in the image causes the OS to become the only reliable source for correct classification. Therefore, the combination of image-based class predictions and the OS labels should outperform the individual components.

4.1 Performance analysis

This section provides more details about several aspects of the proposed method, possible alternatives of the given problem and finally identifies the limitations and future scopes.

Selection of the CNN models In order to select the appropriate components for the proposed ensemble, several CNN models have been explored: VGG-16 [28], Inception [30], AlexNet [13], ResNet50 [9], DenseNet [12] and Xception [2]. These models were trained with the same specifications described previously.

Vision-based visual understanding of the performance can be accomplished with the Gradient-weighted Class Activation Mapping (Grad-CAM) [24] technique. It localizes the attention-map or important regions in the image, which is exploited by the CNN model for classification. Fig. 4 illustrates several examples, where the success/failure of CNN-1 can be explained by the ability to focus on the relevant part of the image for the vehicle-class of interest.

Learning based visual discrimination can be realized with the t-distributed Stochastic Neighbor Embedding (t-SNE) [28] method. A subsample (50%) of the validation set is used to project the 5-dimensional CNN-1 classification scores into the 2-dimensional t-SNE output. Fig. 5 provides the illustration and exhibits the overlaps among several classes: 1-2, 2-3 and 3-4. Indeed, for certain vehicles it is difficult to discriminate the closer classes (*e.g.,* 1-2 and 2-3) when the height measurement acts as the most prominant measure instead of their visual appearance. Likewise, the number of axles is significant to discriminate among the classes 3 and 4. In many cases this feature is partially or fully occluded in the image, which leads to misclassification by CNN-1. These image based limitations is difficult to overcome within the existing paytoll setup and hence left the only choice to use additional data from different sources, such as the OS.

Object of interest classification This paper adopts the *holistic scene based object classification* strategy. In this context, the true class is defined by the class of the *vehicle of interest* when multiple objects appear in the image, see Fig. **??** for an example. However, the common approach (followed by the most VMMR methods) is to apply *vehicle detection followed by classification*. This sub-section provides additional analysis on this. Note that, a problem is encountered with this approach when the object detector fails, *i.e.,* no object of interest is detected

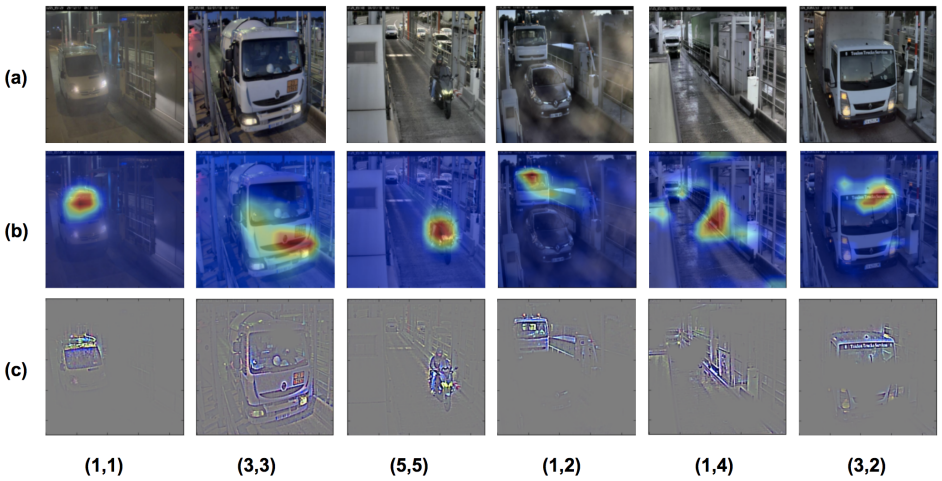


Fig. 4: Illustration of the correctly and incorrectly classified vehicle images. Within each parentheses (t, p), the first value t indicates the *true class label* and the second value p means the *predicted class label*. In (a) the resized image is given; in (b) Grad-CAM illustration; in (c) Guided Grad-CAM.

in the images. In such case, two possibilities are explored: (a) *strategy A*: discard the images and (b) *strategy B:* consider the whole image as the vehicle of interest. The CNN model is used to train and classify with the above strategies and the proposed *holistic scene based* classifier is considered as a benchmark for comparison. Table 6 provides the test set accuracies and shows the best result ($\approx 1\%$ better than the nearest one) is achieved by the proposed *holistic scene based classifier*. For strategy A, two results are obtained: the accuracy only on the nondiscarded images, and the accuracy by considering the discarded images as misclassified. It is observed that, compared to Strategy B the accuracy of strategy A is not better even after discarding the images. This indicates that, in case of the failure of the vehicle detector it is better to consider the entire image to represent the vehicle rather than discarding it. This indeed provide additional evidence to further support the *holistic scene* based classification. Moreover, the significant increase of the computational complexity augmented by the detection method should be taken into account. The combination of both of these facts motivated this research to pursue the *detection-free and holistic scene based classification* approach rather than the *detection followed by classification* approach.

Comparison of the existing solution (OS), scene classification and the out-of-the-box solution This subsection considers a simplified and immediate solution (from the business point of view), which does not need to collect data and train models. The RetinaNet [15] model is selected for this purpose. However,

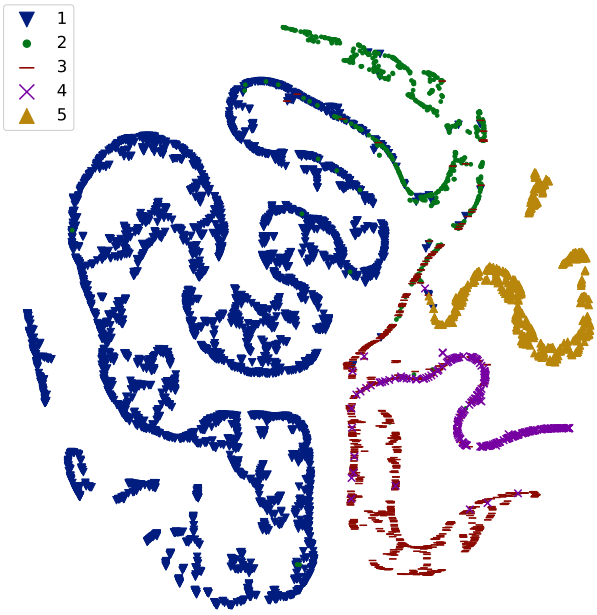


Fig. 5: T-SNE of CNN-1 output scores for the subsample (50%) of the validation set. In (a) colors correspond to the true class of the image, while in (b) to the correct (green) or incorrect (red) classification.

Table 6: Accuracy on the test set for scene classification with CNN-1 model and OIC.

Model	**Acc**
Scene classification	**95.71**
OIC, strategy A (only nondiscarded images)	94.75
OIC, strategy A (all images)	90.26
OIC, strategy B	94.84

it was pretrained on the COCO dataset [16] which does not provide the similar class labels required for the ATC task. Therefore, the number of ATC-classes are reduced to three categories: car, bus/truck, and motorcycle, Table 7 provides further details of the the class labels mapping. Table 8 provides the comparison, where the results are obtained for a subset of the test dataset. This subset (constitutes 94% of the dataset) is constructed by considering the images for which the pre-trained RetinaNet [15] model has detected at least one vehicle

Table 7: Matching classes in the simplified problem version, COCO dataset and VINCI Autoroutes classification guidelines.

Simplified classes	**COCO dataset**	**VINCI Autoroutes**
car	car	class 1, 2
bus/truck	bus, truck	class 3, 4
motorcycle	motorcycle	class 5

Table 8: Analysis of the accuracy (%) with respect to individual vehicle categories.

Class	**PRE-OS**	**POST-OS**	**RetinaNet**	**CNN-1**
car	75.88	97.38	69.92	**99.45**
bus/truck	19.62	44.06	**99.08**	96.47
motorcycle	0.00	83.10	93.43	**99.30**
All	59.10	86.20	77.32	**98.87**

with the score higher than 50%. This approach with RetinaNet achieves good accuracy for classes *bus/truck* and *motorcycle*. However it performs poorly on the *car* class and provides an overall accuracy of 77.32%, which is less than the POST OS performance. The proposed *holistic scene based classification* with CNN-1 model yields the best overall and class-by-class accuracy, except for *bus/truck* class.

Limitations of the proposed method The confusion matrix in Table 5 provides the misclassified cases. Moreover, manual analysis by human observers is performed to visually inspect the reasons for the misclassifications. Fig. 6 illustrates several examples, from which the main causes are identified as: (a) poor light conditions and occlusion, particularly occlusion of the axles and top of the vehicle; (b) class 2 is often misclassified due to the occluded caravan behind it, which causes the vehicle be categorized as class 2 instead of 1; (c) subtle rules in the class estimation, *e.g.,* symbols on the vehicle that transports people with special needs will be categorized as class 1 instead of 2 or 3. These difficulties constitute additional challenges for the proposed method.

The above analyses indicate several weaknesses of the proposed method. Particularly, it exhibits most of the limitations for class 4 which is misclassified as class 3. In future, these errors can be minimized by following several ways: (a) increase training data by collecting more diverse samples, particularly for classes 3 and 4, and the special rules cases; (b) synthesize more data using data augmentation approaches; (c) incorporate efficient pre-processor to tackle the difficult lighting conditions; (d) enhance the efficiency of the classifier by incorporating discriminative loss functions rather than the ordinary softmax loss; and (e) incorporate deeper CNN models. Besides enhancing the efficiency, the proposed method can be evaluated on a variety of similar vehicle classification tasks from different contexts. Moreover, it will be evaluated on the existing car classification

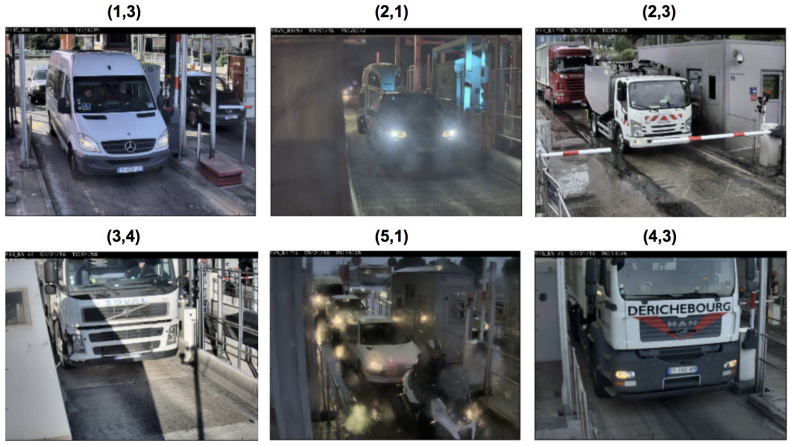


Fig. 6: Illustration of the misclassified vehicle classes. Within each parentheses (t, p), the first value t indicates the *true class label* and the second value p means the *predicted class label*.

datasets [32,17] where the objectives are much different compared to the given ATC based classification.

5 Conclusion

In this paper we proposed a novel vehicle classification method for ATC, which is currently accomplished with several OS and human operators. The proposal consists of a novel multi-classifier fusion-based method, which combines the classification decisions from the OS and the class probabilities estimated from the camera image using two CNN models.

The proposed method significantly outperforms the performance of the existing deployed system by increasing the accuracy from 52.77% to 99.03%.

Additionally, it outperforms several alternative *state-of-the-art* CNN based methods, which could be used for the ATC task.

Obtained results indicate that the proposed approach can be adapted to a large number of vehicle classification problem where the classification decision can be made by fusing the outputs from multiple classifiers. The extensive experiments, analysis and discussions provided in this paper indicate several interesting perspectives and challenges for the future work.

References

1. Biglari, M., Soleimani, A., Hassanpour, H.: A cascaded part-based system for fine-grained vehicle classification. IEEE Transactions on Intelligent Transportation Systems (2017)

2. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017. pp. 1800–1807 (2017)
3. Dlagnekov, L., Belongie, S.J.: Recognizing cars (2005)
4. Dorogush, A.V., Ershov, V., Gulin, A.: Catboost: gradient boosting with categorical features support
5. Fang, J., Zhou, Y., Yu, Y., Du, S.: Fine-grained vehicle model recognition using a coarse-to-fine convolutional neural network architecture. IEEE Transactions on Intelligent Transportation Systems **18**(7), 1782–1792 (2017)
6. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Annals of statistics pp. 1189–1232 (2001)
7. Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., et al.: Recent advances in convolutional neural networks. Pattern Recognition (2017)
8. He, H., Shao, Z., Tan, J.: Recognition of car makes and models from a single traffic-camera image. IEEE Transactions on Intelligent Transportation Systems **16**(6), 3182–3192 (2015)
9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. of IEEE CVPR (2016)
10. Hu, Q., Wang, H., Li, T., Shen, C.: Deep cnns with spatially weighted pooling for fine-grained car recognition. IEEE Transactions on Intelligent Transportation Systems **18**(11), 3147–3156 (2017)
11. Huttunen, H., Yancheshmeh, F.S., Chen, K.: Car type recognition with deep neural networks. In: Intelligent Vehicles Symposium (IV), 2016 IEEE. pp. 1115–1120. IEEE (2016)
12. Iandola, F., Moskewicz, M., Karayev, S., Girshick, R., Darrell, T., Keutzer, K.: Densenet: Implementing efficient convnet descriptor pyramids. arXiv preprint arXiv:1404.1869 (2014)
13. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
14. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
15. Lin, T., Goyal, P., Girshick, R.B., He, K., Dollár, P.: Focal loss for dense object detection. In: IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017. pp. 2999–3007 (2017)
16. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision. pp. 740–755. Springer (2014)
17. Liu, H., Tian, Y., Wang, Y., Pang, L., Huang, T.: Deep relative distance learning: Tell the difference between similar vehicles. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2167–2175 (2016)
18. Loce, R.P., Bernal, E.A., Wu, W., Bala, R.: Computer vision in roadway transportation systems: a survey. Journal of Electronic Imaging **22**(4), 041121 (2013)
19. Maaten, L.v.d., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**(Nov), 2579–2605 (2008)
20. Pearce, G., Pears, N.: Automatic make and model recognition from frontal images of cars. In: Advanced Video and Signal-Based Surveillance (AVSS), 2011 8th IEEE International Conference on. pp. 373–378. IEEE (2011)

21. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proc. of the IEEE conference on computer vision and pattern recognition. pp. 779–788 (2016)
22. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems (NIPS) (2015)
23. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) **115**(3), 211–252 (2015)
24. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: The IEEE International Conference on Computer Vision (ICCV) (Oct 2017)
25. Shalev-Shwartz, S., Ben-David, S.: Understanding machine learning: From theory to algorithms. Cambridge university press (2014)
26. Shvai, N., Meicler, A., Hasnat, A., Machover, E., Maarek, P., Nakib, A.: Ensemble classifiers based classification for automatic vehicle type recognition. In: Proceedings of the IEEE World Congress on Computational Intelligence (2018)
27. Siddiqui, A.J., Mammeri, A., Boukerche, A.: Real-time vehicle make and model recognition based on a bag of surf features. IEEE Transactions on Intelligent Transportation Systems **17**(11), 3205–3219 (2016)
28. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: Proceedings of the International Conference on Learning Representations (2015)
29. Sochor, J., Špaňhel, J., Herout, A.: Boxcars: Improving fine-grained recognition of vehicles using 3-d bounding boxes in traffic surveillance. IEEE Transactions on Intelligent Transportation Systems (2018)
30. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2818–2826 (2016)
31. Wang, J., Zheng, H., Huang, Y., Ding, X.: Vehicle type recognition in surveillance images from labeled web-nature data using deep transfer learning. IEEE Transactions on Intelligent Transportation Systems (2017)
32. Yang, L., Luo, P., Change Loy, C., Tang, X.: A large-scale car dataset for fine-grained categorization and verification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3973–3981 (2015)
33. Yu, S., Wu, Y., Li, W., Song, Z., Zeng, W.: A model for fine-grained vehicle classification based on deep learning. Neurocomputing (2017)

Monitoring Employee and Customer Satisfaction via Social Platforms

Lena Clever[1][0000−0003−0926−6326], Karsten Kraume[1][0000−0003−1762−4170], Christian Grimme[1][0000−0002−8608−8773], and Heike Trautmann[1][0000−0002−9788−8282]

University of Münster, Statistics and Optimization, Münster, Germany
{lena.clever,karsten.kraume,christian.grimme,trautmann}@uni-muenster.de

Keywords: social platforms, data stream mining, employee satisfaction

Leveraging the potential of social platform monitoring Since years, the war for talent is a phenomenon that keeps companies busy: highly skilled employees have become a scarce resource, especially as the COVID pandemic has fueled growth of e(lectronic)-commerce and q(uick)-commerce providers. On top, social platforms such as *indeed, glassdoor* or *kununu* have increased the transparency in the labor market. Employees share their experiences concerning an employer across all stages of the employee life-cycle, which is comparable to the customer life-cycle that is evaluated on platforms such as *google, trustpilot, twitter* and the likes. While customers and employees profit from sharing experiences, of course also companies can derive substantial benefit from social platforms for target-oriented performance optimization. In this paper, we transfer previous work of monitoring *twitter* feeds related to the German Federal Election by means of a sophisticated dashboard into the business context. The core idea is to monitor companies' reputation and appearance on social platforms, which are used by employees for sharing their work experiences and satisfaction level (employee perspective). This will be combined with analyses of the companies' profile on general social media platforms such as *twitter* via stream-mining based topic modeling and sentiment analysis. This integrated approach, currently at a proof-of-concept stage, allows for simultaneously monitoring the customer *and* employee perspective in order to derive recommended actions for optimizing the organization's processes and services. A sophisticated and interactive dashboard will contain informative metrics, statistics, and visualizations from a variety of relevant social platforms and will be filled in an automated, data stream based fashion. As a case-study, we focus on the company `flaschenpost SE`.

Data. Internal surveys are conducted regularly to evaluate the employees' satisfaction. On top, social media channels and employer rating portals are analyzed. For labor marketing, the company is active on channels such as *LinkedIn* and *Xing*. Due to the geographic focus on Germany the portals *kununu, glassdoor, stepstone, indeed* contain relevant data to be analyzed. In addition to quantitative data such as satisfaction indicators based on scales, the platforms also offer qualitative data in the form of comments on various issues. In open text fields, specific textual assessments can be made on aspects of the company,

interaction with employees, the working atmosphere, and job requirements. Data of social media platforms such as *twitter* are not collected at this point in time.

Analytics process. External impact is assessed via quantitative metrics of employer rating portals. Textual data is analysed manually, which is gained from the employer rating portals as well as internal employee surveys and general (social) media. The data available from the above mentioned channels are reviewed daily by an expert of the HR department. On a weekly basis, the manager in charge shares new insights with the respective regional and functional leader. Also, the expert reacts on social media entries.

Related Approaches from Political Social Media Analytics Although the discovery and evaluation of political topics and contexts in social networks primarily aims at identifying manipulative behavior by (possibly) automated or malicious actors [6, 5], the collection of data and identification of issues over time is methodologically related. Access routes for data collection are similar, i.e. platform APIs or crawlers to collect data are used. Also, similar (un)structured data and at the same time analogous data protection requirements can be expected. The analytic approach can certainly also be transferred in many cases. In particular, the data to be collected can be interpreted as a data stream in which topics appear spontaneously and also disappear again [3]. The recognition of topics must therefore be working in near real-time and at the same time be able to evaluate topic importance locally, i.e. depending on the time of observation. The detailed analysis of topics at different points in time plays a central role downstream of the real-time analysis. It is precisely these aspects that various methodological works in the field of political social media analytics focus on: text-based stream clustering techniques [3, 1], two-stage analytics frameworks [2], and their use in the context of monitoring political elections. In this context, the use of a text clustering method for real-time topic discovery and the construction of an information dashboard for the 2021 German federal election are worth highlighting. The dashboard integrated the respective two-phase analytics methodology of real-time topic discovery and downstream detailed analysis to give citizens a transparent picture of the situation regarding election-related topics [2].

A Dashboard for Monitoring Employee and Customer Satisfaction on Social Platforms Within Figure 1, the concept of the interactive dashboard is depicted. The initial idea is, that data is collected via crawlers or specific exports from e.g. *kununu, indeed* or *twitter*. All collected information is stored in an automated fashion within a document comprised data base. Extracted textual data is further analyzed via natural language processing techniques like sentiment analysis. Via a stream clustering approach, discussed topics at different points in time, are extracted and fed into the database again. The storage and initial analysis are done in real-time, so further analytical steps can be applied directly via the interactive dashboard. To enable a structured and detailed view on the data, a mapping layer in terms of an Online Analytical Processing (OLAP) Cube is applied [4]. The mapping combines different data records with

analysis dimensions like time, scores, employee groups, etc. Also, the extracted topics of the stream clustering approach form a dimension of the data matrix. A stakeholder is thus able to use OLAP operations like slicing, dicing or drill downs via the interactive dashboard which shows meaningful metrics, statistics and visualizations derived from the mapping of the OLAP cube. Stakeholder specific exports can be extracted as basis for action recommendations within the organization. Also, an important merit is the possibility to conduct comparisons to business competitors in the market. From the methods perspective, textual data stream clustering algorithms will be further developed and explainability of automated AI tools on semi-structured data will be addressed.

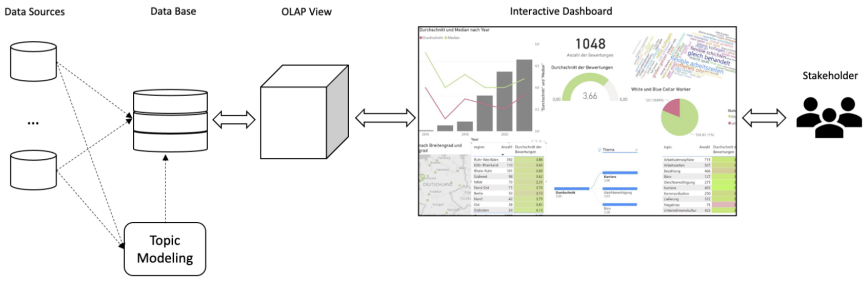


Fig. 1. Concept of interactive dashboard for monitoring social platforms.

Acknowledgements The authors acknowledge support of the European Research Center for Information Systems (ERCIS), the ERCIS Competence Center on Social Media Analytics, and the Topical Program "Algorithmization and Social Interaction".

References

1. Assenmacher, D., Adam, L., Trautmann, H., Grimme, C.: Towards Real-Time and Unsupervised Campaign Detection in Social Media. In: Barták, R., Bell, E. (eds.) Proceedings of the 33rd FLAIRS Conference. pp. 303–306. AAAI Press (2020)
2. Assenmacher, D., Clever, L., Pohl, J.S., Trautmann, H., Grimme, C.: A two-phase framework for detecting manipulation campaigns in social media. In: Meiselwitz, G. (ed.) HCII. pp. 201–214. Springer, Cham (2020)
3. Carnein, M., Assenmacher, D., Trautmann, H.: Stream clustering of chat messages with applications to twitch streams. In: de Cesare, S., Ulrich, F. (eds.) Proceedings of the 36^{th} International Conference on Conceptual Modeling (ER'17). pp. 79–88. Springer, Valencia, Spain (2017). https://doi.org/10.1007/978-3-319-70625-2_8
4. Chamoni, P., Gluchowski, P.: On-Line Analytical Processing (OLAP), pp. 401–444. Gabler Verlag, Wiesbaden (1998). https://doi.org/10.1007/978-3-322-99372-4_13

GRASP-based hybrid search to solve the multi-objective requirements selection problem*

Víctor Pérez-Piqueras[0000−0002−2305−5755], Pablo Bermejo López[0000−0001−7595−910X], and José A. Gámez[0000−0003−1188−1117]

Department of Computing Systems, Intelligent Systems and Data Mining Laboratory (I3A), Universidad de Castilla-La Mancha, Albacete, 02071, Spain
{victor.perezpiqueras,pablo.bermejo,jose.gamez}@uclm.es

Abstract. One of the most important and recurring issues that the development of a software product faces is the requirements selection problem. Addressing this issue is especially crucial if agile methodologies are used. The requirements selection problem, also called Next Release Problem (NRP), seeks to choose a subset of requirements which will be implemented in the next increment of the product. They must maximize clients satisfaction and minimize the cost or effort of implementation. This is a combinatorial optimization problem studied in the area of Search-Based Software Engineering. In this work, the performance of a basic genetic algorithm and a widely used multi-objective genetic algorithm (NSGA-II) have been compared against a multi-objective version of a randomized greedy algorithm (GRASP). The results obtained show that, while NSGA-II is frequently used to solve this problem, faster algorithms, such as GRASP, can return solutions of similar or even better quality using the proper configurations and search techniques. The repository with the code and analysis used in this study is made available to those interested via GitHub.

Keywords: grasp · multi-objective optimization · next release problem · requirements selection · search-based software engineering.

1 Introduction

Software systems are increasing in functionality and complexity over time. This implies that new software projects are potentially more complicated to manage and complete successfully. One of the problematics that can heavily affect the outcome of a project is the planning of a release. In a software project, the product to be delivered is defined by a set of software requirements. These requirements are offered to a group of clients, who will give feedback on which requirements are more important to them. Then, a set of requirements is planned for the release. Selecting the requirements that better fit client interests starts getting complicated when development capacity has to be taken into account.

* This work has been partially funded by the Regional Government (JCCM) and ERDF funds through the project SBPLY/17/180501/000493.

Furthermore, requirements can have dependencies between them. This problem, named requirements selection problem, is very complex and does not have a unique and optimal solution. Two objectives coexist: maximizing the satisfaction of the clients and minimizing the effort of the software developers. Therefore, solutions can range from sets of few requirements with minimal effort and satisfaction, to sets of plenty of requirements, which will imply high client satisfaction but at the cost of a high effort.

Thus, this planning step is critical, especially when applying incremental software development methodologies due to the need to solve the requirements selection problem multiple times, at each iteration. Thus, this problem is candidate to be automated by means of optimization methods. Previous works have studied the applicability of different search techniques, giving preference to evolutionary algorithms, mainly. In our study, we present an algorithm based on the Greedy Randomized Adaptive Search Procedure (GRASP, [7]). We have explored new procedures that allow to improve GRASP performance in the requirements selection problem beyond that of previous studies. The experimentation that we carried out shows that the GRASP metaheuristic can obtain similar results as those of the evolutionary approaches, but reducing drastically its computational cost.

The rest of the paper is structured as follows. In Section 2, a summary of previous works and procedures they applied is made. Section 3 describes our algorithm proposal and defines the solution encoding along with the most important methods and techniques. Then, in Section 4, the evaluation setup is described, along with the algorithms, datasets and methodology used. Section 5 presents and discusses the results of the experimentation. Finally, Section 6 summarizes the conclusions of this study and introduces potential new lines of work for the future.

2 Requirements selection

2.1 Related work

The requirements selection problem is studied in the Search-Based Software Engineering (SBSE) research field, where Software Engineering related problems are tackled by means of search-based optimization algorithms. The first definition of the requirements selection problem was formulated by Bagnall et al. [1]. In their definition of the Next Release Problem (NRP), a subset of requirements has to be selected, having as goal meeting the clients[1] needs, minimizing development effort and maximizing clients satisfaction. In their work, different metaheuristics algorithms, such as simulated annealing, hill climbing and GRASP algorithms were proposed, but all of them combined the objectives of the problem using an aggregate function. The same procedure of single-objective proposals was followed by Greer and Ruhe [9]. They studied the generation of

[1] Although "stakeholder" is a more appropriate term, "client" will be used to keep coherence with previous works present in the literature.

feasible assignments of requirements to increments, taking into account different resources constraints and stakeholders perspectives. Genetic algorithms (GAs) were the optimization technique selected to solve the NRP. Later, Baker et al. [2] demonstrated that metaheuristics techniques could be applied to real-world NRP outperforming expert judgement, using in their study simmulated annealing and greedy algorithms. The works of del Sagrado et al. [5] applied ACO (Ant Colony Optimization). All of these approaches followed a single-objective formulation of the problem, in which the aggregation of the objectives resulted in a biased search.

It was not until the proposal of Zhang et al. [13] that the NRP was formulated as a multi-objective optimization (MOO) problem. This new formulation, Multi-Objective Next Release Problem (MONRP), formally defined in Section 2.2, was based on Pareto dominance. Their proposal tackled each objective separately, exploring the non-dominated solutions. Finkelstein et al. [8] also applied multi-objective optimization considering different measures of fairness. All these studies applied evolutionary algorithms, such as ParetoGA and NSGA-II [4] to solve the MONRP.

Other works that kept exploring evolutionary algorithms to solve the MONRP are those of Durillo et al. [6]. They proposed two GAs, NSGA-II and MOCell (MultiObjective Cellular genetic algorithm), and an evolutionary procedure, PAES (Pareto Archived Evolution Strategy).

2.2 Multi-objective formulation

As mentioned in the introduction, the NRP requires a combinatorial optimization of two objectives. While some studies alleviate this problem by adding an aggregate (single-objective optimization), others tackle the two objectives by using a Pareto front of non-dominated solutions (MOO). Defining the NRP as a multi-objective optimization problem gives the advantage that a single solution to the problem is not sought, but rather a set of non-dominated solutions. In this way, one solution or another from this set can be chosen according to the conditions, situation and restrictions of the software product development. This new formulation of the problem is known as MONRP.

The MONRP can be defined by a set $R = \{r_1, r_2, \ldots, r_n\}$ of n candidate software requirements, which are suggested by a set $C = \{c_1, c_2, \ldots, c_m\}$ of m clients. In addition, a vector of costs or efforts is defined for the requirements in R, denoted $E = \{e_1, e_2, \ldots, e_n\}$, in which each e_i is associated with a requirement r_i . Each client has an associated weight, which measures its importance. Let $W = \{w_1, w_2, \ldots, w_m\}$ be the set of client weights. Moreover, each client gives an importance value to each requirement, depending on the needs and goals that this has with respect to the software product being developed. Thus, the importance that a requirement r_j has for a client c_i is given by a value v_{ij}, in which a zero value represents that the client c_i does not have any interest in the implementation of the requirement r_j. A $m \times n$ matrix is used to hold all the importance values in v_{ij}. The overall satisfaction provided by a requirement r_j is denoted as $S = \{s_1, s_2, \ldots, s_n\}$ and is measured as a weighted sum of

all importance values for all clients. The MONRP consists of finding a decision vector X, that includes the requirements to be implemented for the next software release. X is a subset of R, which contains the requirements that maximize clients satisfaction and minimize development efforts.

3 Proposal

Evolutionary algorithms have been widely applied to solve the MONRP [3,11,13]. Most of the previous studies are based on algorithms such as NSGA-II, ParetoGA or ACO, usually comparing their performance with other algorithms less suited to the problem, e.g. generic versions of genetic or greedy algorithms. However, evolutionary approaches involve a high computational cost, as the algorithms have to generate a population of solutions and evolve each one of the solutions applying many operators and fitness evaluations. For this reason, in this work we have pursued to design an algorithm that can return a set of solutions of similar quality to those obtained by the evolutionary proposals, but reducing the cost of its computation. In this section, it is presented the multi-objective version of a greedy algorithm, along with the solution encoding used. Then, the most relevant procedures of the algorithm and enhancements are presented.

3.1 GPPR: a GRASP algorithm with Pareto front and Path Relinking

GRASP is a multi-start method designed to solve hard combinatorial optimization problems, such as the MONRP. It has been used in its simplest version [11,3] to solve the requirements selection problem.

The basic actions of a canonical GRASP procedure consist of generating solutions iteratively in two phases: a greedy randomized construction and an improvement by local search (see Sections 3.3 and 3.4). These two phases have to be implemented specifically depending on the problem at hand.

We have designed a variant of the GRASP procedure with the goal of solving the MONRP in a hybrid manner, that is, applying both single-objective and multi-objective search methods. The algorithm, named GRASP algorithm with Pareto front and Path Relinking (GPPR), executes a fixed number of iterations, generating at each iteration a set of solutions, instead of only one solution per iteration (which is a different approach from the canonical GRASP that generates one solution per iteration, but in the end works identically). Additionally, we have extended the procedure, updating the Pareto front with the new solutions found after each iteration, and adding a post-improvement procedure known as Path Relinking (see Section 3.5), that will enhance the quality of the Pareto front found by exploring trajectories that lead to new non-dominated solutions. The pseudocode of GPPR is shown in Algorithm 1.

Each one of the operators included in the pseudocode is described in detail in Sections 3.3, 3.4 and 3.5, respectively. As explained previously, GPPR is an algorithm that applies a hybrid approach. It maintains and updates at each

Algorithm 1 GPPR pseudocode

```
procedure GPPR(maxIterations)
    nds ← ∅                                   ▷ empty set of non-dominated solutions
    for i = 0 to maxIterations do
        solutions ← constructSolutions()
        solutions ← localSearch(solutions)
        solutions ← pathRelinking(solutions, nds)
        nds ← updateNDS(solutions)
    end for
    return nds
end procedure
```

iteration a set of non-dominated solutions, returned in the form of a Pareto front at the end of the search. However, it uses an aggregate of the two problem objectives in some phases of the execution (depending on the methods chosen for each phase).

3.2 Solution encoding

Each candidate solution in GPPR is represented by a vector of booleans of length n. Each value of the vector indicates the inclusion or not of a requirement of the set R (see Section 2.2). The satisfaction and effort of each requirement are scaled using a min-max normalization. Each solution is evaluated by means of a *singleScore* value that mixes the scaled satisfaction and effort of the set X of selected requirements in the solution. In this version of the GPPR, we did not model cost restrictions nor interactions between requirements.

3.3 Construction

In this phase a number of solutions are constructed. Their generation can be either randomized or stochastic. We have designed two methods for the construction phase:

- **Uniform.** First, the number x of selected requirements is randomly chosen. Then, x requirements are selected randomly, having each requirement r of the set R of length n a probability $\frac{1}{n}$ of being selected. This construction method works as a random selection of requirements.
- **Stochastic.** The probability of each requirement being selected is proportional to its *singleScore*.

3.4 Local search

This phase is executed after the construction of an initial set of solutions, and it aims to find solutions in the neighbourhood that enhance the former ones. Since GPPR aims to generate solutions fast, it performs a ranking-based forward

search, in which it tries to find and return a neighbour that is better than the initial one. This search method tends to fall into local optima, but it can be corrected increasing the number of executions or applying extra operators after this phase (see Subsection 3.5).

3.5 Path Relinking

One of the adverse characteristics of GRASP is its lack of memory structures. Iterations in GRASP are independent and do not use previous observations. Path Relinking (PR) is a possible solution to address this issue. PR was originally proposed as a way to explore trajectories between elite solutions. In the problem being tackled, elite solutions are the non-dominated solutions. Using one or more elite solutions, trajectories that lead to other elite solutions in the search space are explored, in order to find better solutions. PR applied to GRASP was introduced by Laguna and Martí [10]. It has been used as an intensification scheme, in which the generated solutions of an iteration are relinked to one or more elite solutions, creating a post-optimization phase.

The PR method can be applied either after each iteration, involving a higher computational cost; or at the end of the execution, relinking only the final elite solutions, reducing the effectiveness of this method but speeding up the execution.

In this proposal we have decided to apply PR at each iteration as a third phase, after the local search. The pseudocode is described in Algorithm 2. For each one of the solutions found after the local search, this procedure will try to find a path from each solution to a random elite solution from the set of non-dominated solutions (NDS). This path will help the procedure to find intermediate solutions that can possibly be better than the former ones. For this purpose, each solution in the current set of solutions obtained after the local search is assigned an elite solution from the current NDS. Then, it calculates the Hamming distance of these two solutions. Having the distance value, the procedure finds the bits that are different, that is, the requirements included in one solution that are not in the other. Then, it updates the current solution (flips the bit that returns the highest *singleScore* value) and saves the new path solution in a solution path list, decrementing the distance from the current solution to the elite one. When the distance is zero, the best solution found in the path is appended to a set of best solutions (*bestSols* in Algorithm 2) found by the PR procedure. After finding best path solutions for all the initial solutions, the procedure returns the set of former solutions plus the new solutions found.

4 Evaluation setup

In this section, we present the experimental evaluation. We describe competing approaches used to be compared against our proposal, along with the datasets used to evaluate the algorithms. Our algorithms have been implemented in

Algorithm 2 Path Relinking pseudocode

```
procedure PathRelinking(solutions, nds)
    bestSols ← ∅
    for sol in solutions do
        currSol ← sol                          ▷ Create a copy to be modified
        eliteSol ← getRandomSol(nds)
        distance ← countDistance(currSol, eliteSol)
        pathSols ← ∅
        while distance > 0 do
            diffBits ← findDiffBits(currSol, eliteSol)
            currSol ← flipBestBitSingleScore(diffBits)
            pathSols ← savePath(currSol)
            distance ← distance − 1
        end while
        bestSols ← bestSols ∪ findBestSol(pathSols)
    end for
    return solutions ∪ bestSols
end procedure
```

Python 3.8.8. The source code, experimentation setup and datasets are available at the following repository: https://github.com/UCLM-SIMD/MONRP/tree/ola22.

4.1 Algorithms

To properly compare the effectivity and performance of our proposal, besides GPPR we have included in our experiments the following algorithms: Random search, Single-Objective GA and NSGA-II. The ranges of parameters used in the experimentation for each algorithm are described in Section 4.3, along with their descriptions.

4.2 Datasets

We have tested the performance of the algorithms using a variety of datasets from different sources. Datasets P1 [9] and P2 [11] include 5 clients and 20 requirements, and 5 clients and 100 requirements, respectively.

Due to the privacy policies followed by software development companies, there is a lack of datasets to experiment with. For this reason, we have created synthetically a larger dataset (S3) that includes 100 clients and 140 requirements, in order to evaluate the shift in performance of the algorithms.

4.3 Methodology

We tested a set of configurations for each algorithm and dataset. Each configuration was executed 10 times. For the Single-Objective GA and NSGA-II,

populations were given values among $\{20, 30, 40, 100, 200\}$ and number of generations took values $\{100, 200, 300, 500, 1000, 2000\}$. Crossover probabilities range from $\{0.6, 0.8, 0.85, 0.9\}$. Two mutation schemes were used, *flip1bit* and *flipeachbit*, and mutation probabilites from $\{0, 0.05, 0.1, 0.2, 0.5, 0.7, 1\}$. Both algorithms used a binary tournament selection and a one-point crossover scheme. For the replacement scheme, both Single-Objective GA and NSGA-II applied elitism. The total amount of different hyperparameter configurations executed for each GA and each dataset was 1680. Our GPPR algorithm was tested using a number of iterations from $\{20, 40, 60, 80, 100, 200, 500\}$ and a number of solutions per iteration from $\{20, 50, 100, 200, 300, 500\}$. We tested all combinations of construction methods, local search and PR (including configurations with no local search and no PR methods), which resulted in 1008 different hyperparameter configurations executed for each dataset.

The stop criterion used in other works [13,11,3] is the number of function evaluations, commonly set to 10000. To adapt our experiments to this stop criterion, we restricted the execution of our GAs to: $Pop.\ size \times \#Gens. \leq 10000$; and for the GPPR: $Iterations \times Sols.\ per\ Iteration \leq 10000$.

The GPPR normalizes the satisfaction and effort values, scaling them between 0 and 1. To properly compare its Pareto front solutions against those returned by the GAs, these evolutionary approaches have also used the normalized version of the dataset values. To evaluate the results, we compared the obtained Pareto fronts and a set of quality indicators of the results generated by the algorithms and their efficiency:

- **Hypervolume (HV)**. Denotes the space covered by the set of non-dominated solutions [14]. Pareto fronts with higher HV are preferred.
- **Δ-Spread**. It measures the extent of spread achieved among the obtained solutions [6]. Pareto fronts with lower Δ-Spread are preferred.
- **Spacing**. It measures the uniformity of the distribution of non-dominated solutions [12]. Pareto fronts with greater spacing are preferred.
- **Execution time**. The total time taken by the algorithm to finish its execution. Algorithms with lower execution time are preferred.

Mean values of these metrics have been calculated and compared in a pairwised manner between algorithms using the Wilcoxon rank-sum non-parametric test, which allows to assess whether one of two samples of independent observations tends to have larger values than the other.

5 Results and analysis

5.1 Best configurations

The Single-Objective GA's best hyperparameter configuration includes a population size of 100 individuals, a number of generations of 100 (maximum number to stay under the 10,000 limit) and a $P_c = 0.8$. The mutation operator that showed a better performance was the *flip1bit*. This operator gives a chance of

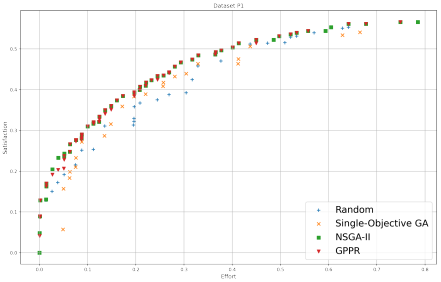


Fig. 1. Pareto front for dataset P1

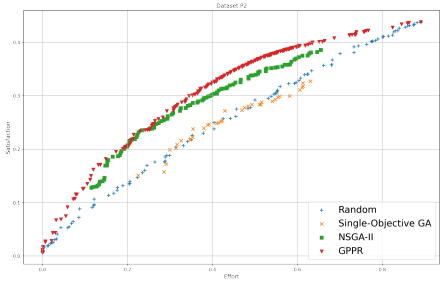


Fig. 2. Pareto front for dataset P2

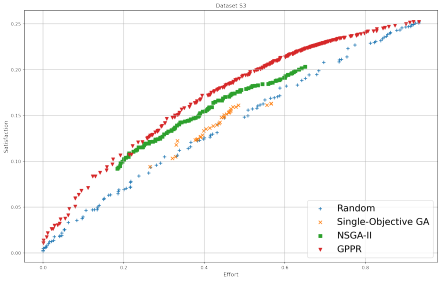


Fig. 3. Pareto front for dataset S3

flipping only one bit of the booleans vector. The best-performing probability is $P_m = 1$, which means that we always mutates one random bit of each individual. That probability is equivalent to using $P_m = \frac{1}{n}$ at gene level, n being the number of genes (scheme used in [6,11]). The best hyperparameter configuration for the NSGA-II used a population size of 100 individuals and 100 generations. The best crossover probability (P_c) was the lowest, 0.6, and the best mutation operator was the *flip1bit*, using a $P_m = 1$. For the GPPR, the ratio between iterations and number of solutions per iteration is less important, as this algorithm does not have memory. Thus, a similar hyperparameter configuration to those of the GAs was used. The construction method that showed a better performance was the *stochastic* one, giving preference to requirements with higher *singleScore*. In all scenarios, hyperparameter configurations with *uniform* construction performed worse.

5.2 Pareto results

Pareto results are shown in Figures 1, 2 and 3. For the sake of space, we omit results of worst-performing hyperparameter configurations.

The Single-Objective GA shows bad performance, being similar to that of the random procedure. This occurs due to the low number of generations set to keep the maximum number of function evaluations. Configuring a number of generations of one or two magnitude orders higher increases the quality of its

Pareto front. Regarding the Pareto front distribution, this GA's aggregation of objectives biases the search, leaving unexplored areas.

The NSGA-II algorithm generates Pareto fronts of better quality: better solutions and more distributed along the search space. As expected, the crowding operator of the algorithm helps exploring the search space. However, as the dataset size increases, its performance decreases significantly. The reason is the limited number of generations, as this algorithm is expected to perform better in larger datasets when compared against other search methods.

Multi-Objective (MO) local search methods do not worsen the solutions, but do not improve them either, as a GPPR without local search is returning similar Pareto fronts. This implies that local search methods that can only explore the neighbourhood of a solution are not able to improve the solutions. Nevertheless, the PR procedure is capable of finding better solutions. In all cases, algorithms applying this methods returned Pareto fronts of higher quality. Regarding the Pareto distribution, as this procedure starts each iteration randomly, it explores the majority of the search space. The most interesting feature of the GPPR is that, while the dataset size grows, its performance is not demeaned. Therefore, unless the search space is much larger, our GPPR proposal can return a Pareto front of acceptable quality very efficiently, while GAs require higher number of iterations, impplying a less affordable computational cost.

5.3 Metrics results

The mean values of the metrics obtained for each algorithm and dataset after 10 independent runs have been statistically compared, as explained in Section 4.3. Each metric mean value has been compared pair-wise between algorithms, denoting the best value in **bold** and indicating the values that are statistically worse ($P < 0.05$) with a $\downarrow$ symbol (see Table 1).

Regarding the HV metric, our GPPR algorithm has obtained significantly better results than the two GAs in datasets P2 and S3, and worse results for dataset P1, whose search space is very small. These results only denote that the extreme solutions of the Pareto front returned by GPPR cover a larger area than those obtained by the GAs.

The Δ-Spread values show that the Single-Objective GA obtains the lowest values, that is, the best Δ-Spread values. Nevertheless, our GPPR proposal obtains values lower than those of the NSGA-II, outperforming it once again.

The spacing values show that, again, the GPPR outperforms the two GAs in the two larger datasets.Comparing the spacing values of the smallest dataset, P1, it is observed that GPPR spacing values decrease, being close to those obtained by the NSGA-II algorithm. However, in datasets P2 and S3, the GPPR spacing values are significantly greater.

Finally, for the execution time, it is important to consider that comparison between our experiments and those made by other studies is only possible if the same software and hardware requirements are met. Otherwise, only the difference in execution time between algorithms of the same study can be analyzed.

The fastest algorithm is the Single-Objective GA, due to the lightweight methods and few generations that it had executed. The NSGA-II obtained the worst values, because of the additional steps executed at each iteration, and despite implementing a fast-sorting method. When compared against our GPPR proposal, the difference is significant, being the NSGA-II almost ten times slower for the smallest dataset (P1), and fairly slower for larger datasets. It is interesting to highlight that, as dataset size grows, the difference between the NSGA-II and our GPPR proposal decreases. However, MONRP instances are not expected to have a scale large enough that the NSGA-II could outperform our GPPR. Therefore, these values demonstrate that our proposal can be applied satisfactorily to MONRP reducing drastically computational cost.

Table 1. Average metrics of the best configurations for each dataset

Dataset	Algorithm	HV	Δ-Spread	Spacing	Exec. time (s)
P1	Single-Objective GA	0.594↓	**0.615**	0.323↓	**17.967**
	NSGA-II	**1.0**	0.963↓	**0.382**	180.991↓
	GPPR	0.909↓	0.644	0.371↓	18.102
P2	Single-Objective GA	0.157↓	**0.637**	0.128↓	**82.713**
	NSGA-II	0.407↓	0.969↓	0.245↓	616.415↓
	GPPR	**0.973**	0.688↓	**0.300**	250.841↓
S3	Single-Objective GA	0.102↓	0.720	0.105↓	**125.702**
	NSGA-II	0.286↓	0.970↓	0.206↓	859.928↓
	GPPR	**0.977**	**0.711**	**0.293**	488.045↓

6 Conclusions and future work

In this paper, we have studied the applicability of a greedy procedure (GPPR) into a multi-objective problem of the Software Engineering field. The MONRP has been tackled previously using, mainly, evolutionary approaches. Few proposals have used GRASP-based methods, usually applying basic instances of it. Our proposal aimed to design a method capable of generating solutions of similar quality than those of the evolutive approaches, but reducing drastically the computational cost. We have explored different combinations of construction and local search methods, and applied post-construction techniques, such as PR, to improve the solutions found. To evaluate our proposal and compare it against classic methods, we have designed an experimentation framework, in which we have used two real-world datasets and created a new one synthetically, setting a rigorous experiment. The comparison have been carried out using a set of quality metrics and comparing the Pareto fronts, obtaining quite good results and showing that our GPPR proposal can outperform more classical and popular methods, in both performance and Pareto front results. Moreover, the code of the algorithms and experiments has been published to be shared by the scientific community.

In future lines of work, we will explore other approaches to the MONRP. It would also be interesting to try combining our GPPR with a post-optimization phase using an evolutive algorithm capable of enhance former solutions. Additionally, it could be interesting to implement interactions between requirements, which is of interest when projects use long-term planning.

References

1. Bagnall, A.J., Rayward-Smith, V.J., Whittley, I.M.: The next release problem. Information and Software Technology **43**(14), 883–890 (2001)
2. Baker, P., Harman, M., Steinhöfel, K., Skaliotis, A.: Search based approaches to component selection and prioritization for the next release problem. In: 2006 22nd IEEE International Conference on Software Maintenance. pp. 176–185 (2006)
3. Chaves-González, J.M., Pérez-Toledano, M.A., Navasa, A.: Software requirement optimization using a multiobjective swarm intelligence evolutionary algorithm. Knowledge-Based Systems **83**(1), 105–115 (2015)
4. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002)
5. del Sagrado, J., del Águila, I.M., Orellana, F.J.: Ant colony optimization for the next release problem: a comparative study. Proceedings - 2nd International Symposium on Search Based Software Engineering, SSBSE 2010 pp. 67–76 (2010)
6. Durillo, J.J., Zhang, Y., Alba, E., Nebro, A.J.: A study of the multi-objective next release problem. Proceedings - 1st International Symposium on Search Based Software Engineering, SSBSE 2009 (2009)
7. Feo, T., Resende, M.: Greedy Randomized Adaptive Search Procedures. Journal of Global Optimization **6**, 109–133 (1995)
8. Finkelstein, A., Harman, M., Mansouri, S.A., Ren, J., Zhang, Y.: A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making. Requirements Engineering **14**(4), 231–245 (2009)
9. Greer, D., Ruhe, G.: Software release planning: An evolutionary and iterative approach. Information and Software Technology **46**, 243–253 (2004)
10. Laguna, M., Marti, R.: GRASP and Path Relinking for 2-Layer Straight Line Crossing Minimization. INFORMS Journal on Computing **11**(1), 44–52 (1999)
11. del Sagrado, J., del Águila, I., Orellana, F.J.: Multi-objective ant colony optimization for requirements selection. Empirical Software Engineering **20**, 577–610 (2015)
12. Schott, J.: Fault tolerant design using single and multicriteria genetic algorithm optimization. Ph.D. thesis, Massachusetts Institute of Technology, M.S., USA (1995)
13. Zhang, Y., Harman, M., Mansouri, A.: The multi-objective next release problem. In: GECCO 2007: Genetic and Evolutionary Computation Conference. pp. 1129–1137 (2007)
14. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Transactions on Evolutionary Computation **3**(4), 257–271 (1999)

Multi-objective hyperparameter optimization with performance uncertainty

Alejandro Morales-Hernández[1,2,3], Inneke Van Nieuwenhuyse[1,2,3], and Gonzalo Nápoles[4]

[1] Core Lab VCCM, Flanders Make
[2] Research Group Logistics, Hasselt University, Agoralaan Gebouw D, Diepenbeek, 3590, Limburg, Belgium
[3] Data Science Institute, Hasselt University, Agoralaan Gebouw D , Diepenbeek, 3590, Limburg, Belgium
[4] Department of Cognitive Science & Artificial Intelligence, Tilburg University, The Netherlands

Abstract. The performance of any Machine Learning algorithm is impacted by the choice of its hyperparameters. As training and evaluating a ML algorithm is usually expensive, the hyperparameter optimization (HPO) method needs to be computationally efficient to be useful in practice. Most of the existing approaches on multi-objective HPO use evolutionary strategies and metamodel-based optimization. However, few methods account for uncertainty in the performance measurements. This paper presents results on multi-objective HPO with uncertainty on the performance evaluations of the ML algorithms. We combine the sampling strategy of Tree-structured Parzen Estimators (TPE) with the metamodel obtained after training a Gaussian Process Regression (GPR) with heterogeneous noise. Experimental results on three analytical test functions and three ML problems show the improvement in the hypervolume obtained, when compared with HPO using stand-alone multi-objective TPE and GPR.

Keywords: hyperparameter optimization · multi-objective optimization · Bayesian optimization · uncertainty

1 Introduction

In Machine Learning (ML), an hyperparameter is a parameter that needs to be specified before training the algorithm: it influences the learning process, but it is not optimized as part of the training algorithm. The time needed to train a ML algorithm with a given hyperparameter configuration on a given dataset may already be substantial, particularly for moderate to large datasets, so the HPO algorithm should be as efficient as possible in detecting the optimal hyperparameter setting.

Many of the current algorithms in the literature focus on optimizing a single (often error-based) objective [2, 13, 10]. In practical applications, however, it is

often required to consider the trade-off between two or more objectives, such as the error-based performance of a model and its resource consumption [7], or objectives relating to different types of error-based performance measures [5]. The goal in multi-objective HPO is to obtain the *Pareto-optimal* solutions, i.e., those hyperparameter values for which none of the performance measures can be improved without negatively affecting any other.

In the literature, most HPO approaches take a deterministic perspective using the mean value of the performance observed in subsets of data (cross validation protocol). However, depending on the chosen sets, the outcome may differ: a single HP configuration may thus yield different results for each performance objective, implying that the objective contains uncertainty (hereafter referred to as *noise*). We conjecture that a HPO approach that considers this uncertainty will outperform alternative approaches that assume the relationships to be deterministic. Stochastic algorithms (such as [3, 4]) can potentially be useful for problems with heterogeneous noise (the noise level varies from one setting to another). To the best of our knowledge, such approaches have not yet been studied in the context of HPO optimization. The main contributions of our approach include:

- Multi-objective optimization using a Gaussian Process Regression (GPR) surrogate that explicitly accounts for the heterogeneous noise observed in the performance of the ML algorithm.
- The selection of infill points according to the sampling strategy of multi-objective TPE (MOTPE), and the maximization of an infill criterion. This method allows sequential selection of hyperparameter configurations that are likely to be non-dominated, and that yield the largest expected improvement in the Pareto front.

The remainder of this article is organized as follows. Section 2 discusses the basics of GPR and MOTPE. Section 3 presents the algorithm. Section 4 describes the experimental setting designed to evaluate the proposed algorithm, and Section 5 shows the results. Finally, Section 6 summarizes the findings and highlights some future research directions.

2 GPR and TPE: Basics

Gaussian Process Regression (GPR) (also referred to as *kriging*, [14]) is commonly used to model an unknown target function. The function value prediction at an unsampled point $\mathbf{x}^{(*)}$ is obtained through the conditional probability $P(f(\mathbf{x}^{(*)})|\mathbf{X},\mathbf{Y})$ that represents how likely the response $f(\mathbf{x}^{(*)})$ is, given that we observed the target function at n input locations $\mathbf{x}^{(i)}, i = 1, \ldots, n$ (contained in matrix $\mathbf{X}$), yielding function values $\mathbf{y}^{(i)}, i = 1, \ldots, n$ (contained in matrix $\mathbf{Y}$) that may or may not be affected by noise. Ankenman et al. [1] provides a GPR model (referred to as *stochastic kriging*) that takes into account the heterogeneous noise observed in the data, and models the observed response value in the r-th replication at design point $\mathbf{x}^{(i)}$ as:

$$f_r(\mathbf{x}^{(i)}) = m(\mathbf{x}^{(i)}) + M(\mathbf{x}^{(i)}) + \epsilon_r(\mathbf{x}^{(i)}) \tag{1}$$

where $m(\mathbf{x})$ represents the mean of the process, $M(\mathbf{x})$ is a realization of a Gaussian random field with mean zero (also referred to as the *extrinsic uncertainty* [1]), and $\epsilon_r(\mathbf{x}^{(i)})$ is the *intrinsic uncertainty* observed in replication r. Popular choices for $m(\mathbf{x})$ are $m(\mathbf{x}) = \sum_h \beta_h f_h(\mathbf{x})$ (where the $f_h(\mathbf{x})$ are known linear or nonlinear functions of $\mathbf{x}$, and the β_h are unknown coefficients to be estimated), $m(\mathbf{x}) = \beta_0$ (an unknown constant to be estimated), or $m(\mathbf{x}) = 0$. $M(\mathbf{x})$ can be seen as a function, randomly sampled from a space of functions that, by assumption, exhibit spatial correlation according to a covariance function (also referred to as *kernel*).

Whereas GPR models the probability distribution of $f(\mathbf{x})$ given a set of observed points ($P(f(\mathbf{x})|\mathbf{X}, \mathbf{Y})$), TPE tries to model the probability of sampling a point that is directly associated to the set of observed responses ($P(\mathbf{x}|\mathbf{X}, \mathbf{Y})$) [2]. TPE defines $P(\mathbf{x}|\mathbf{X}, \mathbf{Y})$ using two densities:

$$P(\mathbf{x}|\mathbf{X}, \mathbf{Y}) = \begin{cases} l(x) \; if & f(x) < y^*, \mathbf{x} \in \mathbf{X} \\ g(x) & o.w \end{cases} \tag{2}$$

where $l(x)$ is the density estimated using the points $\mathbf{x}^{(i)}$ for which $f(\mathbf{x}^{(i)}) < y^*$, and $g(x)$ is the density estimated using the remaining points. The value y^* is a user-defined quantile γ (splitting parameter of Algorithm 1 in [11]) of the observed $f(\mathbf{x})$ values, so that $P(f(\mathbf{x}) < y^*) = \gamma$. Here, we can see l as the density of the hyperparameter configurations that may have the best response. A multi-objective implementation of TPE (MOTPE) was proposed by [11]; this multi-objective version splits the known observations according to their nondomination rank. Contrary to GPR, neither TPE nor MOTPE provide an estimator of the response at unobserved hyperparameter configurations.

3 Proposed algorithm

The algorithm (Figure 1) starts by evaluating an initial set of hyperparameter vectors through a Latin hypercube sample; simulation replications are used to estimate the objective values at these points. We then perform two processes in parallel. On the one hand, we use the augmented Tchebycheff scalarization function [9] (with a random combination of weights) to transform the multiple objectives into a single objective using these training data. Throughout this article, we will assume that the individual objectives need to be minimized; hence, the resulting scalarized objective function also needs to be minimized. We then train a (single) stochastic GPR metamodel on these scalarized objective function outcomes; the replication outcomes are used to compute the variance of this scalarized objective.

At the same time, we perform the splitting process used by [11] to divide the hyperparameter vectors into two subsets (those yielding "good" and "poor" observations) to estimate the densities $l(x)$ and $g(x)$ for each separate input

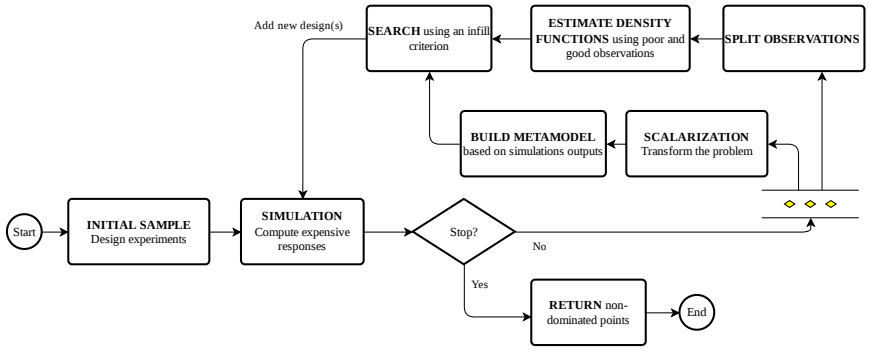


Fig. 1: Proposed multi-objective HPO using GPR with heterogeneous noise and TPE to sample the search space

dimension (Eq. 2). To that end, our approach uses a greedy selection according to the nondomination rank of the observations, and controlled by the parameter γ [5]. The strategy thus preferably selects the HP configurations with highest nondomination rank to enter in the "good" subset.

Using the densities $l(x)$, we randomly select a candidate set of n_c configurations for each input dimension. These individual values are sorted according to their log-likelihood ratio $log\frac{l(x)}{g(x)}$, such that the higher this score, the larger the probability that the input value is sampled under $l(\mathbf{x}_i)$ (and/or the lower the probability under $g(\mathbf{x}_i)$). Instead of selecting the single configuration with highest score on each dimension (as in [2, 11]), we compute the aggregated score $AS(\mathbf{x}) = \sum_{i=1}^{d} log\frac{l(x_i)}{g(x_i)}$ for each configuration, and select the one that maximizes the *Modified Expected Improvement* (MEI) [12] of the scalarized objective function in the set of configurations Q with an aggregated score greater than zero (see Eq. 3).

$$\arg\max_{\mathbf{q}\in Q} (\widehat{Z}_{\min} - \widehat{Z}_{\mathbf{q}})\Phi(\frac{\widehat{Z}_{\min} - \widehat{Z}_{\mathbf{q}}}{\widehat{s}_{\mathbf{q}}}) + \widehat{s}_{\mathbf{q}}\phi(\frac{\widehat{Z}_{\min} - \widehat{Z}_{\mathbf{q}}}{\widehat{s}_{\mathbf{q}}}) \;, Q = \{\mathbf{x} \,|\, AS(\mathbf{x}) > 0\} \quad (3)$$

where $\widehat{Z}_{\min}$ is the stochastic kriging prediction at $\mathbf{x}_{\min}$ (i.e. the hyperparameter configuration with the lowest sample mean among the already known configurations), $\phi(\cdot)$ and $\Phi(\cdot)$ are the standard normal density and standard normal distribution function respectively, the $\widehat{Z}_{\mathbf{q}}$ is the stochastic kriging prediction at configuration $\mathbf{q}$, and $\widehat{s}_{\mathbf{q}}$ is the ordinary kriging standard deviation for that configuration [15]. The search using MEI focuses on new points located in promising regions (i.e., with low predicted responses; recall that we assume that

[5] Notice that both in [11] and in our algorithm, the parameter γ represents a percentage of the known observations that may be considered as "good".

the scalarized objective need to be minimized), or in regions with high meta-model uncertainty (i.e., where little is known yet about the objective function). Consequently, the sampling behavior automatically trades off exploration and exploitation of the configuration search space.

Once a new hyperparameter configuration has been selected as infill point, the ML algorithm is trained on this configuration, yielding (again) noisy estimates of the performance measures. Following this infill strategy, we choose that configuration for which we expect the biggest improvement in the scalarized objective function, among the configurations that are likely to be non-dominated.

4 Numerical simulations

In this section, we evaluate the performance of the proposed algorithm for solving multi-objective optimization problems (GP_MOTPE), comparing the results with those that would be obtained by using GP modelling and MOTPE individually. In a first experiment, we analyze the performance on three well-known bi-objective problems (ZDT1, WFG4 and DTLZ7 with input dimension $d = 5$; see [6]), to which we add artificial heterogeneous noise (as in [4]). More specifically, we obtain noisy observations $\widetilde{f}_p^j(\mathbf{X}_i) = f_j(\mathbf{X}_i) + \epsilon_p(\mathbf{X}_i), p = \{1, \ldots, r\}, j = \{1, \ldots, m\}$, with $\epsilon_p(\mathbf{X}_i) \sim \mathcal{N}(0, \tau_j(\mathbf{X}_i))$. The standard deviation of the noise ($\tau_j(\mathbf{X})$) varies for each objective between $0.01 \times \Omega^j$ and $0.5 \times \Omega^j$, where Ω^j is the range of objective j. In between these limits, $\tau_j(\mathbf{X})$ decreases linearly with the objective value: $\tau_j(\mathbf{X}) = a_j(f_j(\mathbf{X}) + b_j), \forall j \in \{1, \ldots, m\}$, where a and b are the linear coefficients obtained from the noise range [8].

Table 1: Details of the ML datasets

Dataset	ID	Inst. (Feat.)	Dataset	ID	Inst. (Feat.)
Balance-scale	997	625 (4)	Delta_ailerons	803	7129 (5)
Optdigits	980	5620 (64)	Heart-statlog	53	270 (13)
Stock	841	950 (9)	Chscase_vine2	814	468 (2)
Pollen	871	6848 (5)	Ilpd	41945	583 (10)
Sylvine	41146	5124 (20)	Bodyfat	778	252 (14)
Wind	847	6574 (14)	Strikes	770	625 (6)

In a second experiment, we test the algorithm on a number of OpenML datasets, shown in Table 1. We optimize five hyperparameters for a simple (one hidden layer) Multi-Layer Perceptron (MLP), two for a support vector machine (SVM), and five for a Decision Tree (DT) (see Appendix A). In each experiment, the goal is to find the HPO configurations that minimize classification error while simultaneously maximizing recall. In all experiments, we used 20% of the initial dataset as test set, and the remainder for HPO. We apply stratified *k-fold cross-validation* ($k = 10$) to evaluate each hyperparameter configuration.

We used a fixed, small number of iterations (100) as a stopping criterion in all algorithms; this keeps optimization time low, and resembles real-world optimization settings where limited resources (e.g., time) may exist. Table 2 summarizes the rest of the parameters used in the experiments.

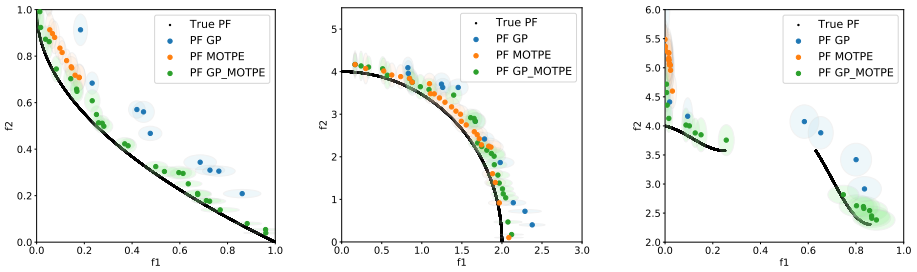
Table 2: Summary of the parameters for the experiments

Setting	Problem	GP	MOTPE	GP_MOTPE
Initial design	Analytical fcts	LHS: $11d-1$		
	HPO	Random sampling: $11d-1$		
Replications	Analytical fcts	50		
	HPO	10		
Acquisition function		MEI	EI_{TPE}	MEI
Acquisition function optimization		**PSO***	Maximization on a candidate set	
Number of candidates to sample		-	$n_c = 1000$, $\gamma = 0.3$	
Kernel		Gaussian	-	Gaussian

* PSO algorithm (Pyswarm library): swarm size = 300, max iterations = 1800, cognitive parameter=0.5, social parameter=0.3, and inertia=0.9

5 Results

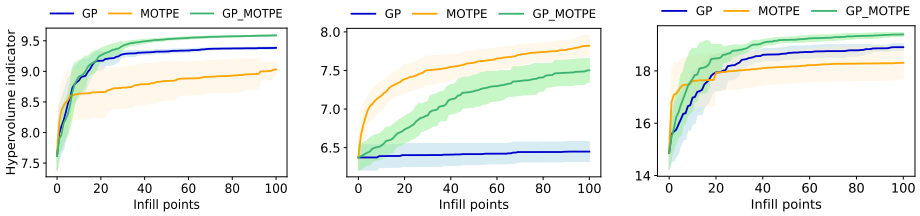
Figure 3 shows the evolution of the hypervolume indicator during the optimization of the analytical test functions. The combined algorithm GP_MOTPE yields a big improvement over both GP and MOTPE algorithms for the ZDT1 and DTLZ7 functions, reaching a superior hypervolume already after a small number of iterations. Results also show that for ZDT1 and DTLZ7, the standard deviation on the final hypervolume obtained by GP and GP_MOTPE is small, which indicates that a Pareto front of similar quality is obtained regardless of the initial design. MOTPE, by contrast, shows higher uncertainty in the hypervolume results at the end of the optimization. For the concave Pareto front of WFG4, MOTPE provides the best results, while GP_MOTPE still outperforms GP.



(a) ZDT1 (b) WFG4 (c) DTLZ7

Fig. 2: Observed Pareto front (PF) obtained at the end of a single macroreplication, for the analytical test functions. The uncertainty of each solution is shown by a shaded ellipse, and reflects the $mean \pm std$ of the simulation replications.

Table 3 shows the average rank of the optimization algorithms according to the hypervolume indicator. The experiments did not highlight significant differences between GP_MOTPE, GP and MOTPE ($p\_value = 0.565 > 0.05$ for the non-parametric Friedman test where H_0 states that the mean hypervolume of the solutions is equal). However, GP_MOTPE has the lowest average rank in the validation set, indicating that on average, the Pareto front obtained with our algorithm tends to outperform those found by GP and MOTPE individually, yielding a larger hypervolume.



(a) ZDT1 $ref = [1, 10]$ (b) WFG4 $ref = [3, 5]$ (c) DTLZ7 $ref = [1, 23]$

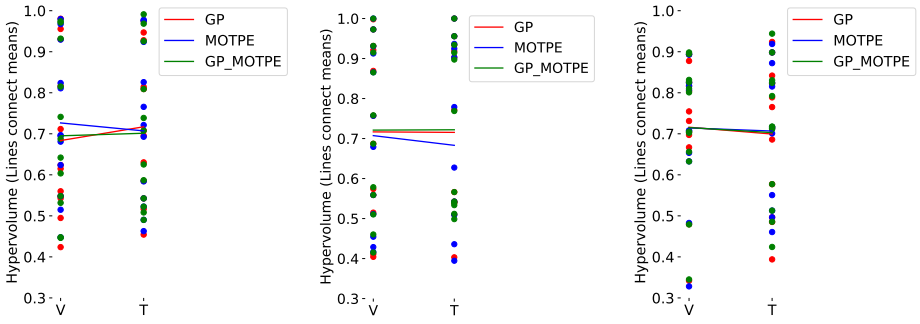
Fig. 3: Hypervolume evolution during the optimization of the analytical test functions. Shaded area represents $mean \pm std$ of 13 macro-replications. Captions contain the reference point used to compute the hypervolume indicator

Once the Pareto-optimal set of HP configurations has been obtained on the validation set, the ML algorithm (trained with those configurations) is evaluated on the test set. The difference between the hypervolume values obtained from the validation and test set can be used as a measure of reliability: in general, one would prefer HP configurations that generate a similar hypervolume in the test set. Figure 4 shows that the difference between both hypervolume values is almost zero when GP_MOTPE is used, for all ML algorithms. In general, MOTPE and GP_MOTPE have the smallest (almost identical) mean absolute hypervolume difference (0.0444 and 0.0445 respectively), compared with that of GP (0.051). However, GP_MOTPE has the smallest standard deviation (0.054), followed by MOTPE (0.066) and GP (0.067).

Table 3: Average rank (given by the hypervolume indicator) of each algorithm

	Validation set			**Test set**		
	GP	**MOTPE**	**GP_MOTPE**	**GP**	**MOTPE**	**GP_MOTPE**
Avg. rank	2.125	1.9861	1.8889	2.1528	1.875	1.9722

It is somehow surprising that the combined GP_MOTPE algorithm does not always obtain an improvement over the individual MOTPE and GP algorithms. By combining both approaches, we ensure that we select configurations that (1)



(a) MLP (b) SVM (c) DT

Fig. 4: Hypervolume generated by the HP configurations found using the validations set (V) and then evaluated with the test set (T)

have high probability to be nondominated (according to the candidate selection strategy), and (2) has the highest MEI value for the scalarized objective. In the individual GP algorithm, (1) is neglected, which increases the probability of sampling a non-Pareto optimal point, especially at the start of the algorithm. In the original MOTPE algorithm, (2) is neglected, which may cause the algorithm to focus too much on exploitation, which increases the probability of ending up in a local optimum. We suspect that the MOTPE approach for selecting candidate points may actually be too restrictive: it will favor candidate points close to already sampled locations, inherently limiting the exploration opportunities the algorithm still has when optimizing MEI.

6 Concluding remarks

In this paper, we proposed a new algorithm (GP_MOTPE) for multi-objective HPO of ML algorithms. This algorithm combines the predictor information (both predictor and predictor variance) obtained from a GPR model with heterogenous noise, and the sampling strategy performed by Multi-objective Tree-structured Parzen Estimators (MOTPE). In this way, the algorithm should select new points that are likely to be non-dominated, and that are expected to cause the maximum improvement in the scalarized objective function.

The experiments conducted report that our approach performed relatively well for the analytical test functions of study. It appears to outperform the pure GP algorithm in all analytical instances; yet, it does not always outperform the original MOTPE algorithm. Further research will focus on why this is the case, which may yield further improvements in the algorithm. In the HPO experiments, GP_MOTPE shows the best average rank w.r.t. the hypervolume computed on the validation set. In addition, it showed promising reliability properties (small changes in hypervolume when the ML algorithm is evaluated on the test set). Based upon these first results, we believe that the combination of GP

and TPE is promising enough to warrant further research. The observation that it outperforms the pure GP algorithm (which used PSO to maximize the infill criterion) is useful in its own right, as the optimization of infill criteria is known to be challenging. Using MOTPE, a candidate set can be generated that can be evaluated efficiently, and which (from these first results) appears to yield superior results.

Acknowledgements

This research was supported by the Flanders Artificial Intelligence Research Program (FLAIR), and by the Research Foundation Flanders (FWO Grant 1216021N).

Appendix A. Setup of hyperparameters in the HPO experiments

HP	Description	Type	Range
Multilayer Perceptron (MLP)			
max_iter	Iterations to optimize weights	Int.	$[1, 1000]$
neurons	Number of neurons in the hidden layer	Int.	$[5, 1000]$
lr_init	Initial learning rate	Int.	$[1, 6]$
b1	First exponential decay rate	Real	$[10^{-7}, 1]$
b2	Second exponential decay rate	Real	$[10^{-7}, 1]$
Support Vector Machine (SVM)			
C	Regularization parameter	Real	$[0.1, 2]$
kernel	Kernel type to be used in the algorithm	Cat.	[linear, poly, rbf, sigmoid]
Decision Tree (DT)			
max_depth	Maximum depth of the tree. If 0, then *None* is used	Int.	$[0, 20]$
mss	Minimum number of samples required to split an internal node	Real	$[0, 0.99]$
msl	Minimum number of samples required to be at a leaf node	Int.	$[1, 10]$
max_f	Features in the best split	Cat.	[auto, sqrt, log2]
criterion	Measure the quality of a split	Cat.	[gini, entropy]

References

1. Ankenman, B., Nelson, B.L., Staum, J.: Stochastic kriging for simulation metamodeling. Operations Research **58**(2), 371–382 (2010). https://doi.org/10.1109/WSC.2008.4736089
2. Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. Advances in neural information processing systems **24** (2011)

3. Binois, M., Huang, J., Gramacy, R.B., Ludkovski, M.: Replication or exploration? sequential design for stochastic simulation experiments. Technometrics **61**(1), 7–23 (2019). https://doi.org/10.1080/00401706.2018.1469433
4. Gonzalez, S.R., Jalali, H., Van Nieuwenhuyse, I.: A multiobjective stochastic simulation optimization algorithm. European Journal of Operational Research **284**(1), 212–226 (2020). https://doi.org/10.1016/j.ejor.2019.12.014
5. Horn, D., Bischl, B.: Multi-objective parameter configuration of machine learning algorithms using model-based optimization. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 1–8. IEEE (2016). https://doi.org/10.1109/SSCI.2016.7850221
6. Huband, S., Hingston, P., Barone, L., While, L.: A review of multiobjective test problems and a scalable test problem toolkit. IEEE Transactions on Evolutionary Computation **10**(5), 477–506 (2006)
7. Igel, C.: Multi-objective model selection for support vector machines. In: International conference on evolutionary multi-criterion optimization. pp. 534–546. Springer (2005). https://doi.org/10.1007/978-3-540-31880-4_37
8. Jalali, H., Van Nieuwenhuyse, I., Picheny, V.: Comparison of kriging-based algorithms for simulation optimization with heterogeneous noise. European Journal of Operational Research **261**(1), 279–301 (2017)
9. Knowles, J.: Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Transactions on Evolutionary Computation **10**(1), 50–66 (2006). https://doi.org/10.1109/TEVC.2005.851274
10. Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., Talwalkar, A.: Hyperband: A novel bandit-based approach to hyperparameter optimization. The Journal of Machine Learning Research **18**(1), 6765–6816 (2017)
11. Ozaki, Y., Tanigaki, Y., Watanabe, S., Onishi, M.: Multiobjective tree-structured parzen estimator for computationally expensive optimization problems. In: Proceedings of the 2020 Genetic and Evolutionary Computation Conference. pp. 533–541 (2020)
12. Quan, N., Yin, J., Ng, S.H., Lee, L.H.: Simulation optimization via kriging: a sequential search using expected improvement with computing budget constraints. Iie Transactions **45**(7), 763–780 (2013)
13. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. Advances in neural information processing systems **25** (2012)
14. Williams, C.K., Rasmussen, C.E.: Gaussian processes for machine learning, vol. 2. MIT press Cambridge, MA (2006)
15. Zhan, D., Xing, H.: Expected improvement for expensive optimization: a review. Journal of Global Optimization **78**(3), 507–544 (2020). https://doi.org/10.1007/s10898-020-00923-x

A learning based matheuristic to solve the two machine flowshop scheduling problem with sum of completion times

Vincent T'kindt[1,2], Romain Raveaux[1]

[1] Université de Tours, Laboratoire d'Informatique Fondamentale et Appliquée de Tours (LIFAT - EA 6300), 64 Avenue Jean Portalis, 37000 Tours, France
tkindt@univ-tours.fr; romain.raveaux@univ-tours.fr
[2] ERL CNRS 7002 Recherche Opérationnelle, Ordonnancement et Transport, 64 Avenue Jean Portalis, 37000 Tours, France

1 Introduction

Consider the problem where n jobs have to be scheduled on two machines organized in a flowshop setting. Each job j is defined by a *processing time* $p_{j,i}$ on machine $i = 1, 2$ and has to be processed first on machine 1 and next on machine 2. Each machine can only process one job at a time and preemption is not allowed. We restrict the search for a solution to the set of permutation schedules. Therefore, the goal is to find a schedule s (permutation) that minimizes the total completion time $\sum_j C_j(s)$ with $C_j(s)$ the completion time of job j in schedule s. When there is no ambiguity, we omit the reference to schedule s when referring to completion times. Following the standard three-field notation in scheduling theory, this problem is referred to as $F2||\sum_j C_j$ and is strongly $\mathcal{NP}$-hard [3].

The $F2||\sum_j C_j$ problem is a challenging problem which has been studied for a long time from both exact and heuristic point of views. In this work, we don't consider exact approaches. Along the years, several heuristic algorithms have been proposed and, to the best of our knowledge, the most efficient one is a matheuristic proposed in [2]. Matheuristics are local search algorithms which explore the neighborhood of an incumbent solution by solving a mathematical programming formulation of the problem. In this work we propose the developpment of a learning based predictor to improve the matheuristic proposed in [2], notably to drive the exploration of large size neighborhoods.

2 A learning based matheuristic

The use of machine learning (ML) techniques within operations research (OR) algorithms is a recent but active and promising research area [1]. To the best of our knowledge, very few contributions of this kind have considered scheduling problems. Our intent is to improve an existing matheuristic referred to as `MATH` ([2]), by using a predictor to drive the exploration of neighborhoods. We first provide the basics about `MATH` heuristic before introducing with the contribution of machine learning.

A solution s of the $F2||\sum_j C_j$ problem can be seen as a sequence of jobs and let $s[k]$ be the job scheduled at position k in s. `MATH` heuristic proceeds by selecting, at each iteration, a *window of positions* $[r; r + h]$ with parameter r being the starting position and parameter h being its width. When r and h are fixed, the problem is to reschedule jobs in positions r to $r + h$ while keeping fixed the partial schedule from positions 1 to $(r - 1)$ and from positions $(r + h + 1)$ to n. This re-optimization is done to optimality by solving a MIP formulation of the problem. Matheuristics are very efficient heuristics which optimally explore large size neighborhoods but rely on a good procedure for selecting relevant windows of positions to reschedule. In the `MATH` heuristic proposed by Della Croce et al. [2], the value r is randomly selected between 1 and $(n - h)$ at each iteration. Besides, the authors report that a good trade-off between time and efficiency is to consider $h = 12$. The latter choice resulting from an experimental evaluation of the average efficiency of their algorithm.

The strengh of MATH relies on the genuine exploitation of an efficient mixed integer formulation of the problem. However, it has several flaws induced by the selection of the r and h values at each iterations. First, two runs on the same instance of the flowshop problem rarely provides the same solution which is induced by the random choice of r. Besides, the convergence towards a good solution can be rather slow as few windows $[r; r+h]$ lead in practice to an improvement of the incumbent solution. At last, the choice of $h = 12$ is debatable as sometimes lower values lead to the same results but faster and as it could be worth trying sometimes larger tractable values. Notice that MATH stops after a given time limit is reached.

In this research we want to demonstrate that machine learning can help improving the MATH heuristic in the choice of relevant rescheduling windows. To our opinion, the ultimate goal in such a perspective is to have a predictor capable of predicting r and h for a given schedule s, if there is a window of positions whose rescheduling leads to a better solution than s. However, this problem is very challenging and we rather focus, as a first approach, in building a predictor capable of deciding if a given window $[r; r+h]$ is worth being rescheduled, *i.e.* if there is an improving solution in this neighborhood. The corresponding learning problem can be formulated as follows: we learn to predict for a given schedule s and values r and h, if a reoptimization leads to an improvement or not. This learning problem is a classification problem. The strengths of this learning based matheuristic are numerous: (1) faster convergence towards a near-optimal solution, (2) removing dependency of the results on randomness, (3) improvement of the quality of the computed solutions.

3 Solution of the learning problem

A deep learning solution was adopted to create our predictor. Deep Learning is about finding a good data representation with respect to an objective thanks to a composition of functions. Composition means that complicated functions are combinations of smaller, simpler functions. Deep learning relies on data. As a consequence, MATH was instrumented to generate a data set $\mathcal{D} = \{x_m, y_m\}_{m=1}^{M}$ from random instances of the scheduling problem. x_m is a triple composed of a job sequence, r and h. y_m is a Boolean value such that $y_m = 1$ if the window $[r; r+h]$ leads to an improvement and $y_m = 0$ otherwise. This raw data are complex and not easy to handle by machine learning techniques. So, we decided to build an embedding function ϕ. Each x_m is projected into a vector space of dimension d thanks to the function $\phi(x_m) \in \mathbb{R}^d$. Key information about the sequence and the window are extracted by the function ϕ which can be seen as a smart preprocessing to feed the predictor with a meaningful representation of a sequence and a window. We have identified 88 potentially relevant features $\phi_j \in \mathbb{R}$.

The predictor predicts whether or not a reoptimization in the window may lead to an improvement. The predictor is a function $p(\phi(x_m), \theta^*) \in [0; 1]$ where parameters $\theta^* \in \Theta$ are found by solving the following learning problem: $\theta^* = arg\min_{\theta \in \Theta} \sum_{(x_m, y_m) \in \mathcal{D}} l(p(\phi(x_m), \theta), y_i)$, l being a loss function gauging the error between the predicted value and the true value (y_m). The learning problem is solved by a well-established gradient descent algorithm called Adam [4].

A key point when learning is to built relevant training, validation and test databases, which must be representative of the reality the predictor will face in the matheuristic. To solve this challenging problem, for each database, we have randomly generated hundreds of instances of the scheduling problem. For each of them, we consider all windows $[r; r+h]$ of the incumbent solutions and systematically reschedule them noting the result as a sample in the database under generation. At the time of solving the learning problem using the training and validation databases, we pay attention to penalize the samples which correspond to no improvement of the incumbent solution as naturally few windows lead to an improvement.

4 Computational experiments

Preliminary results are encouraging but still, works remain to be done to achieve a very accurate predictor. Notably, it seems that the combinatorics induced by scheduling problem (which is a hard

permutation problem) leads to make harder the learning and generalization of a good predictor. To illustrate the potential gain of our approach, we compare two versions of the matheuristic: `MATH`, the original version, and `lMATH` a version embedding a predictor. Heuristic `lMATH` works exactly as `MATH` except that for a given window $[r; r + h]$ `MATH` systematically reschedule while `lMATH` follows the predictor $p(\phi(s), \theta^*)$ to reschedule or not. For a given incumbent schedule s and a window $[r; r + h]$, if $p(\phi(s), \theta^*) \geq 0.5$ then we consider that rescheduling window leads to an improvement over s. Besides, `lMATH` stops whenever, for a given s, no windows lead to an improvement according to the predictor.

The predictor we use in these very preliminary experiments, is built by learning on validation database which is generated by considering 7 instances with $n = 50$ jobs, which the training database is generated by considering 300 instances with $n = 50$ jobs. For 4 out of these 7 instances we report in table 1 the results we obtain. Entries *BestFound* report the best solution value found by a heuristic. Entries *TimeToBest* report the time in seconds a heuristic has consumed to reach its final solution. Entries *TotalTime* report the total CPU time used by each heuristic before stopping. Notice that for `MATH` the time limit has been set to 120s.

	Instance 1		Instance 2		Instance 3		Instance 4	
	`MATH`	`lMATH`	`MATH`	`lMATH`	`MATH`	`lMATH`	`MATH`	`lMATH`
BestFound	55005	55005	62723	62719	48195	48194	51303	51303
TimeToBest (s)	3.37	0.47	5.40	0.39	5.00	6.57	1.00	0.80
TotalTime (s)	120	7.20	120	8.15	120	12.30	120	5.00

Table 1. Comparison of `MATH` and `lMATH`

These results show that building an effective predictor to drive the matheuristic is a research line worth to be explored. Notice that the above results have been obtained on instances with 50 jobs which are instances easy to solve for `MATH`: we can legitimately expect that `lMATH` outperforms even stronger than MATH on instances with more jobs.

References

1. Y. Bengio, A. Lodi and A. Prouvost. Machine Learning for Combinatorial Optimization: A Methodological Tour d'horizon. European Journal of Operational Research, 290(2):405-421, 2021.
2. F. Della Croce, A. Grosso and F. Salassa. A matheuristic approach for the two-machine total completion time flow shop problem. Annals of Operation Research, 213:67-78, 2004.
3. M.R. Garey, D.S. Johnson and R. Sethi. The complexity of flowshop and jobshop scheduling. Mathematics of Operations Research, 1:117-129, 1976.
4. Kingma, D. & Ba, J. Adam: A Method for Stochastic Optimization. *ArXiv E-prints*. pp. earXiv:1412.6980 (2014,12)

Numerical Optimization of the Dirichlet Boundary Condition in the Phase Field Problem

A. Wodecki, P. Strachota, and T. Oberhuber

Department of Mathematics, Faculty of Nuclear Sciences and Physical Engineering, Czech Technical University in Prague. Trojanova 13, 120 00 Praha 2, Czech Republic
wodecale@fjfi.cvut.cz

1 Introduction

The phase field problem is used heavily in the modeling of interfacial dynamics. More specifically, it has found great utility in forecasting the evolution of solidification, fractures or even tumor growth [1–4]. In this contribution, we focus on a variant of the phase field model (PFM) that may be used to simulate the solidification of a pure substance [8]. The distributed optimal control of this PFM with homogeneous Neumann or Dirichlet boundary conditions has been addressed extensively in the past [3, 5–7]. To the authors' best knowledge however, the Dirichlet boundary control for parabolic equations has been addressed only by a handful of publications [9–11]. When working with the weak formulation of these problems the non-homogeneity of the boundary condition causes the problem to not be of variational type. This introduces many difficulties. The common tools that PDE theory provides to deal with such an issue are approximating the Dirichlet boundary condition by Robin boundary conditions or using Dirichlet lifts [12]. Alternatively, one can use the concept of a very weak solution [11]. For the purpose of numerical simulations, we may circumvent these issues and utilize the strong formulation. Using this analytical setting, it is possible to derive an efficient gradient computation method based on the adjoint equations.

2 Problem Formulation

Let $\Omega \subset \mathbb{R}^n$ be a bounded domain and let $T > 0$. The PFM describes the evolution of the phase field $\tilde{y}$ and the temperature field y within the domain Ω. The state of solidification of a pure material in Ω is described by the phase field $\tilde{y}(x,t) \in [0,1]$ for any $t \in [0,T]$ and $x \in \Omega$. The liquid subdomain $\Omega_s(t)$, solid subdomain $\Omega_s(t)$ and the interface $\Gamma(t)$ are identified as

$$\Omega_s(t) = \left\{ x \in \Omega; \tilde{y}(t,x) > \frac{1}{2} \right\}, \Omega_l(t) = \left\{ x \in \Omega; \tilde{y}(t,x) < \frac{1}{2} \right\}, \Gamma(t) = \left\{ x \in \Omega; \tilde{y}(t,x) = \frac{1}{2} \right\}.$$

Our aim is to obtain a Dirichlet boundary condition for the temperature field y that results in a phase field profile $\tilde{y}$ at time T that is as close as possible to a prescribed PF profile $\tilde{y}_f \in L^2(\Omega)$ (i.e., the shape of the solid subdomain $\Omega_s(T)$). This leads us to consider the problem

$$\min J(y,\tilde{y},u) = \frac{1}{2}\int_\Omega |\tilde{y}(T,x) - \tilde{y}_f(x)|^2 dx + \frac{\alpha}{2}\int_0^T \int_{\partial\Omega} |u(t,x)|^2 dt dx \tag{1}$$

$$y_t = \Delta y + H\tilde{y}_t, \qquad \text{in } (0,T)\times\Omega, \tag{2}$$

$$y|_{\partial\Omega} = u \qquad \text{on } [0,T]\times\partial\Omega, \tag{3}$$

$$y|_{t=0} = y_{\text{ini}} \qquad \text{in } \Omega, \tag{4}$$

$$\gamma\xi^2\tilde{y}_t = \xi^2\Delta\tilde{y} \tag{5}$$

$$+ \tilde{y}(1-\tilde{y})\left(\tilde{y} - \frac{1}{2}\right) - \beta\xi(y - y_{\text{mt}}) \qquad \text{in } (0,T)\times\Omega, \tag{6}$$

$$\tilde{y}|_{\partial\Omega} = \tilde{y}_{\text{bc}} \qquad \text{on } [0,T]\times\partial\Omega, \tag{7}$$

$$\tilde{y}|_{t=0} = \tilde{y}_{\text{ini}} \qquad \text{in } \Omega, \tag{8}$$

where α denotes the strength of the regularization term, H is the latent heat of fusion and the corresponding Dirichlet boundary control u is in $C^0([0,T]\times\partial\Omega)$. The parameter y_{mt} denotes the melting temperature, γ, β are dimensionless model parameters and ξ is a parameter related to interface thickness of the phase field $\tilde{y}$ [8].

3 Gradient Computation

We use Lagrangian formalism to derive the adjoint equations for the problem. Labeling $\hat{J}(u) = J(S(u), u)$, where S is the solution operator mapping a control u to the solution of (1)-(8) we arrive at a potentially efficient method for gradient computation

$$\delta\hat{J}(u;s) = \alpha \int_0^T \int_{\partial\Omega} usdtdSt - \int_0^T \int_{\partial\Omega} \nabla p \cdot \vec{n}\,|_{\partial\Omega}\; sdtdS, \tag{9}$$

where $\delta\hat{J}(u;s)$ stands for the directional derivative at point u in direction s, and p is a component of the solution (p,q) of the adjoint system

$$p_t = \Delta p - \beta\xi q \qquad \text{in } (0,T)\times\Omega, \tag{10}$$
$$p|_{\partial\Omega} = 0 \qquad \partial\Omega\times[0,T], \tag{11}$$
$$p|_{t=0} = 0 \qquad \text{in } \Omega, \tag{12}$$
$$\gamma\xi^2 q_t = \xi^2\Delta q + Hp_t - 3\tilde{z}^2 q + 3\tilde{z}q - \frac{1}{2}q \text{ in } (0,T)\times\Omega, \tag{13}$$
$$q|_{\partial\Omega} = 0 \qquad \text{on } \partial\Omega\times[0,T], \tag{14}$$
$$q|_{t=0} = \frac{1}{\gamma\xi^2}\left(\tilde{y}_f - \tilde{y}|_{t=T}\right) \qquad \text{in } \Omega, \tag{15}$$

where $\tilde{z} = \tilde{y}(T-t)$.

4 Numerical Results

We iterate using gradient descent to eventually arive at a local minimum. To solve the primary (1)-(8) and adjoint equations (10)-(15), the finite difference method on a uniform rectangular grid is used along with the forward Euler method for time stepping. The trapezoid rule is applied to approximate the gradient computation (9). Several experiments were performed in one spatial dimension with parameters that do not necessarily correspond to any physical material. In all of these experiments, a control that results in the target profile of the phase field being reached is obtained.

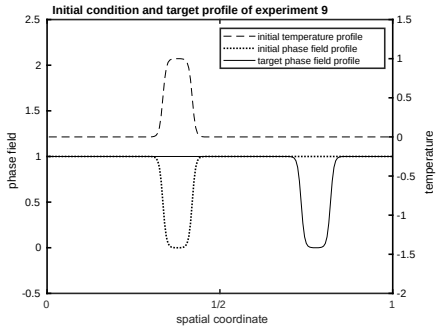
In one of these experiments, we attempt to move a gap between two solid domains. Figure 1a shows how $y_{\text{ini}}, \tilde{y}_{\text{ini}}$ and the target profile $\tilde{y}_f$ are set. Figure 1b shows the resulting temporal control profile of the left and right Dirichlet boundary control. The final states of the temperature and heat field compared to the target phase field profile can be reviewed in Figure 1c. These results demonstrate that even highly non-trivial control profiles can be obtained using this method.

5 Conclusion

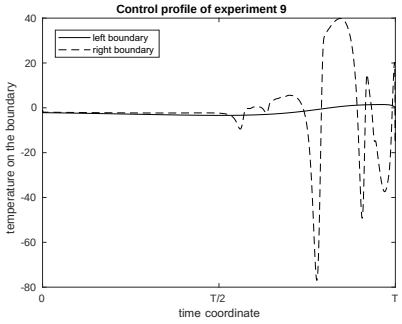
A viable numerical method for the optimization of the Dirichlet boundary condition of the phase field problem was presented. Utilizing the strong formulation of the problem, the adjoint equations were derived. A basic numerical treatment of the resulting problem using the finite difference method and trapezoid rule is shown to yield desirable results. The framework is set up so that combining other more sophisticated numerical solvers (perhaps ones with adaptive time stepping) should be possible.

References

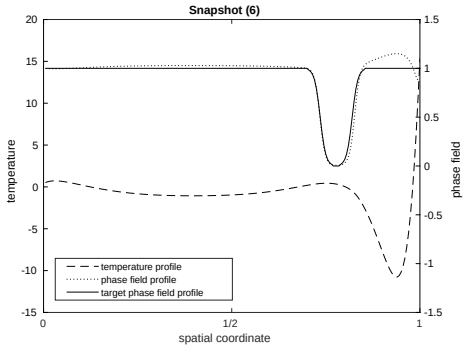
1. Heike Ulmer Christian Miehe, Lisa-Marie Schanzel. Phase field modeling of fracture in multi-physics problems. part i. balance of crack surface and failure criteria for brittle crack propagation in thermo-elastic solids. Computer Methods in Applied Mechanics and Engineering, 2014.
2. Shunsuke Kurima. Asymptotic analysis for cahn hilliard type phase field systems related to tumor growth in general domains. arxiv, 2018.



(a) the initial setup of $y_{\text{ini}}, \tilde{y}_{\text{ini}}$ and the target profile $\tilde{y}_f$



(b) The optimized temporal Dirichlet boundary control profile u



(c) The final states of the temperature and phase fields $y(T), \tilde{y}(T)$ compared to the target profile $\tilde{y}_f$

Fig. 1: A simple numerical experiment demonstrating the capabilities of the method

3. Pierluigi Colli, Andrea Signori, and Jürgen Sprekels. Optimal control of a phase field system modelling tumor growth with chemotaxis and singular potentials. Applied Mathematics and Optimization, 83:2017–2049, 2019.
4. Z. Guo and S. M. Xiong. Study of dendritic growth and coarsening using a 3-D phase field model: Implementation of the Para-AMR algorithm. IOP Conf. Ser.: Mater. Sci. Eng., 84:012067, 2015.
5. Pierluigi Colli, Gianni Gilardi, Gabriela Marinoschi, and Elisabetta Rocca. Optimal control for a conserved phase field system with a possibly singular potential. Evolution Equations and Control Theory, 7(1):95–116, 2018.
6. Karl-Heinz Hoffman and Lishang Jiang. Otimal control of a phase field model for solidification. Numerical Functional Analysis and Optimization 13:1-2, 11-27., 1992.
7. Xiong Zonghong, Wei Wei, Zhou Ying, Wang Yue, and Liao Yumei. Opti- mal control for a phase field model of melting arising from inductive heating. AIMS Mathematics, 7(1):121–142, 2022.
8. Pavel Strachota, Aleš Wodecki, and Michal Beneš. Focusing the latent heat release in 3D phase field simulations of dendritic crystal growth. Modelling Simul. Mater. Sci. Eng., 29:065009, 2021.
9. F. B. Belgacem, C. Bernardi, and H. E. Fekih. Dirichlet boundary control for a parabolic equation with a final observation i: A space-time mixed formulation and penalization. Asymptotic Analysis, 71:101–121, 2011.
10. K. Kunisch and B. Vexler. Constrained dirichlet boundary control in l2 for a class of evolution equations. SIAM J. Control Optim., 46:1726–1753, 2007.
11. Gong, Wei et al. A Finite Element Method For Dirichlet Boundary Control Problems Governed by Parabolic pdes. arXiv: Optimization and Control (2014): n. pag.
12. Graeme Fairweather. Finite element Galerkin methods for differential equations. M. Dekker New York, NY, 1978.

Cooperation-based search of global optima

Damien Vergnet[0000−0003−2948−8807], Elsy Kaddoum, Nicolas Verstaevel, Frédéric Amblard

IRIT, Université de Toulouse, CNRS, Toulouse INP, UT3, **UT1**, **UT2**, Toulouse, France <firstname>.<lastname>@irit.fr

Abstract. A new cooperation-based metaheuristic is proposed for searching gobal optima of functions. It is based on the assumption that the dynamics of the objective function does not change significantly between iterations. It relies on a local search process coupled with a cooperative semi-local search process. Its performances are compared against four other metaheuristics on unconstrained mono-objective optimization problems. Results show that the proposed metaheuristic is able to find the global minimum of the tested functions faster than the compared methods while reducing the number of iterations and the number of calls of the objective function.

Keywords: local cooperation · collective decision · metaheuristic optimization · local search

1 Introduction

The simulation of systems is a powerful tool to understand their behaviors and underline their advantages and limits. Several studies aim at reconstructing virtual systems called digital twins to simulate and verify the behavior of specific systems. Such systems can be used in mobility or natural disaster studies to reproduce specific simulation conditions and understand the reasons of such phenomena [4]. Building a digital twin that reproduces the exact behavior of a real system is not an easy task. As real systems are generaly complex systems with non-linear interdependencies among their parameters, finding the best modeling functions and adapting in real-time their parameters to keep a simulation close to the real behavior of the system is not trivial. Many studies have formalised the calibration problem as an optimization problem where the parameters of the modeling functions are tuned by optimizing an objective function: simulation parameters become decision variables and relevent model outputs are integrated into objective functions [2,8]. This implies the need for a fast optimization system that is able to rapidly adapt to changes that may occur in the real system.

Multiple optimization methods exist that could be used to solve this problem but they present important drawbacks such as a tendency to converge towards local optima or are too slow [6,11,14].

In this paper we propose a new metaheuristic local optimization method named **CoBOpti**, which stands for **Co**operation-**B**ased **Opti**mization. It is

based on an hypothesis of local continuity of the objective function, i.e. the value of the objective function does not vary dramatically when the value of decisions variables varies little. Compared to standard state of the art methods, CoBOpti reaches optimal solutions while reducing the number of iterations and objective function evaluation.

The main contributions of this paper are as follows:

- We introduce a **new local optimization metaheuristic based on an hypothesis of local continuity and cooperation**. This hypothesis allows to model the problem of searching for a global optimum as **a cooperation problem** where a point determines the next point to explore by exploiting the information of its neighbours.
- We experiment and compare our approach on unconstrained mono-objective optimization problems with a single decision variable to demonstrate that **the proposed approach allows to reach a global optimum while minimizing the number of evaluation of the objective function**.

The paper is organised as follow : section 2 discusses the limitations of existing metaheuristics. Section 3 presents our approach and how it gives an answer to these limitations. In section 4, we introduce the results of our experimentation, which is then discussed in section 5 before concluding with limitations and suggest further research.

2 Literature Review

Optimization problems are defined by [3] as finding a vector $\bar{x}_n^* = (x_1^*, ..., x_n^*)$ that optimizes an objective function

$$\bar{f}_k(\bar{x}_n) = (o_1(\bar{x}_n), ..., o_k(\bar{x}_n)) \tag{1}$$

where $\bar{x}_n = (x_1, ..., x_n)$ is a vector of n decision variables.

Many methods exist to solve optimization problems, each making some assumptions on the nature of the problem. One category of such optimization methods is called metaheuristics. [6] defines metaheuristics as methods that perform local and higher level search procedures that are capable of escaping local optima. This definition notably includes methods that employ the notion of neighborhood. The neighborhood of a solution s is the set of all solutions that can be reached from s.

Metaheuristics are interesting for solving optimization problems as they are designed to efficiently explore complex search spaces [6]. Sörensen *et al.* [12] further state that the large majority of real-life optimization problems are more easily solved by metaheuristics, hence our focus on these methods in this paper.

Metaheuristics rely on two important notions: **intensification** and **diversification**. Intensification is a process through which portions of the search space that seem "promising" are explored more thoroughly, i.e. in the neighborhood of the best solutions found yet. Diversification, on the other hand, is a process

aimed at exploring unexplored parts of the search space in hopes to find better solutions. It usually relies on a some form of memory of visited solutions [5].

There are numerous metaheuristics, each with their own hypotheses. As the goal of our proposition is to be used to perform on-line calibration, it needs to rely on fast algorithms and and to be able to handle the set of visited solutions. The presented methods are thus focused around local search and population-based meta-heuristics.

Local search algorithms explore the search space by exploring the immediate neighborhood of the current solution s and selecting the neighbor solution that has a lower objective value than s. In order to escape from local optima, they feature some sort of hill-climbing process that allows degrading the objective value. Such methods include Simulated Annealing (SA), Generalized Simulated Annealing (GSA), Iterated Local Search, Guided Local Search, etc. [6]. The main advantage of these methods is their rapidity, but an important limitation is their tendency to get stuck in local optima [11]. Somes types of local search metaheuristics rely on some kind of memory of visited solutions to try circumvent this limitation such as Tabu Search [6].

Another category of metaheuristics is the **population-based algorithms**. These methods rely on a set of solutions, called the population. The search space is explored by evaluating each solution and modifying them using a set of simple rules. There are two sub-groups in this category: evolutionary and other nature-inspired methods.

Evolutionary algorithms (EA) are iterative methods centered around the notion of *fitness*. The fitness of a solution represents the quality of this solution based on the objective function. During each iteration, called a *generation*, the fitness of each solution is evaluated. Solutions that feature a high enough fitness value are kept for the next generation, all other are discarded. New solutions are generated by stochasticaly crossing over and modifying (mutating) the solutions that were kept after the selection process. This category includes methods such as Genetic Algorithms, Differential Evolution (DE) and Genetic Programming [6,9]. Contrary to local search methods, EAs explore the search space more thoroughly with bigger population sizes and thus are a lot less susceptible to get stuck in local optima. However, they require more computing power and show slower resolution times.

Other population-based methods behave differently from EAs. They still rely on a set of solutions but draw inspiration from complex biological systems such as bird flocking or ant colonies. They feature the same advantage as EAs, i.e. a more thorough exploration of the search space than local search, but still suffer from the same drawbacks of longer computation times and high computing power requirements [6]. Some methods such as Particle Swarm Optimization (PSO) also suffer from a tendency to converge towards local optima because of a poor distribution of information in the population [14].

In our method we propose to combine the speed of local search approaches and the distribution of information of population-based methods. To achieve this goal we borrow the notions of neighborhood and collective reasoning from

these methods. Based on the assumption that **the dynamics of the objective function do not change significantly between two very close points**, we propose a system that **searches for a global optimum through the collective reasoning of already visited solutions**.

Local search and population-based metaheuristics were presented with some of their limitations in the context of optimization for on-line calibration. The next section describes our method, CoBOpti, which is evaluated in section 4.

3 CoBOpti: Cooperation-Based Optimization

In this section, we introduce CoBOpti, a Cooperation-Based Optimization metaheurtistic. The method we propose combines the advantages of both local search and population-based algorithms: the speed of the former and the information distribution of the latter.

Section 3.1 describes the general principle of the approach by giving an overview of the different search phases; section 3.2 details the local search process; section 3.3 details the semi-local search process and how it enables getting out of local minima; finally, section 3.4 describes how points cooperate to solve specific situations.

3.1 General Principle

The goal of CoBOpti is to iteratively explore the surface of an objective function in order to reach a global optimum. During each iteration, the system has to determine the next point to explore. A **point** p_i is defined as a pair $p_i = (x_i, o_i)$ where x_i is the value of the single decision variable and o_i is the value of the objective function at x_i. The succession of visited points is called a **chain**. The algorithm is composed of 4 phases (Figure 1).

The algorithm combines two different heursitics: a **local** one (**phases 1, 2 and 3**), which objective is to discover a local minimum, and **semi-local** one (**phase 4**), which uses the set of local minimum already discovered to look for a global minimum.

The goal of **local search** (**phase 1**) is to find a local minimum. Each iteration t starts with a chain containing some already visited points $p(t)$, $p(t-1)$, etc. Among all the points in the chain, the system choose two points to determine in which direction it needs to go (**phases 2 and 3**). This process continues until a local minimum has been found, i.e. the distance along the x axis between the two points with the lowest objective value of the chain is less than ε_{dist}.

The objective of **semi-local search** is to explore the function towards a global minimum. This process has to decide which point $p(t+1)$ to explore based on already visited local minima (**phase 4**). Every time the semi-local search has decided on which point to explore next, a new chain is created and the local search continues from this new point.

The search stops when a visited local minimum has an objective value less than a predefined threshold ε_{obj}.

The notion of chains is important as it isolates clusters of points (black and red dots in figure 1). It is not desirable that distant points interact during the local search process because of potential higher discrepencies between the actual function value and its estimation. Using chains implies that distant points cannot be used together to compute linear approximations during local search and thus mitigates potential errors. Several chains are created during the optimization process.

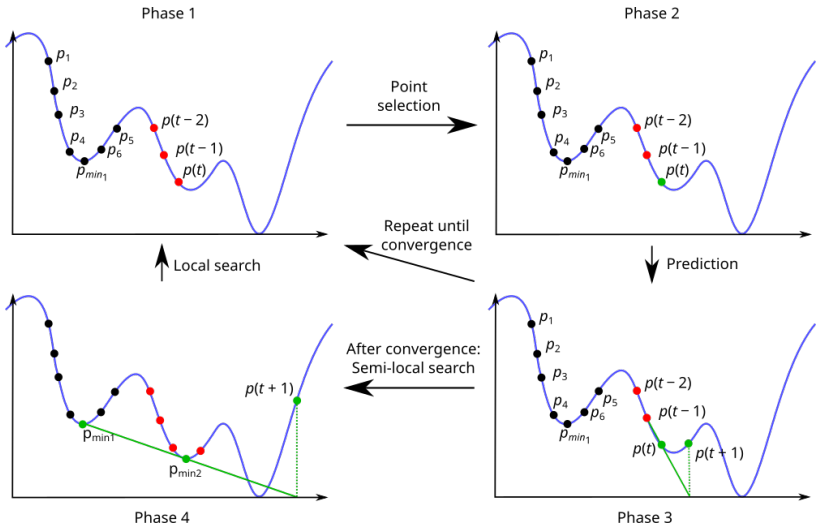


Fig. 1. The search phases of CoBOpti: point selection, local search, higher level search

The following sections detail how points are selected and how $p(t+1)$ is computed. Section 3.2 describes how the local search process selects points to reach a local minima; section 3.3 describes how the system gets out of local minima and searches for a global optimum; finally, section 3.4 describes how points cooperate to solve some difficult situations.

3.2 Local Search

The objective of local search is to follow the curve of the objective function to find a local minimum. At each iteration t, the next point $p(t+1)$ to explore is determined by computing linear approximations of the objective function using two points of the current chain.

Therefore, at each iteration t, two points need to be selected among those in the current chain. The first selected point is the one with the lowest objective value of the chain at time t, noted p_{min}. The second selected point is one of the neighbors of p_{min}. Two points p_1 and p_2 of a chain are said to be **neighbors** if

they are immediately next to each other, i.e. there is no third point p_3 between them along the x axis. A point can have a maximum of two neighbors. For example, in figure 1, points p_1 and p_2 are neighbors but points p_2 and p_4 are not.

As p_{min} is the point with the lowest objective value of its chain, it has either one or two neighbors at any given time.

Phases 2 and 3 of figure 1 illustrate the first situation, where $p(t) = p_{min}$ (green point) has a single neighbor $p(t-1)$. The x component of the next point $p(t+1)$ is computed by a linear approximation of the objective function between $p_{min} = (x_{min}, o_{min})$ and its only neighbor $p(t-1) = p_n = (x_n, o_n)$:

$$x(t+1) = x_n + \frac{-o_n(x_{min} - x_n)}{o_{min} - o_n} \tag{2}$$

This equation returns the x component of the point that would have an objective value of 0 according to the linear approximation of the objective function.

To ensure that the initial assumption on the function's dynamics stays true, the next point cannot be farther than k_{dist} times the distance between p_{min} and p_n. If it is the case, $x(t+1)$ is set to $x_{min} + k_{dist}(x_{min} - x_n)$. In our experiments, $k_{dist} = 5$ was used.

In the second situation, where p_{min} has two neighbors p_l and p_h, as p_{min} is the point with the lowest known objective value, both neighbors have a higher objective value. This implies that a local minimum is somewhere between p_l and p_h. $x(t+1)$ is thus determined by:

$$x(t+1) = \frac{x_{min} + x_n}{2} \tag{3}$$

where x_n is the x component of either p_l or p_h alternatively. Figure 1 shows an example of this situation (black points). The point p_6 was computed this way, using points p_4 as p_{min} and p_5 as its lowest neighbor.

It should be noted that the objective function value does not need to be re-evaluated at the location of the selected neighbor as it is assumed that it has not changed since it was first evaluated.

This whole process repeats until a local minimum is found. The point p_{min} is considered to be a local minimum when the distance to one of its neighbors is less than ε_{dist}.

3.3 Semi-Local Search

The goal of the semi-local search is to find a global minimum. The way points are selected is similar to what was described in the local search process but defers in some key aspects.

In order to compute x component of the next point $p(t+1)$ using linear approximations of the objective function, two points are selected: the latest local minimum $p_{min1} = (x_{min1}, o_{min1})$ found by the local-search process and one of its neighbors. The **neighbors** of a local minimum are the other adjacent local

minima. As with regular points described in section 3.2, local minima have a maximum of two neighbors.

The selected local minimum can have one or two neighbors. Table 1 describes which neighbor is selected depending on the precise situation, where $p_l = (x_l, o_l)$ (resp. $p_h = (x_h, o_h)$) are neighbors of p_{min1} with a lower (resp. higher) x value.

Table 1. Selected neighbor of p_{min1} depending on the situation

	Situation	Selected neighbor
1	One neighbor p_n	p_n
2	Two neighbors, $o_l < o_{min1} < o_h$	p_l
3	Two neighbors, $o_l > o_{min1} > o_h$	p_h
4	Two neighbors, $o_l < o_{min1}$ and $o_{min1} > o_h$	p_l if $o_l < o_h$, otherwise p_h
5	Two neighbors, $o_l > o_{min1}$ and $o_{min1} < o_h$	p_l if $o_l < o_h$, otherwise p_h

For situations 1, 2, 3 and 4, the next point $x(t+1)$ is computed using equation 2, swapping p_{min} for p_{min1} and $p(t-1)$ for the selected neighbor. Phase 4 of figure 1 illustrates this process for situation 1. In this diagram, there are two known local minima, p_{min1} and p_{min2}, the latter being the newly found one. The next point $p(t+1)$ is estimated using a linear approximation between both local minima. As with the local search, $p(t+1)$ cannot be farther than $k|x_{min1} - x_n|$, if it is the case, the same operations are applied as described in section 3.2.

In situation 5, as both neighbors p_l and p_h of p_{min1} have a higher objective value, a global minimum is probably between p_l and p_h. Equation 3 is used again to determine the next point.

Once $x(t+1)$ has been computed, the local search process resumes from this new point with a new chain.

3.4 Cooperation Mechanisms

Sections 3.2 and 3.3 described the nominal behavior of CoBOpti. The system may encounter a number of special situations during both local and semi-local searches. This section presents cooperation rules to detect and solve them.

Case 1. During local search, when a new chain is created, either because it is the first iteration or the semi-local search created a new one, there is a single point inside the chain. This point thus has no neighbors to compute the next point with. Hence, no linear approximation can be estimated and $x(t+1)$ is directly chosen randomly among $\{x_{min}-\delta, x_{min}+\delta\}$ where $\delta = \frac{1}{k_{prop}}|x_{low}-x_{high}|$ and x_{low} (resp. x_{high}) the lower (resp. higher) bounds of the definition domain of x. In our experiment, $k_{prop} = 100$ was used.

Case 2. During semi-local search, a similar situation may occur where there is only one known local minimum. As there are no neighbors to make linear approximations with, a **hill-climbing** process is initiated to escape the local minimum. This process relies on the two points of the latest chain that have the

lowest and highest x value, called extrema. The goal is to climb up the slopes around the local minimum to find another slope of opposite direction.

The search focuses on the slope where the extremum with the lowest objective value is. The next point is computed using equation 4 where $p_e^l = (x_e^l, o_e^l)$ is the extremum with the lowest objective value and $p_e^h = (x_e^h, o_e^h)$ is the other. $p_n = (x_n, o_n)$ is the neighbor of p_e^l. This equation computes the x value of the next point which would have an objective value equal to that of the highest extremum, according to the linear approximation of the objective function between the lowest extremum and its neighbor.

$$x(t+1) = x_e^l + \frac{(o_e^h - o_e^l)(x_n - x_e^l)}{o_n - o_e^l} \tag{4}$$

At the next iteration, if the actual objective value is higher than o_e^h, the process switches sides; if this is not the case, it continues as is. This process is repeated until the actual objective value is lower than o_e^l. The local search process then resumes with a new chain.

During this hill-climbing phase, the distance $|x(t+1) - x_e^l|$ cannot be smaller than a threshold δ_{min} in order to prevent the process from slowing down too much.

Case 3. It may happen that the local search process finds a local minimum that was already discovered in previous iterations. In order to escape a potential search loop, two decisions may occur. If a hill-climbing phase was previously initiated at this local minimum, the next point $x(t+1)$ is computed again and multiplied by a factor of 2, to explore twice as far and explore a new area. On the contrary, if no hill-climbing phase was ever initiated at this local minimum, one is started, in hopes to find a new adjacent valley.

Two local minima are considered to be identical if their distance along the x axis is less than a threshold ε_{same}.

In this section we presented our approach. It relies on the notion of chains of points. We first presented a local search process on a chain that allows finding local optima. When a local optimum is found, a semi-local search process allows finding new regions of the search-space to explore. Cooperation mechanisms were introduced to account for special situations, diversify the solutions and create new chains.

In the next section we evaluate the performances of our method. We compare it to four other local-search and population-based metaheuristics on unconstrained mono-objective optimization problems.

4 Experiments and Results

This section compares the performances of CoBOpti with four other methods cited in section 2: Simulated Annealing (SA), Generalized Simulated Annealing (GSA), Differential Evolution (DE) and Particle Swarm Optimization (PSO).

Section 4.1 presents the different test functions used to test the performances; section 4.2 describes the protocole for comparing the performances of CoBOpti

with other selected methods; section 4.3 presents the results of the experiments; finally, results are discussed in section 5.

4.1 Test Functions

For the performance comparison experiments, four functions have been selected: Gramacy and Lee (domain: $[0.5, 2.5]$), Ackley (parameters: $d = 1$, $a = 20$, $b = 0.2$, $c = 2\pi$; domain: $[-32, 32]$), Rastrigin (parameter: $d = 1$; domain: $[-5.12, 5.12]$) and Levy function (parameter: $d = 1$; domain: $[-10, 10]$). These functions have been chosen because they feature many local minima, a single global minimum, and a single parameter [1,7,10,13].

4.2 Methods Comparison

The performances of each approach (SA, GSA, DE and PSO) are compared against CoBOpti's. They were all implemented in Python 3.8. GSA and DE were implemented using the `scipy.optimize.dual_annealing` and `scipy.optimize.differential_evolution` functions, PSO was implemented with `pyswarm.pso` package, and SA was a custom implementation. For GSA, DE and PSO, all optional parameters excepts those related to bounds, initial state and maximum number of iterations were let to their default value.

Control variables of CoBOpti are set as follows: $\varepsilon_{dist} = 10^{-4}$ (local minimum detection threshold), $\varepsilon_{same} = 0.01$ (minimum distance between local minima), $\delta_{min} = 10^{-4}$ (minimum step size during hill climbing phase), and $\varepsilon_{obj} = 5 \cdot 10^{-3}$ (precision threshold for global minimum objective value).

For every method, except PSO, the initial value v_{init} for each decision variable in a single run is selected by a Sobol Sequence. As values generated by this sequence are all in the $[0, 1]$ interval, they are adjusted to the variable's domain using the formula $v_{init} = s \cdot (d_{max} - d_{min}) + d_{min}$ where s is a value generated by the sequence. We did not specify v_{init} values for PSO as the implementation we used did not allow it.

Three metrics are defined: **success rate**, i.e. the ratio of executions that found the global minimum, **number of iterations**, **number of evaluations of the objective function**.

4.3 Results

Table 2 shows the success rate, mean number of iterations and function evaluations over 200 executions for each method and function, with a maximum of 1000 iterations.

CoBOpti was able to find the global minimum for all four functions. It took on average between 35 and 100 iterations to find the global minimum with a similar number of objective function evaluations.

The constant 1000 iterations for SA and GSA are explained by their stopping criterion. These methods rely on the number of elapsed iterations to compute

probability distributions: the more iterations have passed, the less likely the algorithm is to select a non-improving move. Once the allowed number of iterations has passed, no more non-improving moves can be selected and the algorithm stops. The visited point with the lowest objective value is then returned.

SA did not yield good results, except for Gramacy and Lee's function with nearly 100 % of success rate. It yielded very poor results for Ackley function with only 2 %. These results are coherent with what was described in the review (section 2).

GSA yielded very good results with 100 % on all functions. The number of objective function evaluations was two times higher than SA, around 2000.

DE's success rate is a bit lower than other methods except for SA. However, the mean number of iteration is quite low, staying between 8 and 50.

PSO was able to find the global minimum in all four cases with a low mean number of iterations, between 20 and 50. However, the mean number of function evaluations is higher than other methods, ranging from 2000 to more than 4500.

Table 2. Success rates, average number of iterations and objective function evaluations of tested methods

Method	**Function**	**Success rate**	**# of iterations**	**# of evaluations**
CoBOpti	G. & L.	100 %	49.31	50.31
	Ackley	100 %	95.94	96.94
	Rastrigin	100 %	80.69	81.69
	Levy	100 %	35.3	36.3
SA	G. & L.	99.5 %	1000	1000
	Ackley	2 %	1000	1000
	Rastrigin	10.5 %	1000	1000
	Levy	30 %	1000	1000
GSA	G. & L.	100 %	1000	2035.58
	Ackley	100 %	1000	2124.43
	Rastrigin	100 %	1000	2039.97
	Levy	100 %	1000	2019.60
DE	G. & L.	97.5 %	8.71	154.54
	Ackley	100 %	49.62	801.63
	Rastrigin	94 %	30.91	481.06
	Levy	100 %	50.45	773.75
PSO	G. & L.	100 %	20.61	2008.70
	Ackley	100 %	46.92	4638.51
	Rastrigin	100 %	25.57	2505.57
	Levy	100 %	20.02	1951.32

5 Analysis and Discussion

The initial assumption of continuity in function dynamics has been validated by the experiments on several standard functions. CoBOpti showed better success

rates than SA and DE, and nearly as good as GSA and PSO. Although the number of iterations of CoBOpti is comparable to that of DE and PSO, its number of function evaluations is several orders of magnitude lower.

This low number of objective function evaluations can be attributed to the fact that the objective function is evaluated only once per visited point. This behavior stems from the initial assumption that states that the dynamics of the objective function does not change significantly between two close points.

Execution times were not shown as differences between methods were not significant. This is most likely due to the relatively low complexity of the selected functions.

A sensitivity analysis should be done to test the influence of k_{dist} and k_{prop} on CoBOpti's performances.

CoBOpti was only tested on mono-objective optimization problems with a single decision variable. Further research is needed to generalize this approach to multi-objective global optimization problems with multiple decision variables. The core principle should stay similar to what was presented in this paper. New cooperation mechanisms should be added to select which objective to minimize and which decision variables to tune at each cycle.

Other experiments could be conducted with other complex functions. As real-world applications are subject to noisy data, resilience to such noise has to be tested.

6 Conclusion

In this paper, CoBOpti, a new metaheuristic for global optimization, was presented. It is based on a hypothesis of local continuity of the dynamics of the objective function. CoBOpti explores the search space by relying on the cooperation of visited solutions based on this hypothesis.

This paper focuses on mono-objective global optimization problems with a single decision variable. Experiments showed that CoBOpti needs less objective function evaluations than other common metaheurstic methods while maintaining similar or better success rates on 1D-functions.

CoBOpti is a promising proposition for use in on-line calibration. Indeed, its low number of objective function evaluations would be useful in the context of on-line calibration of complex simulation models with computationally intensive objective functions. This property could help reduce the time required to calibrate these kinds of models.

References

1. Alauddin, M.: Mosquito flying optimization (MFO). In: 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT). pp. 79–84 (Mar 2016). https://doi.org/10.1109/ICEEOT.2016.7754783
2. Arsenault, R., Poulin, A., Côté, P., Brissette, F.: Comparison of Stochastic Optimization Algorithms in Hydrological Model Calibration. Journal of

Hydrologic Engineering **19**(7), 1374–1384 (Jul 2014). https://doi.org/10.1061/(ASCE)HE.1943-5584.0000938, https://ascelibrary.org/doi/abs/10.1061/(ASCE)HE.1943-5584.0000938, publisher: American Society of Civil Engineers
3. Cho, J.H., Wang, Y., Chen, I.R., Chan, K.S., Swami, A.: A Survey on Modeling and Optimizing Multi-Objective Systems. IEEE Communications Surveys Tutorials **19**(3), 1867–1901 (2017). https://doi.org/10.1109/COMST.2017.2698366, conference Name: IEEE Communications Surveys Tutorials
4. Fan, C., Zhang, C., Yahja, A., Mostafavi, A.: Disaster City Digital Twin: A vision for integrating artificial and human intelligence for disaster management. International Journal of Information Management **56**, 102049 (Feb 2021). https://doi.org/10.1016/j.ijinfomgt.2019.102049, https://www.sciencedirect.com/science/article/pii/S0268401219302956
5. Gendreau, M., Potvin, J.Y.: Tabu Search. In: Burke, E.K., Kendall, G. (eds.) Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, pp. 165–186. Springer US, Boston, MA (2005). https://doi.org/10.1007/0-387-28356-0_6, https://doi.org/10.1007/0-387-28356-0_6
6. Gendreau, M., Potvin, J.Y. (eds.): Handbook of Metaheuristics, International Series in Operations Research & Management Science, vol. 146. Springer US, Boston, MA (2010). https://doi.org/10.1007/978-1-4419-1665-5, http://link.springer.com/10.1007/978-1-4419-1665-5
7. Gramacy, R.B., Lee, H.K.H.: Cases for the nugget in modeling computer experiments. Statistics and Computing **22**(3), 713–722 (May 2012). https://doi.org/10.1007/s11222-010-9224-x, https://doi.org/10.1007/s11222-010-9224-x
8. Ma, J., Dong, H., Zhang, H.M.: Calibration of Microsimulation with Heuristic Optimization Methods. Transportation Research Record **1999**(1), 208–217 (Jan 2007). https://doi.org/10.3141/1999-22, https://doi.org/10.3141/1999-22, publisher: SAGE Publications Inc
9. Opara, K.R., Arabas, J.: Differential Evolution: A survey of theoretical analyses. Swarm and Evolutionary Computation **44**, 546–558 (Feb 2019). https://doi.org/10.1016/j.swevo.2018.06.010, https://www.sciencedirect.com/science/article/pii/S2210650217304224
10. Potter, M.A., De Jong, K.A.: A cooperative coevolutionary approach to function optimization. In: Davidor, Y., Schwefel, H.P., Männer, R. (eds.) Parallel Problem Solving from Nature — PPSN III. pp. 249–257. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (1994). https://doi.org/10.1007/3-540-58484-6_269
11. Storn, R., Price, K.: Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. Journal of Global Optimization **11**(4), 341–359 (Dec 1997). https://doi.org/10.1023/A:1008202821328, https://doi.org/10.1023/A:1008202821328
12. Sörensen, K., Sevaux, M., Glover, F.: A History of Metaheuristics. Handbook of Heuristics **to appear** (Jan 2017)
13. Valdez, F., Melin, P.: Parallel Evolutionary Computing using a cluster for Mathematical Function Optimization. In: NAFIPS 2007 - 2007 Annual Meeting of the North American Fuzzy Information Processing Society. pp. 598–603 (Jun 2007). https://doi.org/10.1109/NAFIPS.2007.383908
14. Zhang, Y., Wang, S., Ji, G.: A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications. Mathematical Problems in Engineering **2015**, e931256 (Oct 2015). https://doi.org/10.1155/2015/931256, https://www.hindawi.com/journals/mpe/2015/931256/, publisher: Hindawi

Training large convolutional neural networks through population-based meta-heuristics

M. Lupión[1], N.C. Cruz[2], and P.M. Ortigosa[1]

[1] Dept. of Informatics, University of Almería, ceiA3 Excellence Agri-food Campus, Almería, Spain
marcoslupion@ual.es, ortigosa@ual.es
[2] Dept. of Computer Architecture and Technology–CITIC, University of Granada, Granada, Spain
ncalvo@ual.es

1 Introduction

Nowadays, machine learning algorithms are applied to a huge variety of problems such as data classification, anomaly detection and data generation, obtaining impressive results [14]. Among them, artificial neural networks [6] aim to simulate the behavior of the human brain, and thanks to the current computation capabilities, they can learn to solve complex problems in an acceptable time. More specifically, Convolution Neural Networks (CNN) perform well on images due to the convolution operation carried out during the data processing.

However, these algorithms require training. The most used method is backpropagation, which calculates the gradient of each param of the network to find the values that minimize the error of the system on the dataset. Thus, this method is heavy computationally demanding and may converge to a local optimum [4][5], which makes training a difficult task to accomplish.

In order to solve this problem, population-based meta-heuristic algorithms have been are proposed to train neural networks [9]. A priori, they can find global optimal without being trapped at local ones, so they are being incorporated in some works to optimize the weigths and structure of neural networks [2][12]. More specifically, population-based optimizers consist of individuals (weights in this case), which aim to find the best value in a search space [10]. Among population-based algorithms, Teaching-Learning-Based Optimization (TLBO) simulates a teaching-learning process based on the influence of a teacher on the performance of learners in a class [11]. It may fit for weight optimisation as it supports high-dimensional problems [13]. Also, the user is not required to set any other parameter than the population size and the number of cycles, which differs from other population-based meta-heuristics.

In other works, the meta heuristic algorithms are mainly applied in lightweight neural networks having a small quantity of parameters [1][2]. In those cases, the optimisation algorithms worked well. However, in our case, we aim to find a solution to the problem of big neural networks, having more than 50.000k weights as in real applications, neural networks have a great quantity of parameters.

In this work, the weights of a fully-connected convolutional neural network are optimised by using the TLBO algorithm. They are compared to the performance achieved with backpropagation through stochastic-gradient descent. Finally, a hybrid method is proposed. It uses some solutions of backpropagation to define the initial population of TLBO and lets the latter try to improve them.

2 Methodology

The application of this work is an image classification task. The dataset is CIFAR10, a well-known dataset which consists of 60.000 images; 50.000 are used for training and 10.000 are left for testing. Their dimensions are 32x32x3 and there are 10 classes in total [7]. A CNN neural network adopting the LeNet5-CNN architecture has been adopted in the experiments [8]. It has 71.919 parameters in total, so it is a small network compared to other instances.

The objective function of this algorithm is the cross entropy value of the proposed neural network after evaluating the training dataset. However, only the best individual is changed if the cross entropy value in the validation dataset improves, avoiding overfitting. Regarding the initial population, TLBO initializes its individuals with values in the ranges defined by the user. For

neural networks, the most widespread weights initialization techniques are Xavier and He. These methods take into account the number of neurons before and after the layer, so depending on the layer, the range of the initial weights varies. In this case, the initial population was randomly set according to the Xavier initializer limits. However, after preliminary experimentation, TLBO was confirmed to be unable to find competitive solutions from a completely random start.

Therefore, an hybrid approach (Figure 1) is proposed to adapt the TLBO to this highly dimensional problem.

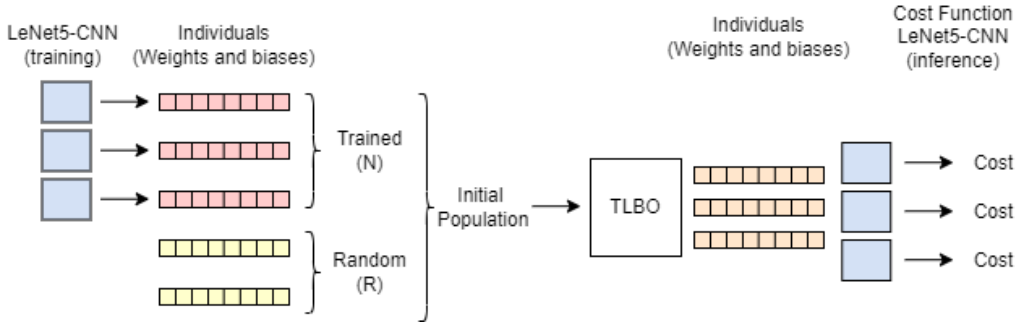


Fig. 1. Hybrid approach

Accordingly, TLBO will start with a population with two sorts of individuals. The first group consists of R individuals randomly generated within the Xavier limits. The second one contains N individuals resulting from instances of the desired neural network trained using backpropagation and stochastic gradient descend. Their training finishes when the model validation accuracy maintains the same value for several epochs, or it starts to decrease. This indicates that the weights are trained and have a good accuracy on the train and validation datasets. However, if the training continues, the network could be overfitted and perform bad on validation dataset.

After initialization, the TLBO executes the teacher and learner stages in its main iteration loop[15]. In the teacher phase, the best individual becomes the teacher and the rest remain as students. Then, a vector containing the mean of each dimension along all the individuals is calculated, and the students are updated taking into account this vector, the values of the teacher, and some random factors. The changes are only kept if they result in a better solution, while non-improving individuals are not ultimately altered. In the learning phase, the students interact with each other in pairs. It has a local scope and lets the best member of the pair act as the teacher. These phases occur as many times as the number of *cycles* defined by the user.

Finally, the best solution achieved so far is returned as the neural network configuration proposed.

The implementation of TLBO used has been obtained from [3]. It is programmed in C and designed to run in parallel by using the MPI and OpenMP libraries in multicores. Concerning the evaluation of the cost function, a neural network framework has been designed and implemented from scratch in C. The previous training of LeNet5-CNN neural networks which build the initial population is carried out using the Tensorflow framework. Therefore, a custom function that translates the weights and biases from Tensorflow to the custom framework in C has been developed.

3 Preliminary results

In order to evaluate the hybrid system, the system was launched with the following configuration: A total of 1.000 images were used to train the neural network and 100 images were left to evaluate. The initial population was set to 1.000, where 50 individuals were the result of 50 neural networks trained with Tensorflow. These neural networks were trained with the images used to train the hybrid system. Their stopping condition was set to 5 epochs without improving the validation loss. In the most cases, the training lasted between 12 and 13 epochs. The validation accuracy in the

trained neural networks had an average of 45%. This is caused by the small quantity of images used to train.

Regarding the configuration of TLBO, the total number of cycles was set to 500. In addition, 15 cycles without improving is set as stopping condition. In the results, the train of the system results to be heavy to computationally expensive. It lasted 76.5 hours, stopping in the cycle number 49. Nevertheless, the final results, i.e., sets of weights and biases found, outperform those achieved by letting the networks train in TensorFlow and backpropagation.

For future work, the system will be tested with different datasets and neural network architectures.

4 Acknowledgments

This research has been funded by the R+D+i project RTI2018-095993-B-I00, financiated by MCIN/AEI/10.13039/501100011033/ and FEDER "Una manera de hacer Europa"; by the Junta de Andalucía with reference P18-RT-1193; by the University of Almería with reference UAL18-TIC-A020-B and by the Department of Computer Science of the University of Almería. Marcos Lupión Lorente is a beneficiary of the program "Formación del Profesorado Universitario" with reference number (FPU19/02756). N.C. Cruz is supported by the Ministry of Economic Transformation, Industry, Knowledge and Universities from the Andalusian government.

References

1. A. Alsaeedi, A. Aljanabi, M. Manna, and L. Adil. A proactive metaheuristic model for optimizing weights of artificial neural network. *Indonesian Journal of Electrical Engineering and Computer Science*, 20, 11 2020.
2. R. Bousmaha, R. M. Hamou, and A. Amine. Optimizing connection weights in neural networks using hybrid metaheuristics algorithms. *International Journal of Information Retrieval Research (IJIRR)*, 12(1):1–21, 2022.
3. N. C. Cruz, J. L. Redondo, J. D. Álvarez, M. Berenguel, and P. M. Ortigosa. A parallel teaching–learning-based optimization procedure for automatic heliostat aiming. *The Journal of Supercomputing*, 73(1):591–606, 2017.
4. P.O. Glauner. Comparison of training methods for deep neural networks. *ArXiv*, abs/1504.06825, 2015.
5. L.K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.
6. N. Kriegeskorte and T. Golan. Neural network models and deep learning. *Current Biology*, 29(7):R231–R236, 2019.
7. A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.
8. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
9. V.K. Ojha, A. Abraham, and V. Snášel. Metaheuristic design of feedforward neural networks: A review of two decades of research. *Engineering Applications of Artificial Intelligence*, 60:97–116, 2017.
10. R. Ramos-Pollán, M. A. Guevara Lopez, and N.G. Posada. Optimizing the area under the roc curve in multilayer perceptron-based classifiers. 2011.
11. R.V. Rao, V.J. Savsani, and D.P. Vakharia. Teaching–learning-based optimization: An optimization method for continuous non-linear large scale problems. *Information Sciences*, 183(1):1–15, 2012.
12. L.M. Rere, M.I. Fanany, and A. Arymurthy. Metaheuristic algorithms for convolution neural network. *Computational Intelligence and Neuroscience*, 2016, 06 2016.
13. S. Satapathy, A. Naik, and K. Parvathi. High dimensional real parameter optimization with teaching learning based optimization. *International Journal of Industrial Engineering Computations*, 3(5):807–816, 2012.
14. N. Sharma, R. Sharma, and N. Jindal. Machine learning and deep learning applications-a vision. *Global Transitions Proceedings*, 2(1):24–28, 2021. 1st International Conference on Advances in Information, Computing and Trends in Data Engineering (AICDE - 2020).
15. J.L. Torres-Moreno, N.C. Cruz, J.D. Álvarez, J.L. Redondo, and A Giménez-Fernandez. An open-source tool for path synthesis of four-bar mechanisms. *Mechanism and Machine Theory*, 169:104604, 2022.

Design and preliminary results of a new stochastic meta-heuristic for derivative-free optimization

N.C. Cruz[1], J.L. Redondo[2], E.M. Ortigosa[1], E. Ros[1], and P.M. Ortigosa[2]

[1] Dept. of Computer Architecture and Technology–CITIC, University of Granada, Granada, Spain
ncalvocruz@ual.es, ortigosa@ugr.es, eros@ugr.es
[2] Dept. of Informatics, University of Almería, ceiA3 Excellence Agri-food Campus, Almería, Spain
jlredondo@ual.es, ortigosa@ual.es

1 Introduction

Optimization problems usually arise in fields such as Architecture, Engineering, and Applied Sciences in general [2–4]. Roughly speaking, this kind of problem requires finding the extreme points (maxima or minima, depending on the type) of a function, which involves different variables and models some aspect. For example, the function, known as the objective function in this field, can represent the cost of a building depending on the sort of materials and quantities. In this situation, it can be assumed that the points sought will be the minima, i.e., the values of the variables that result in the minimum cost. Optimization is also valuable for model tuning: the parameters become variables, and the objective function compares the achieved and expected outputs [3].

Depending on the objective function, constraints and variables (e.g., continuous or discrete), there exist different types of optimization problems and methods to face them. For instance, linear objective functions and constraints with real bounded variables generally result in problems relatively easy to solve [1]. However, this is not always the case, especially when the functions involved do not exhibit a closed analytical form or do not have exploitable mathematical properties (such as linearity and a convex search space). Fortunately, there exist methods with fewer problem requirements. They usually rely on intuitive ideas (heuristics) to obtain acceptable results [7].

The optimization problem that centers the attention of this work can be formulated as follows:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & L_i \leq x_i \leq U_i,\ i = 1, \ldots, N. \end{aligned} \tag{1}$$

where f is a N-dimensional objective function, i.e., $f : \mathbb{R}^N \to \mathbb{R}$, and x refers to any input in $\mathbb{R}^N$ belonging to the domain $[L_1, U_1] \times \ldots \times [L_N, U_N]$. As can be seen, the defined problem only consists of the objective function and the bounds of each variable. There is no additional information about the mathematical properties of f (e.g., convexity, multimodality and smoothness), which can only be evaluated, and the search space is defined by an N-dimensional domain. Therefore, this problem can be classified as a black-box optimization with box constraints [2, 4], and it is suitable for model tuning applications [3, 6], which is one of the expected targets of the proposed method.

In the literature, there are multiple population-based meta-heuristics that can be applied to the problem defined above [7]. Traditional genetic algorithms, Differential Evolution [3], and UEGO [6] are valid examples. However, they have multiple parameters, so they require fine parameter tuning. TLBO [3] avoids this problem as a population-based method that only expects the population size and number of cycles. Regardless, this sort of method will generally need numerous function evaluations to converge due to its haphazard coverage of the search space. Another option especially conceived for black-box optimization is DIRECT [4], which is deterministic in contrast to the previous ones, and it has no other parameters than the number of function evaluations and a tolerance factor that has little effect on its performance. However, its rectangle division behavior is rigid, and depending on the problem, it may require many function evaluations.

This work proposes an optimizer that aims to find a trade-off between the lack of parameters, stochasticity, and a broad exploration of the search space. For this purpose, it combines randomness and the distribution of different search windows linked to a local optimizer from UEGO, with the strict division rules and lack of parameters from DIRECT. The resulting method is called Tangram and can be seen as a rule-based multi-start component linked to a local optimizer. The one selected for now is SASS [5], a stochastic hill-climber with a robust default configuration. Tangram is expected to be effective for low-dimensional problems (less than 20 variables).

2 Method description

Tangram only expects the objective function, whose input is scaled from $\mathbb{R}^N$ to $[0,1]^N$, and the number of function evaluations allowed. As it occurs with DIRECT, normalization allows visualizing the search space as the unit hypercube [4].

The proposed method starts by evaluating the center of the search space, i.e., $(0.5,\ldots,0.5) \in \mathbb{R}^N$, which becomes the current result. This defines its initialization stage. After that, it executes its main loop while there are function evaluations remaining. The loop consists of these stages:

Global phase: The local search component, SASS, is launched from the current result to try to improve it. This optimizer is configured so that the maximum step size is equal to the diameter of the search space. This strategy allows reaching any solution and comes from the way in which UEGO handles its initial or first-level species. Figure 1a depicts this process starting from the initial point, i.e., the center of the search space, for a hypothetical 2D problem.

Division: From the current result, the midpoint between it and each corner of the search space is computed and evaluated. This part aims to emphasize the exploration of the whole search space and allows escaping from local optima. It is vaguely inspired in how DIRECT keeps a representing point of every region of the search space, even though Tangram may not consider fully disjoint zones. Figure 1b shows this phase in the same previous context. Additionally, notice that if the resulting polygons were colored in different colors as if they were solid pieces, they would resemble a Tangram, i.e., the Chinese dissection puzzle [8].

Local phase: The local search component, SASS, is launched from each of the previous midpoints to try to improve them. Those featuring a better initial value for the objective function go first. This order of execution aims to ensure exploring the most promising regions at least, in case that the function evaluation budget is consumed before considering all of them. In contrast to the global phase, for this one, the optimizer is configured not to take steps bigger than the distance between the starting point and the corner used to define it. Regardless, notice that the local search updates its current point every time that it finds a better point, so the division is not rigid as introduced.

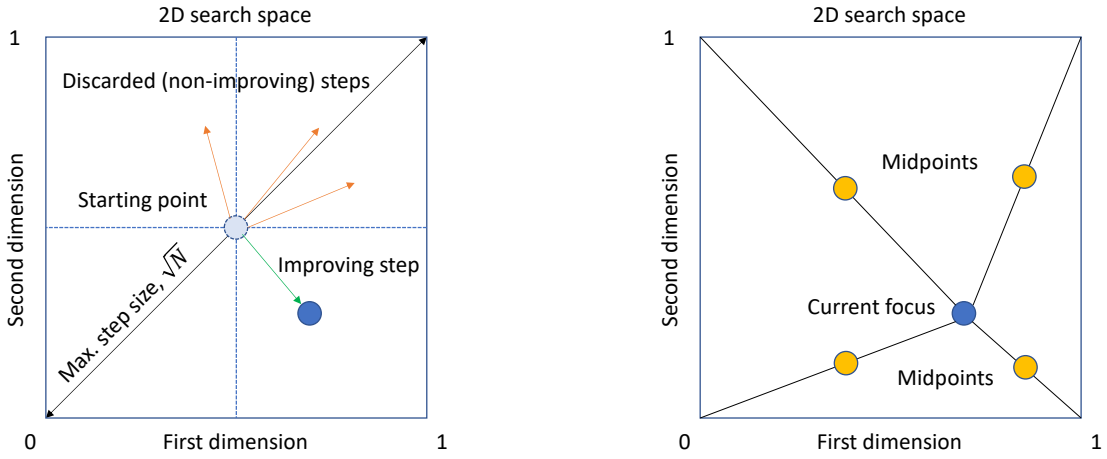
Update: If any of the points found after the division and local searches is better than the current solution, that point replaces it. The method then returns to the global phase and executes another full iteration, as long as there are function evaluations available.

After consuming all the function evaluations allowed, Tangram returns the best solution achieved so far. Notice that the method may run out of function evaluations at any point. Then, it will assign an infinite value to any new point and try to end as soon as possible.

Concerning the local search component, SASS only requires the objective function to be fully defined in the search space. It starts at any given point and randomly decides a direction to move. The jump size cannot exceed a given size, which depends on the phase as previously explained, and it is scaled based on the number of improving and non-improving (discarded) movements. Movements result from adding a normally-distributed random perturbation vector with a standard deviation between 1e−5 and 1, starting at the upper bound and ultimately rescaled by the maximum step size. The standard deviation is doubled after five consecutive successful movements or halved after three consecutive discarded ones. This is the recommended configuration, and it is known to perform well. Similarly, SASS will terminate after 32 iterations, which is assumed enough to converge to the nearby optima according to previous knowledge on the method when used with UEGO. Nevertheless, varying this local budget would just result in a second parameter to tune.

3 Preliminary results and future work

The 20 box-constrained continuous problems proposed in [2], featuring between 1 to 10 variables, have been addressed with Tangram. It is compared to a random search and TLBO, which is known to be simple to tune and effective [3]. The criterion to allow function evaluations in that work has been maintained herein, i.e., $30(N+1)$. As the authors say, these problems are challenging for black-box methods not exploiting any analytical information. They get even harder considering the low number of function evaluations allowed, which is compatible with a context where the cost function is computationally demanding (e.g., simulation-based model tuning). The development



(a) Global phase (b) Division phase

Fig. 1. Depiction of the main concepts of Tangram.

environment used is MATLAB 2020a in Mac OSX (MacBook Pro, Intel i5 2.9 GHz, 8 GB of RAM). Due to stochasticity, each method has been executed 200 independent times for each problem.

Tangram achieved the best average in 17 out of 20 problems, while TLBO was the best in the 3 remaining. More specifically, TLBO only won in those problems with more variables, i.e., one of 8 and two of 10 variables. For them, it seems that Tangram cannot fully explore the regions before running out of function evaluations as the number of corners increases exponentially, yet the budget does linearly. Finally, as expected, the random search did not win any benchmark.

For future work, based on the results achieved, we will try to adjust some parts of Tangram related to the distribution of function evaluations. We also intend to increase the number of benchmark problems and the methods compared, possibly starting with DIRECT and UEGO.

4 Acknowledgments

This work has been supported by the Spanish Ministry of Science through the projects RTI2018-095993-B-I00, financed by MCIN/AEI/10.13039/501100011033/ and FEDER "*Una manera de hacer Europa*", and INTSENSO (MICINN-FEDER-PID2019-109991GB-I00), by Junta de Andalucía through Projects UAL18-TIC-A020-B, CEREBIO (P18-FR-2378) and P18-RT-1193 by the European Regional Development Fund (ERDF). N.C. Cruz is supported by the Ministry of Economic Transformation, Industry, Knowledge and Universities from the Andalusian government.

References

1. S. Boyd, S.P. Boyd, and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.
2. A. Costa and G. Nannicini. RBFOpt: an open-source library for black-box optimization with costly function evaluations. *Mathematical Programming Computation*, 10(4):597–629, 2018.
3. N.C. Cruz, M. Marín, J.L. Redondo, E.M. Ortigosa, and P.M. Ortigosa. A comparative study of stochastic optimizers for fitting neuron models. Application to the cerebellar granule cell. *Informatica*, 32:477–498, 2021.
4. D.R. Jones and J.R.R.A. Martins. The DIRECT algorithm: 25 years later. *Journal of Global Optimization*, 79(3):521–566, 2021.
5. A. Lančinskas, P.M. Ortigosa, and J. Žilinskas. Multi-objective single agent stochastic search in non-dominated sorting genetic algorithm. *Nonlinear Analysis: Modelling and Control*, 18(3):293–313, 2013.
6. M. Marín, N.C. Cruz, E.M. Ortigosa, M.J. Sáez-Lara, J.A. Garrido, and R.R. Carrillo. On the use of a multimodal optimizer for fitting neuron models. Application to the cerebellar granule cell. *Frontiers in Neuroinformatics*, 15, 2021.
7. Saïd Salhi. *Heuristic search: The emerging science of problem solving*. Springer, 2017.
8. F.T. Wang and C.C. Hsiung. A theorem on the Tangram. *The American Mathematical Monthly*, 49(9):596–599, 1942.

Research Issues in Adversarially Robust Stream-Based Federated Learning

Abinash Borah, Dimitrios I. Diochnos, Le Gruenwald, Elaheh Jafarigol, Egawati Panjei, and Theodore B. Trafalis

The University of Oklahoma
{abinashborah, diochnos, ggruenwald, elaheh.jafarigol, egawati.panjei, ttrafalis}@ou.edu

1 Introduction

During the last fifteen years, we have seen a tremendous increase of computing devices in our everyday lives. This influx of devices has allowed novel applications of machine learning and has raised issues that were not originally foreseen when the field of machine learning was being formed. In particular, we can see applications that combine learning in a networked environment (*federated learning*) as well as learning through continuous streams of data (*stream-based learning*). In this work we want to identify research issues by taking into account one more dimension of modern machine learning which has not been explored in this joint context: *adversarially robust* models.

Federated Learning. In a federated learning scenario [5] we have different computing nodes, called *clients*, interconnected in a network. These nodes are trying to accomplish a common supervised-learning task using a common predictive mechanism (e.g., a neural network) by forming *local models* that are obtained using *local training data*. The local data typically follow different distributions on the different nodes. The goal is to learn a global model by sharing information about the locally learnt models between the clients, but not by sharing local training data as this could violate privacy constraints. The most common approach for federated learning is that of a star topology (see Figure 1), where a central server orchestrates the learning process of a global model by receiving information from local models that reside on the clients, which in turn have been trained from local *private* data. It is also possible to have tree-like hierarchical or even decentralized topologies with the same ultimate goal of learning a common global model; however, these approaches have been less explored.

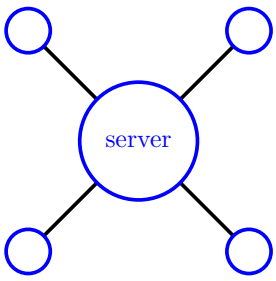


Fig. 1. Typical topology in a federated learning setup. The central node plays the role of the *server* that maintains a global model, while the *client* nodes at the edges of the star topology maintain a local model that relies on data that follow distributions that are specific to those nodes.

Stream-Based Learning. In a stream-based learning scenario [6] the learner is faced with a continuous stream of training data. This continuous stream of data poses two main issues to the learning process: (a) the learner cannot possibly hold all the training data in the main memory (because of the vast amount of the data in the long run), and (b) the learner should be able to deal with *concept drift* (changes in the data-generating distribution). In addition, the arrival rate of the training data poses further processing issues to the learning mechanism, should the learner want to update the learnt model by using the information of all the data that is presented to them.

Applications of Stream-Based Federated Learning. There are several applications under the umbrella of stream-based federated learning. Such applications include text prediction on mobile devices [8], healthcare applications using wearable devices [3], control of signal lights using computing devices and cameras on traffic lights [11], anomaly detection on sensors used for *smart homes*, and *smart applications* in general [17], to name a few.

2 The Setup: Adversarially Robust Stream-Based Federated Learning

Another property that is sought for in modern machine learning classifiers is that of *adversarial robustness*. While the field has its roots in studying (rigorously or experimentally) different noise models (see, e.g., [18] for some popular noise models), nevertheless, in the last decade the field has grown by including broader *training-time* as well as *test-time* attacks that are orchestrated by adversaries of varying strengths; see, e.g., [16], for a taxonomy of these recent methods.

Training-time attacks are typically known as *poisoning attacks* as an adversary is contaminating the training set that is subsequently used by the learner and the purpose of the adversary is to cause an increase in the error rate (or, lower the confidence) for the predictions of the trained model. Such attacks can be either *targeted* or *indiscriminate*, depending, respectively, (a) on whether the adversary aims to increase the error rate (and/or lower the confidence) of the trained model *on a particular (set of) test examples* that the adversary has in mind, or (b) the adversary aims to increase the error rate (lower the confidence) *in general*, as test data is drawn from some underlying distribution. Test-time attacks are known as *evasion attacks*, or as the field of *adversarial examples*. Furthermore, adversaries may coordinate the attacks between the training phase and the test phase, resulting in either *backdoor attacks* [7], or *hybrid attacks* [4]. However, the important assumption in the joint setting that we want to study is the following one.

Assumption 1 *Due to the distributed nature of federated learning, adversaries may have under their control a certain fraction of the total number of clients of the network and as a consequence these nodes can be vulnerable to noise, poisoning attacks, or adversarial examples.*

The above assumption, together with the special circumstances and constraints that stream-based learning induces in the whole learning process, raise several issues regarding adversarial robustness of the global and the local models, which we discuss in the following section.

3 Core Research Issues

Studying *training-time attacks* in networked environments has led to some interesting results. For example, while in traditional single-source learning a learner cannot tolerate malicious noise that has a rate of at least $\frac{\varepsilon}{1+\varepsilon}$ [9], in the case of multiple sources the learner can still generate a model that has error rate less than ε as long as less than half of the sources are maliciously changed [10], assuming that the distributions on the different sources are identical and that the nodes may share actual examples in order to learn the global model. In a similar setting, [12] has shown that clean-label attacks are an effective attack mechanism in a federated learning setting. Another line of work is concentrated on exploring the potential and limitations of *specific* learners in settings where training data are corrupted as dictated by certain noise models or attack methods; e.g., [13, 2]. Furthermore, while test-time attacks have not yet been investigated a lot in a federated setting, nevertheless, the joint setting of backdoor attacks has received some attention; e.g., [1].

The above lines of work may operate in a passive or online learning setting, but have not been explored in a stream-based setting. Thus it is unclear, for example, how limited memory may affect positive and negative results, as well as what additional benefits a federated learning system that allows concept drift can enjoy. In addition, how the detection of poisoned data at local sites can be integrated with stream-based learning at local sites to address other important characteristics of data streams, such as data transiency (data become less important as time goes), temporal context, and data uncertainty. These characteristics also need to be considered when creating the global model and broadcasting it to the participating local sites in the federated learning system.

Another line of work has to do with defense mechanisms that we can embed into stream-based federated learning methods so that we can make the models resistant to adversarial situations mentioned above. For example, how far one can go with sanitization methods and outlier detection mechanisms, whether outlier detection mechanisms should be used to detect outliers at local devices only or whether they should also be applied at the server to detect outliers in the local models sent to the server (for example, outliers in the local model parameters), and what kind of metrics one can use so that we can uncover attacks that occur on the stream of data that is presented to the learner. Furthermore, in real-world settings, such defense mechanisms and removal of harmful data would require not only the identification of the poisoned data in the stream, but also an explanation

to be provided to the individual users, thus promoting trust in the task that is executed by the global model. In addition, we would like the defense and explanation mechanisms not only to be effective, but also to be *lightweight* and be able to operate in *real-time* as the stream of data arrives at the learner. A similar explanation mechanism for stream outlier detection has been proposed in [15]. Concluding, one may also care about complex performance measures [14] and it is unclear what can be accomplished for imbalanced distributions under the setup of Section 2.

4 Conclusions

We identified research issues that are novel for the framework of adversarially robust stream-based federated learning and anticipate stimulating investigations in the future. Finally, we note that studying regression problems would give an additional dimension to the issues raised in Section 3.

References

1. Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How To Backdoor Federated Learning. In *AISTATS*, volume 108 of *Proceedings of Machine Learning Research*, pages 2938–2948. PMLR, 26–28 Aug 2020.
2. Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin B. Calo. Analyzing Federated Learning through an Adversarial Lens. In *ICML 2019*, volume 97 of *Proceedings of Machine Learning Research*, pages 634–643. PMLR, 2019.
3. Yiqiang Chen, Xin Qin, Jindong Wang, Chaohui Yu, and Wen Gao. FedHealth: A Federated Transfer Learning Framework for Wearable Healthcare. *IEEE Intelligent Systems*, 35(4):83–93, 2020.
4. Dimitrios I. Diochnos, Saeed Mahloujifar, and Mohammad Mahmoody. Lower Bounds for Adversarially Robust PAC Learning under Evasion and Hybrid Attacks. In *ICMLA 2020*, pages 717–722. IEEE, 2020.
5. Peter Kairouz et. al. Advances and Open Problems in Federated Learning. *Foundations and Trends in Machine Learning*, 14(1-2):1–210, 2021.
6. Heitor Murilo Gomes, Jesse Read, Albert Bifet, Jean Paul Barddal, and João Gama. Machine learning for streaming data: state of the art, challenges, and opportunities. *SIGKDD Explorations*, 21(2):6–22, 2019.
7. Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. *CoRR*, abs/1708.06733, 2017.
8. Andrew Hard, Kanishka Rao, Rajiv Mathews, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated Learning for Mobile Keyboard Prediction. *CoRR*, abs/1811.03604, 2018.
9. Michael J. Kearns and Ming Li. Learning in the Presence of Malicious Errors. *SIAM Journal on Computing*, 22(4):807–837, 1993.
10. Nikola Konstantinov, Elias Frantar, Dan Alistarh, and Christoph Lampert. On the Sample Complexity of Adversarial Multi-Source PAC Learning. In *ICML 2020*, volume 119 of *Proceedings of Machine Learning Research*, pages 5416–5425. PMLR, 2020.
11. Xiaofeng Lu, Yuying Liao, Pietro Liò, and Pan Hui. Privacy-Preserving Asynchronous Federated Learning Mechanism for Edge Network Computing. *IEEE Access*, 8:48970–48981, 2020.
12. Saeed Mahloujifar, Mohammad Mahmoody, and Ameer Mohammed. Universal multi-party poisoning attacks. In *ICML 2019*, volume 97 of *Proceedings of Machine Learning Research*, pages 4274–4283. PMLR, 09–15 Jun 2019.
13. El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. The Hidden Vulnerability of Distributed Learning in Byzantium. In *ICML 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3518–3527. PMLR, 2018.
14. Harikrishna Narasimhan, Harish G. Ramaswamy, Aadirupa Saha, and Shivani Agarwal. Consistent Multiclass Algorithms for Complex Performance Measures. In *ICML 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2398–2407. JMLR.org, 2015.
15. Egawati Panjei, Le Gruenwald, Eleazar Leal, and Christopher Nguyen. Micro-clusters-based outlier explanations for data streams. In *ANDEA 2021, co-located with KDD 2021*, 2021.
16. Nikolaos Pitropakis, Emmanouil Panaousis, Thanassis Giannetsos, Eleftherios Anastasiadis, and George Loukas. A taxonomy and survey of attacks against machine learning. *Computer Science Review*, 34, 2019.
17. Raed Abdel Sater and A. Ben Hamza. A Federated Learning Approach to Anomaly Detection in Smart Buildings. *ACM Transactions on Internet of Things*, 2(4):28:1–28:23, 2021.
18. Robert H. Sloan. Four Types of Noise in Data for PAC Learning. *Information Processing Letters*, 54(3):157–162, 1995.

Improving the accuracy of vehicle routing problem approximation using the formula for the average distance between a point and a rectangular area

HASEGAWA Daisuke [1][0000-0002-4854-6665] , HONMA Yudai[2][0000-0002-6458-0767]

SHIONO Naoshi [3][0000-0003-0913-821X] and TOKI Souma [4][0000-0003-3603-3798]

[1]The University of Tokyo, Bunkyo, Tokyo, Japan
hasega60@iis.u-tokyo.ac.jp
[2]The University of Tokyo, Bunkyo, Tokyo, Japan
yudai@iis.u-tokyo.ac.jp
[3]Kanagawa Institute of Technology, Atsugi, Kanagawa, Japan
na-shiono@ic.kanagawa-it.ac.jp
[4] Tokyo gas co., ltd, Minato, Tokyo, Japan.
toki.s@tokyo-gas.co.jp

Abstract. The continuous approximation model of VRP analyzes cost at the planning and strategic analysis stage in the delivery and logistic field. Most previous studies used the tour distance between customers and the linehaul distance between the depot and the customers. This study focused on the linehaul distance, and we applied the formula for the average length in a right triangle and presented the average distance between depot and service area. Our proposed model is tested in instances with different locations of the depot and the service area, trucks, and customers. Regression results indicate that the approximation model improves accuracy if the depot locates outside the service area. Our results can be applied when planning deliveries, for cases where the depot is located at the edge of the city and outside the delivery area, and planning area segmentation.

Keywords: Vehicle routing problem, Distance estimation, Continuous approximation approach.

1 Introduction

The travel salesman problem (TSP) and vehicle routing problem (VRP) are methods for minimizing the cost of transportation using single or multiple vehicles from one location to a customer. This problem is extremely important not only in the logistics sector but also in the public transportation sector, where new services such as ride-sharing and car-pooling are being developed. With the improvement in computers and algorithms, the number of possible optimal solutions has increased; however, there are many cases where an approximate solution is required. For example, if the number of demand points is too high for an optimal solution, the number of vehicles will be too

high, or the specific location of the demand points will be unknown. Some mathematical models (commonly called continuous approximation models) can be used to approximate the optimal solution. Currently, numerous studies have been conducted on continuous approximation models for the TSP and VRP. Some of these models include the average distance between the depot and the delivery area, considering the case where the depot, which is a base for vehicles, is not at the center of the delivery area. Many previous studies have used the linehaul distance for such cases. However, they used simplifying assumptions and did not strictly reflect the shape and location of a region. This study aims to improve the approximation accuracy of the continuous approximation model using an analytically derived value of the average distance between a depot and the delivery service area.

The rest of this paper is organized as follows. Section 2 provides a literature review of the VRP and TSP length calculations obtained using the continuous approximation approach. Section 3 presents the approximation result of the average distance between points and a rectangular area and its application to the continuous approximation model of the VRP. Section 4 describes the experimental design and estimation results. Finally, Section 5 concludes the paper.

2 Literature review

Most of the continuous approximation models for the TSP and VRP have been based on the study conducted by Beardwood et al. (1959) [1], who proved the result generally known as the BHH formula. For a set of n random points in a i-dimensional space $\mathbb{R}^i$, the length of an optimal tour through n points D^* satisfies:

$$\lim_{n\to\infty} \frac{D^*}{n^{(i-1)/i}} = k_i i^{\frac{1}{2}} [v(\Psi)]^{\frac{1}{i}}, \tag{1}$$

where the measure of the Lebesgue-measurable set Ψ is denoted by $v(\Psi)$, and k_i is a constant that depends on i. This formula is quite complicated, and Eilon et al. (1971) [6] showed a simple explanation of the planar case ($i = 2$), where S is planar area of and k is used instead of k_2 :

$$\frac{D^*}{\sqrt{n}} \to k\sqrt{S} \qquad if\ n \to \infty, \tag{2}$$

and if $n \in \mathbb{N}$, Eq. (2) can be rewritten as:

$$D^* \approx k\sqrt{nS}. \tag{3}$$

The value of k is an unknown constant; nevertheless, it has been estimated in several previous studies. In the case of the Manhattan distance metric, Jaillet (1988) [11] estimated $k = 0.97$, and Stein (1978) [4] estimated $k = 0.765$ using the Euclidean distance metric. Cook et al. (2011) [3] showed that k is correlated to n and estimated it to be $0.625 \leq k \leq 0.920$ for n values between 100 and 2000.

In addition, the BHH formula can be extended to a VRP that has capacitated vehicles based on the depot visit customers in the area. Daganzo (1984a) [4] proposed a simple formula for the optimal length of VRP D_m^* with m routes:

$$D_m^* = k\sqrt{nS} + 2Rm, \tag{4}$$

where the value of m is obtained from the number of customers n and the maximum capacity of vehicle C ($m = n/C$); parameter $k = 0.57$, and R is the distance from the depot to a random point in the area. In this formula, the first term refers to the tour length estimated by the BHH formula, and the second term is generally called the linehaul distance, which is the distance from the depot to the customer. The linehaul distance contributes to the estimation when the depot is not located at the center of the service area. Some studies have developed regression tests to estimate the TSP optimal length D^* and VRP optimal length D_m^*. Chien (1992) [2] suggested this approximation for a depot located at the corner of the area. Let B be the size of the boundary box enclosing all points; in this case, the total distance in the case of one truck D_1^* (TSP route length) can be expressed as:

$$D_1^* = 0.67\sqrt{nB} + 2.1R. \tag{5}$$

Figliozzi (2008) [7] presented the following as a model of the VRP containing the linehaul distance:

$$D_m^* = k\left(\frac{n-m}{n}\right)\sqrt{nS} + 2Rm, \tag{6}$$

where the parameter k is estimated to be $0.62 \leq k \leq 0.90$, and average $k = 0.77$ by linear regression, with a high accuracy for approximating D_m^*.

Most previous studies have focused on the tour length, indicated by the first term, to improve the accuracy. Therefore, the assumption of the linehaul distance R is simplified. In Equation (4) proposed by Daganzo (1984a) [4], R is determined to check whether the depot o is located in service area $\mathcal{D}$ as follows:

$$R = \begin{cases} \left(\sqrt{S}/6\right)\left(\sqrt{2} + \log\tan(3\pi/8)\right) \cong 0.382\sqrt{S}, & (o \in \mathcal{D}) \\ \hat{d} & (o \notin \mathcal{D}). \end{cases} \tag{7}$$

where $\hat{d}$ is the distance from the depot o to the center of gravity of the service area $\mathcal{D}$. Moreover, Franceschetti et al. (2017) [8] used the closest point in the service area. Huang et al. (2013) [9] used the expected distance to reach the first point by taking the square root of the sum of the square of the expected longitudinal distance and the square of the expected transverse distance.

However, the tour start point exists randomly in the service area, and we expect that calculating the average distance by considering the area and shape of the service area can help improve the estimation accuracy of D_m^*. For this purpose, we tested using two different linehaul distances for variable R in Daganzo's model (4) and Figlilozzi's model (6). One is the distance from the center of gravity of the area used in previous studies, and the other is the average distance between the point and the service area calculated using a probabilistic approach.

3 Approximation of the average distance between a point and an area

In this section, we first describe the problem setting. Next, we present the methods for determining the distance between points and an area.

3.1 Problem setting

In the continuous approximation approach for TSP and VRP reported in previous studies, the service area was simplified to a rectangular shape (Daganzo, 1984a ; Franceschetti et al., 2017; Huang et al., 2013 [4, 8, 9]), circular/elliptical shape (Robusté et al., 2004[16]), and a ring-radial network (Newell and Daganzo, 1986; Jabail et al., 2012[15, 10]). However, the shape of the area does not significantly affect the quality of the approximation (Daganzo, 1984b; Koshizuka, 2019[5, 13]). Thus, we model the service area as a rectangular shape for ease of numerical experiments and assume that the depot position is fixed and that the service area can be flexibly changed to account for the differences in the linehaul distance. We assume a rectangular service area $\mathcal{D}$ with horizontal length A and vertical length B. The n number of customers in $\mathcal{D}$ are distributed in the form of a continuous uniform distribution and are picked up by m trucks based at depot o (Fig. 1). We calculate the average distance of a right triangle area using the model proposed by Koshizuka and Kurita (1991); Kurita (2013) [12, 14] and apply it to the rectangular regions.

3.2 Model

Let the right triangle δ, shown in Fig. 2, be a depot o as an acute vertex. α is the edge from o to the right angle a, and β is the other edge from o to b. The probability density function $\varphi(x)$ of the distance from o to a point x in δ can be expressed as:

$$\varphi(x) = \frac{L(x)}{S} \tag{8}$$

$$S = \frac{\alpha}{2}\sqrt{\beta^2 - \alpha^2} \tag{9}$$

Here, S is the size of δ, and $L(x)$ is the perimeter of the fan shapes with x radius and is obtained from cases shown in Figs. 2 (i) and (ii):

$$L(x) = \begin{cases} \text{(i) } x \arccos\frac{\alpha}{\beta} & (0 \le x \le \alpha) \\ \text{(ii) } x\left(\arccos\frac{\alpha}{\beta} - \arccos\frac{\alpha}{x}\right) & (\alpha < x \le \beta) \end{cases} \tag{10}$$

where $\arccos \alpha/\beta$ is the angle θ between α and β. Thus, the average distance d_δ^* can be expressed as:

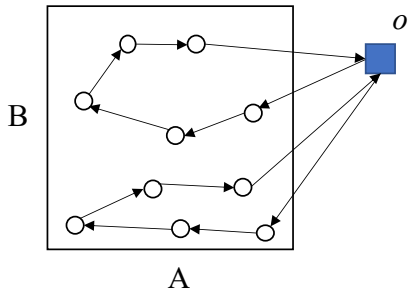


Fig. 1. Service area and depot.

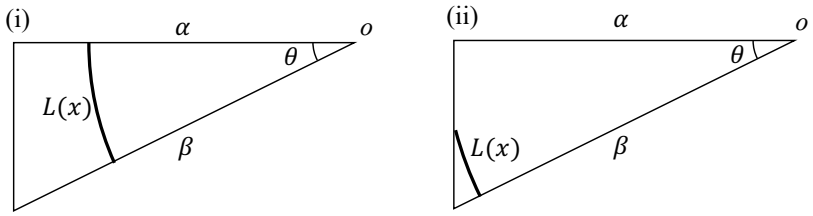


Fig. 2. Location of service area depot.

$$d^*_{oab} = \int_0^{\beta} x\,\varphi(x)\mathrm{d}x \tag{11}$$

Equation (11) can be reorganized as follows:

$$d^*_{oab} = \frac{1}{3}\left(\beta + \frac{\alpha^2}{\sqrt{\beta^2-\alpha^2},} ln\frac{\beta+\sqrt{\beta^2-\alpha^2}}{\alpha^2}\right) \tag{12}$$

Subsequently, we can calculate the average distance to the rectangular area by combining the right triangles. There are four cases (I to IV in Fig. 3) depending on whether the location of *o* is within the horizontal or vertical extent of the area. Fig. 4 shows the calculation method for the average distance between Cases I and IV.

For a rectangular area $\mathcal{D}_{abcd}$, the average distance d^* from o to $\mathcal{D}_{abcd}$ can be expressed as:

$$d^* = \frac{LD_{abcd}}{S_{abcd}} \tag{13}$$

where S_{abcd} is the size of $\mathcal{D}_{abcd}$, and the total distance LD_{abcd} is calculated by combining the LD of the rectangles with *o* as its vertex. The intersection points of each edge and perpendicular line from depot *o* are *e*, *f*, *g*, and *h*. The total distance LD_{aego} from *o* to the rectangular area $\mathcal{D}_{aego}$ (upper left in Case I) with *o* as its vertex can be expressed as:

$$LD_{aego} = S_{aeo}d^*_{aeo} + S_{ago}d^*_{ago} \tag{14}$$

where the size of the right triangle is denoted by S_{aeo}. In Case I, where o is in $\mathcal{D}_{abcd}$, LD_{abcd} is obtained by summing the LD of the rectangles around o. In Cases II, III, and IV, e, f, g, and h are the intersections of the line extending each edge and perpendicular line, and we removed the unnecessary part (the gray area in Fig. 4 (IV)) from the rectangle with o as its vertex that is greater than $\mathcal{D}_{abcd}$. Thus, let the coordinates of a be (a_1, a_2), and LD_{abcd} can be expressed as in Eq. (15).

$$LD_{abcd} = \begin{cases} (\mathrm{I}) & LD_{aego} + LD_{ebof} + LD_{goch} + LD_{ofdh}, \\ (\mathrm{II}) & \mathrm{LD}_{ofgd} + \mathrm{LD}_{ebof} - \left(LD_{ohgc} + LD_{eaoh}\right) \\ (\mathrm{III}) & LD_{aeho} + LD_{ebof} - \left(LD_{\mathrm{cgho}} + LD_{gdof}\right) \\ (\mathrm{IV}) & LD_{ebof} - \left(LD_{eaoh} + LD_{gdof} - LD_{gcof}\right) \end{cases} \tag{15}$$

We used d^* as the linehaul distance R for models (4) and (6):

$$D_m^* = k\sqrt{nS} + 2d^*\, m \tag{4-1}$$

$$D_m^* = k\left(\frac{n-m}{n}\right)\sqrt{nS} + 2d^*\, m, \tag{4-2}$$

In the next section, we compare the estimation accuracies of models (4) and (4-1), and (6) and (6-1) by numerical experiments.

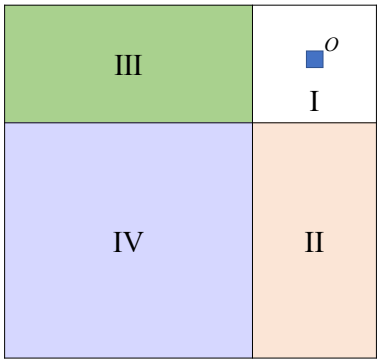


Fig. 3. Calculation condition for d^*.

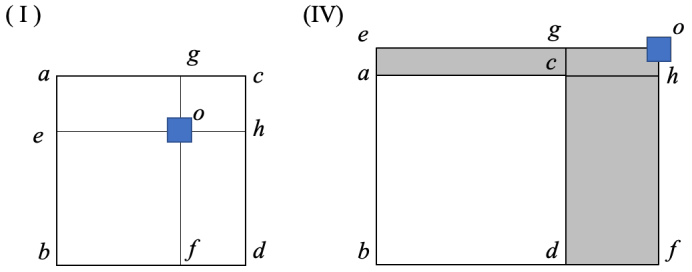


Fig. 4. Calculation method for d^*(Cases I and IV)

4 Experimental setting and results

In this section, we first explain the numerical experimental setup. Next, we evaluate the accuracy of the estimates obtained using the continuous approximation model.

4.1 Experimental setting

Numerical experiments were conducted to verify the accuracy of the continuous approximation model defined in Section 3. There are many instances of TSP and VRP, such as TSPLib, CVRPLib, and instance data from Solomon (1987)[20, 19, 17, 21]. For these examples, problems and solutions are available, including the spatial distribution of customers, vehicle capabilities, customer requirements, and customer time settings. However, in many instances, the depot location is included in the range of the demand distribution, and we cannot evaluate the difference in the linehaul distance, which is the subject of discussion in this study. Therefore, route optimizations with different area sizes and locations, demand patterns, and number of vehicles were obtained using mixed-integer linear programming (MILP).

Table 1. Experimental conditions.

Items		Values
Area settings	Area length (A, B)	(10000,10000), (20000,10000), (30000,10000)
	Depot location	(25000, 25000), (50000, 25000), (75000, 25000)
Layout patterns		9
Trucks		2, 3, 4, 5
Demands		10, 20, 30, 40, 50 points per trucks

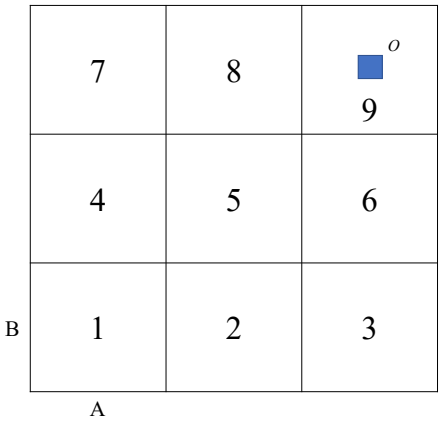


Fig. 5. Layout pattern of the depot and service area.

Table 1 shows the details of the experimental conditions. Fig. 5 shows the layout pattern of the depot o and service area $\mathcal{D}$. The area length is the length of the edges with $\mathcal{D}$ (Fig. 4, (A, B)); three types with different aspect ratios are assumed, and the position of o is changed on the basis of the aspect ratio of $\mathcal{D}$. The area layout has nine patterns (Fig. 5, Nos. 1 to 9) for the layout of o and $\mathcal{D}$. In the case of No. 9, o is located at the centroid of $\mathcal{D}$, which is the pattern with the shortest linehaul distance; conversely, in case No. 1, it is the pattern with the farthest distance from o. The number of trucks is assumed to range from 1 to 5, and the number of demands is assumed to range from 10 to 50 per truck, which is the same as the truck capacity.

We solved 540 instances (three area settings, nine layouts, four trucks, and five demands) 10 times by randomly changing the demand points within the area.

We used LocalSolver 9.5 to solve the VRP, and the CVRP algorithms for the problems that have been tested for performance by LocalSolver [22]. The computational tests were performed on two Windows 10 machines (3.5 GHz Intel Core i9 processor, with 64 GB RAM; 2.5 Xeon GHz processor with 64 GB RAM). The maximum running time was set to 300 s. However, for the pattern with the largest number of combinations (five trucks, demand 50 per truck), the optimality gap with the upper bound was less than 5%.

4.2 Results

To evaluate the prediction accuracy, the R-square value, root-mean-squared error (RMSE), and mean absolute percentage error (MAPE) were used. The RMSE and MAPE were calculated as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_i (y_i - f_i)^2} \tag{16}$$

$$MAPE = \frac{100}{n}\sum_i \left|\frac{(y_i - f_i)}{y_i}\right| \tag{17}$$

where the actual distance, which is calculated by MILP for instance i, is denoted y_i, and the estimated distance is denoted by f_i. The RMSE indicates the absolute error value for a specific distance. However, the longer the total distance, the greater the error. Therefore, the MAPE is used to determine the relative error.

Table 2. Model fit comparison.

Model	k	R^2	RMSE	MAPE
(4)	0.570	0.995	10087.0	3.4%
(4-1)	0.529	0.998	6830.2	2.5%
(6)	0.570	0.995	9944.8	3.5%
(6-1)	0.549	0.998	6883.5	2.5%

Table 2 presents the fitting results for the linehaul distance and average distance. In the estimation, for model (4), we used $k = 0.57$, the value given in the original paper, and for model (6), we used the same value. In Figliozzi (2008), k was fitted with real data (Solomon (1987)), and the conditions of the numerical experiments were different from those in this study. For models (4-1) and (6-1), which are the focus of this study, parameter k was fitted using the maximum likelihood estimation method. In Table 2, all the models have good R^2 values. However, models (4-1) and (6-1), which use a continuous approximation term for the linehaul distance, have a better RMSE and MAPE performance than models (4) and (6).

Table 3. Model fit comparison by number of trucks.

	RMSE				MAPE			
	trucks				trucks			
	2	3	4	5	2	3	4	5
Model								
(4)	7794.2	9350.5	10723.6	11992.3	3.9%	3.5%	3.1%	2.9%
(4-1)	6870.8	6627.1	6694.2	7118.0	3.6%	2.6%	2.0%	1.8%
(6)	8347.9	9500.4	10356.9	11330.7	4.4%	3.7%	3.1%	2.9%
(6-1)	6775.1	6688.2	6838.3	7220.4	3.7%	2.7%	2.1%	1.8%

Next,wecompare the performance of each model based on the number of trucks and the layout of the area and depot to analyze the factors that can help improve the accuracy. Table 3 shows the comparison results of the number of trucks. The number of trucks is proportional to the linehaul distance, as shown in the second term of each model. Therefore, the greater the number of trucks, the better the RMSE and MAPE performance. However, the difference is highest when the number of vehicles is four, and above that, the performance tends to saturate.

7			8			9		
	RMSE	MAPE		RMSE	MAPE		RMSE	MAPE
(4)	9534.3	2.3 %	(4)	8803.4	3.3 %	(4)	8210.7	6.1 %
(4-1)	5945.0	1.4 %	(4-1)	6063.7	2.3 %	(4-1)	9031.0	7.1 %
4			**5**			**6**		
	RMSE	MAPE		RMSE	MAPE		RMSE	MAPE
(4)	9584.3	2.2 %	(4)	8055.7	2.9 %	(4)	11613.6	6.5 %
(4-1)	5957.1	1.3 %	(4-1)	5559.7	2.0 %	(4-1)	6283.2	3.4 %
1			**2**			**3**		
	RMSE	MAPE		RMSE	MAPE		RMSE	MAPE
(4)	8704.2	1.8 %	(4)	6474.8	1.9 %	(4)	8522.0	3.4 %
(4-1)	5655.3	1.1 %	(4-1)	5010.0	1.6 %	(4-1)	7206.8	2.8 %

Fig. 6. Model fit comparison by layout pattern of depot and service area (Models (4) and (4-1))

10

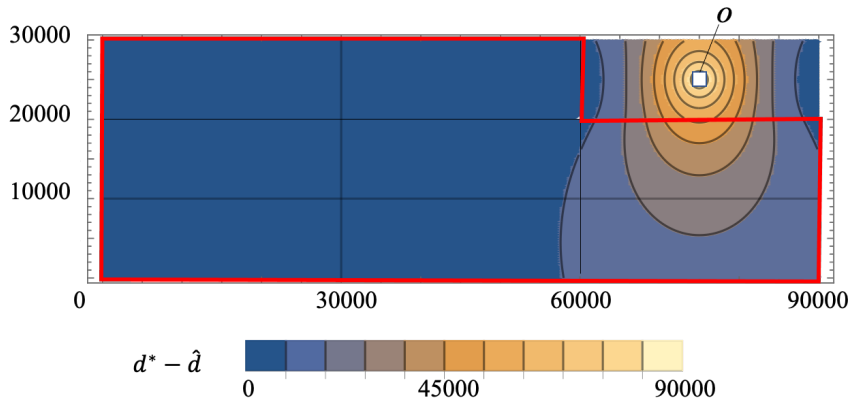


Fig. 7. Mapping of difference values between $\hat{d}$ and d^* .

Fig. 6 shows a comparison of the RMSE and MAPE values of models (4) and (4-1) for different area layouts, as shown in Fig. 5. The performance of Nos. 1 to 8, which are the area layouts away from the depot, is improved. In particular, No. 6 exhibits the best performance. Fig. 7 shows the reason for this. This figure shows the difference in the distance from the depot to the center of gravity $\hat{d}$ or rectangular areas d^* when the depot is located at (75000, 25000), and the area edges A = 30000 and B = 10000 (thus, the aspect ratio is 3.0) are moved. When the area does not include the depot (in the red frame), a large difference value is located at No. 6. In other words, the lower the $\hat{d}$ value and the longer the edge that intersects the line, the greater the error.

In other areas, the RMSE was in the range of 1300–3600, and the MAPE values were reduced by 0.3%–1.0%. However, in the region containing the depot (No. 9), the performance is worse because the adjustment of the tour length is sufficient and the linehaul distance is over-adjusted. These results indicate that our proposed method of calculating the linehaul distance is useful for regions where the depot is far from the service area, contributing to the improvement in the estimation accuracy.

5 Conclusions

In this study, we analyzed approximations of the linehaul distance between the depot and rectangular service area to improve the accuracy of estimating the total VRP distance. The VRP approximation formula is useful for the strategic and planning analyses of transportation and logistics problems, where the number and location of customers change daily. We defined a continuous approximation formula to find the average distance of a right triangle with the depot as an acute angle and the average distance between a point and a rectangular region at any given location. This approximation formula was used as the linear distance in the VRP model. In addition, we computed the optimal route for many instances with different demands, number of

trucks, and locations that can be considered in the VRP approximation model. The numerical solutions were estimated using the approximation model. The results showed an improvement in the accuracy of the approximation equation for service areas far from the depot. Our results can be applied when planning deliveries, for cases where the depot is located at the edge of the city and outside the delivery area and to the planning of area segmentation represented by the strip strategy (e.g., Daganzo (1984a), Franceschetti et al. (2017)[4, 8]). The linehaul distance approximation is also useful for estimating the distance traveled by cabs and kickboards without tours. In the future, we plan to apply the developed model to analyze the above problems, while considering the time constraints of the VRP and the effects of area segmentation on the delivery efficiency.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Numbers JP21K14314 and Obayashi Foundation. We appreciate their support.

References

1. Beardwood, J., Halton, J. H., Hammersley, J. M.: The shortest path through many points. Mathematical Proceedings of the Cambridge Philosophical Society, 55(4), 299–327 (1959).
2. Chien, T. W.: Operational estimators for the length of a traveling salesman tour. Computers and Operations Research, 19(6), 469–478 (1992).
3. Cook, W. J., Applegate, D. L., Bixby, R. E., Chvátal, V.: The Traveling Salesman Problem. Princeton University Press (2011).
4. Daganzo, C. F.: The distance traveled to visit N Points with a maximum of C Stops per vehicle: An analytic model and an application. Transportation Science, 18(4), 331–350 (1984a).
5. Daganzo, C. F.: The length of tours in zones of different shapes. Transportation Research Part B, 18(2), 135–145 (1984b).
6. Eilon, S., Watson-Gandy, C. D. T., Christofides, N.: Distribution Management: Mathematical Modelling and Practical Analysis. Charles Griffin, London (1971).
7. Figliozzi, M. A.: Planning approximations to the average length of vehicle routing problems with varying customer demands and routing constraints. Transportation Research Record, 2089(1) ,1–8 (2008).
8. Franceschetti, A., Honhon, D., Laporte, G., Woensel Van, T. V., Fransoo, J. C.: Strategic fleet planning for city logistics. Transportation Research. Part B, Methodological, 95, 19–40 (2017).
9. Huang, M., Smilowitz, K. R., Balcik, B.: A continuous approximation approach for assessment routing in disaster relief. Transportation Research. Part B, Methodological, 50, 20–41 (2013).
10. Jabali, O., Gendreau, M., Laporte, G.: A continuous approximation model for the fleet composition problem. Transportation Research. Part B, Methodological, 46(10), 1591–1606 (2012).
11. Jaillet, P.: A priori Solution of a Traveling Salesman Problem in which a Random Subset of the Customers Are Visited. Operations Research, 36(6), 929–936 (1988).

12. Koshizuka, T., Kurita, O.: Approximate formulas of average distances associated with regions and their applications to location problems. Mathematical Programming, 52(1–3), 99–123 (1991).
13. Koshizuka, T.: Integral Geometry for Applications - Measures of Figures. Kindai Kagaku Sha (2019). (in Japanese).
14. Kurita, O.: Mathematical Models of Cities and Regions - Mathematical Methods in Urban Analysis. Kyoritsu Shuppan Co., Ltd (2013). (in Japanese).
15. Newell, G. F., Daganzo, C. F.: Design of multiple-vehicle delivery tours—I a ring-radial network. Transportation Research Part B, 20(5), 345–363 (1986).
16. Robusté, F., Estrada, M., López-Pita, A.: Formulas for estimating average distance traveled in vehicle routing problems in elliptic zones. Transportation Research Record, 1873(1), 64–69 (2004).
17. Solomon, M. M.: Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research, 35(2), 254–265 (1987).
18. Stein, D. M.: An asymptotic, probabilistic analysis of a routing problem. Mathematics of Operations Research, 3(2), 89–101 (1978).
19. CVRPLIB, http://vrp.atd-lab.inf.puc-rio.br/index.php/en/, last accessed 2022/01/20.
20. TSPLIB, http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/, last accessed 2022/01/20.
21. The instances of Solomon, http://web.cba.neu.edu/~msolomon/problems.htm, last accessed 2022/01/20.
22. Benchmark–Capacitated vehicle routing problem (CVRP), Localsolver, https://www.localsolver.com/benchmarkcvrp.html, last accessed 2022/01/20.

Optimal Delivery Area Assignment for the Capital Vehicle Routing Problem Based on a Maximum Likelihood Approach

MARUYAMA Junya[1][0000-0002-6703-3259] , HONMA Yudai[2][0000-0002-6458-0767] ,

HASEGAWA Daisuke[3][0000-0002-4854-6665] , TOKI Soma[4][0000-0003-3603-3798]

and SHIONO Naoshi[5][0000-0003-0913-821X]

[1] The University of Tokyo, Bunkyo, Tokyo, Japan
juntama0826@g.ecc.u-tokyo.ac.jp
[2]The University of Tokyo, Bunkyo, Tokyo, Japan
yudai@iis.u-tokyo.ac.jp
[3]The University of Tokyo, Bunkyo, Tokyo, Japan
hasega60@iis.u-tokyo.ac.jp
[4] Tokyo Gas Co., Ltd., Minato, Tokyo, Japan
toki.s@tokyo-gas.co.jp
[5] Kanagawa Institute of Technology, Atsugi, Kanagawa, Japan
na-shiono@ic.kanagawa-it.ac.jp

Abstract. In this study, we constructed an optimization model for the maximum likelihood estimation of delivery areas from a capacitated vehicle routing problem. The aim is to develop a method that combines the advantages of two methods of delivery planning: the efficiency of the routing software-based method and the flexibility of the area-in-charge method. We first conduct computer experiments to derive the optimal cycling plan for each stochastic demand pattern. We then solve the optimal delivery area assignment that is globally consistent with the data from these experiments. We focused on whether the optimal route for each demand pattern was contained in the same area and found the assigning area that maximized the probability. This model is designed for daily use because it is an easy-to-interpret area map, while the optimization of the circulation problem is solved using computers in advance. In experiments using the data, we confirmed that the model can provide correct area creation.

Keywords: Delivery plan, Area assignment, Maximum likelihood estimation.

1 Introduction

Planning an effective delivery route is one of the most important topics discussed in logistics, such as postal delivery services and meter readings. In recent years, its importance has increased. Figure 1 shows the distribution of customers for gas cylinder delivery. About 5,000 customers exist. Distributors deliver 30 to 70 cylinders every

day. That is, daily demand points are stochastic. For distributors, there are mainly two effective strategies for stochastic delivery route planning: one is to find the best route through routing software, and the other is to assign the driver's area to each driver.

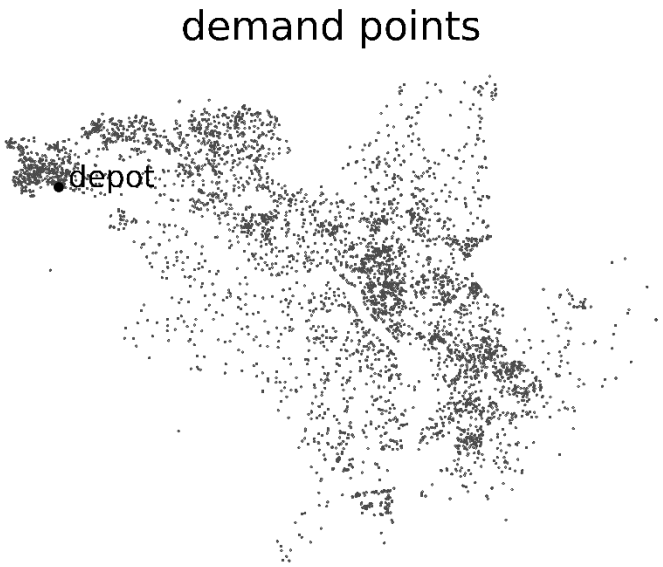


Fig. 1. Demand points

The routing software-based method of determining the optimal route is more efficient than the method of assigning a driver's area. The method of searching for the optimal route has been studied as the Vehicle Routing Problem (VRP). On the other hand, solving the VRP is computationally demanding, so it is not practical to find the optimal route. In actual delivery, distributors face sudden customer changes, such as a customer's request due to running out of gas or bad weather. In these cases, a manager change driver's scheduling by hang as applying the software takes much time.

Alternatively, in daily delivery, plans are often made based on the area assignment for each driver. For many distributors, these areas are determined by discussions among drivers or by the experience of managers without any actual quantitative analysis. Assignment areas for each driver are often inefficient. On the other hand, such a delivery plan has the advantage of being able to respond flexibly to sudden changes in the delivery destination or the absence thereof. Such irregular events are common in everyday delivery.

As shown above, the two methods have trade-offs in terms of computational time, traveling time, and flexibility in actual daily delivery when we make a delivery route plan.

Previous attempts in logistics have been made to assign areas appropriately. In the planning and strategic phases, a successive approximation approach with simple assumptions (uniform random demand distribution, simple shapes) estimated the impact of zoning. Newell and Daganzo [1] analyzed the effect on distance when a rectangular area was divided into equal parts and found the optimal number of divisions. Applying these approaches, Ouyang [2] proposed an algorithm in which geometric features determine the vehicle routing zone for any demand distribution. In a more practical approach, Galvão et al. [3] focused on the density distribution of demand and tried to determine the area using the Voronoi diagram approach. They relaxed the predetermined boundaries of the partition and iteratively modified them until they converged to minimize the distance traveled by vehicles. Ayala et al. [4] developed a demand point allocation scheme that minimizes the distance from the depot to the network connecting the demand point to the depot or the demand points to each other. Zhong et al. [5] divided the field into areas by preparing cells, which are sets of demand points, and considered the optimal clustering of cells by adding a cost to each. In recent research, Sung et al. [6] used a meta-heuristics approach to zoning aerial vehicles by bearing from depots to demand points. However, there is no study that solves the VRP for the entire field and naturally divides the distributor's area into driver's areas.

The purpose of this research is to propose a new optimization model to find the area assignment for the capacitated vehicle routing problem, which maintains both efficiency (by calculating the optimal route) and flexibility (by defining the area assignment). To achieve this purpose, we first conducted computer experiments to derive the optimal cycling plan for each stochastic demand pattern. We then identified the optimal delivery area assignment that was globally consistent with the data from these experiments. We introduce a clustering method in the work of Honma et al [7], which is a maximum likelihood estimation of areas from transition information.

One of the features of this research is that we aimed to produce output in the form of a map for use in actual delivery scenarios. It is frequently difficult to handle and calculate complex mathematical models in the field. Even in such cases, the output in the form of a map provides an easy and intuitive way to use the results of this research. In addition, because of the map format, the delivery personnel do not have to significantly alter what they have been doing. Thus, it can be said that the output of this research can achieve efficiency with less effort.

In addition, by using area assignment, the system can handle demands that occur randomly every day. In actual delivery, demand occurs randomly among a number of potential demand points, and it is necessary to create a delivery plan that corresponds to the demand. This is much easier to operate than finding a solution through software.

This paper is organized as follows. In Section 2, we formulate the problem. We propose a mathematical model for achieving efficient area assignment, which is the objective of this study and is estimated by taking the maximum likelihood from the optimal route and constructing a mathematical model for assigning an area. In Section 3, we introduce the data for the computer experiments used to test the certainty of the model, and in Section 4, we verify the model use actual data. Section 5 discusses the model based on the results of Section 4, and finally, Section 6 provides conclusions and future perspectives for further study.

2 Formulation

2.1 Our concept

In this study, we deal with the capacitated Vehicle Routing Problem (CVRP). To assign the optimal area for each driver, we iteratively calculate the optimal traveling routes on software under stochastic demands and then estimate the assignment areas based on the solutions to be as consistent as possible with the computer experiments. This means that we assign a label of area information for each demand point. Thereby, this problem can be rephrased as the problem of labeling area information at the demand points on the same circular route in computer experiments in as few areas as possible.

First, we present the conditions for achieving this objective mathematically. In this study, we consider an area assignment that satisfies the following conditions that an area is as consistent as possible.

i. Demand points that belong to the same area should be delivered by the same vehicle as much as possible.
ii. Demand points that belong to different areas should be delivered by different vehicles as much as possible.

These show the properties that an optimal area assignment should satisfy. For these properties, we conduct a large number of computer experiments because we assume that daily demand points are random. Therefore, we apply probabilities i and ii. An area assignment that satisfies these properties is achieved by combining the following procedures, each corresponding to the above conditions:

i'. Maximize the probability of delivering any given demand points in the same area with the same vehicle
ii'. Minimize the probability of delivering any given demand points in different areas with the same vehicle

By achieving these conditions at the same time, the solution should be obtained such that the same drivers deliver as much as possible within the same area, conversely, the same drivers do not deliver as much as possible in different areas. When solving our area assignment problem, a constraint is employed in that each vehicle has the same capacity. This is to keep the quantity transported by each delivery vehicle constant. The optimization problem is formulated as a maximum likelihood estimation by partitioning a field as described above.

2.2 Formulation based on demand points

Our area assignment problem is solved as a quadratic assignment problem in terms of the label of area information [8].

Let I be the set of demand points and A be the set of areas for all drivers. The presence or absence of delivery to demand points $i(\in I)$ and $j(\in I)$ is determined by repeated computer experiments. We now introduce a 0-1 variable as follows:

$$z_{ia} = \begin{cases} 1 & (\text{If demand point i belongs to area } a\ (\in A)) \\ 0 & (\text{otherwise}) \end{cases} \tag{1}$$

where the number of areas is determined by the number of vehicles used for daily delivery. The purpose of this study is to divide the entire target field into areas without overlapping by multiple areas and to assign a demand point to one area. Then, the following relationship is established:

$$\sum_{a\in A} z_{ia} = 1 \qquad \forall i \in I \tag{2}$$

Let K be the set of trials for computer experiments and I_k be the set of demand points at which demand points occur in a certain trial $k(\in K)$. We denote by $C(I_k)$ the set of functions of I_k that extracts from I_k every combination that takes two demand points. Then, $z_{ik}z_{jk}$ indicates the presence or absence of a path connecting any demand points i and j belonging to an area $a(\in A)$ in a certain trial $k(\in K)$. The total number of paths between demand points belonging to the same area in all trials is expressed as follows:

$$\sum_{a\in A}\sum_{(i,j)\in C(I_k)}\sum_{k\in K} z_{ia}z_{ja} \tag{3}$$

The path across the different areas, except for the upper one, can be expressed as follows:

$$\sum_{a\in A}\sum_{(i,j)\in C(I_k)}\sum_{k\in K}(1 - z_{ia}z_{ja}) \tag{4}$$

In addition, we also introduce the following 0-1 variables:

$$s_{ij}^{k} = \begin{cases} 1(\text{If demand points i and j are delivered} \\ \quad \text{by the same vehicle in trial } k(\in K)) \\ 0(\text{otherwise}) \end{cases} \tag{5}$$

Using these variables, the total number of routes delivered to the same vehicle in each area can be shown as follows:

$$\sum_{a\in A}\sum_{(i,j)\in C(I_k)}\sum_{k\in K} s_{ij}^{k} \times z_{ia}z_{ja} \tag{6}$$

Similarly, the total number of deliveries between any demand points in different areas by the same vehicle can be shown as follows:

$$\sum_{a\in A}\sum_{(i,j)\in C(I_k)}\sum_{k\in K} s_{ij}^{k} \times \left(1 - z_{ia}z_{ja}\right) \tag{7}$$

The probability f_{in} of delivering to any demand point in the same area with the same vehicle and the probability f_{out} of delivering to any demand point in a different area with the same vehicle, which are the conditions that appeared in Section 2.1, can be expressed using the above variables as follows:

$$f_{in} = \frac{\sum_{a\in A}\sum_{(i,j)\in C(I_k)}\sum_{k\in K} s_{ij}^{k}\times z_{ia}z_{ja}}{\sum_{a\in A}\sum_{(i,j)\in C(I_k)}\sum_{k\in K} z_{ia}z_{ja}} \tag{8}$$

$$f_{out} = \frac{\sum_{a\in A}\sum_{(i,j)\in C(I_k)}\sum_{k\in K} s_{ij}^{k}\times(1-z_{ia}z_{ja})}{\sum_{a\in A}\sum_{(i,j)\in C(I_k)}\sum_{k\in K}(1-z_{ia}z_{ja})} \tag{9}$$

Furthermore, in this mathematical optimization, we set the constraint for capacity, i.e., the number of demand points belonging to each area, using the number of vehicles N as follows. The purpose of this is to adjust the delivery volume of each vehicle for practical applications:

$$\left\lfloor\frac{|I|}{N}\right\rfloor \le \sum_{i\in I} z_{ia} \le \left\lceil\frac{|I|}{N}\right\rceil \qquad \forall a \in A \tag{10}$$

Since satisfying conditions i and ii in Section 2.1 simultaneously means minimizing $f_{in} - f_{out}$, the following mathematical optimization problem is established:

$$max \quad f_{in} - f_{out} \tag{11}$$

$$s.t. \quad \sum_{a\in A} z_{ia} = 1 \qquad \forall i \in I \tag{12}$$

$$\left\lfloor\frac{|I|}{N}\right\rfloor \le \sum_{i\in I} z_{ia} \le \left\lceil\frac{|I|}{N}\right\rceil \qquad \forall a \in A \tag{13}$$

$$z_{ia}, s_{ij}^{k} \in \{0,\ 1\} \qquad \forall i,j \in I, \forall a \in A, \forall k \in K \tag{14}$$

2.3 Formulation based on zones

In Section 2.2, we formulated the equation based on the demand points. However, there are two issues. First, quadratic assignment problems with over 5,000 points cannot be solved in a short time. Second, considering practical applications, it is more convenient for the same vehicles to visit the same town.

For these reasons, we introduce “zone” aggregated demand points. The concept of the zone is illustrated in Figure 2. The zones are sets of demand points. The number of demand points for each zone is nearly equivalent. For example, zones are created by zip code. In addition, zones can be obtained by solving the *p*-median problem [9]. In this study, 100 zones are created, and each demand point is assigned to one zone. In the previous section (2.2), we formulated the equation in such a way that demand points are allocated to areas, and in this section (2.3), we formulate it in such a way that zones are allocated to areas.

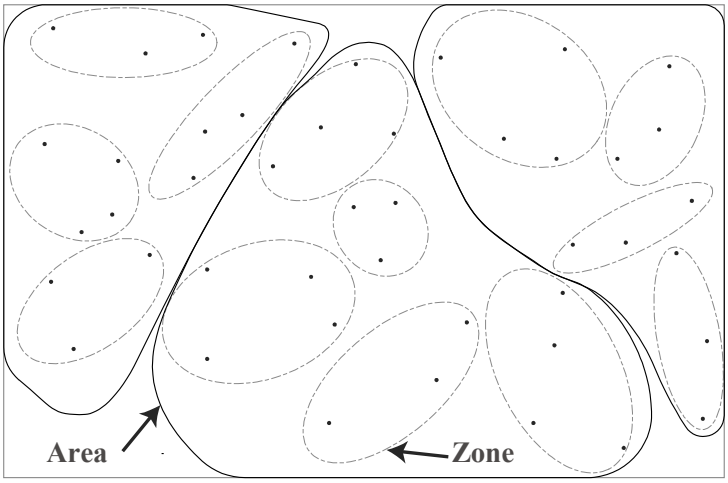


Fig. 2. Concept of zone

Let P be the set of zones. To propose a formulation that includes zones, we first introduce $p(\in P)$, and then we introduce a 0-1 variable that indicates whether zone p belongs to area $a(\in A)$:

$$x_{pa} = \begin{cases} 1 & (\text{If demand zone p belongs to area } a\ (\in A)) \\ 0 & (\text{otherwise}) \end{cases} \tag{15}$$

As shown in Section 2.2, we denote by $C(P)$the set of functions of P that extracts from P every combination that takes two demand points. At this time, let n_{pq}^{k} be the number of combinations such that one demand point is in zone p and another is in zone q among all combinations $C(I_k)$ of demand points I_k visited in trial k. The total number of deliveries in zones p and q of the same area a patrolled in a certain trial k can be expressed as $n_{pq}^{k} \times x_{pa}x_{qa}$, and the total number of deliveries within that same area is:

$$\sum_{a\in A}\sum_{(p,q)\in C(P)}\sum_{k\in K} n_{pq}^{k} \times x_{pa}x_{qa} \tag{16}$$

As in Section 2.2, the total number of routes that cross different areas is as follows:

$$\sum_{a\in A}\sum_{(p,q)\in C(P)}\sum_{k\in K} n_{pq}^{k} \times \left(1 - x_{pa}x_{qa}\right) \tag{17}$$

Furthermore, let t_{pq}^{k} be the number of combinations of all combinations $C(I_k)$ of demand points I_k visited in trial k, where one demand point is in zone p and the other in zone q, and where they are serviced by the same vehicle. Among those whose routes are in the same area, the total number delivered by the same vehicle is expressed:

$$\sum_{a\in A}\sum_{(p,q)\in C(P)}\sum_{k\in K} t_{pq}^{k} \times x_{pa}x_{qa} \tag{18}$$

In the same way as in Section 2.2, those items whose route is not in the same area, and which are delivered by the same vehicle, are shown as follows:

$$\sum_{a\in A}\sum_{(p,q)\in C(P)}\sum_{k\in K} t_{pq}^{k} \times \left(1 - x_{pa}x_{qa}\right) \tag{19}$$

Using the above, the f_{in} of delivering to any zone in the same area with a same vehicle and the probability and f_{out} of delivering to any zone in a different area with the same vehicle, which replace demand points to zones as they appeared in Section 2.2, can be shown as follows:

$$f_{in} = \frac{\sum_{a\in A}\sum_{(p,q)\in C(P)}\sum_{k\in K} t_{pq}^{k}\times x_{pa}x_{qa}}{\sum_{a\in A}\sum_{(p,q)\in C(P)}\sum_{k\in K} n_{pq}^{k}\times x_{pa}x_{qa}} \tag{20}$$

$$f_{out} = \frac{\sum_{a\in A}\sum_{(p,q)\in C(P)}\sum_{k\in K} t_{pq}^{k}\times\left(1-x_{pa}x_{qa}\right)}{\sum_{a\in A}\sum_{(p,q)\in C(P)}\sum_{k\in K} n_{pq}^{k}\times\left(1-x_{pa}x_{qa}\right)} \tag{21}$$

When the number of zones is constant, the number of demand points will also be constant, thus guaranteeing that the capacity of each zone is constant. Using this fact, and adding that the number of each zone should be as equal as possible, so that $f_{in} - f_{out}$ is maximized, the mathematical optimization problem can be shown as follows:

$$max \quad f_{in} - f_{out} \tag{22}$$

$$s.t. \quad \sum_{a\in A} x_{pa} = 1 \qquad \forall p \in P \tag{23}$$

$$\left\lfloor\frac{|P|}{N}\right\rfloor \le \sum_{p\in P} x_{pa} \le \left\lceil\frac{|P|}{N}\right\rceil \qquad \forall a \in A \tag{24}$$

$$x_{pa} \in \{0,\ 1\} \qquad \forall p \in P, \forall a\ \in A \tag{25}$$

3 Data Preparation

The optimization problem formulated in Section 2.3 was solved using the following experimental data. We considered the distribution of demand points as shown in Figure 1. There are 5,000 candidate demand points. There is only one depot in the upper left, and all delivery vehicles originate from here.

First, we randomly select demand points in accordance with the number of vehicles and the number of demand points delivered by one vehicle in one day. This means the distribution of demand for a hypothetical day. Next, for these demand points, we solve the CVRP using the LocalSolver package [10]. This is equivalent to calculating the optimal route minimizing the total distance for one day's demand distribution. We repeat these procedures for 300 trials, which means one year of delivery operations. In this way, computational experiments were conducted to find the optimal route for a year of demand. Thus, we obtained 300 patterns of hypothetical optimal delivery plans as computer experiments for solving the optimal problem.

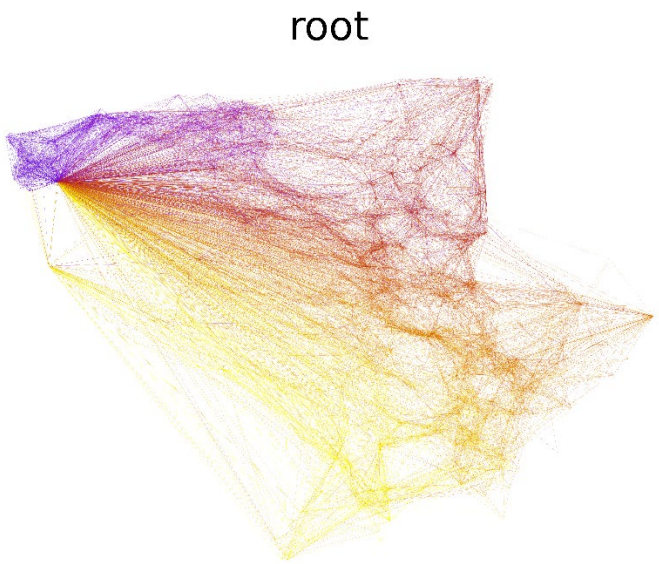


Fig. 3. Assembly of routes

By overlaying these computer experiment data, we can intuitively grasp the area distribution as shown in Figure 3. This figure shows a case with six delivery vehicles, and routes delivered by the same vehicle are shown in the same color. Such areas are created by the maximum likelihood estimated in (22)-(25).

4 Results

The following are the results of obtaining the solution using LocalSolver 10.5[11] with the equation in Section 2.3 and the data prepared in Section 3. The area assignment of demand points for the cases of 6 and 14 vehicles is shown in Figures 4 and 5 so that the same area has the same color. In addition, the white dot in the figures is the depot (starting point) of the delivery.

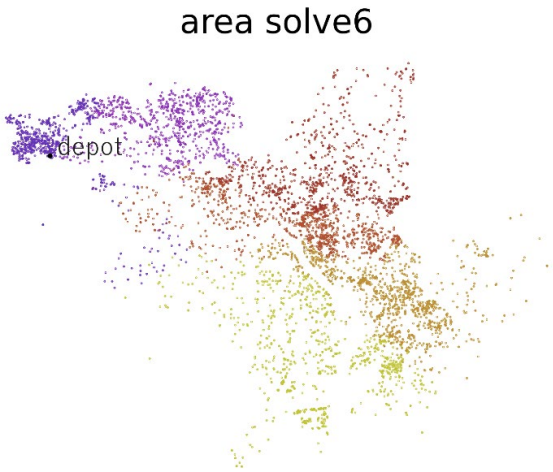


Fig. 4. Area solve (6 cars = 6 areas)

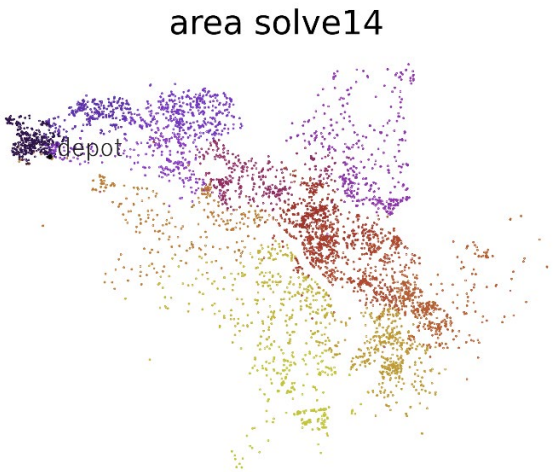


Fig. 5. Area solve (14 cars = 14 areas)

5 Discussion

From the results in the previous section, we can see that the formulation of this study allows us to make a clear assignment of areas, thus confirming that the area assignment

was reasonable. These results can be obtained regardless of the number of vehicles. Therefore, we can say that the formulation and constraints used in the process are appropriate. On the other hand, since we imposed a very strict condition on the capacity, it is necessary to examine how it changes when this condition is relaxed.

In this area assignment, the area spreads radially. This is an obvious solution since all the computer experiments start from the depot, but it also works positively in terms of efficiency in the actual delivery scenarios. In this sense, we can say that we have succeeded in creating a map that is suitable for the actual delivery scenarios.

6 Conclusion

In this study, we developed a model for the maximum likelihood estimation of area assignment from optimal routes. From the results, it can be inferred that the model is appropriate. In the next step, it will be necessary to verify whether the area assignment obtained in this study is superior in terms of efficiency to the currently used delivery areas.

References

1. Newell, F., Gordon, D., Carlos, F.: Design of multiple vehicle delivery tours-II other metrics. Transport Res B-Meth **20**(5), 365–376 (1986).
2. Ouyang, Y.: Design of vehicle routing zones for large-scale distribution systems. Transport Res B-Meth: Methodological **41**(10), 1079–1093 (2007).
3. Galvão, L.C., Novaes, A.G.N., Souza de Cursi, J.E., Souza, J.C.: A multiplicatively-weighted Voronoi diagram approach to logistics districting. Comput Oper Res **33**, 93–114 (2006).
4. García-Ayala, G., González-Velarde, J.L., Ríos-Mercado R.Z., Fernández, E.: A novel model for arc territory design: promoting Eulerian districts. Int Trans Oper Res **23**, 433–458 (2016).
5. Zhong, H., Hall, R.W., Dessouky, M.: Territory Planning and Vehicle Dispatching with Driver Learning. Transp. Sci. **4**(1), 74–89 (2007)
6. Sung, I., Nielsen, P.: Zoning a Service Area of Unmanned Aerial Vehicles for Package Delivery Services. J. Intell. Robot. Syst. **97**(3), 719–731 (2020).
7. Yudai, H., Atsushi, S.: New Clustering method to Estimate Overlapped Exchange Area from Regional Flow Matrix –Characteristics of population migrations and freight flow-. Journal of the City Planning Institute of Japan **55**(3), 475–481 (2020).
8. Mikio, K., Pedroso, J.P., Masakazu, M., Rais, A.: Mathematical Optimization: Solving Problems using Gurobi and Python. 2nd edn. Kindai Kagaku sha Co., Ltd., Japan (2013).
9. Densham, P.J., Rushton, G.: A more efficient heuristic for solving large *p*-median problems. The Journal of the RSAI **71** (3), 307–329 (1992).
10. Capacitated Vehicle Routing (CVRP) package, LocalSolver, https://www.localsolver.com/docs/last/exampletour/vrp.html, last accessed 2022/01/20.
11. LocalSolver, https://www.localsolver.com/, last accessed 2022/01/20.

A Comparative Study of Machine Learning Algorithms for Padel Tennis Shot Classification

Guillermo Cartes Domínguez[1], Evelia Franco Álvarez[2], Alejandro Tapia Córdoba[3] and Daniel Gutierrez Reina[1]

[1] University of Seville, Spain
guillecartes@hotmail.com, dgutierrezreina@us.es
[2] Comillas Pontifical University, Madrid, Spain
efalvarez@comillas.edu
[3] Universidad Loyola Andalucía, Seville, Spain
atapia@uloyola.es

1 Introduction

During the last few years, the rapid development in low-power semiconductor technology, in combination with the wide-adoption of mobile phone platforms, has fostered an astonishing growth of wearables, which have become especially relevant in the sport practice. The use of MEMS in sport activity represents an efficient way of monitoring, capable of generating a huge amount of data that can be further processed [1]. In addition, Machine Learning algorithms can leverage such datasets to develop supervised learning applications, such as classifiers and regressors, related to sport activities.

In this work, a comparison of Machine Learning algorithms for padel shot classification is proposed. Padel has recently experienced significant worldwide growth. For instance, in Spain, the number of practitioners has doubled in less than 9 years, surpassing tennis in 2020 and becoming one of the most popular sports at the national level[4]. Therefore, this paper aims to combine two growing disciplines: Machine Learning and padel.

The main contributions of this work are:

- The creation of the first padel shot dataset in the literature.
- The development of a padel shot classifier with the capacity to distinguish 13 different strokes.

2 Related Work

Shot classification has been proposed for other racket sports such as tennis and table tennis. In [2], a comparative study of table tennis strokes using Deep Learning is presented. The dataset employed included 1570 shots from 16 participants. Regarding tennis studies, in [3] the authors present a classification system for tennis shots. The system is developed by using two sensors placed, respectively, on the wrist and waist of the athlete. Up to four shot types have been classified and the achieved results reached a 99.25% accuracy. Similarly, in [4], a classification of 9 different tennis shots is performed using Machine Learning with a dataset generated by 19 different athletes. In this work, a total of 28.582 strokes were collected using an Inertial Measurement Unit (IMU) placed on the wrist of the athletes. Six different Machine Learning models were trained, among which the cubic-kernel Support Vector Machine (SVM) showed the best performance, reaching a 93.2% accuracy. Although previous works on racket sports are promising, to the best of the authors knowledge this is the first work focused on padel shot classification. Furthermore, the dataset created is the first one related to padel sport that is available in the literature.

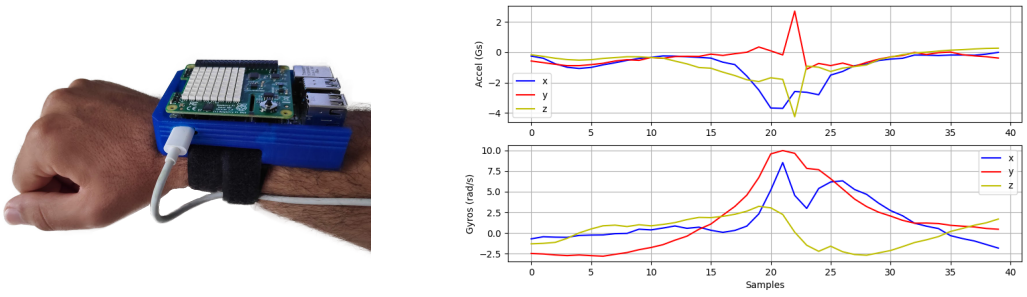
3 Padel dataset

The first challenge regarding the classification of padel tennis shots lies in the generation of a representative dataset from experimental data. As the dataset is to be used for training and

[4] Data of sport practitioners in Spain are avaliable in $https://www.csd.gob.es/es/federaciones-y-asociaciones/federaciones-deportivas-espanolas/licencias$.

testing supervised learning algorithms, the different shots are required to be properly labeled. A set of experiments have been carefully designed aiming at an efficient generation, processing and labeling of the experimental data. The procedure has been divided into three different stages:

1. **Data acquisition:** Each participant has performed the shots in 13 separate tests[5], each of them consisting on iteratively repeating a single type of shot. During the whole test, the players have worn the data acquisition device in the dominant hand, responsible for measuring both the velocities and accelerations experienced by the player's hand during a shot. The device consists of a LSM9DS1 IMU, which is included in the Sense Hat of the Raspberry Pi (see Figure 1a), and fixed into a tailored 3D printed case designed to ergonomically fit the player's wrist. Once activated, the device system collects data from the player on a continuous basis at a fixed rate of 20 Hz. It is relevant to note that, after the raw data has been captured, it consists of different time series containing a sequence of independent shots, and thus they are required to be identified and segmented.
2. **Shot detection and segmentation:** The shot detection algorithm developed is based on speed and acceleration thresholds in the three axis measured by the IMU. Once the data tests have been collected, a valid system for both detecting the precise length of a shot and removing the piece of data not related to the padel shot, must be applied. From a visual inspection of the tests, it has been observed that the 6 Degrees of Freedom (DOFs) undergo significant and easily identifiable changes when a shot occurs, reaching localized peak values in both accelerometer and gyroscope axes. For the shot detection, a threshold has been defined for each DOF in such a way that, if any of the accelerometer and gyroscope axis exceeds its corresponding threshold in the same sample, a shot can be considered to have occurred. However, it has been detected that the accelerometer (3 Gs) and gyroscope (5 rad/s) thresholds could be exceeded more than once during the same shot. To achieve the highest possible accuracy, a flag has been used to disable the search for a shot, as long as the last detected shot has not been completely saved, i.e. as long as the current sample belongs to a time interval of a previously detected shot. The stroke duration has been set to 2 seconds. The adjustment of both the thresholds and the duration has been conservative, as avoiding false positives is a priority to reach a precision as high as possible. In Fig. 1b, an example of a forehand stroke once the shot has been detected and singled out is shown.
3. **Dataset creation:** The dataset is finally built, collecting data from 12 real players (7 male and 5 female) aged between 22 and 50 years old. Participants' levels of play are different, and range from beginner to professional level. The final dataset contains a total of 2328 strokes representing the following 13 padel shot types: forehand ground, backhand ground, forehand wall, backhand wall, forehand lob, backhand lob, forehand lob wall, backhand lob wall, forehand volley, backhand volley, tray, smash, and serve.



(a) Data capture device. (b) Forehand shot data sample.

Fig. 1: Data capture device and Forehand shot data sample.

[5] Some of the experimental tests carried out can be seen in $https://youtu.be/UO_W Z40g1Io$

4 Methodology and results

Five different Machine Learning algorithms have been considered for this study. They can be divided into model and instance-based algorithms. Among the model-based algorithms, the one proposed are (fully connected) Neural Network (NN), one-dimensional Convolutional Neural Network (1D CNN), decision tree and SVM. On the other hand, a K-Nearest Neighbors (K-NN) has been considered as instance based algorithm. The classification of these algorithms is performed through two different types of input: the complete shot time series, on one hand, and feature engineering, on the other. For this last, the maximum, minimum and average values of each DOF of the shot time series have been considered. The input dataset has been divided into train (64%), validation (16%) and test (20%). In cases where validation is not carried out, the dataset has only been divided into train (70%) and test (30%). For a fair comparison between algorithms, a hyper-parametrization of each algorithm has been first carried out in order to correctly adjust each algorithm to the problem posed. This hyper-parametrization has been carried out through a grid search. The results obtained are summarized in Table 1, where it can be seen that the 1D CNN can achieve up to 93.35% accuracy. It is also relevant to note that only the results for time series input are shown, since they achieved better results than the feature engineering case.

Table 1: Parameters used for the machine learning algorithms and achieved test results.

Classifier	**Hyperparameters**	**Accuracy (%)**
NN	3 layers (1000 500 and 13 filters) 70 epochs 40 batch size	92.06
1D-CNN	1 convolutional layer (256 filters) 2 NN layers (1000 and 13 filters) 70 epochs 70 batch size	93.35
Decision Tree	40 max depth Min samples split 4 Min samples leaf 1 Entropy criterion	62.09
SVM	Kernel rbf C = 10	91.85
K-NN	k = 1	81.84

5 Conclusion

The work presented in this paper involves the creation of the first padel shot dataset and contributes to existing literature with the first classification of padel shots. For this purpose,the results of up to 5 different Machine Learning algorithms are compared resulting in an accuracy of up to 93%, achieved by the 1D Convolutional Neural Network.

References

1. Jonas Lutz, Daniel Memmert, Dominik Raabe, Rolf Dornberger, and Lars Donath. Wearables for integrative performance and tactic analyses: opportunities, challenges, and future directions. *International journal of environmental research and public health*, 17(1):59, 2020.
2. Sahar S Tabrizi, Saeid Pashazadeh, and Vajiheh Javani. Comparative study of table tennis forehand strokes classification using deep learning and svm. *IEEE Sensors Journal*, 20(22):13552–13561, 2020.
3. Luis Benages Pardo, David Buldain Perez, and Carlos Orrite Urunuela. Detection of tennis activities with wearable sensors. *Sensors*, 19(22):5004, 2019.
4. David Whiteside, Olivia Cant, Molly Connolly, and Machar Reid. Monitoring hitting load in tennis using inertial sensors and machine learning. *International journal of sports physiology and performance*, 12(9):1212–1217, 2017.

Sweep Algorithms for the Vehicle Routing Problem with Time Windows

Philipp Armbrust[1], Kerstin Maier[1], and Christian Truden[2]

[1] Department of Mathematics, Universität Klagenfurt, Klagenfurt, 9020, Austria
armbrust.philipp@aau.at, kerstinma@edu.aau.at
[2] Lakeside Labs GmbH, Klagenfurt, 9020, Austria
truden@lakeside-labs.at

Abstract. Attended home delivery services like online grocery shopping services require the attendance of the customers during the delivery. Therefore, the Vehicle Routing Problem with Time Windows occurs, which aims to find an optimal schedule for a fleet of vehicles to deliver goods to customers. In this work, we propose three sweep algorithms, which account for the family of cluster-first, route-second methods, to solve the Vehicle Routing Problem with Time Windows. In the first step, the customers are split into subsets such that each set contains as many as possible customers that can be served within one tour, e.g., supplied with one vehicle. The second step computes optimal tours for all assigned clusters. In our application, the time windows follow no special structure, and hence, may overlap or include each other. Further, time windows of different lengths occur. This gives additional freedom to the company during the planning process, and hence, allows to offer discounted delivery rates to customers who tolerate longer delivery time windows. Our sweep algorithms differ in the clustering step. We suggest a variant based on the standard sweep algorithm and two variants focusing on time window length and capacity of vehicles. In the routing step, a Mixed-Integer Linear Program is utilized to obtain the optimal solution for each cluster. The paper is concluded by a computational study that compares the performance of the three variants. It shows that our approach can handle 1000 customers within a reasonable amount of time.

Keywords: Vehicle routing· time windows· sweep algorithm· attended home delivery· transportation· logistics.

1 Introduction

The popularity of Attended Home Delivery (AHD) services, e.g., online grocery shopping services, increased within the last years. Especially, due to the current Covid-19 pandemic, this trend is continuing. For example in Western Europe, see [11], the share of online buyers is predicted to increase from 67% in 2020 to 75% in 2025. Therefore, the online share of groceries changed from 3.4% in 2019 to 5.3% in 2020 and is expected to reach 12.6% by 2025.

All AHD services have in common that the customer must attend the delivery of the goods or the provision of the booked service. To manage this in an effective manner, customers can typically choose among several time windows during which he or she is

available to receive the ordered goods or to supervise the service provision. We consider an application where different time window lengths occur. This allows an operator of a delivery service to offer discounted delivery rates to customers who tolerate longer delivery time windows as such offers are easier to include within a tour. Typically, the company aims to minimize the overall delivery costs and therefore, a variant of the Vehicle Routing Problem (VRP), the so-called Vehicle Routing Problem with Time Windows (VRPTW), occurs. A comprehensive introduction to the VRP can be found in [13]. The authors give an overview of different VRP types with the help of applications, case studies, heuristics, and integer programming approaches. Another overview and classification of recent literature considering varieties of the Vehicle Routing Problem can be found in [2]. For an overview of exact methods for the VRPTW we refer to [1]. Heuristic methods are reviewed in [3] and [4], and a compact review of exact and heuristic solving approaches for the VRPTW can be found in [6].

Due to the fact that in AHD services mostly a large number of customers are delivered, solving this problem to optimality within reasonable time is rarely possible. In practice, heuristics are applied to produce delivery schedules of high quality within a short amount of time. A common approach is the *cluster-first, route-second* method. In the first step, the customers are split into subsets, so-called clusters, such that each set contains as many as possible customers that can be served within one tour, e.g., supplied with one vehicle. In the second step, an optimal tour for each cluster is computed. [7] originally introduced the *Sweep Algorithm* for the VRP. Using the analogy of clock hands, the depot is placed at the center of the plane. A clock hand then sweeps across the plane. The angle of the hand increases while a customer is inserted in the current cluster, if a feasible insertion is possible, or otherwise within a new cluster. A recent study, see [5], applies the Sweep Algorithm for a VRP occurring in an install and maintenance service for smart meter devices. In [12], the sweep algorithm only considers time windows when computing the routes for each vehicle but not during the clustering step. [8] consider time windows already in the clustering step. However, they make use of a special time window structure, where the time windows are non-overlapping.

In this work, we propose the following three variants of performing the clustering step of a sweep algorithm for a VRPTW:

- a variant based on the standard sweep algorithm [7],
- a variant that takes the length of the delivery time windows into account, and
- a variant considering both the lengths of the delivery time windows and the vehicles' capacities.

We consider time windows with no special structure, and hence, they can overlap or contain each other. A Mixed-Integer Linear Program (MILP), which is stated in [9], is applied for deciding the feasibility of a cluster of customers, and for obtaining the optimal tour for each cluster. Considering non-overlapping time-windows, like in [8], would lead to a more efficient MILP formulation (see [9] for a comparison of both MILP approaches). However, most modern routing applications use time windows, which can overlap each other.

For the computational study, we use a large variety of benchmark instances that have been carefully constructed such that they resemble real-world data. We consider different capacities of the vehicles and therefore, the tour lengths can differ. This study shows

that our approach is capable of finding good initial solutions for instances containing up to 1000 customers within a short amount of time.

2 Mathematical Formulation

Now let us introduce the notation required to define the VRPTW. An instance of the VRPTW is defined by:

- A set of time windows $\mathscr{W} = \{w_1, \dots, w_q\}$, where each $u \in \mathscr{W}$ is defined through its start time s_u and its end time e_u with $s_u < e_u$.
- A set of customers $\mathscr{C}$, $|\mathscr{C}| = n$, with corresponding order weight function $c\colon \mathscr{C} \to]0, C]$, where $C \in \mathbb{R}_{>0}$ denotes the given vehicle capacity, and a service time function $s\colon \mathscr{C} \to \mathbb{R}_{>0}$.
- A function $w\colon \mathscr{C} \to \mathscr{W}$ that assigns a time window to each customer during which the delivery vehicle must arrive.
- A depot d from which all vehicles depart from and return to, $\overline{\mathscr{C}} \coloneqq \mathscr{C} \cup \{d\}$.
- A travel time function $t\colon \overline{\mathscr{C}} \times \overline{\mathscr{C}} \to \mathbb{R}_{\geq 0}$.

In the following, we state some basic definitions and we define the following notation $[u] \coloneqq [1, \dots, u]$, where $u \in \mathbb{N}$.

A *tour* of n customers consists of a set $\mathscr{A} = \{a_1, a_2, \dots, a_n\}$ and the indices of the costumers refer to the order of the customers. Each customer in the tour has a corresponding arrival time α_{a_i}, $i \in [n]$, during which the vehicles are scheduled to arrive. Furthermore, each tour has assigned *start* and *end times* that we denote as $start_{\mathscr{A}}$ and $end_{\mathscr{A}}$, respectively. Hence, the vehicle executing tour $\mathscr{A}$ can leave from the depot d no earlier than $start_{\mathscr{A}}$ and must return to the depot no later than $end_{\mathscr{A}}$.

We denote a tour $\mathscr{A}$ as *capacity-feasible*, if $\sum_{i=1}^{n} c(a_i) \leq C_{\mathscr{A}}$. The special case that the capacity of a single customer exceeds the capacity limit of the vehicles cannot occur.

If for each customer of a tour holds that the delivery of the goods occurs within the time window $u = w(a_i)$, i.e., $s_{w(a_i)} \leq \alpha_{a_i} \leq e_{w(a_i)}$, and there is enough time for fulfilling the order and traveling to the next customer, i.e., $\alpha_{a_{i+1}} - \alpha_{a_i} \geq s(a_i) + t(a_i, a_{i+1})$, for all $i \in [n]$, then we call it *time-feasible*.

A *feasible* tour is capacity- and time-feasible and a schedule $\mathscr{S} = \{\mathscr{A}, \mathscr{B}, \dots\}$ consists of feasible tours where each customer occurs exactly once.

Each element in $\overline{\mathscr{C}}$ has geographical coordinates in the two-dimensional plane and we assume that the travel times are correlated to the geographical distances. In general, the travel times are somehow related to, but not completely determined by the geographical distances. In our application, we typically deal with asymmetric travel time functions, for which the triangle inequalities, i.e., $t(a,c) \leq t(a,b) + t(b,c),\ a,b,c \in \overline{\mathscr{C}}$, do not hold. In general, $t(a,b) = t(b,a)$, where $a, b \in \overline{\mathscr{C}}$, is not guaranteed for an asymmetric travel time function.

The length of a time window $u \in \mathscr{W}$ is defined as $e_u - s_u$. Each customer can choose among time windows of length 10, 20, 30, 60, 120, or 240 minutes.

Two time windows u and v are overlapping, if $u \cap v \neq \emptyset$, and non-overlapping, if $u \cap v = \emptyset$. Time window u contains time window v, if $s_u \leq s_v$, and $e_v \leq e_u$. If time

window u contains time window v, then these time windows overlap each other. It must not hold that a time window s which overlaps t, also contains t.

In this work we consider three objectives, namely $\lambda_1, \lambda_2, \lambda_3$:

- The first one is the *number of vehicles used*, i.e., $\lambda_1(\mathscr{S}) := |\mathscr{S}|$.
- Secondly, the sum of all *tour durations* is denoted by the *schedule duration* $\lambda_2(\mathscr{S})$, i.e., $\sum_{\mathscr{A}\in\mathscr{S}} \lambda_2(\mathscr{A})$, where $\lambda_2(\mathscr{A}) := t(d,a_1) + \alpha_{a_n} - \alpha_{a_1} + s(a_n) + t(a_n,d)$.
- Thirdly, the sum of all *tour travel times* is denoted by the *schedule travel time* $\lambda_3(\mathscr{S})$, i.e., $\sum_{\mathscr{A}\in\mathscr{S}} \lambda_3(\mathscr{A})$, where $\lambda_3(\mathscr{A}) := t(d,a_1) + \sum_{i=1}^{n-1} t(a_i,a_{i+1}) + t(a_n,d)$.

Similar to [12] and [8], we aim to minimize these three objectives with respect to the lexicographical order $(\lambda_1, \lambda_2, \lambda_3)$, since providing a vehicle is usually the most expensive cost component, followed by the drivers' salaries, and the costs for fuel.

3 Sweep Strategy

A sweep algorithm is based on the polar coordinate angles of the customers $\mathscr{C}$, where the depot d lies in the center of the grid and $\theta(a) \in [0, 2\pi[$ denotes the angle component of a customer $a \in \mathscr{C}$.

We choose the zero angle, i.e., the starting angle, according to [8] such that it splits the two neighboring customers with the largest angle gap, i.e., $\max_{a,b\in\mathscr{C}} \theta(a) - \theta(b)$.

We propose the following general strategy to obtain solutions for a given VRPTW instance consisting of the following steps:

1. Apply one of the methods that obtain a feasible clustering of $\mathscr{C}$.
2. Compute the optimal route for each cluster.

In both steps, the *Traveling Salesperson Problem with Time Windows* (TSPTW) occurs as a subproblem to check time-feasibility or to obtain the optimal solution of a single tour. Next, we give further information about the two steps.

Clustering: We apply the following variants of clustering algorithms in Step 1.

- Traditional Sweep
- Sweep algorithm depending on time window length
- Sweep algorithm depending on time window length and overall capacity

Each of the three algorithms is described in more detail below. Moreover, we illustrate each variant with a toy example. The depot is located in the center of the coordinate system and the capacity of each vehicle is 10. In the visualizations we choose the direction of the zero angle $\theta_0 = 90^\circ$ as three o'clock and increase the angle in each step counterclockwise. Depending on the selected sweep algorithm we check in each iteration the capacity-feasibility or time- and capacity-feasibility. Note that the MILP is only used if the tour is capacity-feasible and the current tour candidate is not time-feasible, i.e., cannot feasible inserted without changing the order of the already placed customers. Then, the exact approach tries to find any time-feasible tour including the current costumers.

After obtaining an initial clustering using one of the heuristics, we determine efficient tours for each vehicle.

Routing: In Step 2, a tour for each cluster is obtained by solving the TSPTW-MILP with lexicographical objective (λ_2, λ_3) to optimality. We apply an MILP formulation that has been proposed in [9] for solving the TSPTW. Following the lexicographical order, the MILP first minimizes the tour duration $\lambda_2(\mathscr{A})$, and then the tour travel time $\lambda_3(\mathscr{A})$ while keeping $\lambda_2(\mathscr{A})$ fixed.

In the following subsections, we describe the different clustering steps in more detail.

3.1 Traditional Sweep

The first variant reflects the traditional algorithm by [7] except checking for both, time- and capacity-feasibility. It works as follows:

1. Compute the zero angle θ_0 and sort the customers according to their polar coordinate angles starting with θ_0.
2. Start with an empty cluster.
3. We check, if the next customer from the sorted set can be feasibly inserted within the current cluster.
   - If capacity- and time-feasibility holds, we add the new customer to the current cluster.
   - Otherwise, we initiate a new cluster with the current customer.
4. We repeat step 3 until all given customers are assigned within a cluster.

Therefore, the result of the traditional sweep algorithm is a set of capacity- and time-feasible clusters.

A simple example of the traditional sweep algorithm with five customers is depicted in Figure 1. The weights of the customers are given in their boxes, the capacity of each vehicle is 10, a depot in the center and we start with the zero angle at three o'clock (90°). In each step, we increase the angle counterclockwise, such that a new customer is added to the current cluster. Then we check the capacity- and time-feasibility of it and if necessary, we increase the number of clusters.

3.2 Sweep algorithm depending on time window length

In this subsection, we describe a sweep algorithm that takes the lengths of the delivery time windows into account. First, we sort all customers in ascending order of their time window lengths. Due to the high number of customers there are many customers with the same time window length and therefore, we start with the following procedure:

- The customer with the lowest time window length is added to the first cluster. If there is more than one customer with the same length, the sweep algorithm increases the angle beginning from the zero angle and selects the first customer of the set.
- In each sweep iteration: We increase the angle and consider the next customer within the current time window length and try to add this customer to the current cluster:

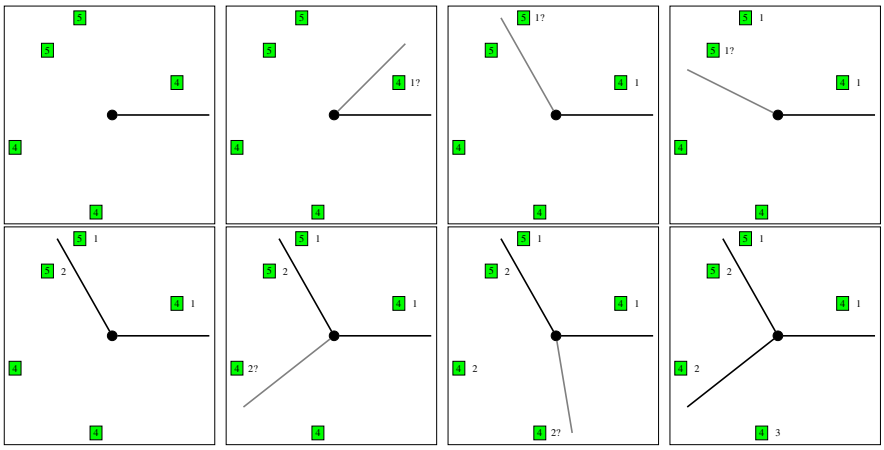


Fig. 1. Example of the traditional sweep method with one depot in the center. The capacity of each vehicle is 10 units, the weight of each customer is given in their boxes, and the number on the right side of the boxes denotes the number of the scheduled cluster.

  - If the tour is time- and capacity-feasible, add the customer to the current cluster.
  - Else, increase the number of clusters and add the customer to the next cluster.
- If no customer remains, increase the current time window length and start with the sweep iteration again, until all customers are scheduled within a cluster.

For further clarifying this algorithm, we consider a toy example with the customers given in Table 1.

Table 1. Toy example of customers sorted by their time window length for sweep algorithm depending on time window length.

		Time window properties		
	Required capacity	Start time	End time	Length
Customer 1	5	09:00	10:00	60
Customer 2	4	14:00	15:00	60
Customer 3	4	09:30	11:30	120
Customer 4	5	12:00	14:00	120

In our example, the capacity of each vehicle is 10. We start with the lowest time window length. There are two customers (customer 1 and customer 2) with the same time window length. Due to the given angles, customer 1 is added to the first cluster. In the following step, we consider customer 2, which we try to add to the current cluster, and therefore we check, if the tour remains time- and capacity-feasible after an insertion. In this case, the current tour satisfies both conditions, and hence, we add customer 2 to the first cluster. Next, we consider customer 3, which cannot be added to the current cluster,

due to the capacity limit of the vehicle. Therefore, we increase the number of clusters and add customer 3 to the second cluster. Then we try to add the last customer in the toy example, and hence, the time- and capacity-feasibility of the second tour with the next customer is checked. The insertion is possible, and hence, customer 4 is added to the current cluster. Our algorithm terminates with two clusters consisting of two customers each.

3.3 Sweep algorithm depending on time window length and overall capacity

Again we sort the customers by their time window lengths, and afterward by their polar coordinate angles. In contrast to Subsection 3.2, we now reserve half of the vehicle capacity for customers with longer time window lengths. The remaining procedure stays the same. As a result, this algorithm returns clusters, which are up to a half filled with customers having a time window length of 10, 20, or 30 minutes and the remaining capacity is filled with customers having a time window length of 60, 120, or 240 minutes. The idea behind this allocation is to ensure that the created tours have a proper mixture of short and long time windows. This property shall increase the robustness of the tours against delays occurring during the delivery process. The guaranteed portion of long delivery time windows introduces some slack such that late deliveries become less likely.

Now, we continue with a toy example and all information about the customers are given in Table 2.

Table 2. Toy example of customers sorted by their time window length for sweep algorithm depending on time window length and overall capacity constraints.

		Time window properties		
	Required capacity	Start time	End time	Length
Customer 1	5	09:00	09:10	10
Customer 2	4	14:00	14:30	30
Customer 3	5	15:00	15:30	30
Customer 4	4	09:30	10:30	60
Customer 5	4	11:30	13:30	120
Customer 6	5	12:00	14:00	120

First, we add the first customer (with the shortest time window length) to the first cluster. As we reserve half of the capacity for customers with time windows longer than 30 minutes, customer 3 (selected due to the polar coordinate angles of customer 2 and customer 3) is added to a new cluster, and the same procedure applies to customer 2. According to the polar coordinate angles, we try to add customer 5 to the first tour, and therefore, we check, if the cluster is time- and capacity-feasible. Both constraints are satisfied, and customer 5 is added to the first cluster. Next, we further increase the angle of the hand, and we consider customer 6 to be next. We notice that there is not sufficient capacity left in the first cluster. Hence, we assign customer 6 to the second cluster after

checking the constraints. Finally, customer 4 can be feasibly inserted into the third cluster. Therefore, the algorithm returns three clusters containing two customers each.

4 Computational Results

In this work, we introduce a benchmark set that imitates urban settlement structures. Customers can choose from time windows of different lengths, i.e, 10, 20, 30, 60, 120, or 240 minutes. The operation time of the vehicles is ten hours a day and the time windows are sampled randomly between the operation times. The customers are distributed on a 20 km $\times$ 20 km square grid that is roughly of the same size as the city of Vienna, Austria. Furthermore, 80% of the customers are arranged within randomly selected clusters and only 20% are uniformly distributed on the grid. The Euclidean distance is used to calculate the distance between two customers. We assume an average travel speed of 20 km/h (as proposed by [10]) to calculate the travel times.

The service time at each customer is set to five minutes and the order weights are sampled from a truncated normal distribution centered around five units. In the computational study, we consider different numbers of customers, namely $|\mathscr{C}| = \{250, 500, 750, 1000\}$, and the capacity of the vehicles is set to $C_{\mathscr{A}} = 200$, $\mathscr{A} \in \mathscr{S}$. The instances are available at http://dx.doi.org/10.13140/RG.2.2.20934.60480. We use an Ubuntu Mint 20 machine equipped with an Intel Xeon $E5-2630V3$@2.4 GHz 8 core processor and 132 GB RAM and Gurobi 8.1.1 in single-thread mode to solve the Mixed-Integer Linear Programs. For each instance, we apply our three methods and the average results over 10 instances for the different numbers of customers are given in Tables 3 - 6. We observe similar behavior for all instance sizes considered. The Traditional Sweep is much faster compared to the two other methods for both steps in the sweep algorithm. As the first method obtains up to four times the number of clusters, the number of customers within a tour is very low, and therefore, the routing step is very efficient using an MILP approach. However, the number of vehicles is crucial, and thus, the methods two and three perform in a costefficient manner. There is no big difference in the results between the two methods with and without considering the overall capacity. However, it is expected that the third sweep variant produces more robust tours as described in Subsection 3.3. Thus, we are able to improve robustness without any loss of quality with respect to all three objectives. Further, the traditional sweep badly performs with respect to λ_2 but slightly improves λ_1 compared to the other two variants. We are able to find clusters for all algorithms within eight minutes for up to 1000 customers. The runtime for the routing step is up to one and a half hours for variants two and three.

5 Conclusion

In this paper, we considered several sweep algorithms, which belong to cluster-first, route-second methods, for the Vehicle Routing Problem with Time Windows. The first step of the algorithm clusters the customers according to their polar coordinate angles originating from the depot as the center point of the grid. Secondly, the tour for each cluster is determined. Considering the clustering step, we introduced a variant based on the standard sweep algorithm and two variants focusing on time window length and

Table 3. Average results with 250 customers over 10 instances each. We denote the runtime by t. λ_1 depicts the number of vehicles used, λ_2 gives the schedule duration, and λ_3 denotes the total travel time.

	Clustering		Routing		
Runtime/Objectives	t	λ_1	t	λ_2	λ_3
Units	[m:ss]		[mm:ss]	[hhh]	[hh]
Traditional Sweep	0:11	32.7	00:01	222	50
Time window length	1:29	11.7	14:56	46	67
Length & overall capacity	1:36	11.9	14:29	46	68

Table 4. Average results with 500 customers over 10 instances each. We denote the runtime by t. λ_1 depicts the number of vehicles used, λ_2 gives the schedule duration, and λ_3 denotes the total travel time.

	Clustering		Routing		
Runtime/Objectives	t	λ_1	t	λ_2	λ_3
Units	[m:ss]		[mm:ss]	[h]	[h]
Traditional Sweep	0:46	65.9	00:01	460	101
Time window length	3:22	19.8	33:57	76	116
Length & overall capacity	3:48	20.0	40:27	77	116

Table 5. Average results with 750 customers over 10 instances each. We denote the runtime by t. λ_1 depicts the number of vehicles used, λ_2 gives the schedule duration, and λ_3 denotes the total travel time.

	Clustering		Routing		
Runtime/Objectives	t	λ_1	t	λ_2	λ_3
Units	[m:ss]		[h:mm:ss]	[h]	[h]
Traditional Sweep	1:51	101,3	0:00:02	714	144
Time window length	6:27	29,4	0:57:51	111	165
Length & overall capacity	6:31	29,8	1:00:40	114	155

Table 6. Average results with 1000 customers over 10 instances each. We denote the runtime by t. λ_1 depicts the number of vehicles used, λ_2 gives the schedule duration, and λ_3 denotes the total travel time.

	Clustering		Routing		
Runtime/Objectives	t	λ_1	t	λ_2	λ_3
Units	[m:ss]		[h:mm:ss]	[h]	[h]
Traditional Sweep	1:53	149.6	0:00:03	1040	196
Time window length	7:26	37.1	1:16:02	151	203
Length & overall capacity	7:49	36.9	1:23:27	152	200

capacity of vehicles. Further, a benchmark set with different customer sizes is provided. Each customer chooses the length of the time window within a given set, namely 10, 20, 30, 60, 120, and 240 minutes. Our computational study showed that the heuristic is able to cluster 1000 customers within eight minutes. In the routing step, the optimal tours are calculated by a Mixed-Integer Linear Program, which results in runtimes of up to one and a half hours. It remains for future work to apply a heuristic approach that gathers good quality solutions in a fraction of time.

Acknowledgments This work was supported by Lakeside Labs GmbH, Klagenfurt, Austria, and funding from the European Regional Development Fund and the Carinthian Economic Promotion Fund (KWF) under grant 20214/31942/45906.

References

1. Baldacci, R., Mingozzi, A., Roberti, R.: Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. European Journal of Operational Research **218**(1), 1–6 (2012)
2. Braekers, K., Ramaekers, K., Nieuwenhuyse, I.V.: The vehicle routing problem: State of the art classification and review. Computers & Industrial Engineering **99**, 300–313 (2016)
3. Bräysy, O., Gendreau, M.: Vehicle routing problem with time windows, Part I: Route construction and local search algorithms. Transportation Science **39**(1), 104–118 (2005)
4. Bräysy, O., Gendreau, M.: Vehicle routing problem with time windows, Part II: Metaheuristics. Transportation Science **39**(1), 119–139 (2005)
5. Bucur, P.A., Hungerländer, P., Jellen, A., Maier, K., Pachatz, V.: Shift planning for smart meter service operators. In: Haber, P., Lampoltshammer, T., Mayr, M., Plankensteiner, K. (eds.) Data Science – Analytics and Applications. pp. 8–10. Springer Fachmedien Wiesbaden, Wiesbaden (2021)
6. El-Sherbeny, N.A.: Vehicle routing with time windows: An overview of exact heuristic and metaheuristic methods. Journal of King Saud University **22**, 123–131 (2010)
7. Gillett, B.E., Miller, L.R.: A heuristic algorithm for the vehicle-dispatch problem. Operations Research **22**(2), 340–349 (1974)
8. Hertrich, C., Hungerländer, P., Truden, C.: Sweep algorithms for the capacitated vehicle routing problem with structured time windows. In: Fortz, B., Labbé, M. (eds.) Operations Research Proceedings 2018. pp. 127–133. Springer International Publishing, Cham (2019)
9. Hungerländer, P., Truden, C.: Efficient and easy-to-implement mixed-integer linear programs for the traveling salesperson problem with time windows. Transportation Research Procedia **30**, 157 – 166 (2018)
10. Pan, S., Giannikas, V., Han, Y., Grover-Silva, E., Qiao, B.: Using customer-related data to enhance e-grocery home delivery. Industrial Management & Data Systems **117**(9), 1917–1933 (2017)
11. Savills: European Food and Groceries Sector, European Commercial, Savills Commercial Research (04 2021), https://pdf.euro.savills.co.uk/european/europe-retail-markets/spotlight---european-food-and-groceries-sector---2021.pdf, [Online, accessed 19.12.2021]
12. Solomon, M.M.: Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research **35**(2), 254–265 (1987)
13. Toth, P., Vigo, D. (eds.): The Vehicle Routing Problem: Problems, Methods, and Applications. Society for Industrial and Applied Mathematics, 2nd edn. (2014)

Adaptive Continuous Multi-Objective Optimization using Cooperative Agents

Quentin Pouvreau[1,2][0000−0003−0700−5149], Jean-Pierre Georgé[1][0000−0002−4255−236X], Carole Bernon[1][0000−0002−7602−141X], and Sébastien Maignan[1]

[1] IRIT, Université de Toulouse, CNRS, Toulouse INP, UT3, Toulouse, France
www.irit.fr
[2] ISP System, Vic-en-Bigorre, France
www.isp-system.fr
{name.surname}@irit.fr

Abstract. Real-world optimization of complex products (*e.g.*, planes, jet engines) is a hard problem because of huge multi-objective and multi-constrained search-spaces, in which many variables are to be adjusted while each adjustment essentially impacts the whole system. Since the components of such systems are manufactured and output values are obtained with sensors, these systems are subject to imperfections and noise. Perfect *digital twins* are therefore impossible. Furthermore simulating with sufficient details is costly in resources, and the relevance of Population-based optimization approaches, where each individual is a whole solution to be evaluated, is severely put in question. We propose to tackle the problem with a Multi-Agent System (MAS) modeling and optimization approach that has two major strengths : 1) a natural representation where each agent is a variable of the problem and is perceiving and interacting through the real-world topology of the problem, 2) a cooperative solving process where the agents continuously adapt to feedback, that can be interacted with, can be observed, where the problem can be modified on-the-fly, that is able to directly control these variables on a real-world product while taking into account the specifics of the components. We illustrate and validate this approach in the *Photonics* domain, where a light beam has to follow a path through several optical components so as to be transformed, modulated, amplified, etc., at the end of which sensors give feedback on several metrics that are to be optimized. Robotic arms have to adjust the 6-axis positioning of the components and are controlled by the Adaptive MAS we developed.

Keywords: Continuous Optimization · Multi-Objective Optimization · Adaptive Multi-Agent Systems · Robotics Control · Photonics.

1 Problem Statement and Positioning

This study mainly concerns optimization problems from real-world applications, especially robotics control command based on sensor feedback. These applications go from system configuration based on test bench feedback to real-time

feedback control of an automated system. In the first case we want to optimize the response to an input. In the second case we mainly want to minimize the distance between sensor feedback and objectives by positioning, orienting, aligning one or more components. We define a robot, an actuator or any automated system as a composition of one or more axes, which are associated with the degrees of freedom of the system. Such problems have a wide variety of external constraints like limited resolution time or limited number of moves, predetermined components positions to respect, etc. We intend to develop an optimization system able to adapt to multiple robotics applications. The application domain we focus on is the photonics domain as illustrated in Fig. 1.

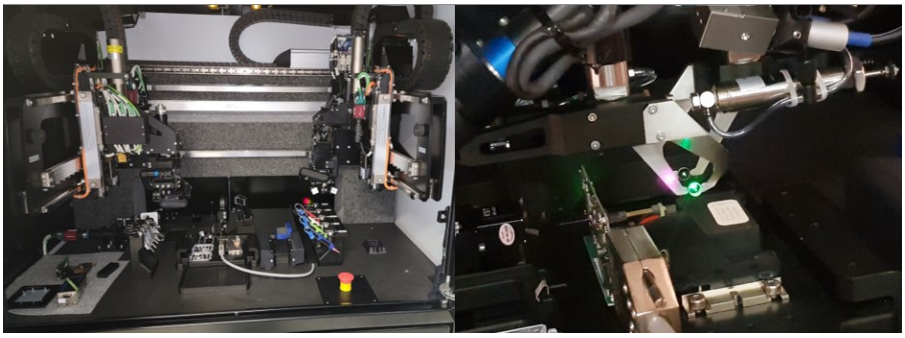


Fig. 1. Example of a real photonics application we are working on (credit ISP System)

1.1 Search Space Topology

Optimization problems are divided into domains depending on their search space topology. Search space dimensions are defined by decision variables, and potentially limited by hard constraints. For example the number of robots and the number of axes per robot increase the dimensions. On the other hand, limits on a component assembly reduce the possibilities on one or more axes, so one or more dimensions get constrained in the corresponding search space. These constraints, alongside one or more expert-given objectives, most of the time antinomic, make appear a Pareto front in the search space. As constraints can generally be transposed into objectives dealing with the distance to a threshold (the further away from the threshold, the better, or defining a cost regarding an acceptable violation), these problems can be defined as Multi-Objective Optimization (MOO) problems, also called Pareto optimization problems.

Another factor of dimensioning of the search space is the decision variable domain. Depending on whether the domain is discrete or continuous, optimization problems are combinatorial or continuous. Since the precision of robotic systems is continuously increasing, robots positioning problems can be considered as continuous optimization problems.

Indeed, the axis resolution, meaning the minimal step with accuracy guarantee, is often very small compared to the range of values it can achieve. Furthermore, axes resolutions, ranges and other robots features can change from an application to another, changing therefore the decision variable domains.

A continuous problem has a potentially infinite number of solutions when a combinatorial problem has only all possible combinations of its discrete variables. This gap can have a significant influence in terms of calculation cost. However, those properties do not mean that combinatorial problems are trivial and continuous problems are not. It means that search space topology is quite different from a category to another so the employed method might not have the same results.

In this study, we focus on Continuous Multi-Objective Optimization problems and we consider that reliable problems with only one objective are a particular subdivision of these problems and can be processed in the same way.

1.2 Multi-Objective Optimization Approaches

This section introduces the main approaches used in the optimization domain before focusing on the most suited for our concern.

Analytical approaches are the most accurate but they are time-consuming and may not also be suitable for large-scale optimization problems. Arithmetic programming approaches on the contrary have fast computation performances since they are based on simplifications and sequential linearizations. However, they are very weak in handling multi-objective nonlinear problems and may converge to local optima, non-necessarily satisfying enough [16].

Meta-heuristic optimization algorithms are extensively used in solving Multi-Objective Optimization problems since they can find multiple optimal solutions in a single run, and improve the ratio between accuracy and computational cost. They are problem-independent optimization techniques which provide, if not optimal, at least satisfying solutions by stochastically exploring and exploiting search spaces iteratively [17]. The following sections focus on these algorithms.

Population-Based Heuristics are a large part of the state of the art in Meta-heuristic Optimization Algorithms. The main principle is to simultaneously handle a population of solutions spread randomly or not in the search space. The population can evolve and select the best solutions iteratively, or converge to an optimum following a set of influence rules. These algorithms can also be used in hybrid solutions alongside more classical algorithms like simulated annealing [1] to balance their weaknesses.

It is difficult to be exhaustive about all works in this domain. For instance, Genetic Algorithms [9] and Particle Swarm Optimization Algorithms [12] have together more than 3000 publications per year [18].

A major limitation of Population-Based approaches is their potential computational cost. Such algorithms need to evaluate a relatively large number of

candidates in order to create a good population of solutions. Computing all the candidate solutions can be prohibitive for computationally expensive problems.

The type of applications we are studying here requires an adaptation each time we get a sensor feedback. As a result, an evolution process would require a huge amount of resources. As each new candidate in the population needs to be evaluated, the robot needs to reconfigure the whole experimental setting for each proposed configuration. An adaptive algorithm modifying and proposing for testing a unique configuration at each feedback is a more suitable strategy (i.e. a resolution process forming a single trajectory in the search space).

Moreover, when scaling up the number of objectives, the Pareto-dominance relation essentially loses the ability to distinguish desirable solutions, since nearly all population members are non-dominated at an early stage of the search. In fact, a large majority of the usual Population-Based methods (evolutionary algorithms, swarm intelligence) have been shown to degrade when the number of objectives grows beyond three, and moreover beyond eight [10]. This particularity explains why a part of the literature about these approaches focuses on Many-Objective Optimization, that is Multi-Objective Optimization problems with more than three objectives [13]. The need for this category of problems to have more relevant indicators than Pareto-dominance makes the Population-Based Heuristics to specify additional calculation for solution comparison.

The types of problems we are interested in require an optimization system able to converge without maintaining and computing a large number of solutions, especially when solution comparison becomes non-trivial. Furthermore, we need a decentralized real-time control of robots arms to actually optimize the real world system being processed, taking into account errors and noise.

Multi-Agent Systems (MAS) are a decentralized approach, based on self-organisation mechanisms [22], where the calculation task is distributed over *agents* which are virtual or physical autonomous entities[19].

Each agent has only a local point of view of the problem it is solving, corresponding to a local objective function. The global objective function of a problem is then the sum of all these local functions. This particularity enables to easily distribute calculation tasks in the resolution process and consequently reduces computational costs. That is why multi-agent approaches are preferred where centralized approaches have limited flexibility and scalability.

Multi-Agent Systems are used in a wide variety of theoretical [15] and real-world application domains of distributed optimization [21]: classification optimization algorithms [3], power systems [7] , complex networks and IoT [5] , smart manufacturing [2] or multi-robot systems [20].

Distributed Constraint Optimization Problems (DCOPs) are a well-known class of combinatorial optimization problems prevalent in Multi-Agent Systems [4]. DCOP is a model originally developed under the assumption that each agent controls exactly one variable.

This model was designed for a specific type of problems where the difficulty resides in the combination of multiple constraints. These problems are supposed to be easily decomposable into several cost functions, where the cost values associated with the variables states are supposed to be known. This major assumption does not stand for complex continuous optimization problems, where the complexity of the models and their interdependencies cause this information to be unavailable in most cases. It is important to remark that some works tried to extend DCOP model to continuous optimization problems but the state of art about those works remains scattered for now [8].

2 AMAS Theory for Optimization

As seen before, a MAS is a problem-independent solution making it possible to have a natural representation where each agent is a variable of an MOO problem. These agents, which perceive their environment and interact, are also a means to continuously adapt to real-world feedback and provide a "solution" anytime especially when the problem can be modified on-the-fly.

We propose to adopt the Adaptive Multi-Agent Systems (AMAS) theory where cooperation [6] is the engine that drives the adaptation of an agent and the emergence of a global functionality. This cooperation relies on three mechanisms : an agent may adjust its internal state to modify its behavior (tuning), may modify the way it interacts with its neighborhood (reorganization) or may create other agents or self-suppress when there is no other agent to produce a functionality or when a functionality is useless (evolution).

2.1 Natural Domain Modeling

As we previously stated, when solving complex continuous problems existing techniques usually require a transformation of the initial formulation, in order to satisfy some requirements for the technique to be applied. Beside the fact that correctly applying these changes can be a demanding task for the designers, imposing such modifications changes the problem beyond its original, natural meaning. What we propose here is an agent-based modeling where the original structure/meaning of the problem, is preserved. Indeed it represents the formulation which is the most natural and easiest for the expert to manipulate. We call this modeling *Natural Domain Modeling for Optimization* (NDMO) [11].

In order to represents the elements of a generic continuous optimization model, we identified five classes of interacting entities: *models*, *design variables*, *outputs*, *constraints* and *objectives*. Briefly: given the values of the design variables, certain models will calculated output values, which will enable other models to calculate other outputs and so on, until constraints and objectives can be calculated, in a sort of calculus network. In general, three elements need to be *agentified*: the design variables (because they need to be optimised and that constitutes the solving process), the constraints and objectives. The last two

are there to model the requirements or statements of the problem, i.e. what the solving process has to achieve.

To this end, we use a mechanism based on a specific measure called *criticality*. This measure represents the state of dissatisfaction of the agent regarding its local goal. Each agent is in charge of estimating its own criticality and providing it to the other agents. The role of this measure is to aggregate into a single comparable value all the relevant indicators regarding the state of the agent. Having a single indicator of the state of the agent is interesting as it simplifies the reasoning of the agents. However the system designer has the difficult task to provide the agents which adequate means to calculate their criticality.

2.2 Agent Internal State and behavior

Our system is a work-in-progress implementation based on the AMAK Framework [14]. A main system representing the AMAS handles an environment representation and a collection of agents interacting with the environment. Each agent controls a parameter of the system, consequently a decision variable of the problem. The environment and the set of agents execute an iteration to update their state one at a time. The agents iteration order is randomly updated at the beginning of each cycle. All the agents have the same three-phase algorithm:

- Perception phase: the agent gets an observation of the environment, that is a set of *criticalities* calculated from the distance to the objective.
- Decision phase: the agent processes its new data and follows a decision tree to adjust its internal state.
- Action phase: the agent acts following its decision by changing its parameter.

The agent decision phase aims at increasing or decreasing the value of the decision variable it is responsible for (a predefined variation step depending on its characteristics), and is thus at the core of the process. Except in one case that we will explain below, a decision is always repeated a stochastically chosen number of iterations, from one to ten. This *momentum* mechanism allows to desynchronize the agents decisions to prevent them from being trapped in what we call *non cooperative synchronisations* (basically when two agents try to "help" at the same time, thus hindering each other).

When starting the resolution, each agent has only a set of criticalities given by the last environment update. Since it does not have any idea of which action is the best, its first decision is to act randomly. If the agent observes the criticalities decreased beyond a configurable threshold, it will repeat its decision. It is generally useful to converge fast when the current region of the search space is relatively regular. On the contrary, if the criticalities increased beyond the same threshold, the agent will make only one step in the opposite way. These two rules make a first decision process we can qualify as reactive. When it is not possible to reactively spot an adequate decision, the agent will process data it registered in the last perception phases. This more cognitive process consists in interpolating the variation of the criticalities according to its own value. The

goal is to identify a region (plus or minus) where the integrative is the lower. It appears that the momentum mechanism is also useful to this decision process since it made the agent do what we might call a stochastic scanning of its local area. In the rare case the second decision process fails to give a decision, the agent acts randomly.

3 Photonics Problem Modeling and Implementation

The environment of the AMAS has to update the system state, apply the changes from the AMAS and calculate the input variables used by the agents to take their decisions. The simulator we developed is shown in Fig. 2.

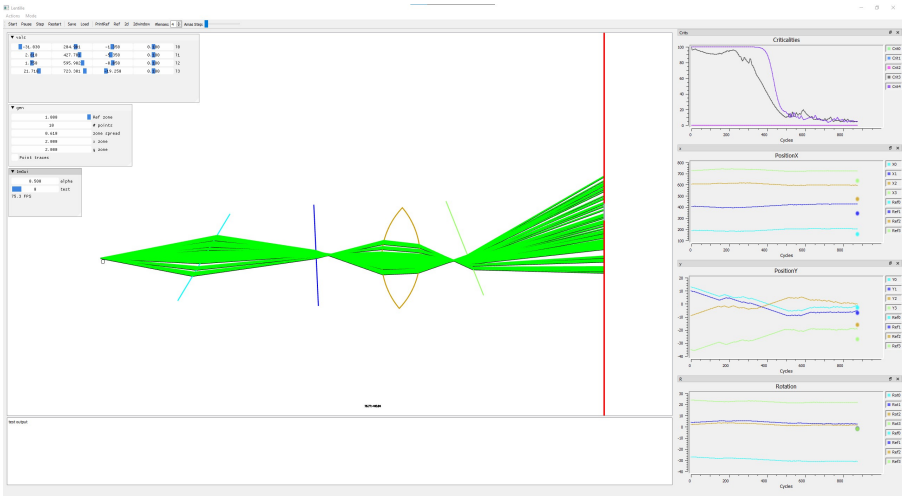


Fig. 2. Screenshot of the simulator at runtime

The system is a 2D-world composed of a light source, several lenses (L_i with i in $[1, N]$) and a screen. The light source emits a number of rays (R_j with j in $[1, M]$) in a conic shape. If a ray intersects with a lens it is refracted using Snell's law of refraction *i.e.* $n1.sin(\theta_1) = n2.sin(\theta_2)$ (with each θ as the angle measured from the normal of the boundary, and n as the refractive index of the respective medium). So assuming a ray goes through all lenses in the system (which is the desired state) we have a mathematical suite of operations applied to its position and orientation $R_j(pos_{j,i}, \theta_{j,i}) = L_i(R_j(pos_{j,i-1}, \theta_{j,i-1}))$ with i in $[1, n]$ and j in $[1, m]$.

Test cases for the AMAS are generated so that all rays pass through all lenses as follows: a set of lenses with various characteristics (thin or cylindrical, refraction index, focal length) are randomly placed on the axis between the source and the screen (X). A set of rays is generated parallel to the X axis

so that each ray goes through all lenses and reaches the screen. Repeatedly, the lenses are shifted and rotated randomly as well as the direction of the rays, while keeping all rays going through all lenses. After a number of cycles, the state of the system is set as the reference to reach for the AMAS. Then the lenses are randomly placed and rotated. This new state is used as a starting point for the AMAS to work with.

A lens L_i is represented by its type (thin or cylindrical), its position $P_i(x, y)$ and rotation angle T_i, its refraction index n_i and focal length F_i. For cylindrical lenses the radius of each face are also necessary $R1_i$ and $R2_i$. From these parameters only P_i and T_i can change during the run and are controlled by the AMAS.

A ray is a more complex structure since it is represented by a path. A path is an ordered list of positions and directions describing the intersection points with the various lenses it goes through $\{p_{j,k}(x, y)\}$ with k the index of the intersection, and the direction of the ray at these points represented as an angle with X $\{ang_{j,k}\}$.

Rays are not directly known by the AMAS. Only the last position and direction of each ray (when it reaches the screen) is used to derive information to send to the AMAS as a feedback to its actions.

The derived information can be the results of various computations. The most straightforward is to form a set of M differences between current rays and reference rays: $\{|p_{j,last}(y) - p^{ref}_{j,last}(y)|, |ang_{j,last} - ang^{ref}_{j,last}|\}$. This gives the AMAS quite a lot of precise information which is not often readily available in real life systems.

The second type of derived information are root mean square deviations ($rmsd$) of the positions and angles: $R_{pos} = \sqrt{(\sum((p_{j,last}(y) - p^{ref}_{j,last}(y))^2)/M)}$ and $R_{ang} = \sqrt{(\sum((ang_{j,last} - ang^{ref}_{j,last})^2)/M)}$

These two values are more representative of what can be perceived on a real system like the global intensity on the screen.

For each of the experiments presented hereafter a set of parameters are given which represents the setup of the run: first, the sequence of lenses, from source to screen, present in the system with C for a cylindrical lens and T for a thin lens. Then the number of rays, the percentage of maximum step for lenses moves and the type of information sent to the AMAS (full or $rmsd$). So for an experiment with 3 lenses, 10 rays, 50% of maximum step and using the $rmsd$, this set would be $\{CTC, 10, 50\%, rmsd\}$.

4 Experiments

First we checked the ability to manage different types of lenses, as for instance, a cylindrical lens has more complex interactions with rays than a thin lens.

We generated two experiments $\{T, 100, 10\%, rmsd\}$ and $\{C, 100, 10\%, rmsd\}$ (Fig. 3 and Fig. 4) in which all the other parameters were equal. As visible on the graphs, the evolution of criticality is more chaotic with a cylindrical lens.

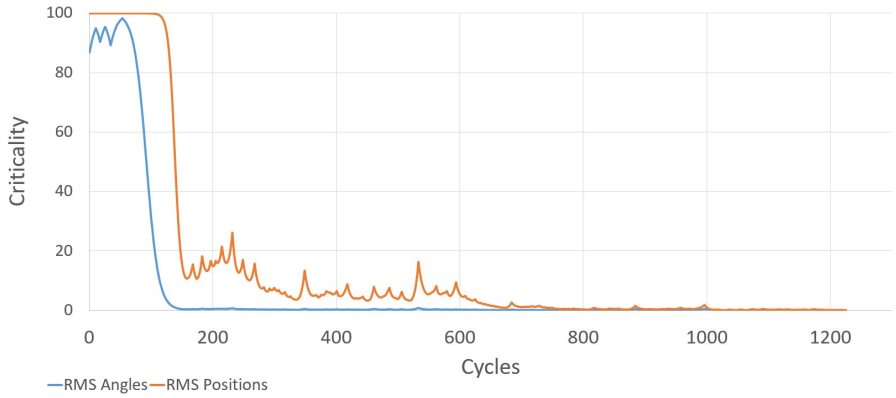


Fig. 3. Resolution with one thin lens $\{T, 100, 10\%, rmsd\}$

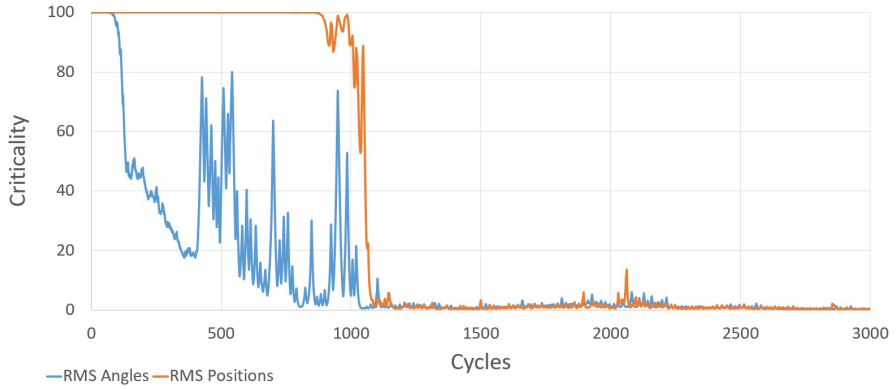


Fig. 4. Resolution with one cylindrical lens $\{C, 100, 10\%, rmsd\}$

We remark in the one lens experiments that curves make some peaks repetitively. These are the consequence of the momentum mechanism seen in section 2.2 and do not impact the convergence that remains globally continuous.

The other experiments (Figs. 5 and 6) show that the system seems to be scalable in terms of number of decision variables. In these cases, the peaks mentioned earlier disappeared. The results of the moves of each axis agent are more softened as the number of interactions between rays and optical surfaces grows.

This proof of concept shows promising results: the resolution process succeeds and no divergence from a satisfying area of the search space has been observed. However, some adjustments that have to be explored yet could greatly improve the resolution process. The difficulties encountered are mainly due to the problem itself: almost all positioning values impact all criticalities, and in a non-linear way. The parameters consequently are strongly interconnected: the current posi-

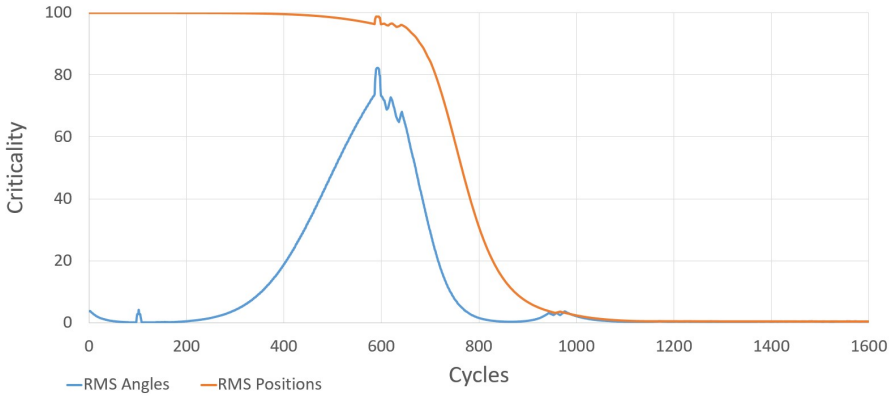


Fig. 5. Resolution with two lenses $\{CT, 100, 1\%, rmsd\}$

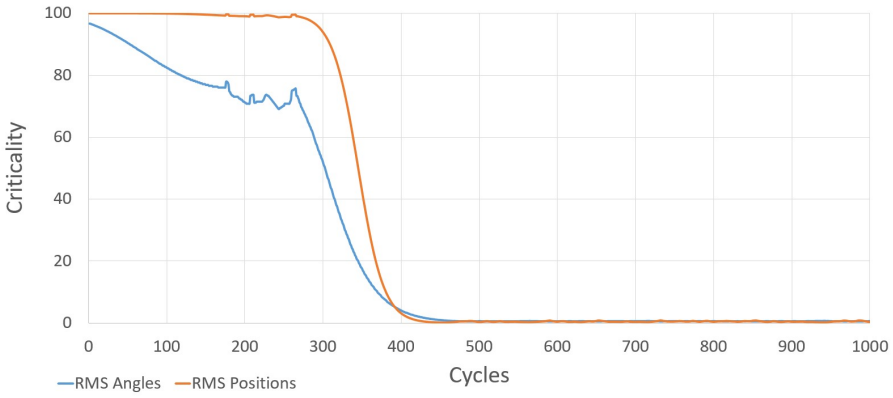


Fig. 6. Resolution with three lenses $\{TTT, 100, 1\%, rmsd\}$

tion of one axis agent moves the target position of one or more others. Therefore, the system is chaotic and the agents can collectively hinder themselves. At this point the optimization problem becomes a cooperation problem.

It has to be noted that the examples presented here are voluntary more difficult for the AMAS than a real case, where the starting positions are nearer from the optimum. Starting further away lets us test how the system behaves while crossing the vast search space of the problem. In this way we can observe that it does not suffer divergence from any achievement it has already made. Further work will be done to optimise the number of cycles needed to bring the criticalities down near zero.

5 Conclusion and Perspectives

State of the art in multi-objective optimization is dominated by Population-Based and DCOP approaches. However, it can be difficult for those methods to be used for real-world applications, especially when the context brings new constraints like narrowed computation time or resources. This is even worse in real world robotics control where only one "current solution" can be manipulated by the robots, and no *digital twin* is possible.

Moreover, we identified a limitation of current continuous optimization methods regarding the handling of complex problems with a multi-dimensional continuous search space. Problems of this category are usually too complex to be solved by classical optimization methods due to multiple factors: the inter-dependencies of their objectives, their heavy computational cost, their non-linearities, etc.

This limitation has been the motivation to propose a new decentralized approach. We designed a solver fitted for a large panel of real-world applications with miscellaneous search space topologies. This approach also permits to easily scale up problem complexity, in terms of number of parameters as well as number of objectives. Its aim is to naturally model a real-world optimization problem as a cooperative resolution problem and to satisfy as much as possible expert given objectives at a reasonable computation cost. The first results obtained with a proof of concept are promising. Enhancing the optimization process will now rely on enriching the cooperation capabilities of the agents.

Acknowledgements This work has been financially supported by the *Région Occitanie* (`www.laregion.fr`) as part of the READYNOV 2019-2020 research program. Quentin Pouvreau has been co-funded by the *Association nationale de la recherche et de la technologie (ANRT)* (`www.anrt.asso.fr`) and by *ISP System*(`www.isp-system.fr`), who also provided the test cases and expert knowledge on photonics.

References

1. Assad, A., Deep, K.: A hybrid harmony search and simulated annealing algorithm for continuous optimization. Information Sciences **450**, 246–266 (2018)
2. Bendul, J.C., Blunck, H.: The design space of production planning and control for industry 4.0. Computers in Industry **105**, 260–272 (2019)
3. Couellan, N., Jan, S., Jorquera, T., Georgé, J.P.: Self-adaptive support vector machine: A multi-agent optimization perspective. Expert systems with Applications **42**(9), 4284–4298 (2015)
4. Fioretto, F., Pontelli, E., Yeoh, W.: Distributed constraint optimization problems and applications: A survey. Journal of Artificial Intelligence Research **61**, 623–698 (2018)
5. Fortino, G., Russo, W., Savaglio, C., Shen, W., Zhou, M.: Agent-oriented cooperative smart objects: From iot system design to implementation. IEEE Transactions on Systems, Man, and Cybernetics: Systems **48**(11), 1939–1956 (2017)

6. Georgé, J.P., Gleizes, M.P., Camps, V.: Cooperation. In: Serugendo, G.D.M., Gleizes, M.P., Karageorgos, A. (eds.) Self-organising Software, pp. 193–226. Natural Computing Series book series (NCS), Springer (2011)
7. González-Briones, A., De La Prieta, F., Mohamad, M.S., Omatu, S., Corchado, J.M.: Multi-agent systems applications in energy optimization problems: A state-of-the-art review. Energies **11**(8), 1928 (2018)
8. Hoang, K.D., Yeoh, W., Yokoo, M., Rabinovich, Z.: New algorithms for continuous distributed constraint optimization problems. In: Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (2020)
9. Holland, J.H.: Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press (1992)
10. Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Evolutionary many-objective optimization: A short review. In: 2008 IEEE congress on evolutionary computation (IEEE world congress on computational intelligence). pp. 2419–2426. IEEE (2008)
11. Jorquera, T., Georgé, J.P., Gleizes, M.P., Régis, C.: A Natural Formalism and a MultiAgent Algorithm for Integrative Multidisciplinary Design Optimization. In: IEEE/WIC/ACM International Conference on Intelligent Agent Technology - IAT. pp. 146–154. Atlanta, United States (2013)
12. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95-international conference on neural networks. vol. 4. IEEE (1995)
13. Maltese, J., Ombuki-Berman, B.M., Engelbrecht, A.P.: A scalability study of many-objective optimization algorithms. IEEE Transactions on Evolutionary Computation **22**(1), 79–96 (2016)
14. Perles, A., Crasnier, F., Georgé, J.P.: Amak-a framework for developing robust and open adaptive multi-agent systems. In: International Conference on Practical Applications of Agents and Multi-Agent Systems. pp. 468–479. Springer (2018)
15. Sghir, I., Hao, J.K., Jaafar, I.B., Ghédira, K.: A multi-agent based optimization method applied to the quadratic assignment problem. Expert Systems with Applications **42**(23), 9252–9262 (2015)
16. Shaheen, A.M., Spea, S.R., Farrag, S.M., Abido, M.A.: A review of meta-heuristic algorithms for reactive power planning problem. Ain Shams Engineering Journal **9**(2), 215–231 (2018)
17. Sharma, M., Kaur, P.: A comprehensive analysis of nature-inspired meta-heuristic techniques for feature selection problem. Archives of Computational Methods in Engineering **28**(3) (2021)
18. Wang, Z., Qin, C., Wan, B., Song, W.W.: A comparative study of common nature-inspired algorithms for continuous function optimization. Entropy **23**(7) (2021)
19. Weiß, G.: Multiagent Systems, A modern Approach to Distributed Artificial Systems. MIT Press (1999)
20. Yan, Z., Jouandeau, N., Cherif, A.A.: A survey and analysis of multi-robot coordination. International Journal of Advanced Robotic Systems **10**(12), 399 (2013)
21. Yang, T., Yi, X., Wu, J., Yuan, Y., Wu, D., Meng, Z., Hong, Y., Wang, H., Lin, Z., Johansson, K.H.: A survey of distributed optimization. Annual Reviews in Control **47**, 278–305 (2019)
22. Ye, D., Zhang, M., Vasilakos, A.V.: A survey of self-organization mechanisms in multiagent systems. IEEE Transactions on Systems, Man, and Cybernetics: Systems **47**(3), 441–461 (2016)

A reinforcement learning-based local search

W. Benzineb[1], L. Loukil[1], B. Amrane A.[1], and H. Benyamina[1]

University of Oran 1
benzineb.walid@edu.univ-oran1.dz,
loukil.lakhdar@univ-oran1.dz,amrane.bakhta@univ-oran1.dz,benyamina.hassen@univ-oran1.dz

Abstract. Local search methods (LSM) are known to be potent optimization metaheuristics that allow for efficient exploration of large and complex search space. They are, however, 'amnesiac' search process in that they do not harness the data generated from previous executions which we consider to be a wealth of information that can be used to improve the search process. In this work, we propose a reinforcement learning-based LSM that combines LSM with reinforcement learning (RL) paradigm to improve the search process. The proposal was tested on the quadratic 3-dimensional assignment problem (Q3AP). Experiments showed that LSM combined with RL significantly improved the performance of LSM in terms of solution quality and execution time.

keywords: Metaheuristics, Local search, Reinforcement learning, Tabu search, Simulated annealing.

1 Introduction

Local search methods (LSM) are widely known as powerful optimization metaheuristics that allow for efficient exploration of large and complex search space. They initiate the search process from a feasible solution of the targeted problem, carry out a succession of moves until a predefined stopping criterion is met and then return the best solution found so far. The most well-known local search methods are, but not limited to, simulated annealing (SA) [5], tabu search (TS) [3], variable neighborhood search (VNS) [4] and iterated local search (ILS) [6].

The move selection procedure from the set of potentially eligible moves is a key component of local search methods and therefore the selection procedure of a move amongst the potentially acceptable ones is crucial for their performance in terms of quality of solutions and execution time. A common practice in using a local search in optimization is to perform several independent search processes (sequentially or in parallel) and take as (sub-)optimal solution a statistical measure (usually the *mean*) of the solutions returned by each independent search. Each of these local search processes starts from scratch and does not exploit data from previous searches. We consider that data gathered from previous searches is a valuable mine of knowledge that can be harnessed to make a local search more effective.

Machine learning (ML) is a field of artificial intelligence that focuses on developing systems that learn from a dataset and leverages the knowledge mined from this dataset to improve the performance of the targeted system. Machine learning-based approaches have been approved to be strong methods to solve large scaled or difficult problems where explicit algorithms do not show good performance [1]. Reinforcement learning (RL) is a machine learning paradigm that aims at taking suitable actions that would maximize the total cumulative reward of the RL agent. A RL agent takes an action, moves to a new state and receives a reward that determines the quality of the action taken. The RL agent's goal is to maximize the accumulation of rewards over time.

The remaining of the paper is organized as follows. Section 2 describes the reinforcement learning-based LSM. Section 3 reports the results of the experiments. We end with a conclusion.

2 Reinforcement learning approach for local search method

As noted previously, the move selection is a key component of local search metaheuristics. Our proposal is a move scoring algorithm that assigns scores to moves so that the more interesting the move, the higher the score. The move scoring procedure is depicted by Algorithm 1. A `MAXSCORE` value is set and will be assigned to the best solution encountered during tests. All moves of a local

search leading to a solution `sol` are recorded in a the so-called `solMoveList`. First, we begin by scoring the solution `sol`: if the cost of `sol` (i.e. f(`sol`)) is better than the best cost found so far, then it is assigned `MAXSCORE` otherwise SolScoring function is invoked to compute the score of `sol` (Algorithm 1, lines 4-9). Now that the solution score is set, we update the score list of this solution and we proceed to the scoring of its moves by querying the function MovesScoring by passing to it the list `solMoveList` of the moves carried out between `initSol` and `sol`, the score `scoreSol` of the solution `sol` as well as the list `moveScoreList` of all the moves. To score a move in `solMoveList`, the procedure MovesScoring proceeds as follows. It first tests if the move is new in which case a default score `DEFAULTSCORE` is assigned to it. If not, the score of the move is set to its current score increased by the value (`solScore` $\times\delta$) (Algorithm 1, lines 32-33). The list of move scores, `moveScoreList`, is then updated and the same process is performed for the other moves.

Algorithm 1 Moves scoring pseudo-code.

```
 1: procedure Scoring (initSol, sol, bestSol, solMoveList, solScoreList, moveScoreList)
 2:     Input: initSol, sol, bestSol, solMoveList, solScoreList, moveScoreList;
 3:     Output: solScoreList, moveScoreList;
 4:     if ( f(sol) < f(bestSol) ) then
 5:         bestSol = sol; solScore = MAXSCORE;
 6:     else
 7:         SolScoring (sol, bestSol, solScore);
 8:     end if
 9:     UpdateSolScoreList (solScore, solScoreList);
10:     MovesScoring (initSol, sol, solScore, solMoveList, moveScoreList);
11: end procedure
12:
13: procedure SolScoring (sol, bestSol, solScore)
14:     Δ = (f(sol)×100)/f(bestSol)); solScore = MAXSCORE ×Δ;
15: end procedure
16:
17: procedure MovesScoring (initSol, sol, solScore, solMoveList, moveScoreList)
18:     i = 1; s_0 = initSol; Δ = f(sol) - f(s_0);
19:     while (solMoveList[i] ≠ NULL) do
20:         if (not(solMoveList[i] ∈ moveScoreList)) then
21:             moveScore = DEFAULTSCORE
22:         else
23:             moveScore=getScoremove(solMoveList[i],moveScoreList);
24:         end if
25:         δ = f(s_i) - ((f(s_{i-1}) × 100)/Δ);
26:         moveScore = moveScore + (solScore × δ);
27:         UpdateMoveScoreList (moveScore, moveScoreList);
28:         i = i + 1;
29:     end while
30: end procedure
```

Algorithm 2 describes how the proposed scoring technique is implemented into a local search template to improve search quality and execution time. At each move selection phase, a list of potential moves is first generated from the list of move scores returned by our machine learning-based scoring approach. Each potential move is indexed with a number equal to his score so that higher score moves are more likely to be selected. If the set of potential moves is not empty, a move is randomly selected from this set otherwise the selection method inherent to the local search is applied.

3 Experiments and discussion

In order to experiment our proposal, we implemented two well-known local search metaheuristics: simulated annealing (SA) and variable neighborhood search (VNS). SA and VNS have been used to

Algorithm 2 Template of a local search procedure with scoring method.

```
1: procedure LS(P, moveScoreList)
2:     s_0 =GenerateInitSol(P);
3:     k = 0;
4:     while (stopping criterion not met) do
5:         potentialMoveList = SelectPossibleMoves (moveScoreList,s_k);
6:         if ( potentialMoveList ≠ ∅) then
7:             move(k) = Rand ( potentialMoveList);
8:         else
9:             move(k) =SelectAccepatbleMove(s_k);
10:        end if
11:        s_{k+1} = MakeMove(s_k, move(k));
12:        k = k + 1;
13:    end while
14:    return s_k;
15: end procedure
```

optimize the quadratic 3-dimensional assignment problem (Q3AP) [7]. For SA, the cooling schedule function is the geometric schedule ($T_{k+1} = \alpha T_k$, where $\alpha \in [0, 1]$). For VNS, 4 neighborhood functions have been implemented:

- `DoubleSwap`: Randomly swap two elements to create a new solution.
- `TripleSwap`: A random permutation between three elements.
- `OptheaderSwap`: We take two random elements not including the first two elements and we swap them with the latter.
- `DichotomySwap`: We take the left elements of the center and swap them all with the right side.

Our approach has been tested on a set of Q3P instances derived from QAPLIB [2]: 9 instances derived from Nugent benchmarks and 2 instances derived from Hadley benchmarks. The maximum number of iterations of the LS was set to 100. Table 1 reports the results of the experiments. Let

Table 1: Experimental results

	Algo	Nug13	Nug15	Nug18	Nug20	Nug22	Nug25	Nug27	Nug30	Had16	Had18
BKV		1912	2230	17836	25590	42467	37716	13266	69704	52980	84932
BFV	SA/RL	1912/1912	2230/2230	17836/17836	25590/25590	43108/42467	37716/37716	14813/13266	72006/69704	52980/52980	84932/84932
	VNS/RL	1912/1912	2230/2230	17836/17836	25590/25590	43016/42467	37716/37716	14516/13266	72180/69704	52980/52980	84932/84932
#hits	SA/RL	31/74	4/37	0/9	1/11	0//7	2/27	0/16	0/8	7/46	8/34
	VNS/RL	28/79	7/52	2/23	2/29	0/11	1/24	0/19	0/11	7/49	9/41
AvgBC	SA/RL	2473/2108	2861/2421	20894/18952	28461/26433	48007/44641	39018/38461	16898/14108	74181/71962	55873/53746	87818/85843
	VNS/RL	2538/2046	2846/2381	20431/18463	28413/25998	48131/44382	39156/38721	16789/14081	74087/71067	55793/53567	87801/85838
AvgTime	SA/RL	431/181	198/114	891/507	1448/989	2716/2101	2466/1798	3087/2188	4889/3997	401/279	585/397
	VNS/RL	467/153	196/108	867/464	1403/891	2756/1989	2468/1816	3085/2154	4884/3887	403/259	581/381

- `BKV`: Best known value in the literature for the Q3AP instance.
- `BFV`: Best found value by the LS.
- `#hits`: number of times the LS has caught the `BKV`.
- `AvgBC`: Average of the best costs returned by the LS.
- `AvgTime (sec)`: Average of the execution time.

us note that the local search combined with our scoring approach based on reinforced learning has achieved good performances in terms of solution quality and execution time compared to a standard local search. Let's particularly note a significant improvement in the number of hits in LS+R compared to LS alone, particularly for big instances. Furthermore, for `Nug22`, `Nug27` and `Nug30`, we notice the failure of basic local search to catch the best known value (`BKV`) while the local search combined with the scoring method succeeded in finding the optimal value.

4 Conclusion

In this work, we have proposed and evaluated a reinforcement learning method based on a score system to enhance local search algorithms. Summing up the results, we see that the method was

successful in upgrading our local searches by obtaining better results in terms of both the quality of the solution and the execution time thanks to more efficient search space exploitation. In the future, we would like to apply the scoring method to more local searches for further studies and test it on even higher benchmark sizes.

Acknowledgements

This paper is supported by the PRIMA programme under grant agreement No 1821, project WATERMED4.0. The PRIMA programme is supported by the European Union."

References

1. K. P. Bennett and E. Parrado-Hernández. The interplay of optimization and machine learning research. *Journal of Machine Learning Research*, 7(46):1265–1281, 2006.
2. R. E. Burkard, S. E. Karisch, and F. Rendl. QAPLIB - A quadratic assignment problem library. *J. of Global Optimization*, 10(4):391â403, jun 1997.
3. F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.
4. P. Hansen and N. Mladenović. *Variable Neighborhood Search*, pages 211–238. Springer US, Boston, MA, 2005.
5. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.
6. H. R. Lourenço, O. C. Martin, and T. Stützle. *Iterated Local Search*, pages 320–353. Springer US, Boston, MA, 2003.
7. W. P. Pierskalla. Letter to the editorâthe multidimensional assignment problem. *Operations Research*, 16(2), 1967.

Assessing Similarity-Based Grammar-Guided Genetic Programming Approaches for Program Synthesis

Ning Tao[1,2], Anthony Ventresque[1,2][0000−0003−2064−1238], and Takfarinas Saber[1,3][0000−0003−2958−7979]

[1] Lero – the Irish Software Research Centre, Ireland
[2] School of Computer Science, University College Dublin, Dublin, Ireland
ning.tao@ucdconnect.ie, anthony.ventresque@ucd.ie
[3] School of Computer Science, National University of Ireland, Galway, Ireland
takfarinas.saber@nuigalway.ie

Abstract. Grammar-Guided Genetic Programming is widely recognised as one of the most successful approaches for program synthesis, i.e., the task of automatically discovering an executable piece of code given user intent. Grammar-Guided Genetic Programming has been shown capable of successfully evolving programs in arbitrary languages that solve several program synthesis problems based only on a set of input-output examples. Despite its success, the restriction on the evolutionary system to only leverage input/output error rate during its assessment of the programs it derives limits its scalability to larger and more complex program synthesis problems. With the growing number and size of open software repositories and generative artificial intelligence approaches, there is a sizeable and growing number of approaches for retrieving/generating source code based on textual problem descriptions. Therefore, it is now, more than ever, time to introduce G3P to other means of user intent (particularly textual problem descriptions). In this paper, we would like to assess the potential for G3P to evolve programs based on their similarity to particular target codes of interest (obtained using some code retrieval/generative approach). We particularly assess 4 similarity measures from various fields: text processing (i.e., FuzzyWuzzy), natural language processing (i.e., Cosine Similarity based on term frequency), software clone detection (i.e., CCFinder), plagiarism detector(i.e., SIM). Through our experimental evaluation on a well-known program synthesis benchmark, we have shown that G3P successfully manages to evolve some of the desired programs with three of the used similarity measures. However, in its default configuration, G3P is not as successful with similarity measures as with the classical input/output error rate at evolving solving program synthesis problems.

Keywords: Program Synthesis · Grammar-Guided Genetic Programming · Code Similarity · Textual Description · Text To Code

1 Introduction

Genetic Programming (GP [16]) is an efficient approach to evolve code using high-level specifications, hence it is the most popular approach to tackle program synthesis problems (i.e., the task of automatically discovering an executable piece of code given user intent) for software engineering [25, 24] and testing [26]. Various GP systems with different representations have been designed over time to tackle the diverse program synthesis problems.

PushGP [22] is one of the most efficient GP systems. PushGP evolves programs in the specially purpose-designed Push language. (i.e., a stack-based language designed specifically for program synthesis task). In Push, every variable type (e.g. strings, integers, etc.) has its own stack, which facilitates the genetic programming process. Despite its efficiency, PushGP's dependence to Push (a language that is not commonly used in practice and that is hard to interpret) hinders its exploitability and lowers our ability to improve upon it.

Grammar-Guided Genetic Programming (G3P [7]) system is another efficient GP system that evolves programs based on a specified grammar syntax. Besides its efficiency at solving program synthesis problems, the use of a syntax grammar enables G3P to produce programs that are syntactically correct with respect to any arbitrary programming languages definable through a grammar. The use of a grammar makes G3P particularly easy to move from one system to another and to adapt from one language to another [7]. This flexibility elevates G3P to be widely recognised as one of the most successful program synthesis approaches.

A recent comparative study [6] has evaluated the ability of both G3P and PushGP to solve several program synthesis problems from a well-studied program synthesis benchmark [11, 10] based only on a set of input-output examples. The study found that G3P achieves the highest success rate at finding correct solutions when it does find any. The study also found that PushGP is able to find correct solutions for more problems than G3P, but PushGP's success rate for most of the problems was very low. However, despite G3P's and PushGP's successes, the restriction on the evolutionary systems to only leverage the input/output error rate during their assessment of the programs they derive limits their scalability to larger and more complex program synthesis problems.

Following on the big data trend [4] and the growing number and size of open software repositories (i.e., databases for sharing and commenting source code) and generative artificial intelligence approaches (generative deep learning) there is a sizeable and growing number of approaches for retrieving/generating source code based on textual problem descriptions. Therefore, it is now, more than ever, time to introduce G3P to other means of user intent (particularly textual problem descriptions). Code retrieval and code generation techniques might output several incomplete snippets or not fully fit for purpose codes–which often makes them impossible to exploit in their form. Therefore, in this work, we propose an approach whereby such code guides the search process towards programs that are similar.

In this paper, we would like to assess the potential for G3P to evolve programs based on their similarity to particular target codes of interest which would have

been retrieved or generated using some particular text to code transformation. We particularly assess 4 similarity measures from various fields: text processing (i.e., FuzzyWuzzy), natural language processing (i.e., Cosine Similarity), software clone detection (i.e., CCFinder), plagiarism detector(i.e., SIM). The ultimate goal is the ability to identify the most suitable program similarity measure to guide the program synthesis search/evolutionary process when introducing code retrieval/generation from textual problem descriptions.

Through our experimental evaluation on a well-known program synthesis benchmark, we show that G3P successfully manages to evolve some of the desired programs with three of the used similarity measures. However, in its default configuration, G3P is not as successful with similarity measures as with the classical input/output error rate at evolving solving program synthesis problems. Therefore, in order to take advantage of textual problem descriptions and their subsequent text to code approaches in G3P, we need to either design better-fitted similarity measures, adapt our evolutionary operators to take advantage of program similarities, and/or combine similarity measures with the traditional input/output error rate.

The rest of the paper is structured as follows: Section 2 summarises the background and work related to our study. Section 3 describes our approach and details the similarity metrics used as code similarity in our evaluation. Section 4 details our experimental setup. Section 5 reports and discusses the results of our experiments. Finally, Section 6 concludes this work and discusses our future study.

2 Background and Related Work

In this section we present the material which forms our research background.

2.1 Genetic Programming

Genetic programming (GP) is an evolutionary approach that enables us to devise programs. GP starts with a population of random programs (often not very fit for purpose), and iteratively evolves it using operators analogous to natural genetic processes (e.g., crossover, mutation, and selection). Over the years, a variety of GP systems have been proposed–each with its specificity (e.g., GP [16], Linear GP [2], Cartesian GP [19]).

2.2 Grammar-Guided Genetic Programming

While there is a variety of GP systems, G3P is among the most successfully GP systems. What is unique to G3P is its use of a grammar as a guideline for syntactically correct programs throughout the evolution. Grammars are widely used due to their flexibility as they can be defined outside of the GP system to represent the search space of a wide range of problems including program synthesis [20], evolving music [17], managing traffic systems [31] evolving aircraft

models [3] and scheduling wireless communications [29, 18, 28, 30, 27]. Grammar-Guided Genetic Programming is a variant of GP that use grammar as the representation with most famous variants are Context-Free Grammar Genetic Programming (CFG-GP) by Whigham [32] and grammatical evolution [21].

The G3P system proposed in [7] puts forward a composite and self-adaptive grammar to address different synthesis problems, which solved the limitation of grammar that has to be tailored/adapted for each problem. In [7], several small grammars are defined–each for a data type that defines the function/program to be evolved. Therefore, G3P is able to reuse these grammars for different problems while keeping the search space small by not including unnecessary data types.

2.3 Problem Text Description to/from Source Code

The ability to automatically obtain source code from textual problem descriptions or explain concisely what a block of code is doing have challenged the software engineering community for decades.

The former (i.e., source code from textual description) was aimed at automating the software engineering process with a field mostly divided into two parts: (i) Program Sketching which attempts to lay/generate the general code structure and let either engineers or automated program generative approaches fill the gaps (e.g., [14]), and (ii) Code Retrieval which seeks to find code snippets that highly match the textual description of the problem in large code repositories.

The latter (i.e., textual description from source code) was mostly to increase the readability of source code and assist software engineering with their debugging, refactoring, and porting tasks. Several works have attempted to either provide meaningful comments for specific lines/blocks (e.g., [13]) or to generate brief summaries for the source code (e.g., [1]).

3 Similarity-Based G3P

In this section, we report on how similarity-based G3P perform on selected three problems from [10]. The G3P system in [7] uses error rate fitness function based on given input and output data for evolving the next generation, while similarity-based G3P presented in this section uses code similarity value to the given correct program.

3.1 Proposed Approach

Our ultimate goal is to exploit textual descriptions of user intent in the program synthesis process in combination with current advances in code retrieval/generation (even if such techniques potentially generate multiple incomplete or not fully fit for purpose programs) to guide the search process of G3P. To this end, in this work, we devise a similarity-based G3P system, which uses code similarity to evaluate the fitness of evolved programs against a target source code instead of input/output error rate. The focus of this particular work is not on generating

target source code but on (i) assessing the capability of G3P to evolve programs using similarity measures and (ii) identifying the most suitable measure.

3.2 Program Similarity Assessment Approaches

Measuring similarity between source code is a fundamental activity in software engineering. It has multiple applications including duplicate/clone code location, plagiarism detection, code search, security bugs scanning, vulnerability/bugs identification [9] and code recommendation [12]. There have been proposed dozens of similarity detection algorithms since the last few decades, which can be classified into metrics, text, token, tree, and graph-based approaches based on the representation [23]. We selected four top-ranked similarity measures to evaluate their code synthesis proneness when used within G3P.

Cosine Similarity In addition to the standard code similarity detector, we also used cosine similarity to measure the similarity between two source codes. The following steps illustrate how we measured similarity using cosine similarity:

1. Prepossessing: The source program is tokenized by removing indentation information, including white spaces, brackets, newline characters, and other formatting symbols. Arithmetic operators and assignment symbols were kept as they can provide meaningful structural information.
2. Frequency Computation: For each token sequence of the source program, we compute the frequency of each token.
3. Cosine Similarity Computation: We calculate the similarity score with the cosine formula based on the token frequencies of each source code.

FuzzyWuzzy [5] is a string matching open-source python library based on difflib python library. It uses Levenshtein Distance to calculate the differences between sequences. The library contains different similarity functions including *TokenSortRatio* and *TokenSetRatio*. Ragkhitwetsagul et al. [23] surprisingly found that the string matching algorithm also works pretty well for measuring code similarity. *TokenSortRatio* function first tokenizes the string by removing punctuation, changing capitals to lowercase. After tokenization, it sorts the tokens alphabetically and then joins them together to calculate the matching score. While *TokenSetRatio* takes out the common tokens instead of sorting them.

CCFinder [15] is a token-based clone detection technique designed for large-scale source code. The technique detects the code clone with four steps:

1. Lexical Analysis: Generates token sequences from the input source code files by applying a lexical rule of a particular programming language. All source files are tokenized into a single sequence to detect the code clone with multiple files. White spaces and new line characters are removed to detect clone codes with different indentation rules but with the same meaning.

2. Transformation: The system applies transformation rules on token sequence to format the program into a regular structure, allowing it to identify code clones even in codes written with different expressions. Furthermore, all identifiers (e.g., variables, constants, and types) are replaced with special symbols to detect clones with different variable names and expressions.
3. Clone Matching: The suffix-tree matching algorithm is used to compute the matching of the code clones.
4. Formatting: Each clone pair is reported with line information in the source file. This step also contains reformatting from the token sequence.

CCFinder was designed for large-scale programs. Since the codes involved in our evaluation are elementary, the following modifications and simplifications are made to the original tool:

- Given that we are only interested in obtaining a similarity score between two pieces of code, we divide the length of the code clone by the maximum between the lengths of the source files:

$$Sim(x, y) = \frac{Len(Clone(x, y))}{Max(Len(x), Len(y))} \tag{1}$$

  where $Clone(x, y)$ denotes the longest code clone between x and y.
- The matching of code clones using the suffix-tree matching algorithm is simplified by getting the length of the longest common token sequence using a 2D matrix (each dimension representing the token sequence).
- The mapping information between the token sequence and the source code is removed since reporting the line number is no longer needed in our study.

SIM [8] is a software tool for measuring the structural similarity between two C programs to detect plagiarism in the assignment for lower-level computer science courses. It is also a token-based plagiarism detection tool that uses a string alignment technique to measure code similarity.

The approach comprises two main functions, generating tokens with formatting and calculating the similarity score using alignment. Each source file is first passed through a lexical analyzer to generate a token sequence. Like the common plagiarism detection system, the source code is formatted to standard tokens with white space removal representing arithmetic or logical operators, different symbols, constant or identifiers. After tokenization, the token sequence of the second program is divided into multiple sections, each representing a piece of the original program. These sections are then aligned with the token sequence of the first source code separately, which allows the tool to detect the similarity even the program is plagiarised by modifying the order of the functions.

4 Experiment Setup

4.1 General Program Synthesis Benchmark Suite

Helmuth and Spector [11, 10] introduced a set of program synthesis problems. These problems were based on coding problems that might be found in intro-

ductory computer science courses. Helmuth and Spector provide a textual description as well as two sets of input/output pairs for both training and testing during the program synthesis process. Table 1 describes the characteristics of each of the program synthesis problems considered in our evaluation.

Table 1. Description and characteristics of the selected program synthesis problems

Problem	Textual Description	# Input/Output Pair	
		Training	Testing
Number IO	Given an integer and a float, print their sum.	25	1000
Median	Given 3 integers, print their median.	100	1000
Smallest	Given 4 integers, print the smallest of them.	100	1000

4.2 Target Programs

To evolve our programs through G3P, we consider an oracle that computes the similarity measure of each evolved program to a target program code obtained using some text to code transformation. In this work, we wish to focus our analysis on the similarity measures and reduce the varying elements in our experiments (particularly in terms of ability to obtain a target program of good quality). Therefore, we consider the theoretical case where the oracle is aware of a code that solves the problem, but it is only reporting the similarity of the evolved code to it. While this assumption is not applicable in real life (i.e., if we know the correct code, then the problem is already solved without requiring any evolution), we hope to get enough insight from it on the capability of G3P to reproduce a program only based on a similarity measure.

Listings 1.1, 1.2, and 1.3 depict the target programs for the oracle assessment of program similarity for Number IO, Smallest, and Median respectively.

```
def numberIO(int1, float1):
    result = float(int1 + float1)
    return result
```

Listing 1.1. Target program for Number IO

```
def smallest(int1, int2, int2, int3):
    result = min(int1,min(int2,min(int3,int4)))
    return result
```

Listing 1.2. Target program for Smallest

```
def median(int1, int2, int3):
    if int1 > int2:
        if int1 < int3:
            median = int1
        elif int2 > int3:
            median = int2
        else:
            median = int3
    else:
        if int1 > int3:
```

```
            median = int1
        elif int2 < int3:
            median = int2
        else:
            median = int3
    return median
```

Listing 1.3. Target program for Median

4.3 G3P Parameter Settings

In our evaluation, we use the same parameter settings as those defined for G3P [7]. We only introduce a unique varying element (i.e., the fitness function based on a particular similarity measure). We repeat our evaluations 30 times for each problem and each G3P version (each version with its specific similarity measure). The general settings for the G3P system are shown in Table 2.

Table 2. Experiment parameter settings

Parameter	Setting
Runs	30
Generation	200
Population size	1000
Tournament size	7
Crossover probability	0.9
Mutation probability	0.05
Node limit	250
Variable per type	3
Max execution time	1 s

5 Results

In this section, we report and discuss the results of our evaluations. First, we start by comparing the performance of G3P using each of the similarity measures, then we compare them against the traditional error-rate based G3P.

5.1 Comparison of Similarity Measures

The result of the similarity-based G3P is reported in this subsection. The goal of this experiment is to assess G3P's ability to evolve a program solving a program synthesis problem (only known to an oracle) based on the similarity measure.

Figure 1 shows the number of runs (out of 30) where G3P manages to evolve the correct program for each of the program synthesis problems while using one of the four considered similarity measures as the fitness function.

We see from Figure 1 that G3P was able to able to evolve the correct programs for Number IO and Smallest at least once with Cosine, CCFinder and SIM. However, G3P did not manage to evolve any correct program for Median. G3P manages to find the correct program for Number IO in most runs (i.e., 27 out of 30) while using Cosine Similarity. However, the same program fails to find any correct program for Smallest. Similarly, G3P manages to find the correct program with Smallest in 17 runs out of 30 while using SIM, but the same program fails to find any correct program for Number IO. Alternatively G3P with CCFinder finds correct programs for both Number IO and Smallest, but in fewer runs. Overall, we could say that G3P has the potential to evolve programs for synthesis problems using similarity measures. However, there is no similarity measure that seems to work better than the rest and we need to consider combining similarity measures to increase the effectiveness of the approach.

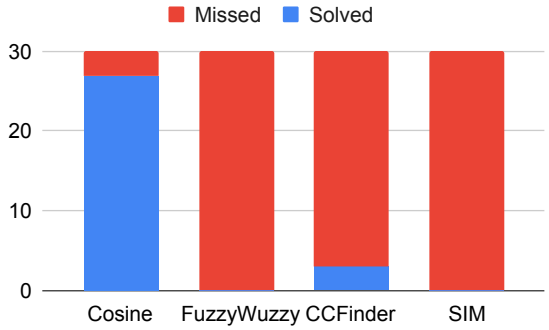
5.2 Comparison Against Error Rate-Based G3P

While we have seen that G3P has the capability to evolve correct programs for some program synthesis, we would like to assess how efficient is this process at evolving correct programs in comparison with the use of input/output error rate.

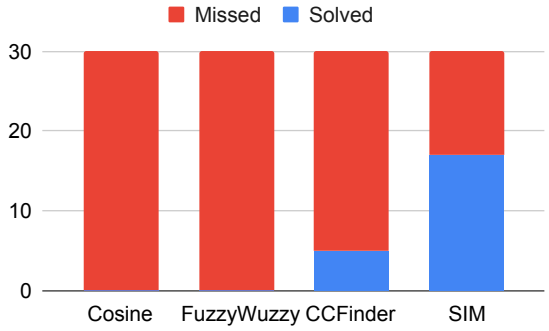
Figure 2 shows the performance of G3P with the input/output error rate to evolve correct programs for each of the considered programs over 30 distinct runs. We see that G3P with input/output error rate is capable to evolve correct problems to all the considered program synthesis problems. Furthermore, it is also capable of finding a correct program in more runs than the different G3P approaches using any similarity measure. Therefore, while we have seen that similarity measures seem promising to guide the G3P search for correct programs to program synthesis problems, they are not reaching the performance level of the traditional input/output error rate. This difference could be explained by the long amount of research that has been carried out to refine and optimise the G3P process with input/output error rate (particularly in terms of designing fit for purpose crossover and mutation operators).

6 Conclusion and Future Work

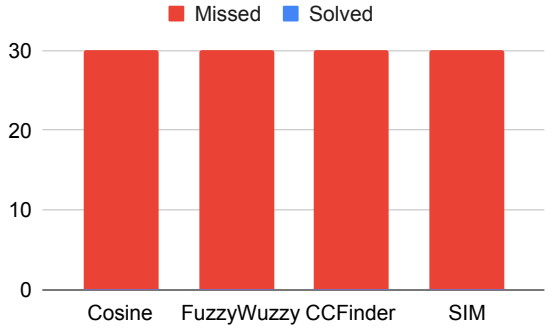
In this paper, we have assessed the potential for G3P to evolve programs based on their similarity to particular target codes of interest. The ultimate goal of this work is the ability to exploit textual descriptions of program synthesis problems as a guide to the evolutionary process in place of the traditional input/output error rate. We particularly assessed the capability of G3P to evolve correct programs using 4 similarity measures from various fields (i.e., Cosine, FuzzyWuzzy, CCFinder, and SIM). Our experimental evaluation on a well-known benchmark dataset has shown that G3P is able to evolve correct programs for some of the considered program synthesis problems. However, we have found that the performance of G3P using similarity measures is lower than G3P with the traditional input/output error rate. Out future work will focus on trying to improve the



(a) Number IO



(b) Smallest



(c) Median

Fig. 1. Number of iterations (out of 30) where G3P manages or fails to evolve the target program with each of the similarity measures.

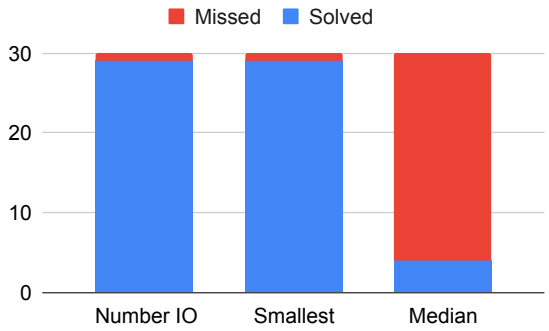


Fig. 2. Number of iterations (out of 30) where G3P with input/output error rate manages/fails to evolve the target program.

performance of G3P to take full advantage of such similarity measures alongside the traditional input/output error rate.

Acknowledgement: Supported, in part, by Science Foundation Ireland grant 13/RC/2094_P2.

References

1. Alexandru, C.V.: Guided code synthesis using deep neural networks. In: ACM SIGSOFT. pp. 1068–1070 (2016)
2. Brameier, M., Banzhaf, W., Banzhaf, W.: Linear genetic programming, vol. 1. Springer (2007)
3. Byrne, J., Cardiff, P., Brabazon, A., et al.: Evolving parametric aircraft models for design exploration and optimisation. Neurocomputing **142**, 39–47 (2014)
4. Ciritoglu, H.E., Saber, T., Buda, T.S., Murphy, J., Thorpe, C.: Towards a better replica management for hadoop distributed file system. In: IEEE BigData Congress (2018)
5. Cohen, A.: Fuzzywuzzy: Fuzzy string matching in python (2011)
6. Forstenlechner, S.: Program synthesis with grammars and semantics in genetic programming. Ph. D. dissertation (2019)
7. Forstenlechner, S., Fagan, D., Nicolau, M., O'Neill, M.: A grammar design pattern for arbitrary program synthesis problems in genetic programming. In: EuroGP. pp. 262–277 (2017)
8. Gitchell, D., Tran, N.: Sim: a utility for detecting similarity in computer programs. ACM SIGCSE Bulletin **31**(1), 266–270 (1999)
9. Hartmann, B., MacDougall, D., Brandt, J., Klemmer, S.R.: What would other programmers do: suggesting solutions to error messages. In: SIGCHI. pp. 1019–1028 (2010)
10. Helmuth, T., Spector, L.: Detailed problem descriptions for general program synthesis benchmark suite. University of Massachusetts Amherst (2015)

11. Helmuth, T., Spector, L.: General program synthesis benchmark suite. In: GECCO. pp. 1039–1046 (2015)
12. Holmes, R., Murphy, G.C.: Using structural context to recommend source code examples. In: ICSE. pp. 117–125 (2005)
13. Hu, X., Li, G., Xia, X., Lo, D., Jin, Z.: Deep code comment generation. In: IEEE/ACM ICPC. pp. 200–20010 (2018)
14. Jeon, J., Qiu, X., Foster, J.S., Solar-Lezama, A.: Jsketch: sketching for java. In: ESEC/FSE. pp. 934–937 (2015)
15. Kamiya, T., Kusumoto, S., Inoue, K.: Ccfinder: A multilinguistic token-based code clone detection system for large scale source code. IEEE Transactions on Software Engineering **28**(7), 654–670 (2002)
16. Koza, J.R., et al.: Genetic programming II, vol. 17. MIT press Cambridge, MA (1994)
17. Loughran, R., McDermott, J., O'Neill, M.: Tonality driven piano compositions with grammatical evolution. In: IEEE CEC. pp. 2168–2175 (2015)
18. Lynch, D., Saber, T., Kucera, S., Claussen, H., O'Neill, M.: Evolutionary learning of link allocation algorithms for 5g heterogeneous wireless communications networks. In: GECCO. pp. 1258–1265 (2019)
19. Miller, J.F., Harding, S.L.: Cartesian genetic programming. In: GECCO. pp. 2701–2726 (2008)
20. O'Neill, M., Nicolau, M., Agapitos, A.: Experiments in program synthesis with grammatical evolution: A focus on integer sorting. In: CEC. pp. 1504–1511 (2014)
21. O'Neill, M., Ryan, C.: Grammatical evolution: Evolutionary automatic programming in a arbitrary language, volume 4 of genetic programming (2003)
22. Pantridge, E., Spector, L.: Pyshgp: Pushgp in python. In: GECCO. pp. 1255–1262 (2017)
23. Ragkhitwetsagul, C., Krinke, J., Clark, D.: A comparison of code similarity analysers. Empirical Software Engineering **23**(4), 2464–2519 (2018)
24. Saber, T., Brevet, D., Botterweck, G., Ventresque, A.: Is seeding a good strategy in multi-objective feature selection when feature models evolve? IST (2017)
25. Saber, T., Brevet, D., Botterweck, G., Ventresque, A.: Milpibea: Algorithm for multi-objective features selection in (evolving) software product lines. In: EvoCOP. pp. 274–280 (2020)
26. Saber, T., Delavernhe, F., Papadakis, M., O'Neill, M., Ventresque, A.: A hybrid algorithm for multi-objective test case selection. In: IEEE CEC (2018)
27. Saber, T., Fagan, D., Lynch, D., Kucera, S., Claussen, H., O'Neill, M.: A hierarchical approach to grammar-guided genetic programming the case of scheduling in heterogeneous networks. In: TPNC. pp. 118–134 (2018)
28. Saber, T., Fagan, D., Lynch, D., Kucera, S., Claussen, H., O'Neill, M.: Multi-level grammar genetic programming for scheduling in heterogeneous networks. In: EuroGP. pp. 118–134 (2018)
29. Saber, T., Fagan, D., Lynch, D., Kucera, S., Claussen, H., O'Neill, M.: Hierarchical grammar-guided genetic programming techniques for scheduling in heterogeneous networks. In: CEC (2020)
30. Saber, T., Fagan, D., Lynch, D., Kucera, S., Claussen, H., O'Neill, M.: A multi-level grammar approach to grammar-guided genetic programming: the case of scheduling in heterogeneous networks. GPEM pp. 1–39 (2019)
31. Saber, T., Wang, S.: Evolving better rerouting surrogate travel costs with grammar-guided genetic programming. In: IEEE CEC. pp. 1–8 (2020)
32. Whigham, P.A.: Grammatical bias for evolutionary learning. (1997)

Audio Spectrogram Recognition with Neural Networks

Farah Medjahed[1], Philippe Devienne[2], Abou El Hassan Benyamina[1] and Mohammed Kamel Benhaoua[3]

(medjahed.farah@edu.univ-oran1.dz, Philippe.Devienne@univ-lille.fr, benyamina.hassen@univ-oran1.dz, k.benhaoua@univ-mascara.dz)

1. LAPECI Laboratory, Computer Science Department, Oran1 University, Ahmed Ben Bella, Oran, Algeria

2. Univ. Lille, CNRS, Centrale Lille, UMR 9189 – CRIStAL, F-59000 Lille, France

3. University of Mascara - Department of Computer Science, Algeria

Keyworks : Automatic Sound Classification, Convolutional Neural Network (CNN), Spiking Neural Networks (SNN), Natural Computing.

This work is in part supported by the European Prima S1\WATERMED 4.0 project for agriculture in semi-arid areas and the French ANR-21-CE04-0020 ULP COCHLEA for nano acoustic sensors and biodiversity.

I Introduction

Artificial neural networks interest many researchers for recent years, they are employed in a variety of sectors, including image processing, signal processing, handwriting recognition, and facial recognition. Sound recognition is a popular application of neural networks, which has sparked a lot of interest.

In this article, we will elaborate in the first part the audio classification, by presenting the different techniques of audio classification that are used. In the second part, we will present the Convolutional Neural Network and the Spiking Neural Network. Finally, we will conclude by citing the current work which is in progress related to audio classification using both these two types of neural networks.

II Audio classification

Audio classification is a hot topic that occupies the attention of researchers as it is used in different applications in different fields : telecommunication, robotics, marine and agricultural fields as well as other fields. Audio classification consists of automatically determining the nature of the sound signal. Several works have addressed this subject and different classification techniques have been used. Dhanalakshm and al. [1] have used the GMM (Gaussian Mixture Models) method as a method of classification which is 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 densities. Temko and al. [2] worked on SVMs (Support Vector Machines) which transform

data into a high-dimensional space, this convert the classification problem into a simpler one which can use linear discriminant functions. L.R. Rabiner and al.[3] used the HMM method in their work which is widely used classification models in speech recognition. 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. However, recent works on audio classification use Convolutional Neural Networks [[4],[5], [6], [7]...]. The advantage of this technique is that it can classify a large amount of data and increase the accuracy for image classification.

CNN (Convolutional Neural Network) is a feed-forward neural network which is able to classify images based on feature extraction. CNN is composed of three main layers : convolution and pooling layers, which perform feature extraction, and a fully connected layer, which maps the extracted features into the appropriate class. Several CNN models have been proposed each of them has a specific architecture but their purpose is the same is to increase the classification accuracy.[8]

SNN (Spiking Neural Network) [9] is the type of neural network that mimics the brain the most, it is more biologically plausible than other traditional neural networks 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 Work in progress

First, our work is to do the audio classification using convolutional neural networks by transforming the audio signal into spectrograms and use these later as an input of a Convolutional Neural Network to do the classification. Second, the same classification is made using Spiking Neural Networks based on the CNN-SNN conversion method named Spkeras which was proposed by Dengyu Wu et al. [5] which consists on detecting the activation layer in CNN to create SpikeActivation layer.

The dataset that we use include 20 animals and instruments sound [10]. This dataset is constructed using Animal Sound Data and Instrument Data. Each audio is split into multiple samples. These samples are divided into three parts: Train, Validation and Test sets which are disjoint. The train set is composed of 16,636 samples, the validation set contains 3,249 samples and the test set contains 3,727 samples. The code of SpKeras uses Tensorflow and Keras.

This work consists on training of the train set using a Convolutional Neural Network then, we use SpKeras to convert CNN into SNN after that we evaluate SNN model.

The objective is to prove the utility of using an Spiking Neural network than a Convolutional Neural Network, in addition, SNN are highly computationally and energy-efficient model and it can be exploited in a neuromorphic hardware device such as SpiNNaker. The following table presents the results of the experiments which shows the change in accuracy related to the number of epochs.

Number of epochs	CNN accuracy	SNN accuracy
50	80%	71%
100	77%	73%
200	79%	75%
300	80%	77%

IV Conclusion

In this brief study, we sought to provide an overview of audio classification and the most used classification techniques that exist and introduction of Convolutional and spiking neural network, as well as a glimpse into our ongoing work on audio classification using convolutional and spiking neural networks. As a complementary work, we will use a neuromorphic hardware in order to have an ultra-low power acoustic classifier and for better performance.

References

[1] P. Dhanalakshmi, Sengottayan Palanivel, and Vivekanandan Ramalingam. Classification of audio signals using aann and gmm. *Applied Soft Computing*, 11:716–723, 01 2011.

[2] Andrey Temko and Climent Nadeu. Classification of acoustic events using svm-based clustering schemes. *Pattern Recogn.*, 39(4):682–694, April 2006.

[3] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[4] Heart sound classification based on improved mfcc features and convolutional recurrent neural networks. *Neural Networks*, 130:22–32, 2020.

[5] Xiaowei Huang Dengyu Wu, Xinping Yi. A little energy goes a long way: Energy-efficient, accurate conversion from convolutional neural networks to spiking neural networks. 6 Mar 2021.

[6] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron Weiss, and Kevin Wilson. Cnn architectures for large-scale audio classification. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017.

[7] Zohaib Mushtaq and Shun-Feng Su. Efficient classification of environmental sounds through multiple features aggregation and data enhancement techniques for spectrogram images. *Symmetry*, 12(11), 2020.

[8] Shouheng Peng Zewen Li, Wenjie Yang and Fan Liu. A survey of convolutional neural networks: Analysis, applications, and prospects. 1 Apr 2020.

[9] Hammouda Elbez, Benhaoua Kamel, Philippe Devienne, and Pierre Boulet. Vs2n : Interactive dynamic visualization and analysis tool for spiking neural networks. pages 1–6, 06 2021.

[10] Cheng-Hao Tu Yi-Ming Chan. Sound20. `https://github.com/ivclab/Sound20`, 2018.

Data-Table Simulation-Optimization (DTSO): An efficient approach to optimize simulation models

Mohammad Dehghanimohammadabadi

Department of Mechanical and Industrial Engineering, Northeasntern Univeristy

m.dehghani@northeastern.edu

ABSTRACT

Simulation-optimization (SO) is instrumental to solve stochastic problems with complexity. Over the past half-century, SO methods have progressed theoretically and methodologically across different disciplines. The majority of commercial simulation packages - to some degree - offer a SO tool that allows decision-makers to conveniently determine an optimal or near-optimal system design. With the latest advancements in simulation techniques, such as data-driven models and Digital Twins, SO tools need a redesign to include new capabilities. This paper proposes a Data-Table Simulation-Optimization (DTSO) platform to narrow this gap. By considering data-tables as a decision variable (control), DTSO can systematically generate new tables, run experiments, and determine the best table entries to optimize the model. To implement DTSO, three software packages (MATLAB, Simio, and MS Excel) are integrated via a customized coded interface, called Simio-API. The applicability of this SO tool is tested in two experimental settings to evaluate its effectiveness and provide some insights for future extensions. The DTSO initial results are promising and should stimulate further research in academia and industry.

Keywords: Data-generated Modeling, Digital Twins, Simheuristics, Intelligent Simulation, Simio

1 Introduction

One of the major goals of using simulation modeling is to obtain the ideal configuration of a system. This is achievable by integrating the simulation model with an optimization module. In this approach, the optimizer explores the solution space in order to find the best input values to optimize the model design. And its performance measures. A general algebraic form of the simulation optimization (SO) problem can be defined as,

$$\min \quad \mathbb{E}_v[f(x,y,v)]$$

$$s.t.$$

$$\mathbb{E}_v[g(x,y,v)] \preccurlyeq 0$$
$$h(x,y) \preccurlyeq 0$$
$$x_l < x < x_u$$
$$y_l < y < y_u$$

$$x \in \mathbb{R}^n$$
$$y \in \mathbb{D}^m$$

where f is a real-valued objective function evaluation or the model output, which without loss of generality, is minimized. The performance measure of the model is calculated based on the expected output value with respect to input variables. Discrete and continuous input parameters of the simulation model are defined by x and y, respectively, with known lower and upper boundaries. Vector v is the realization of the associated random variables in the model. Function g is a real vector-valued function of stochastic constraints of the model to account for uncertainty. Constraint h represents a real vector-valued for deterministic constraints that are not affected by the uncertain parameters.

The above-mentioned formulation is general enough to represent all kinds of SO problems. Choosing the appropriate simulation and optimization design is crucial for practical applications and highly depends on the problem characteristics [1]. Commercial simulation software packages often use provably convergent algorithms proposed by the research community with metaheuristics to help their users deploy SO for their models [2]. In order to use these optimization algorithms embodied in simulation packages, a user needs to create the preliminary design of the optimization problem. This design includes simulation inputs (controls), the objective functions (responses), and constraints. The optimizer explores a series of simulation configurations by changing the model controls and tries to obtain the optimum or close-to-optimum set of input parameters. When the simulation model inputs are large or the simulation model is complex, this search becomes computationally expensive and resource demanding. Therefore, the optimizer needs to be efficient and adaptive enough to find the best input variable values among all possibilities without explicitly evaluating each possibility in simulation optimization [3].

In the existing SO tools provided by discrete event simulation (DES) packages, the model controls can be defined either as binary, integer, or real numbers. A user can easily define the type of inputs and their boundaries, and then such as lower-bound and upper-bounds as needed. SO tool will generate scenarios based on different combinations of inputs and will optimize the simulation configuration. However, to the best of the Author's knowledge, none of the existing SO tools are capable of handling *data-table inputs* and optimize the model based on table entries.

Nowadays, using data-table inputs is very essential in developing DES models with large amounts of data. Using data-tables makes simulation model development, execution, and experimentation efficient and easy to implement. Instead of defining an abundant number of parameters and variables, all the required data for the simulation modeling can be stored in a data-table format. The data-table values can be manually entered by the user or bound to a data structure framework such as MS Excel or databases. This feature provides a highly flexible approach to handling large data inputs for modeling needs. Data-tables could efficiently include any type of model information with a large number of data points. This could include entities' information (i.e. entity types, arrival times, processing time(s), sequence and priorities, etc.), resources' data (i.e. schedules, maintenance plans, locations on the layout, etc.), or even transportation networks. Based on the new advancement in DES commercial packages, these data can be accessed sequentially, randomly, directly, and even automatically [4]. This emphasizes the importance of using data-tables with the simulation model creation and experimental analysis.

Although enhancing a simulation model with data-table inputs simplifies the model development, there is not a trivial way to optimize the simulation models based on data-table inputs. Existing commercial SO tools are designed to include a limited number of *numerical* controls (binary, integer, or real) for optimization purposes, but are essentially incapable of using data-table inputs. This becomes more challenging when data-tables are non-numerical, e.g. categorical, Date/Time, etc. Therefore, this paper aims to remedy this lack and introduce *Data-Table Simulation-Optimization (DTSO)* platform with the following contributions:

- include the ability to run simulation model experiments based on data-table inputs
- propose a simulation-optimization structure to optimize data-table entries
- demonstrate its application in three case studies to reveal some insights into the present and future works

It needs to be noted that this article does not attempt to develop a new simulation-optimization algorithm. The novelty of this work is the platform that allows simulation users to optimize the model's performance by deploying simulation experiments with respect to data-table inputs with no data format restrictions. For instance, a user can optimize *"Patient's Arrival Table"* (Data/Time format) in a healthcare system, *"Product Mix Table"* (categorical format) in a manufacturing setting, or *"Destinations/Nodes Sequence Table"* (integer format) for a set of transporters/vehicles in a supply chain network. This platform benefits simulation model extensibility, scenario creation, and experiment repeatability. So, the original form of the SO problem can be modified as follows:

$$\min \quad \mathbb{E}_v[f(x, y, t, v)]$$

$$s.t.$$
$$\mathbb{E}_v[g(x, y, z, v)] \preccurlyeq 0$$
$$h(x, y) \preccurlyeq 0$$
$$x_l < x < x_u$$
$$y_l < y < y_u$$

$$x \in \mathbb{R}^n$$
$$y \in \mathbb{D}^m$$
$$t \in \mathbb{T}$$

where t represents the data-table inputs used towards simulation modeling. The values stored in the table could have any format and structure, which all of these formats are represented by $\mathbb{T}$.

The rest of this paper is organized as follows. Section 2 provides a brief introduction of SO and highlights the motivation of this work. In Section 3, the DTSO platform is explained in detail and its implementation aspects are discussed. To demonstrate the applicability of the proposed framework, two experiments are designed and analyzed in Section 4. This work is wrapped up in Section 5, and future work directions are presented in the end.

2 Background and Motivation

The desirability of seeking better solutions is the main driver for developing SO techniques. As a definition, SO is a *systematic* search process to find the best configuration of a stochastic system in order to optimize objective function(s). This search needs to be efficient to minimize the resources spent while maximizing the information obtained in a simulation experiment [3]. With a long and illustrious history, SO is arguably the ultimate aim of most simulationists [5]. With a huge advancement in both research and practice, SO is considered as one of the main streams of simulation studies and has received considerable attention from both simulation researchers and practitioners [6]. Many studies applied SO to address problems in healthcare [7]–[10], manufacturing [11]–[14], and supply chain [15]–[18].

Nowadays, SO is a vibrant field and various subdisciplines are evolved from different communities such as systems and control, statistics and design-of-experiments, math programming, and even computer science [5]. With the advancements in the SO literature, many of the simulation vendors provide some sort of automatic experimental generators or optimization tools. Based on a survey conducted by [19], 40 out of 55 software packages are featured with SO tools. Most of these SO tools such as OptQuest [20] and SimRunner [21] are designed based on Simheuristics structure. In Simheuristics, a Metaheuristic algorithm is coupled with the simulation environment [22] to perform an iterative search through parameter values to

obtain improved responses. These tools do not offer guarantees optimality, but the provided solutions are near-to-optimal and realistic.

Almost in all the existing SO tools, the decision variables are restricted to numerical parameters without considering other important elements of the model. For instance, in a healthcare system, one can easily evaluate a hospital model based on different numbers of resources' (i.e. physicians, nurses, beds, etc.) and determine the best numeric combination of these values. However, within the same model, it is not trivial to systematically change the layout of the hospital, schedule of physicians/nurses, or even individual patients' arrival time. To evaluate each of the above-mentioned scenarios, a tremendous amount of effort is required to manually make changes and customize the simulation model. The same concept relates to a manufacturing setting. For example, finding the best number of workers, transporters, servers, etc. is easily attainable using the existing commercial or non-commercial SO tools, while optimizing workers' schedule, transporters' network, and servers' location (layouts) is not a straightforward task.

The proposed DTSO in this article is a perfect alternative in which numerical and data-table inputs can be optimized simultaneously. This platform is general enough to include any type of table entries with multiple data formats. Therefore, DTSO is a promising SO framework and can introduce a significant opportunity to the simulation community to solve problems efficiently on a larger scale.

3 The proposed DTSO platform

To obtain a practical and ideal DTSO model, an integrated framework is developed using three modules, namely (i) optimization, (ii) simulation, and (iii) data-exchange. As illustrated in Figure 1, both simulation and optimization modules are bound to an external data source. In this iterative scheme, the optimization module provides a new solution and updates the data-table input(s) in each iteration, and then, the simulation module runs the model based on the new provided data settings. Using this new framework, users can easily build a simulation model with data-tables and optimize it in an efficient manner.

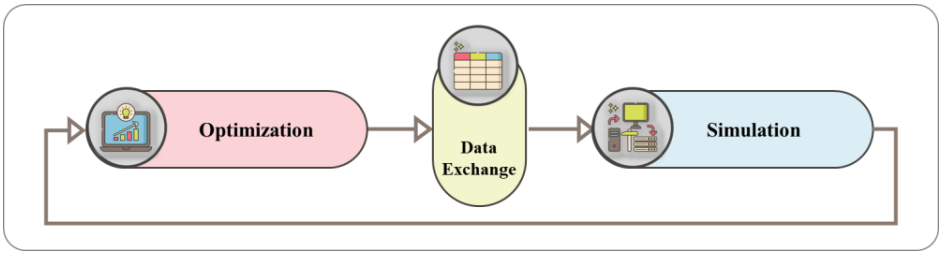


Figure 1: The proposed DTSO platform structure

The main goal is to make this framework general enough to be used in any simulation settings with different applications. To implement the proposed DTSO framework, three software packages are linked together. The Optimization Module is deployed in MATLAB, and the Simulation Module is designed in Simio. These two modules are linked together via MS Excel for data-table input exchange.

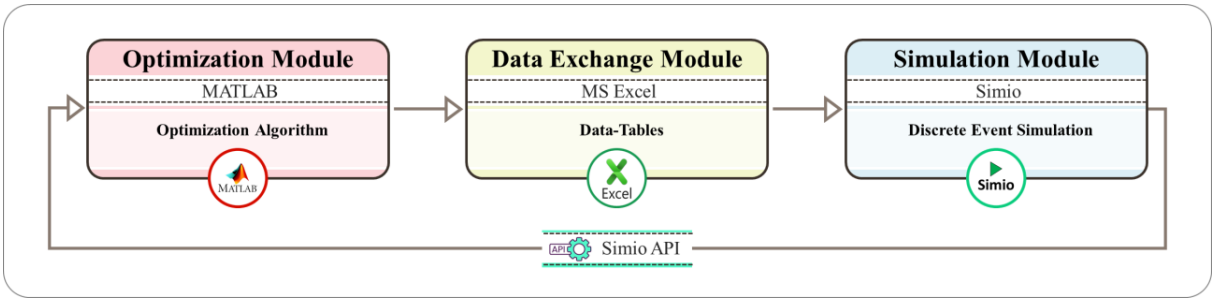


Figure 2: Implementation components of the DTSO framework using MATLAB, Simio, and Excel

MATLAB is a powerful tool and is adequate for addressing heavy computing needs such as optimization problems. MATLAB can be easily linked with external software for advanced calculations and its programmable environment allows users to design and customize their algorithms. It has enhanced optimization capabilities and allows its user to choose between the existing optimization libraries or their own developed algorithms. As a result, the applied optimization algorithm in this work is coded in MATLAB.

Simulation models often require large amounts of data to define different elements of the model such as entities, objects, networks, schedules, etc. Simio can represent data in simple tables and allows users to match the data schema for the manufacturing data (e.g. an ERP system) [23]. This simulation software is flexible enough to model complex systems with different operational needs. Another major benefit of Simio is its API capability which helps developers to extend Simio's access to external software packages. By taking advantage of this feature, a customized API is coded in C# to assist with the TDSO idea. This API connects MATLAB with Simio and provides a scheme to exchange data between the two. This enables users to connect Simio with other programming languages such as Python, R, or Julia.

The third component of this framework is MS Excel which is compatible with both MATLAB and Simio. This Data Exchange Module is the central piece of the framework and facilitates the data transfer between simulation and optimization packages. In Simio, data-tables can be bound to MS Excel and be accessed sequentially, randomly, directly, and even automatically. This important feature makes the development of DTSO feasible and effective intervention. A schematic illustration of DTSO is shown in Figure 2 and its pseudocode is provided in Figure 3.

Algorithm 1: Data-table Simulation Optimization (DTSO)

Data: Define simulation model parameters: $x \in R^n$, $y \in D^m$
Data: Define simulation model data-table inputs $t \in$T

initialization;
while *not the end of optimization algorithm* **do**
  Generate a new data-table solution (Optimization Module)
  Update data with the new data-table solution (Data-Exchange Module)
  Trigger the simulation model and run experiments (Simio API)
  for $r \leftarrow 1$ **to** $MaxReplications$ **do** (Simulation Module)
    Replicate the simulation model
  Calculate the expected value of simulation responses $E_v[f(x, y, t, v)]$
  Update objective function values
Result: Optimal/near-to-optimal data-table input

Figure 3: Pseudocode of Data-table Simulation Optimization (DTSO)

4 Experimental Analysis: DTSO for Job Scheduling and Sequencing

Using data-tables can significantly facilitate simulation modeling development, execution, and improvement. Data can be imported, exported, and even bound to external resources. While reading and writing disk files interactively during a run can reduce the execution speed, tables hold their data in memory and so execute very quickly [24]. Applying DTSO can harness the advantages of data-tables and place more emphasis on their use. To reveal some insights into the present and future works, this section demonstrates the applicability of DTSO in two experimental settings.

To maintain the focus of the paper on the DTSO advantages, the following experiments in this section introduce typical manufacturing settings with nominal operations. However, without loss of generality, DTSO can be utilized in any simulation models in Simio with different levels of complexities. Also, the applied optimization algorithm is Particle Swarm Optimization (PSO) which is manually coded in MATLAB to optimize data-table entries. Again, this does not limit the applicability of DTSO, and different users can leverage a variety of optimization tools and algorithms to solve their problems. These experiments are discussed as follows. To maintain paper's flow and its readability, details of PSO, its operations and pseudocode are provided in Appendix A.

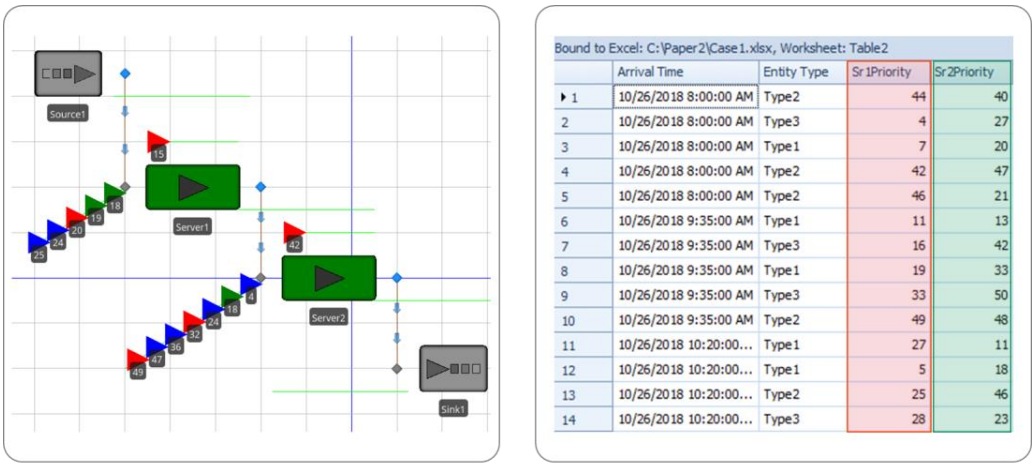
4.1 Experiment 1: Job Scheduling with DTSO

In this study considers a flow shop model where 50 jobs (entities) are processed sequentially by two (2) servers. This model includes three (3) types of jobs which randomly arrive in the system in batches of five (5). The model assumptions are:

- All machines are ready to be scheduled in time zero.
- Preemption of operations of each job is not allowed.
- Different job types have different distributions for processing time and due dates.
- Setup time is job dependent (setup time varies from one job type to another on each server.)
- Each machine can process only one operation at a time.

Figure 4-a depicts the simulation environment where the flow shop model is developed based on the assumptions provided above. This model has two objective functions: i) minimizing the average Time In

the System (TIS), and ii) minimizing the Total Tardiness Cost (TTC) of all jobs. The goal is to find the best prioritization of jobs in both servers in such a way that all objective functions are optimized. This model is simulated in Simio and a data-table is created to implement its operational logic.



(a) Simio model

Bound to Excel: C:\Paper2\Case1.xlsx, Worksheet: Table2

	Arrival Time	Entity Type	Sr1Priority	Sr2Priority
1	10/26/2018 8:00:00 AM	Type2	44	40
2	10/26/2018 8:00:00 AM	Type3	4	27
3	10/26/2018 8:00:00 AM	Type1	7	20
4	10/26/2018 8:00:00 AM	Type2	42	47
5	10/26/2018 8:00:00 AM	Type2	46	21
6	10/26/2018 9:35:00 AM	Type1	11	13
7	10/26/2018 9:35:00 AM	Type3	16	42
8	10/26/2018 9:35:00 AM	Type1	19	33
9	10/26/2018 9:35:00 AM	Type3	33	50
10	10/26/2018 9:35:00 AM	Type2	49	48
11	10/26/2018 10:20:00...	Type1	27	11
12	10/26/2018 10:20:00...	Type1	5	18
13	10/26/2018 10:20:00...	Type2	25	46
14	10/26/2018 10:20:00...	Type3	28	23

(b) Jobs' priority in Server1 and Server2

Figure 4: Job Scheduling simulation model and the data-table input structure

Figure 4-b depicts a snapshot of the data-table entry which stores entities' information such as arrival time, entity type, and priority numbers in Servers 1 and 2. The highlighted columns (*Sr1Priority* and *Sr2Priority*) indicate the priority of jobs on each server. This Optimization Module (i.e. PSO algorithm) in DTSO treats this table as a *decision variable* (control) and improves its entries sequentially until the desired solution is obtained. In every iteration, the optimization module in DTSO changes job priorities (values in Sr1Priority and Sr2Priority columns). Then, the simulation is triggered to simulate the model based on new data-table entries and runs replications to calculate the expected value of objective functions. These results are transferred to the Optimization Module to generate new solutions. This cycle repeats until the stopping criteria (which are usually set by the user) are met.

To analyze the performance of DTSO, its results are compared with two heuristic methods available in the literature for solving flow shop scheduling.

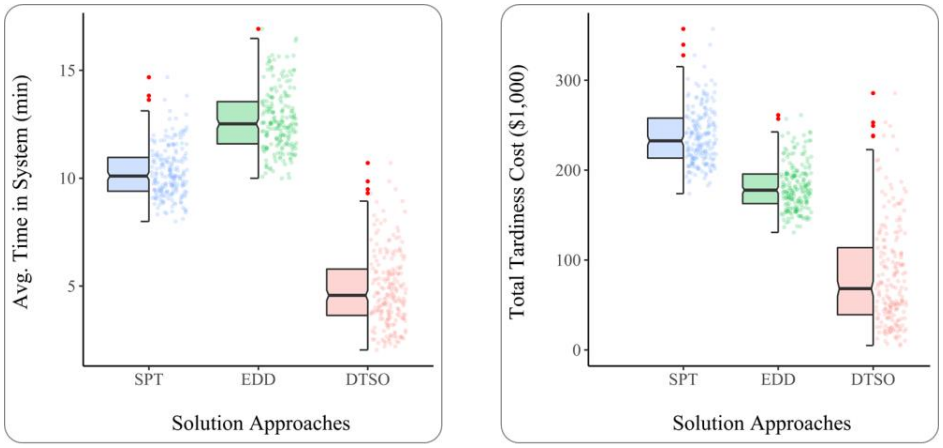
- **Heuristic 1- SPT:** Shortest Processing Time or SPT has shown superior performance for job scheduling in many research investigations [25]. By ranking jobs based on the ascending order of their processing times, SPT minimizes the total completion time.
- **Heuristic 2- EDD:** The second heuristic is EDD (Earliest Due Date) which arranges job orders to minimize the total tardiness cost of jobs [26].

The applied PSO algorithm in DTSO uses a weighted average of both objective functions to solve the problem (Equation 1).

$$\min \left[w_1 \times \mathbb{E}_v[TIS(x,y,t,v)] + w_1 \times \mathbb{E}_v[TTC(x,y,t,v)]\right] \quad (1)$$

The performance of the calculated optimal solution provided by DTSO is compared with heuristic results provided by SPT and EDD. Each of these solutions is replicated 200 times and the ultimate results are plotted in Figure 5. These results suggest the superiority of DTSO over SPT and EDD in terms of both

objective functions (time in the system and tardiness cost). With a smooth and straightforward implementation, DTSO could efficiently improve the prioritization of jobs and provide competitive results.



(a) Average time in system (b) Total tardiness cost

Figure 5: Experimental results of the flow shop scheduling study

Insight 1: The applicability of DTSO can be easily extended to a flow shop model with more servers. To include a new server in the simulation model, a new column needs to be added to the data-table to represent jobs' priority on that server (i.e. Sr3Priorirty). In the optimization algorithm, the size of decision variables $(nVar)$ directly depends on the number of jobs (n) and the number of servers (m) in the model $(nVar = n \times m)$. So, adding a new server or more jobs just needs a slight change in the optimization algorithm and updating $nVar$ parameter.

4.2 Experiment 2: Job Sequencing with DTSO

The second experiments demonstrate the applicability of DTSO in solving a job sequencing problem in a multi-stage flow shop system (known as assembly flow shop). In this case, 90 jobs are released to the floor with 3 stages and 5 servers in each. The model assumptions are:

- Each machine can process only one operation at a time.
- Assembly or post-processing stages begin readily after all previous stage operations are completed.
- All machines are ready to be scheduled in time zero.
- Preemption of operations of each job is not allowed.
- Setup time is zero.

Figure 6 depicts the simulation environment where this assembly shop is developed. Objective functions are to minimize both i) average Time In the System (TIS), and ii) Total Tardiness Cost (TTC) of all jobs. In all stages, the processing time of jobs is set to $normal(20,2)$ min, and due dates are uniformly distributed using $uniform(50,250)$ min. There are five (5) homogenous servers in each stage and each job has to follow a sequence of tasks to complete the assembly.

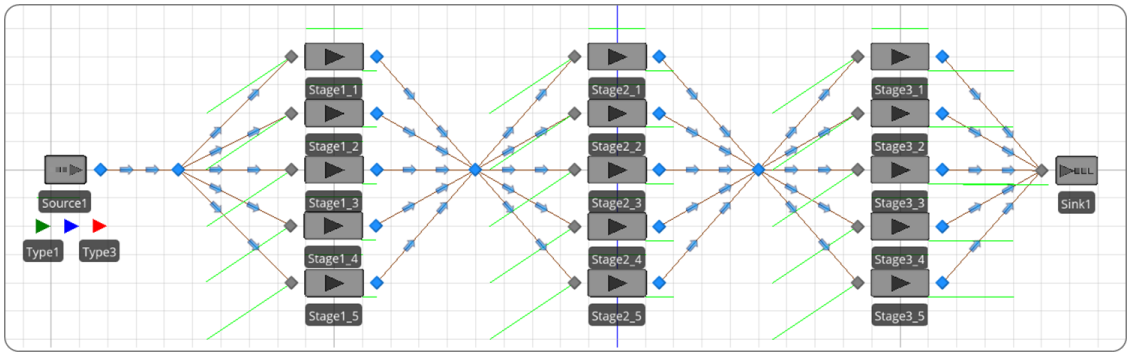


Figure 6: Multi-stage flow shop system developed in Simio

The goal is to find the best sequence of each job in each stage. In other words, the ideal solution should determine which server is selected in each stage to process a given job. To deploy this, a data-table is utilized in the simulation model to set up the sequencing logic. Figure 7 shows a screenshot of this table where each row represents a given job arrival time and its sequence in different stages. Three highlighted columns (Stage1Sr, Stage2Sr, and Stage3Sr) indicate the server IDs (1 to 5) that each job has to go through sequentially from stage 1 to 3 before leaving the system.

Bound to Excel: Case2.xlsx, Worksheet: Table2
Data has not been imported or is of an unknown age

	Arrival Time	Stage1Sr	Stage2Sr	Stage3Sr
1	10/26/2018 8:00:00 AM	3	3	5
2	10/26/2018 8:00:00 AM	1	3	2
3	10/26/2018 8:00:00 AM	2	3	3
4	10/26/2018 8:00:46 AM	4	5	5
⋮	⋮	⋮	⋮	⋮
89	10/26/2018 1:31:27 PM	3	5	3
90	10/26/2018 1:32:37 PM	1	4	4

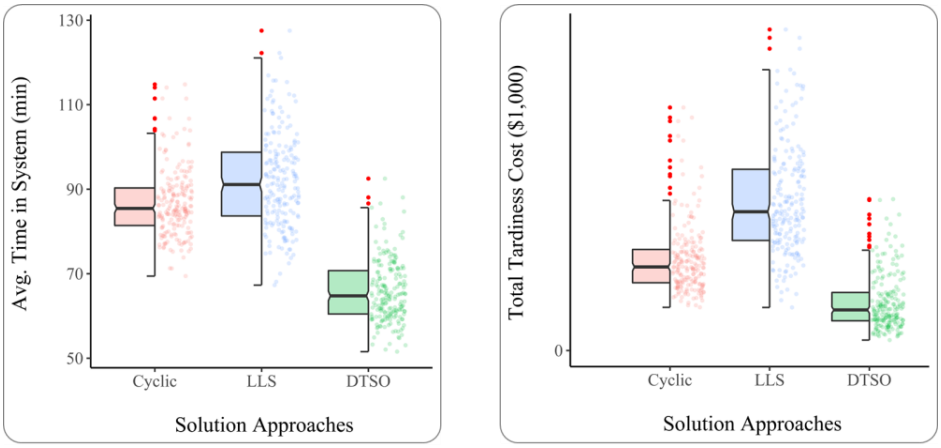
Figure 7: Data-table input structure for the job sequencing problem

To optimize this problem, the DTSO platform explores different combinations of sequences for all jobs and provides a solution that minimizes objective functions. Two heuristics rules are considered to evaluate the quality of DTSO results. These heuristics are:

- **Heuristic 1- Cyclic:** In each stage, this rule selects servers cyclically to carry out new jobs.
- **Heuristic 2- LLS:** This routing rule, selects a server with the lowest service load (LLS) upon a new job arrival to the stage.

The obtained solutions of these three approaches are simulated with 200 replications to estimate the expected value of objective functions, TIS, and TTC. The jitter-boxplots of these results are presented in Figure 8, where DTSO performance is adequately better than heuristics. The optimization algorithm in DTSO could find a solution with higher quality and less variability. The significance of this difference is

tested using one-way ANOVA for TIS and TTC objectives (Table 1 and Table 2). The p-value of both tests is low ($p < 0.001$), which appears that DTSO's superiority is statistically significant.



(a) Average time in system (b) Total tardiness cost

Figure 8: Experimental results of the multi-stage flow shop job sequencing study

Table 1: One-way ANOVA results for the average time in the system results

Source of Variation	SS	df	MS	F	P-value	F critical
Between Groups	76580.58	2	38290.29	474.10	5.2E-124	3.010815
Within Groups	48215.91	597	80.76			
Total	124796.5	599				

Table 2: One-way ANOVA results for the total tardiness cost results

Source of Variation	SS	df	MS	F	P-value	F critical
Between Groups	2.63E+08	2	1.32E+08	309.3843	6.31E-93	3.010815
Within Groups	2.54E+08	597	425708.2			
Total	5.18E+08	599				

Insight 2: In this experiment, DTSO solved the problem of sequencing for 90 jobs in 3 stages ($nVar = 270$). To solve this problem, OptQuest requires at least 180 controls (properties) to find the solution; whereas, DTSO takes the data-table as a decision variable and evolves its values until the desired solution is achieved.

5 Conclusion and future works

For more than fifty years, simulation has been extensively applied by researchers and developers. Traditionally, it is appealing to equip simulation with optimization to tackle stochasticity and complexity of problems. In spite of tremendous advancements in SO techniques, designing well-established models are still demanding for today's standards [1]. In addition, simulation software packages fail to incorporate human decision-making analysis or highly computational support tools [27].

This article introduced an innovative interaction between simulation and optimization to carry out the decision-making process based on table-table inputs. Unlike the existing commercial software packages,

the proposed DTSO framework can efficiently solve problems with a large set of parameters and data-tables. By developing an application programming interface, called Simio-API, this framework connects three modules together, simulation, optimization, and data-exchange. This integration has the ability to run simulation experiments with multiple data-table settings, evolve their values, and achieve satisfactory results.

The usefulness of the proposed framework is demonstrated in two experimental scenarios, 1) job scheduling in a flow shop system, and 2) Job sequencing in a multi-stage flow shop system. These experiments demonstrated DTSO's applicability and efficiency. More importantly, some insights provided to show how the new model can be implemented and extended to new works.

Introducing DTSO offers new opportunities to the community and paves a new avenue of research for theory and practice. Nowadays, many simulation software developers value data-driven models. With the emergence of new simulation techniques such as *data-generated* modeling and *Digital Twins*, the usefulness of DTSO can become more obvious. The concept of data-generated modeling is on the basis of creating a simulation model automatically using data-tables. This populates a complete simulation model from scratch by adding objects to the environment and mapping them to the tables. These tables can include all of the modeling needs such as resources' information, entities, networks, transports, schedules, tasks, etc. Lately, Simio announced a newly added feature to its software to build data-generated models [28]. By leveraging DTSO, one can optimize different components of the model systematically without the need for manual changes. For instance, optimizing system layouts (i.e. hospitals, manufacturing systems, etc.) are traditionally limited to a few scenarios suggested by layout designers. By taking advantage of data-generated modeling in Simio and DTSO, one can easily change the layout, make simulation models instantly, and experiment with an abundance of layouts. As shown in Figure 9-a, this requires to define object coordinates as decision variables (XLocation and ZLocation columns) for DTSO and let it solve the problem. Another example could be optimizing manufacturing orders where orders' *release date* and *priority* need to be optimized (Figure 9-b).

Resources

	Resource Name	XLocation (Meters)	ZLocation (Meters)	Object Type
1	Cut	-8	5	SchedTransferNode
2	Cut1	-4	2	SchedServer
3	Cut2	-4	7	SchedServer
4	Shape	5	-1	SchedTransferNode
5	Shape1	1	-2	SchedServer
6	Shape2	9	-2	SchedServer
7	Weld	5	11	SchedTransferNode
8	Weld1	1	12	SchedServer
9	Weld2	9	12	SchedServer
10	Finish	12	5	SchedTransferNode

(a) Resources data-table

Manufacturing Orders

	Order Id	Material Name	Release Date	Due Date	Order Status	Priority
1	Order01	FinishedGoodA	12/1/2019 12:00:00 AM	12/10/2019 4:00:00 PM	WIP	1
2	Order02	FinishedGoodA	12/1/2019 12:00:00 AM	12/5/2019 4:00:00 PM	New	1
3	Order03	FinishedGoodB	12/1/2019 12:00:00 AM	12/7/2019 4:00:00 PM	WIP	1
4	Order04	FinishedGoodB	12/1/2019 12:00:00 AM	12/5/2019 4:00:00 PM	WIP	1
5	Order05	FinishedGoodC	12/1/2019 12:00:00 AM	12/6/2019 4:00:00 PM	WIP	1
6	Order06	FinishedGoodC	12/1/2019 12:00:00 AM	12/3/2019 4:00:00 PM	New	1
7	Order07	FinishedGoodC	12/1/2019 12:00:00 AM	12/3/2019 4:00:00 PM	New	1
8	Order08	FinishedGoodA	12/1/2019 12:00:00 AM	12/12/2019 11:00:00 PM	New	1
9	Order09	FinishedGoodA	12/1/2019 12:00:00 AM	12/14/2019 9:00:00 PM	New	1
10	Order10	FinishedGoodB	12/1/2019 12:00:00 AM	12/13/2019 4:00:00 PM	New	1

(b) Manufacturing orders data-table

Figure 9: Examples of data-generated modeling in Simio

Another future research direction is to explore other optimization techniques in the model. To keep the focus of the paper on the platform, just one algorithm (PSO) is used in experiments. However, there are plenty of SO techniques that can be borrowed and tested within DTSO design. Multiple metaheuristic algorithms can be utilized to develop data-table Simheuristics and solve the problems. By increasing the size of data-tables (and decision variables), neural networks can be added to the optimizer and make the model computationally efficient by approximating objective functions.

References:

[1] G. Figueira and B. Almada-Lobo, "Hybrid simulation–optimization methods: A taxonomy and discussion," *Simul. Model. Pract. Theory*, vol. 46, pp. 118–134, Aug. 2014, doi: 10.1016/j.simpat.2014.03.007.

[2] S. Amaran, N. V. Sahinidis, B. Sharda, and S. J. Bury, "Simulation optimization: a review of algorithms and applications," *4OR*, vol. 12, no. 4, pp. 301–333, 2014.

[3] Y. Carson and A. Maria, "Simulation optimization: methods and applications," 1997, pp. 118–126.

[4] J. S. Smith, D. T. Sturrock, and W. D. Kelton, *Simio and Simulation: Modeling, Analysis, Applications: 4th Edition - Economy*, 4 edition. CreateSpace Independent Publishing Platform, 2017.

[5] M. C. Fu and S. G. Henderson, "History of seeking better solutions, AKA simulation optimization," in *2017 Winter Simulation Conference (WSC)*, 2017, pp. 131–157.

[6] S. Ólafsson and J. Kim, "Simulation optimization," in *Proceedings of the winter simulation conference*, 2002, vol. 1, pp. 79–84.

[7] A. Azadeh, M. P. Ahvazi, S. M. Haghighii, and A. Keramati, "Simulation optimization of an emergency department by modeling human errors," *Simul. Model. Pract. Theory*, vol. 67, pp. 117–136, 2016.

[8] M. Rezaeiahari and M. T. Khasawneh, "Simulation optimization approach for patient scheduling at destination medical centers," *Expert Syst. Appl.*, vol. 140, p. 112881, 2020.

[9] A. Gupta, G. W. Evans, and S. S. Heragu, "Simulation and optimization modeling for drive-through mass vaccination–A generalized approach," *Simul. Model. Pract. Theory*, vol. 37, pp. 99–106, 2013.

[10] M. Dehghanimohammadabadi and N. Kabadayi, "A COMBINED MULTI-OBJECTIVE SIMULATION-OPTIMIZATION AND AHP APPROACH: A HEALTHCARE CASE STUDY," *Int. J. Anal. Hierarchy Process*, vol. 12, no. 1, 2020.

[11] R. Aiassi, S. M. Sajadi, S. M. H. Molana, and A. Z. Babgohari, "Designing a stochastic multi-objective simulation-based optimization model for sales and operations planning in built-to-order environment with uncertain distant outsourcing," *Simul. Model. Pract. Theory*, p. 102103, 2020.

[12] J. Seif, M. Dehghanimohammadabadi, and A. J. Yu, "Integrated preventive maintenance and flow shop scheduling under uncertainty," *Flex. Serv. Manuf. J.*, pp. 1–36, 2019.

[13] F. Amiri, B. Shirazi, and A. Tajdin, "Multi-objective simulation optimization for uncertain resource assignment and job sequence in automated flexible job shop," *Appl. Soft Comput.*, vol. 75, pp. 190–202, 2019.

[14] N. Azouz and H. Pierreval, "Adaptive smart card-based pull control systems in context-aware manufacturing systems: Training a neural network through multi-objective simulation optimization," *Appl. Soft Comput.*, vol. 75, pp. 46–57, 2019.

[15] D. Drenovac, M. Vidović, and N. Bjelić, "Optimization And Simulation Approach To Optimal Scheduling Of Deteriorating Goods Collection Vehicles Respecting Stochastic Service And Transport Times," *Simul. Model. Pract. Theory*, p. 102097, 2020.

[16] M. D. Phung and Q. P. Ha, "Motion-encoded particle swarm optimization for moving target search using UAVs," *Appl. Soft Comput.*, vol. 97, p. 106705, 2020.

[17] A. A. Vieira, L. Dias, M. Y. Santos, G. A. Pereira, and J. Oliveira, "Are Simulation Tools Ready For Big Data? Computational Experiments with Supply Chain Models Developed in Simio," *Procedia Manuf.*, vol. 42, pp. 125–131, 2020.

[18] F. Goodarzian, H. Hosseini-Nasab, J. Muñuzuri, and M.-B. Fakhrzad, "A multi-objective pharmaceutical supply chain network based on a robust fuzzy model: A comparison of meta-heuristics," *Appl. Soft Comput.*, p. 106331, 2020.

[19] J. J. Swain, "Simulated worlds," *ORMS Today*, vol. 42, no. 5, pp. 36–49, 2015.

[20] M. Laguna, "Optimization of complex systems with OptQuest," *White Pap. OptTek Syst. Inc*, 1997.

[21] D. L. Heflin and C. R. Harrell, “Simulation modeling and optimization using ProModel,” in *1998 Winter Simulation Conference. Proceedings (Cat. No. 98CH36274)*, 1998, vol. 1, pp. 191–197.
[22] A. A. Juan, J. Faulin, S. E. Grasman, M. Rabe, and G. Figueira, “A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems,” *Oper. Res. Perspect.*, vol. 2, pp. 62–72, 2015.
[23] C. D. Pegden, “Introduction to SIMIO,” in *2008 Winter Simulation Conference*, 2008, pp. 229–235.
[24] D. T. Sturrock, “Traditional Simulation Applications in Industry 4.0,” in *Simulation for Industry 4.0*, Springer, 2019, pp. 39–54.
[25] G. Jules, M. Saadat, and S. Saeidlou, “Holonic goal-driven scheduling model for manufacturing networks,” in *2013 IEEE International Conference on Systems, Man, and Cybernetics*, 2013, pp. 1235–1240.
[26] M. Dehghanimohammadabadi, “Iterative Optimization-based Simulation (IOS) with Predictable and Unpredictable Trigger Events in Simulated Time,” WESTERN NEW ENGLAND UNIVERSITY, 2016.
[27] M. Dehghanimohammadabadi and T. K. Keyser, “Intelligent simulation: Integration of SIMIO and MATLAB to deploy decision support systems to simulation environment,” *Simul. Model. Pract. Theory*, vol. 71, pp. 45–60, 2017.
[28] D. T. Sturrock, “Using Commercial Software to Create a Digital Twin,” in *Simulation for Industry 4.0*, Springer, 2019, pp. 191–210.
[29] J. Kennedy and R. Eberhart, “Particle swarm optimization (PSO),” in *Proc. IEEE International Conference on Neural Networks, Perth, Australia*, 1995, pp. 1942–1948.
[30] M. A. Shaheen, H. M. Hasanien, and A. Alkuhayli, “A novel hybrid GWO-PSO optimization technique for optimal reactive power dispatch problem solution,” *Ain Shams Eng. J.*, 2020.
[31] M. Usman, W. Pang, and G. M. Coghill, “Inferring structure and parameters of dynamic system models simultaneously using swarm intelligence approaches,” *Memetic Comput.*, vol. 12, no. 3, pp. 267–282, 2020.
[32] K. Park, “A Heuristic Simulation–Optimization Approach to Information Sharing in Supply Chains,” *Symmetry*, vol. 12, no. 8, p. 1319, 2020.
[33] J. F. Marchesi, S. Hamacher, and J. L. Fleck, “A stochastic programming approach to the physician staffing and scheduling problem,” *Comput. Ind. Eng.*, vol. 142, p. 106281, 2020.
[34] R. H. Caldeira and A. Gnanavelbabu, “A simheuristic approach for the flexible job shop scheduling problem with stochastic processing times,” *SIMULATION*, p. 0037549720968891, 2020.

Appendix A

Particle Swarm Optimization or PSO is a population-based Metaheuristic algorithm developed by Kennedy and Eberhart in 1995 [29]. PSO is a swarm-based algorithm and by moving particles in a specific exploration field [30]. Due to its effective balancing of exploration and exploitation [31], PSO has been widely used in the development of Simheuristic models and solving SO problems. Recent examples include using PSO to deal with stochastic models in the supply chain management [32], healthcare systems [33], and manufacturing [34]. The general pseudocode of PSO is shown in Figure 10.

Algorithm 2 Pseudo-code of Particle Swarm Optimization (PSO)

// Initialization ▷ Generate particles
for $i = 1 : N_s$ **do**
    Initialize $s_i(t = 0)$
    Initialize $v_i(t = 0)$
    $P_i^{best} \leftarrow s_i$
end for

// PSO loop
$G^{best} = 0$
for $t = 1 : Max_{it}$ **do**
    for $i = 1 : N_s$ **do**
        $v_i(t+1) = wv_i(t) + c_1 r_1[P_i^{best} - s_i(t)] + c_2 r_2[G^{best} - s_i(t)]$
        $s_i(t+1) \leftarrow s_i(t) + v_i(t+1)$
        Evaluate $fitness s_i(t+1)$
        if $fitness(P_i^{best}) < fitness(s_i(t+1))$ **then**
            $P_i^{best} \leftarrow s_i(t+1)$ ▷ Update Personal best
            if $fitness(G^{best}) < fitness(P_i^{best})$ **then**
                $G^{best} \leftarrow P_i^{best}$ ▷ Update Global best
            end if
        end if
    end for
    $t \leftarrow t + 1$
end for

Notations:

N_s : Swarm size, $i = 1, 2, ..., N_s$: Particles index

s_i : Solution (particle), v_i : Velocity, w : Inertia weight

P_i^{best} : Personal best solution, c_1 : Personal learning factor

G^{best} Global best, c_2 : Global learning factor

r_1, r_2 : Random numbers $\sim u(0, 1)$

MAX_{it} : Max number of iterations, $t : 0, 1, 2, ..., MAX_{it}$: Iteration index

Figure 10: Pseudocode of Particle Swarm Optimization (PSO)

Interference Classification in WiFi Environments using commercial Access Points

J. Candelario[1], F. Salvago[1], P. Aguilera[2], S. Yanes[1], F. Peralta[1], D. G. Reina[1] and S. Toral[1]

1. E. T. S. Ingeniería, Universidad de Sevilla
storal@us.es
2. Galgus, Camas, Sevilla. España
pablo.aguilera@galgus.net

Keywords : Interference Detection, Classification, Machine Learning, 1D CNN.

1 Introduction

There are currently numerous commercial devices working with wireless protocols operating in the 2.4 GHz environment. In the coming years, the number of wireless devices is even expected to increase considerably due to the rise of IoT devices, mostly equipped with wireless communications. This saturation of the wireless space poses a serious challenge to the reliability of communications due to the problem of interference between technologies. However, this problem can be mitigated if the source of the interference is known. In the case of WiFi environments, access points can apply strategies such as changing the channel or modifying the channel bandwidth. It is also possible, once the sources of interference are known, to modify the location or the number of access points to achieve better performance.

Nowadays, there is a multitude of equipment available to perform this type of task, such as spectrum analysers, interference classifiers, etc. But all of them are specific purpose devices with a high cost. The objective of this work is to incorporate into commercial access points a specific software to scan the medium and, once the data is collected, process them by using classifiers so that to interferences can be categorized according to their origin.

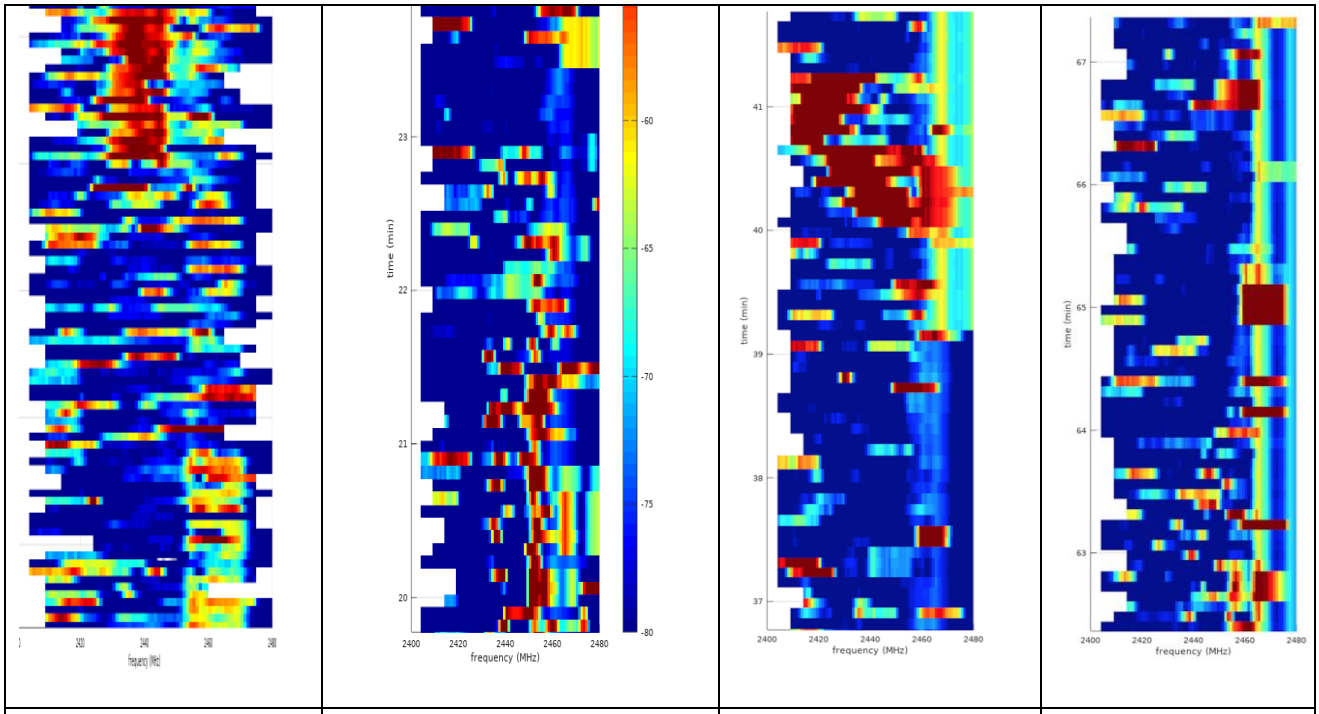
2 Related work

WiFi access points (APs) use various mechanisms in the 802.11 MAC to detect and avoid interference from other Wifi sources but usually they ignore non-Wifi sources. However, given the continued growth of non-Wifi activity in this shared unlicensed band and the resulting impact on Wifi performance, hardware vendors and network administrators are increasingly exploring techniques to better detect non-Wifi sources of interference. In [1], authors use hidden repeating patterns in the signals to build a unique signature to detect signal types and their spectral and spatial parameters. Monitoring packets on heterogeneous wireless networks to efficiently map signals to protocols has also been proposed as a possible solution [2]. Channel State Information (CSI) [3] and RSSI [4] have also been proposed as the basic features to perform the categorization of interferences in conjunction with several machine learning classifiers such as kNN or SVM.

Unlike these previous works that focus on certain signal characteristics or on the packets of the different protocols, this work proposes to use the raw data in combination with convolutional neural networks. The main advantage of using neural networks is that it is the network itself that searches for the relevant signal features that optimize the final classification.

3 Methodology and Results

Qualcomm/Atheros 802.11xx chipset is part of many APs and can be used to scan in passive mode and report spectral samples using Discrete Fourier Transform (DCT) tool to compute Fast Fourier transform (FFT) in baseband, which could be used to detect wireless signals with inclusion of non-802.11 signals. Once the dataset is stored, the maximum and the mean power values for each frequency are collected. The interference detection problem is then formulated as a 5-class classification problem in a WiFi scenario with the following classes: no interference (clean environment), WiFi interference, Bluetooth interference, Jammer interference and microwave interference. It is assumed that only one interference is present at the same time. The dataset was collected using commercial APs that continuously scan the environment under the presence of each type of interference. As a result, each spectral scan can be represented as a time/frequency diagram, as illustrated in Figure 1. The horizontal axis represents the 100 frequencies between 2401 and 2500 and the vertical axis is time.



(a) WiFi	(b) Bluetooth	(c) Jammer	(d) Microwave

Figure 1. Time/Frequency diagrams for the four types of interferences

The set of classifiers detailed in Table 1 will be considered for the interference detection. The continuous scan is split in 500 spectral scans of 10 timesteps for each class, which leads to a total dataset of 2500 samples. 2000 out of 2500 samples were randomly selected for training, 150 for validation and 150 for testing. The training dataset was used to train the parameters of the set of classifiers detailed in Table 1. For each algorithm, several hyperparameters also detailed in Table 1 are optimized using the validation dataset. The final accuracy of each algorithm is reported with the test dataset.

The first proposed classifier is a 1D convolutional neural network (CNN1D), which applies the convolution operation along the time dimension. The rest of classifiers are traditional machine learning classifiers. In this case, the mean value along the 10 timesteps is taken as the input features for each algorithm. The second column of Table 1 provides the optimum values of the most important hyperparameters related to each algorithm. The main difference between 1D CNN and the rest of machine learning classifiers is that the former considers sequences of 10 timesteps so that filters can detect temporal dependencies along time. This possibility is not considered by machine learning classifiers.

Table 1. List of classifiers, hyperparameters and final accuracy

Classifier	Hyperparameters (Optimum value)	Result (test accuracy)
1D convolutional neural network (CNN 1D)	Number of filters (16) Kernel size (4) Dropout layers prob. (0.2)	0.82
k-NN	k value (1)	0.58
Logistic regression (LG)	NA	0.46
Random Forest (RF)	Number of estimators (100) Max depth (30)	0.72
Gradient Boosting (GB)	Number of estimators (100) Max depth (10)	0.73
Support Vector Machines (SVM)	Regularization factor (0.3)	0.49
Multilayer perceptron (MLP)	Number of neurons per layer (16,16,8)	0.57

The comparative analysis presented in the last column of Table 1 shows that the best classifier is the one implemented by means of CNN1D with an accuracy value of 80%, followed by the RF and GB classifiers.

4 Conclusions

In this paper, we propose using a CNN1D to process the time/frequency diagram collected by commercial APs in a WiFi environment. Its main advantage over traditional machine learning classifiers is its ability to find time patterns over the spectral data of the environment, improving the results provided by alternative classifiers.

Acknowledgments

This work has been partially funded by the Spanish "Ministerio de Ciencia, Innovación y Universidades" under the grant RTI2018-098964-B-I00 funded by MCIN/AEI/ 10.13039/501100011033, and by the regional government Junta de Andalucía under the Projects "Desarrollo de nuevas tecnologías WiFi inteligentes en entornos móviles y con alta densidad de usuarios PAIDI 2020 P18-TP-1520.", "Despliegue Inteligente de una red de Vehículos Acuáticos no Tripulados para la monitorización de Recursos Hídricos US-1257508"' and "Despliegue y Control de una Red Inteligente de Vehículos Autónomos Acuáticos para la Monitorización de Recursos Hídricos Andaluces PY18-RE0009".

References

[1] Hong, S. S., & Katti, S. R. (2011, August). DOF: A local wireless information plane. In Proceedings of the ACM SIGCOMM 2011 Conference (pp. 230-241).

[2] Lakshminarayanan, K., Sapra, S., Seshan, S., & Steenkiste, P. (2009, December). Rfdump: an architecture for monitoring the wireless ether. In Proceedings of the 5th international conference on Emerging networking experiments and technologies (pp. 253-264).

[3] Iyer, V., Hermans, F., & Voigt, T. (2015, February). Detecting and avoiding multiple sources of interference in the 2.4 ghz spectrum. In European Conference on Wireless Sensor Networks (pp. 35-51). Springer, Cham.

[4] Yang, Z., Wang, Y., Zhang, L., & Shen, Y. (2018, December). Indoor interference classification based on wifi channel state information. In International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage (pp. 136-145). Springer, Cham.

Machine learning models for quality-of-service estimation and anomaly detection over real wireless network data

Abraham Pérez-Hernández, Maydelis N. Barreras-Martín, and Pablo Aguilera.

Galgus Research
{abraham.perez}{maydelis.barreras}{pablo.aguilera}@galgus.net

1 Introduction

Modern WiFi networks have become complex systems with multiple entities interacting with each other, trying to provide the best quality of service to all their connected users [1]. Meanwhile, the demand for multimedia streaming services, ultra-high-resolution video, and real-time gaming has increased exponentially. Due to this demand, the maintenance and troubleshooting of such networks become a task reserved for technical experts. These tasks become costly and must be performed continuously so that connected devices do not experience the quality of service (QoS) degradation. In addition, there are multi-factor anomalies that are very difficult to detect by the human eye, or that it is too late to act when they are detected.

In this paper, we study various machine learning models for the supervised classification of large amounts of collected data [2]. The objective is to obtain the modulation and coding scheme (MCS), one of the most reliable network metrics to assess the QoS that a user is experiencing [3]. The resulting models, once trained with real data, can be deployed in corporate WLAN networks to provide insights on: (I) The expected MCS based on the rest of the observed parameters. (II) The most likely cause why the MCS has reached that value. (III) Whether this value is considered an anomaly concerning what should have been obtained in the current state of the system [4].

The paper is structured as follows. In Section II, the general architecture of the system is presented and the remote data collection process is discussed. Section III presents the machine learning models to be analyzed, as well as the experiments and their results.

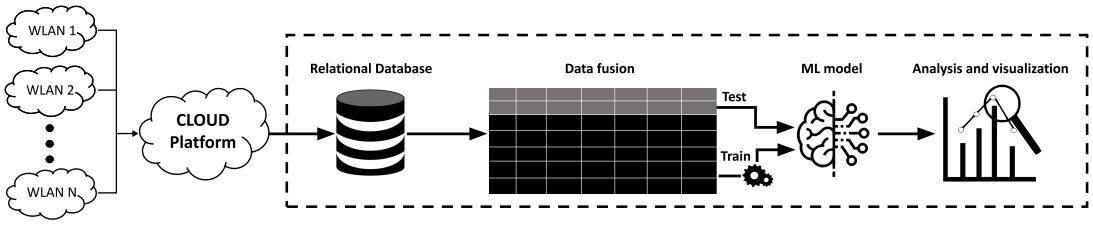


Fig. 1. Scheme of the system.

2 System overview

A WLAN network generally consists of access points (APs), switches to aggregate traffic, and a router to forward the data to/from the Internet, in addition to Ethernet cabling and end devices (smartphones, tablets, laptops, etc.) that connect wirelessly to the APs. Furthermore, modern WiFi networks usually have a management and configuration platform in the cloud, where technicians can collect statistics and modify settings on the remote machines, either via API or through graphical interfaces. The most direct way to collect information on the various network metrics is to capture various features in the APs, which are the most critical equipment in the network. Indeed, through their radio interfaces, they are in direct communication with the end-users, who are the ones who suffer or enjoy the quality of the network the most. Figure 1 shows an overview of the system of our approach. As can be seen, firstly, data is collected from different WLAN networks. This process consists of a series of queries and API calls to stimulate the APs and to gather their responses so that it is passive for the users. This way they don't have to be disturbed by installing apps and

granting permissions, and it is more flexible. Specifically, the metrics we captured from the APs are shown in Table 1, on three levels: (I) about their internal state, (II) about their connection to the Internet, (III) referring to their link with connected clients. All this data is then aggregated and stored on a remote machine, which will also oversee the offline post-processing. This data is initially stored in a relational database and then merged into a single table pivoting on the fundamental metrics. Once the feature table is built, different analyses can be performed in parallel to test its classification and predictive capabilities.

Table 1. List of collected metrics.

(I) - Minutes since the last hardware reboot - Averaged load in the last minutes - Used/free/buffered/available memory - CPU status - 2.4 GHz and 5 GHz radio channels and their occupation time	(III) - 802.11 Modulation and Coding Scheme - Signal-to-Noise Ratio (SNR) - 802.11 protocol: A/B/G/N/AC/AX - Upload/download throughput - Failed and total transferred packets - Roaming: 802.11K/R/V protocols - Channel bandwidth - Number of spatial streams (MIMO)
(II) - Latency to Google's DNS (`8.8.8.8`) - Upload/download bitrate speed in Mbps	

3 Experiments

A series of experiments have been carried out with three different models: decision tree (DT), random forest (RF), neural network (NN). Although regression tests have also been performed on the MCS, here we show the results on classification tests, as this problem is more consistent with the nature of the MCS as a QoS discrete metric. Recalling that the MCS is a discrete value ranging from 1 to 9, we consider a model to be a good classifier if it obtains a low root mean squared error (RMSE), or a high success rate. To provide context, each sample was taken approximately every 6 minutes on each AP of each WLAN. The 15495 samples captured during multiple weeks have been divided into two independent sets: a training set (80%) and a test set (20%). Missing memory data were filled in using linear regression, which showed 99.97% accuracy. Data rates and throughput were shown to have a negligible influence on the models, and have therefore been ignored for the final analysis. Also, features like the *transmission mode* that could distort the results were removed because of their artificial correlation with the MCS due to standard restrictions [5].

3.1 Hyperparameter selection

After some experimental tests, the chosen values for the hyperparameters of the DTs are a maximum branch depth of 6 and a minimum number of samples per leaf of 0.5%. In the RF models, a number of estimators (trees) of 800 is added to the previous parameters, as it produced the best results. Finally, NNs are much more flexible, as there is a wide variety of choices, and after some testings, the final NN is conformed by a total of five layers with 72, 100, 100, 48, and 9 neurons, respectively. Moreover, a 10% dropout has been applied to avoid overfitting.

3.2 MCS root cause analysis

On the one hand, it is possible to identify it on the decision paths over the trained DT and RF, as well as the influence of each input feature on the final estimated MCS. The NN, on the other hand, does not directly show the influence of each feature, which is its main disadvantage compared to the previous models. Thus, in order of importance, the features that have the greatest influence on the decision are: (I) Bandwidth (24%). (II) Signal to Noise Ratio (12%). (III) Spatial streams (12%). (IV) Used and available memory (7%). (V) Uptime (5.5%). The remaining features influence less than 5%. This allows to immediately solve two very useful problems for network administrators when troubleshooting: (I) Given an observed MCS, choose as the most likely causal branch the one that leads to a leaf that maximizes the number of samples. (II) Given an observed and an estimated MCS, determine whether it is an anomaly or not concerning the rest of the dataset. In other words, whether or not this value can be explained by the model.

3.3 Impact of the size of the training set

Finally, the number of samples to train the three models will be increased from 1% to 100% of the set reserved for training. For each iteration, the samples to evaluate the success rate of classification are chosen randomly (until the corresponding percentage is reached), to avoid biases due to transitory phenomena in the dataset. Figure 2 shows the performance of the three models in terms of the success rate concerning the percentage of training data used. Results have been averaged 100 times to make the output more robust. It can be seen how the RF and DT models perform better for small training datasets, while the NN outperforms them with a larger corpus. Moreover, it is not worth training with more than 6000 samples (half of those available), thereafter achieving an RMSE of 1.5 and a success rate of 81.25%.

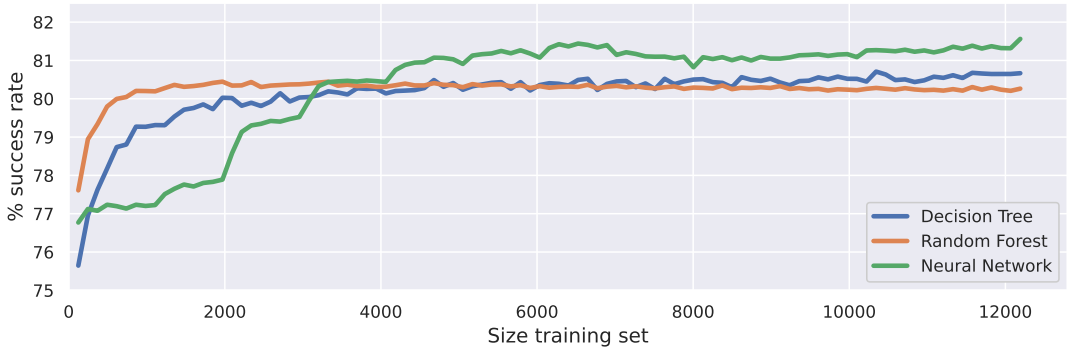


Fig. 2. The averaged test success rate of the three models versus the size of the training data set.

4 Conclusions

In this work, three models for the classification of the MCS have been built and evaluated using real data from active WiFi networks. This allows estimating the quality-of-service of each connected user, as well as detecting anomalies and explaining the root cause of the values found. The data collection and processing architecture has also been presented. The experiments carried out show that the neural network needs many more training samples to reach its minimum. Of the three models chosen, the one with the lowest error rate is the neural network, but as we already know, it hinders the interpretability of the decisions. Thus, the best compromise between model explainability and performance is obtained in the random forest.

References

1. Wi-Fi Alliance, "Global Economic Value of Wi-Fi 2021-2025," Sep. 2021. [Online]. Available: https://www.wi-fi.org/downloads-public/Global_Economic_Value_of_Wi-Fi_2021-2025_202109.pdf/37347
2. M. Abusubaih, "Intelligent wireless networks: Challenges and future research topics," *Journal of Network and Systems Management*, vol. 30, no. 1, pp. 1–29, 2022.
3. D. D. Coleman and D. A. Westcott, *CWNA Certified Wireless Network Administrator Study Guide: Exam CWNA-108*. John Wiley & Sons, 2021.
4. C. V. Murudkar, R. D. Gitlin *et al.*, "Machine learning for QoE prediction and anomaly detection in self-organizing mobile networking systems," 2019.
5. A. Mcgregor and D. Smithies, "Rate adaptation for 802.11 wireless networks: Minstrel," *Submitted to ACM SIGCOMM*, 2010.

Towards an Online Water Quality Monitoring system of Dynamic Environments using an Autonomous Surface Vehicle

F. Peralta[1], D. Gutierrez Reina[1] and S. Toral[1]

University of Seville, Spain
fperalta@us.es
dgutierrezreina@us.es
storal@us.es

Abstract. The usage of Autonomous Surface Vehicles enhances the monitoring of large-scale water environments. In this work-in-progress, we evaluate the idea of monitoring the continuously changing behaviors of water quality parameters via Bayesian Optimization and Gaussian Process that account for time-varying behaviors. Results of this novel formulation show good monitoring regarding three implemented strategies for the monitoring system.

1 Introduction

Water Quality Parameters (WQPs), such as Dissolved Oxygen, pH, turbidity, etc., are used to describe the physicochemical properties of water bodies. Bad practices involving these waters can result in polluted waters that are no longer safe to harbor a balanced ecosystem, meaning that their WQPs have undesirable values and are harmful for fish, and even humans. Therefore, a monitoring system of these WQPs can be helpful for maintaining or recovering healthy waters. Mar Menor in Spain is a salt lagoon that is known for having polluted waters for numerous reasons [1], hence it is a good candidate for monitoring.

There exist a set of fixed monitoring stations across the lagoon [1], but its size (135 km^2) is too large to let the fixed system alone to provide models with low levels of uncertainty. Moreover, there exist zones whose WQPs are always only approximated since they are never measured. It is inefficient just to increase the number of measurement locations and, more importantly, not every location might be available for continuously measuring. Therefore, in this work, we use Autonomous Surface Vehicles (ASVs) to solve these issues. ASVs are mobile robots that can travel on the surface of a water body and, in this case, perform WQP measurements using on-board WQP sensor systems. Monitoring water environments using ASVs has other advantages considering their safety, robustness, and quick reusability.

Systems that use ASVs to perform monitoring can be found in the literature [2, 3], some of them focus on patrolling the surface of the water body [2], others seek to obtain approximate static models [3], and others on optimizing [4] (finding minimum/maximum), but none of them consider that the environment is continuously changing, which is an expected behavior of WQPs. Considering this, the monitoring to be proposed in this work-in-progress considers this dynamism when performing the mission and is designed to produce dynamic models of WQPs.

This work-in-progress discusses the main system that will efficiently select Water Quality measurement locations for an ASV. The system will employ a time-varying surrogate model as the core component and utility functions to select measurement locations so that this model is reliable and includes knowledge of a non-static scenario. The main contribution of the current work is: the first system, to the best known of the authors, to propose a time-varying surrogate modeling system for WQPs using an ASV. Section 2 describes the problem and the proposed approach and Section 3 presents the current results and provides a framework for future work.

2 Description of the Proposed Approach

To efficiently utilize the offline data, which consists of 12 real measurements performed, we use Gaussian Process (GPs) as the surrogate model, which will include information derived from these

measurements. GPs are stochastic models based on Multivariate Normal Distributions that can include known information to efficiently describe approximated models [4]. Moreover, GPs yield the expected output values and confidence measures around them, making their use evident for cases like this, in which we will never know the real behavior of a WQP. To produce a regression, GPs use i) a covariance function between inputs, usually called kernel function, and ii) pairs of input and output data.

In this work, we aim at obtaining WQP values (output) considering a location x inside the lagoon (input). Therefore, the kernel function is described in terms of covariance between measurement locations (i.e., how far apart can be two locations so they are correlated?). In [4], the authors have shown that the Radial Basis Function (RBF) kernel can be used to model WQPs efficiently. RBF depends on a single hyper-parameter ℓ, named length scale, and has the form of $k(x, x') = \exp\left(-||x - x'||^2/2\ell^2\right)$. In this work, we consider the measurements performed by the mentioned fixed stations to approximate the length scale value. Finally, the GP regression for all lagoon locations $x \in X$ can be obtained in terms of the mean value $\mu(x)$ and the uncertainty $\sigma(x)$ around it using the next expression:

$$\mu(x) = k(x, \mathbf{p})\big(K + \sigma_\epsilon^2 I\big)^{-1} y \tag{1}$$

$$\sigma(x) = k(x, x) - k(x, \mathbf{p})\big(K + \sigma_\epsilon^2 I\big)^{-1} k(\mathbf{p}, x) \tag{2}$$

where $\mathbf{p} \subset X$ are the locations in which measurements have been performed, hence their y values are also known, K is the covariance matrix of all locations $\mathbf{p}$ and σ_ϵ^2 is the expected noise value. Considering this regression, the objective is to define new measurement locations according to the GP model and a utility function (based on $\mu(x), \sigma(x)$), which corresponds to the truncated Expected Improvement (tr-EI(x)) [4], so that a new measurement location is obtained using:

$$x^* = \arg\max \big(\text{tr-EI}(x)\big) \;\forall\; x \in X \tag{3}$$

This function ensures that the vehicle is only able to travel a maximum distance between measurements. Next, using the methods for finding maximum/minimum time-varying functions described in [5]. We implemented the following strategies for efficient time-varying WQP monitoring: i) ignore strategy, ii) time as third feature strategy, and iii) time as noise amplitude strategy.

2.1 Ignore Strategy (IS)

As the name suggests, in the first strategy, we ignore that the measured model is dynamic and the GP is fitted to the measured data as is. This method resembles the work in [4], where the behavior is expected to be stationary. This method serves as a baseline and can be used for stationary parameters, but will produce unreliable outputs for time-varying behaviors.

2.2 Time as third feature Strategy (TaTFS)

Expanding the dimension of the input is probably the most compelling strategy. In this strategy, we expand the input x to include the time t. Consequently, the input space is defined by $x = (\text{x}, \text{y}, t)$, in which t is defined by the difference in hours between the time of measurement and the time of the first measurement performed t_s. Thus, t is a monotonically increasing value.

Since an additional feature is now present in the GP, an additional hyper-parameter must be precalculated so that the GP can make efficient use of this dimension. In this approach, we propose to derive an approximate value considering the time and output difference between two available measures. With $\ell = \big(\ell, \ell, \mathbf{t}\big)$ and all previous expressions, the new GP will vary its output according to the time in which the measurement has been performed.

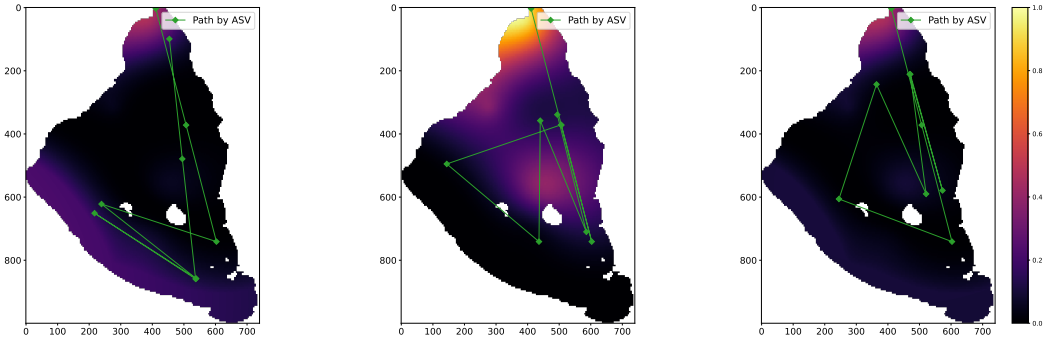
2.3 Time as Noise amplitude Strategy (TaNS)

For continuous monitoring, considering that newer measurements are more relevant than older ones is also a valid strategy. In this strategy, we maintain the location (x, y) as input but change the noise σ_ϵ^2 to account for "aging" of measurements. This strategy makes explicit use of the advantage of including noise in the GP, since the value is now time-varying, hence not the same

for all performed measurements. In that sense, the noise value $\sigma_\epsilon^2(t)$ is now dependent on t and is updated with each evaluation of the GP. The dependency is a subject of study and must be carefully defined according to the expected reevaluation time frame.

3 Current Results and Future Work

Initial results include evaluation of the offline error according to synthetic data of one WQP. Using the web page `https://marmenor.upct.es/`, we selected to monitor the turbidity in two dates. For the lengthscale values, we used the map and obtained a value that would satisfy the reads (measurements). Considering the initial and final readings, we can also derive a lengthscale of time for the second strategy whose value is also useful for the third strategy. Using as lengthscales, $\ell_x, \ell_y, \ell_t = [1998.9, 1358.457, 27.68]$ [pixels, pixels, hours] and defining a maximum distance of 382 [pixels], according to [4], Figure 1 show current results for the offline error at time t_e, which considers the final synthetic map available in the same webpage as the ground truth.



(a) Ignore Strategy. (b) Time as third feature Strategy. (c) Time as Noise Amplitude Str.

Fig. 1: Offline Squared Error for the three different implemented strategies.

Current results show that, for the slow rate of change of the synthetic data, the IS and TaNS can efficiently obtain good surrogate models with low levels of offline error, having Mean Squared Errors (MSE) of 0.0396 and 0.0288, respectively. Surprisingly, the TaTFS shows the least favorable behavior ($MSE = 0.082$). Future work will focus on time-varying acquisition functions so that the selection of next measurement locations also includes the time dependence. Additional future work will also include fusion with fixed station sensor values and evaluation considering the error over time instead of the static offline final error currently being used.

References

1. M. Erena, J. A. Domínguez, F. Aguado-Giménez, J. Soria, and S. García-Galiano. Monitoring coastal lagoon water quality through remote sensing: The mar menor as a case study. *Water*, 11(7):1468, 2019.
2. M. Arzamendia, D. Gregor, D. Gutierrez Reina, and S. Toral. A path planning approach of an autonomous surface vehicle for water quality monitoring using evolutionary computation. In *Technology for Smart Futures*, pages 55–73. Springer, Switzerland, 2018.
3. Micaela Jara Ten Kathen, Isabel Jurado Flores, and Daniel Gutiérrez Reina. An informative path planner for a swarm of asvs based on an enhanced pso with gaussian surrogate model components intended for water monitoring applications. *Electronics*, 10(13):1605, 2021.
4. F. Peralta, D. Gutierrez Reina, S. Toral, M. Arzamendia, and D. Gregor. A bayesian optimization approach for multi-function estimation for environmental monitoring using an autonomous surface vehicle: Ypacarai lake case study. *Electronics*, 10(8):963, 2021.
5. Sergio Morales-Enciso and Juergen Branke. Tracking global optima in dynamic environments with efficient global optimization. *European Journal of Operational Research*, 242(3):744–755, 2015.

This work has been partially funded under the grant RTI2018-098964-B-I00 funded by MCIN/AEI/ 10.13039/501100011033, by the regional government Junta de Andalucía under the Projects "Despliegue Inteligente de una red de Vehículos Acuáticos no Tripulados para la monitorización de Recursos Hídricos US-1257508" and "Desarrollo de nuevas tecnologías WiFi inteligentes en entornos móviles y con alta densidad de usuarios PAIDI 2020 P18-TP-1520."

MORL/D: Multi-Objective Reinforcement Learning based on Decomposition

Florian Felten[1], El-Ghazali Talbi[2,3], and Grégoire Danoy[1,3]

[1] SnT, University of Luxembourg
[2] CNRS/CRIStAL, University of Lille
[3] FSTM/DCS, University of Luxembourg
{florian.felten, gregoire.danoy}@uni.lu, el-ghazali.talbi@univ-lille.fr

1 Introduction

Many real life problems involve multiple objectives. The established way to resolve these is called Multi-Objective Optimization (MOO). This field has been extensively studied for over 50 years, producing a wide variety of optimisation approaches. These rely on a set of concepts such as Pareto dominance, indicators or scalarisation. Yet most of these approaches require a complete knowledge of the environment dynamics [6].

Reinforcement Learning (RL) is a machine learning technique which aims at training an agent to behave optimally in some possibly unknown environment [6]. Its extension to environments containing multiple objectives is called Multi-Objective Reinforcement Learning (MORL). One of the existing approaches to solve MORL problems, called *outer loop* multi-policy MORL, aims at learning various optimal compromises between objectives. Its strategy is to decompose the problem into various subproblems by using different scalarisation functions. These algorithms then apply standard RL on the scalarised subproblems as to target various optimal policies. Other MORL approaches have been published lately; nevertheless, this field remains understudied when compared to MOO or RL [2].

At the same time, MOEA/D [8], a framework allowing to use the decomposition technique in Evolutionary Algorithms (EAs) has been introduced in the MOO literature. Similar to outer loop MORL, this framework converts the multi-objective problem into various single-objective problems (SOPs) by using various scalarisations. An interesting feature of MOEA/D is that the algorithm proposes to search for different solutions simultaneously by using the concept of neighborhood: subproblems which rely on close weight parameters for their scalarisations have good chances of sharing similarities in their solutions' components, see for example Figure 1. Since its introduction in [8], this algorithm has been the subject of a lot of studies and variants [7].

Although outer loop algorithms have a similar structure to MOEA/D, to the best of our knowledge no work has clearly identified those similarities yet. It seems likely to us that techniques coming from decomposition based algorithms could be good candidates to be applied to outer loop MORL. Given the abundance of decomposition related research published over the last years, the field of MORL could clearly benefit from the acquired knowledge of the MOO community. This paper presents an initial attempt to design an outer loop MORL framework which could easily be extended in order to ease the transfer of MOEA/D variants' concepts into MORL.

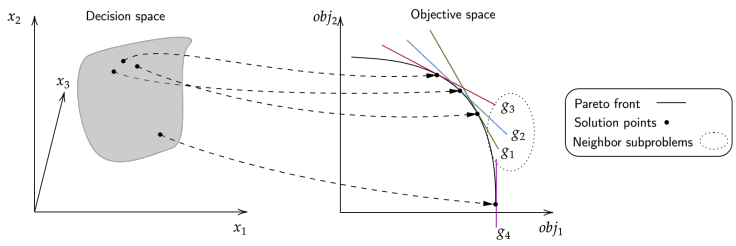


Fig. 1. The MOEA/D idea: split the multi-objective problem into various single-objective problems g_n. g_1, g_2, g_3 are considered to be neighbors since their associated weight vectors are close while g_4 is not considered to be in the neighborhood.

Algorithm 1: MORL/D: outer loop MORL framework using decomposition.

Input: Stopping criterion *stop*, Environment *env*, Number of scalarisations n, Neighborhood size T, Sub problem stopping criterion *SOStop*.
Output: The trained policies $\pi_1, ..., \pi_n$.

```
1  begin
2      Λ ⟵ InitializeWeights(n)
3      Π ⟵ InitializePolicies(n)
4      for i = 1 to n do
5          B(i) ⟵ {i_1, ..., i_T} | Λ^{i_1}, ..., Λ^{i_T} are the closest neighbors of Λ^i in the euclidean space
6      while ¬stop do
7          for i = 1 to n do
8              Π^i ⟵ SOLearn(Π^i, Scal, Λ^i, env, SOStop)
9              UpdateNeighbors(B(i), Π^i)
10     return Π
```

2 A Decomposition Framework for MORL

Algorithm 1 presents a high-level method for learning multiple policies using decomposition. It is inspired from the seminal work by Zhang et al. [8], and is designed to be easily extended.

The algorithm starts by initializing the weights and policies (lines 2–3)[1]. Then the neighborhoods are computed (lines 4–5). A single-objective RL algorithm is applied on the environment to improve the current policy Π^i. The vectorial rewards are converted to scalar using the scalarisation function *Scal*, as well as the weight vector associated to the policy Λ^i (line 8). The learning can be propagated to the neighbor policies in $B(i)$ (line 9). The loops ensure all target policies are trained until a stopping criterion. Finally, the trained policies, representing various behaviors are returned (line 10).

Multiple parts of this algorithm could be replaced in order to enhance the performance of the overall algorithm. Some candidates, based on variants of MOEA/D, categorized by the parts they modify are presented below.

Neighborhoods *$(B, UpdateNeighbors)$* One of the important aspects of MOEA/D is the concept of neighbor solutions. It allows MOEA/D to mix multiple neighbor solutions to find potentially better ones. The definition of neighborhood and crossing methods have been the subject of various subsequent studies. For example, some algorithms add constraints to the SOPs in the same neighborhood to prevent them from falling onto the same optimal point in order to preserve the diversity of the solution set. Others have shown that dynamic neighborhood size might be beneficial on some instances [7].

In MORL, multiple ways to share information between neighbors can be imagined. Some work already use such mechanisms to reduce the training time when learning multiple policies. For example, [4] proposes to initialize new policies from close, already trained policies. Other methods such as sharing experience buffer have been introduced [1]. However, such studies are currently limited to the weighted sum scalarisation and the effectiveness of neighbor sharing with non-linear scalarisation remains an open question.

Decomposition methods *(Scal)* Over time, various scalarisation methods have been proposed for MOEA/D. The original ones presented in [8] have been enhanced or replaced in subsequent studies. Additionally, some MOEA/D variants propose to adapt scalarisation methods during the search, or to combine them in order to benefit from their advantages. Finally, some approaches aiming at decomposing the decision space instead of the objective space have also been proposed in MOO.

Multiple salarisation methods have been studied in MORL [2]. The non-linear cases have been shown to be more challenging since the Markov property does not hold in such environments [3]. However, to the best of our knowledge, no work has been dedicated to combining or adjusting scalarisations methods, or decomposing the decision space in MORL.

[1] It is possible to also include the learned values to apply for example an actor-critic learning algorithm.

Weight vectors generation *($InitializeWeights, \Lambda$)* The choice of weight vectors in such techniques can influence both the training time and the performance of the algorithm. Again, multiple solutions have been proposed as extensions to decomposition-based EAs. Uniform design aims at distributing the weight vectors uniformly in the objective space while other algorithms propose to dynamically adjust the weight vectors to focus on less crowded regions in the objective space [7].

In MORL, most of the work relies on uniform design. Though an interesting approach to generate weights based on the potential improvement of the convex coverage set of learned policies is proposed in [5]. Again, this technique is restricted to linear scalarisation technique.

3 Conclusion

This article presented an initial attempt to build a framework allowing to transpose concepts coming from MOO, and in particular MOEA/D variants, to inner loop MORL. It is able to learn multiple policies, leading to various compromises between the objectives in the problem, and is designed to be modular and extendable.

From there, some existing works have been briefly presented and leads for MORL/D variants have been identified. (i.) To the best of our knowledge no MORL work has currently been conducted to share knowledge between neighbors using non-linear scalarisation. (ii.) Many ways to share information between neighbors are yet to be studied. (iii.) Combination or adaptation of decomposition methods has not been tried in MORL yet. (iv.) Decision space decomposition is also a promising lead to ensure diversity of resulting policies. (v.) Weight adjustment techniques show potential gains as well.

In the future, we aim to use this framework and propose variants based on the identified leads, or other existing decomposition based techniques. Such algorithms will be used to control robots in real time.

Acknowledgement

This work is funded by the Fonds National de la Recherche Luxembourg (FNR), CORE program under the ADARS Project, ref. C20/IS/14762457.

References

1. Diqi Chen, Yizhou Wang, and Wen Gao. Combining a gradient-based method and an evolution strategy for multi-objective reinforcement learning. *Applied Intelligence*, 50, October 2020.
2. Conor Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa Zintgraf, Richard Dazeley, Fredrik Heintz, Enda Howley, Athirai Irissappane, Patrick Mannion, Ann Nowe, Gabriel Ramos, Marcello Restelli, Peter Vamplew, and Diederik Roijers. *A Practical Guide to Multi-Objective Reinforcement Learning and Planning.* March 2021.
3. Diederik Roijers, Denis Steckelmacher, and Ann Nowe. *Multi-objective Reinforcement Learning for the Expected Utility of the Return.* July 2018.
4. Diederik Roijers, Shimon Whiteson, and Frans Oliehoek. *Point-Based Planning for Multi-Objective POMDPs.* July 2015.
5. Diederik M. Roijers, Shimon Whiteson, and Frans A. Oliehoek. Linear support for multi-objective coordination graphs. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, AAMAS '14, pages 1297–1304, Richland, SC, May 2014. International Foundation for Autonomous Agents and Multiagent Systems.
6. Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* Adaptive Computation and Machine Learning series. A Bradford Book, Cambridge, MA, USA, 2 edition, November 2018.
7. Qian Xu, Zhanqi Xu, and Tao Ma. A Survey of Multiobjective Evolutionary Algorithms Based on Decomposition: Variants, Challenges and Future Directions. *IEEE Access*, 8:41588–41614, 2020. Conference Name: IEEE Access.
8. Qingfu Zhang and Hui Li. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, December 2007. Conference Name: IEEE Transactions on Evolutionary Computation.

Integer Linear Programming reformulations for the linear ordering problem

Nicolas Dupin[1]*[0000-0003-3775-5629]

Université Paris-Saclay, LISN, 91405, Orsay, France
nicolas.dupin@universite-paris-saclay.fr

Abstract. This article studies the linear ordering problem, with applications in social choice theory and databases for biological datasets. Integer Linear Programming (ILP) formulations are available for linear ordering and some extensions. ILP reformulations are proposed, showing relations with the Asymmetric Travel Salesman Problem. If a strictly tighter ILP formulation is found, numerical results justify the quality of the reference formulation for the problem in the Branch&Bound convergence. The quality of the continuous relaxation allows to design rounding heuristics, it offers perspectives to design matheuristics.

Keywords : Optimization; Integer Linear Programming; Linear ordering; Median of permutations; Consensus ranking; Polyhedral analysis

1 Introduction

A bridge between optimization and Machine Learning (ML) exists to optimize training parameters of ML models, using continuous optimization and metaheuristics [17, 18]. Discrete and exact optimization, especially Integer Linear Programming (ILP), is also useful to model and solve specific variants of clustering or selection problems for ML [5, 6]. In this paper, another application of ILP to learning is studied: the Linear Ordering Problem (LOP). LOP aims to define a common and consensus ranking based on pairwise preferences, which is used in social choice theory. If many applications deal with a small number of items to rank, bio-informatics applications solve large size particular instances of LOP as medians of permutations [2, 3]. An ILP formulation is available for LOP with constraints defining facets [10, 11]. An extension of LOP considering ties relies on this ILP formulation [2]. Current and recent works focus on consensus ranking for biological datasets, and use specific data characteristics of these median of permutation problems for an efficient resolution [1, 13]. This paper analyzes the limits of state-of-the art ILP solvers to solve LOP instances. Several alternative ILP formulations are designed using recent results on the Asymmetric Traveling Salesman Problem (ATSP) from [14]. Comparison of Linear Programming (LP) relaxations illustrates and validates polyhedral analyses, as in [14]. The practical implication of polyhedral work is analyzed on the resolution using modern ILP solvers, as in [4]. Lastly, the quality of LP relaxation is used to design first variable fixing matheuristics, as in [7]. Variants of variables choices and ILP Formulations are recalled in Tables 1 and 2.

Table 1. Definitions of variables in the ILP formulations

Variables	Definitions
$x_{i,j} \in \{0,1\}$	$x_{i,j} = 1$ iff item $i \neq j$ is ranked before j.
$y_{i,j} \in \{0,1\}$	$y_{i,j} = 1$ iff item $i \neq j$ is ranked immediately before j.
$f_i \in \{0,1\}$	$f_i = 1$ iff item i is the first item of the ranking.
$l_i \in \{0,1\}$	$f_i = 1$ iff item i is the last item of the ranking.
$n_i \in [0, N-1]$	$n_i - 1$ gives the position of item i in the ranking.
$z_{i,j,k} \in \{0,1\}$	$z_{i,j,k} = 1$ for $i \neq k$ and $j \neq k$ iff i is ranked before j and item j is ranked immediately before k.
$z'_{i,j,k} \in \{0,1\}$	$z'_{i,j,k} = 1$ for $i \neq j \neq k$ iff i is ranked before j and j is before k.

2 Problem statement and reference ILP formulation

LOP consists in defining a permutation of N items indexed in $[\![1; N]\!]$, while maximizing the likelihood with given pairwise preferences. $w_{i,j} \geqslant 0$ denotes the preference between items i and j: i is preferred to j if $w_{i,j}$ is higher than $w_{j,i}$. A ranking is evaluated with the sum of $w_{i,j}$ in the $\frac{N(N-1)}{2}$ pairwise preferences it implies. Each permutation of $[\![1; N]\!]$ encodes a solution of LOP, there are thus $N!$ feasible solutions. The reference ILP formulation, given and analyzed in [10, 11], uses binary variables $x_{i,j} \in \{0,1\}$ such that $x_{i,j} = 1$ if and only if item i is ranked before item j in the consensus permutation. With such encoding, one computes the rank of each item i with $1 + \sum_{j \neq i} x_{j,i}$. ILP formulation from [11] uses $O(N^2)$ variables and $O(N^3)$ constraints:

$$\max_{x \geqslant 0} \quad \sum_{i \neq j} w_{i,j} x_{i,j} \tag{1}$$

$$x_{i,j} + x_{j,i} = 1 \qquad \forall i < j, \tag{2}$$

$$x_{i,j} + x_{j,k} + x_{k,i} \leqslant 2 \qquad \forall i \neq j \neq k, \tag{3}$$

Constraints (2) model that either i is preferred to j, or j is preferred to i. Constraints (3) ensure that $x_{i,j}$ variables encode a permutation: if i is before j and j before k, i.e. $x_{i,j} = 1$ and $x_{j,k} = 1$, then i must be before k, i.e. $x_{i,k} = 1$ which is equivalent to $x_{k,i} = 0$ using (2). Constraints (2) and (3) are proven to be facet defining under some conditions [11].

Note that some alternative equivalent ILP were formulated. Firstly, the problem is here defined as a maximization, whereas it is considered as a minimization of disagreement in [2]. Considering $w'_{i,j} = w_{j,i}$ or $w'_{i,j} = M - w_{i,j}$, where M is an upper bound of weights $w_{i,j}$, allows to transform minimization into maximization. Secondly, equations (3) are equivalently written as $x_{i,k} - x_{i,j} - x_{j,k} \geqslant -1$ in [2]. Formulation (3) is symmetrical, and was also used for the ATSP [16]. To see the equivalence, we use that $-x_{i,k} = x_{k,i} - 1$:

$$\begin{aligned} x_{i,k} - x_{i,j} - x_{j,k} \geqslant -1 &\iff -x_{i,k} + x_{i,j} + x_{j,k} \leqslant 1 \\ x_{i,k} - x_{i,j} - x_{j,k} \geqslant -1 &\iff x_{k,i} - 1 + x_{i,j} + x_{j,k} \leqslant 1 \\ x_{i,k} - x_{i,j} - x_{j,k} \geqslant -1 &\iff x_{k,i} + x_{i,j} + x_{j,k} \leqslant 2 \end{aligned}$$

3 From ATSP to consensus ranking, tighter formulations

LOP and ATSP feasible solutions may be encoded as permutations of $[\![1; N]\!]$, order matters for cost computations. If any LOP solution is a permutation, ATSP solutions are Hamiltonian oriented cycles. LOP solutions can be projected in a cycle structure, adding a fictive node 0 such that $w_{0,i} = w_{i,0} = 0$ opening and closing the cycle: $x_{0,i} = 1$ (resp $x_{i,0} = 1$) expresses that i is the first (resp last) item of the linear ordering. This section aims to use polyhedral work from ATSP to tighten the reference formulation for LOP [14]. For ATSP, binary variables $y_{i,j} \in \{0, 1\}$ are defined such that such that $y_{i,j} = 1$ if and only if j is next item immediately after item i, for $i \neq j \in [\![1; N]\!]$. Equivalently to consider a fictive node 0, we define binary variables $f_i, l_i \in \{0, 1\}$ such that $f_i = x_{0,i} = 1$ (resp $l_i = x_{i,0} = 1$) denotes that item i is the first (resp last) in the linear ordering. Having these variables x, y, l, f induces another ILP formulation for LOP, denoted SSB for ATSP [16]:

$$\max_{x,y,f,l} \quad \sum_{i \neq j} w_{i,j} x_{i,j} \tag{4}$$

$$x_{i,j} + x_{j,k} + x_{k,i} \leqslant 2 \qquad \forall i \neq j \neq k, \tag{5}$$

$$y_{i,j} \leqslant x_{i,j} \qquad \forall i \neq j, \tag{6}$$

$$x_{i,j} + x_{j,i} = 1 \qquad \forall i < j, \tag{7}$$

$$\textstyle\sum_i f_i = 1 \tag{8}$$

$$\textstyle\sum_i l_i = 1 \tag{9}$$

$$l_i + \textstyle\sum_{j \neq i} y_{i,j} = 1 \qquad \forall i, \tag{10}$$

$$f_i + \textstyle\sum_{j \neq i} y_{j,i} = 1 \qquad \forall i, \tag{11}$$

Objective function differ from ATSP, where a weighted sum of y, f, l variables is minimized [16]. x variables were used only to cut sub-tours for ATSP, whereas there are necessary for LOP to write the objective function. The constraints are identical for ATSP and LOP once variables x, y, f, l are used. Constraints (5) and (7) are identical with (2) and (3). Constraints (10) and (11) are ATSP elementary flow constraints: for each item there is a unique predecessor and a unique successor, 0 as node successor or predecessor implies using variables f_i, l_i. Unicity constraints (8) and (9) are ATSP elementary flow constraints arriving to and leaving from the fictive node 0. SSB can be tightened in the SSB2 formulation, replacing constraints (5) by tighter constraints (12) from [16]:

$$\forall i \neq j \neq k, \quad x_{i,j} + y_{i,j} + x_{j,k} + x_{k,i} \leqslant 2 \tag{12}$$

Sub-tours between two cities (or items) may have a crucial impact in the resolution, as in [5]. Having variables x and constraints (5) and the tighter variants implies the other sub-tours between two items. Indeed, $y_{i,j} + y_{j,i} \leqslant x_{i,j} + x_{j,i} = 1$. Constraints (13) are sub-tour cuts between node 0 and each item $i > 0$, these are known to tighten strictly SSB2 formulation [16]:

$$\forall i, \quad f_i + l_i \leqslant 1 \tag{13}$$

Another formulation was proposed for ATSP without constraints (5), but with linking constraints $y_{i,j} - x_{k,j} + x_{k,i} \leqslant 1$, to induce the same set of feasible solution [9]. These constraints can be tightened in two different ways:

$$\forall i \neq j \neq k, \quad y_{i,j} + y_{j,i} - x_{k,j} + x_{k,i} \leqslant 1 \tag{14}$$

$$\forall i \neq j \neq k, \quad y_{i,j} + y_{k,j} + y_{i,k} - x_{k,j} + x_{k,i} \leqslant 1 \tag{15}$$

Tightening only with (14) and (15) induce respectively GP2 and GP3 formulations for ATSP [9]. A strictly tighter formulation, denoted GP4, is obtained with both sets of constraints [14]. A strictly tighter formulation is also obtained adding (12) to (14) and (15) for ATSP. Numerical issues are to determine whether the quality of LP relaxation is significantly improved after tightening.

4 Other ILP reformulations

In this section, alternative ILP reformulations for LOP are provided, adapting other formulations from ATSP. Firstly, a formulation with $O(N^2)$ variables and constraints is given, before three-index formulations with $O(N^3)$ variables.

4.1 ILP formulation with $O(N^2)$ variables and constraints

Similarly with MTZ formulation [12], $O(N)$ additional variables $n_i \in [\![0, N-1]\!]$ can directly indicate the position of the item in the ranking :

$$\max_{x,n \geqslant 0} \quad \sum_{i \neq j} w_{i,j} x_{i,j} \tag{16}$$

$$x_{i,j} + x_{j,i} = 1 \qquad \forall i < j, \tag{17}$$

$$n_j + N \times (1 - x_{i,j}) \geqslant n_i + 1 \quad \forall i \neq j, \tag{18}$$

$$n_i + \sum_{j \neq i} x_{i,j} = N - 1 \qquad \forall i, \tag{19}$$

$$n_i \in [0, N-1] \qquad \forall i \tag{20}$$

Note that as for MTZ formulation, variables n_i can be declared as continuous, feasibility of (18) and bounds (20) implies $n_i \in [\![0, N-1]\!]$. Objective function (16) and constraints (17) are unchanged. Constraints (18) are similar with MTZ constraints: if i is ranked before j, i.e. $x_{i,j} = 1$, then it implies $n_j \geqslant n_i + 1$, N is a "big M" in this linear constraint. If (17) and (18) are sufficient to induce feasible solutions for the ILP, constraints (19) complete (18) without using any "big M". Indeed, $c_i = \sum_{j \neq i} x_{i,j}$ counts the number of items after i, so that for each i, $c_i + n_i = N - 1$. As big M constraints are reputed to be weak and inducing poor LP relaxations, a numerical issue is to determine the difference with the continuous relaxation when relaxing also constraints (18) and (19). Note that this relaxation has trivial optimal solutions, considering $x_{i,j} = 1$ and $x_{j,i} = 0$

for $i \neq j$ such that $w_{i,j} \geqslant w_{j,i}$. Hence, following upper bound is valid, and also larger than any LP relaxation for LOP:

$$UB = \sum_{i<j} \max(w_{i,j}, w_{j,i}) \tag{21}$$

4.2 Three-index Flow formulation

Another three index formulation, tighter than GP2, GP3 and GP4, was proposed for ATSP [9]. Adapting this formulation to LOP, one uses binary variables $z_{i,j,k} \in \{0,1\}$ for $i \neq k$ and $j \neq k$ defined with $z_{i,j,k} = 1$ if and only if i is ranked before j (not necessarily immediately before) and j is ranked immediately before k. First and last items are still marked with binaries $f_i, l_i \in \{0,1\}$. Binaries $x_{i,j}, y_{i,j} \in \{0,1\}$ are then defined by $x_{i,j} = \sum_k z_{i,j,k} + l_j$ and $y_{i,j} = z_{i,i,j}$.

$$\max_{z,l,f \geqslant 0} \quad \sum_{i \neq j} w_{i,j} \left(l_i + \sum_k z_{i,j,k} \right) \tag{22}$$

$$l_i + \textstyle\sum_k z_{i,j,k} + l_j + \sum_k z_{j,i,k} = 1 \quad \forall i < j, \tag{23}$$

$$\textstyle\sum_i f_i = 1 \tag{24}$$

$$\textstyle\sum_i l_i = 1 \tag{25}$$

$$l_i + \textstyle\sum_{j \neq i} z_{i,i,j} = 1 \quad \forall i, \tag{26}$$

$$f_i + \textstyle\sum_{j \neq i} z_{j,j,i} = 1 \quad \forall i, \tag{27}$$

$$z_{i,j,k} \leqslant z_{j,j,k} \quad \forall i, j, k, \tag{28}$$

Constraints (23) and (24)-(27) are respectively constraints (7) and (8)-(11) replacing x, y occurrences by the linear expressions using z, l variables. A similar operation allows to write the objective function using z, l variables. Constraints (28) model that $z_{i,j,k} = 1$ implies that j is ranked just before k and thus $z_{j,j,k} = 1$. This formulation has $O(N^3)$ variables and $O(N^3)$ constraints only because of constraints (28). It is possible to preserving the validity of the ILP while having only $O(N^2)$ constraints replacing flow constraints (28) by the aggregated version:

$$\forall j, k, \quad \sum_i z_{i,j,k} \leqslant N z_{j,j,k} \tag{29}$$

4.3 Another three-index flow formulation

$z'_{i,j,k} \in \{0,1\}$ defined for $i \neq j \neq k$ with $z'_{i,j,k} = 1$ if and only if items i, j, k are ranked in this order. In this ILP formulation, we keep variables $x_{i,j}$, it induces the valid ILP formulation for LOP:

$$\max_{x,z \geqslant 0} \quad \sum_{i \neq j} w_{i,j} x_{i,j} \tag{30}$$

$$3z'_{i,j,k} \leqslant x_{i,j} + x_{j,k} + x_{i,k} \qquad \forall i \neq j \neq k \tag{31}$$

$$z'_{i,j,k} + z'_{i,k,j} + z'_{j,i,k} + z'_{j,k,i} + z'_{k,j,i} + z'_{k,i,j} = 1 \quad \forall i \neq j \neq k, \tag{32}$$

Constraints (31) are linking constraints among variables x, z: $z'_{i,j,k} = 1$ implies $x_{i,j} = x_{j,k} = x_{i,k} = 1$. Constraints (32) express that each triplet $i \neq j \neq k$ is assigned in exactly one order in a permutation, replacing constraints of type $x_{k,i} + x_{i,j} + x_{j,k} \leqslant 2$. Constraints (32) induce that this ILP formulation has also $O(N^3)$ variables and $O(N^3)$ constraints. Note that a similar constraint can be defined as cut for the previous ILP formulation, with an inequality:

$$z_{i,j,k} + z_{i,k,j} + z_{j,i,k} + z_{j,k,i} + z_{k,j,i} + z_{k,i,j} \leqslant 1 \tag{33}$$

Table 2. Summary of implemented formulations, their denomination, the sets and asymptotic number of variables and constraints

Formulation	Variables	Constraints	nbVariables	nbConstraints
LOP_ref	x	(2), (3)	$O(N^2)$	$O(N^3)$
LOP_SSB2	x, f, l, y	(6) - (11), (12, (13)	$O(N^2)$	$O(N^3)$
LOP_GP3	x, f, l, y	(6) - (11), (15)	$O(N^2)$	$O(N^3)$
LOP_MTZ	x, n	(17), (19)	$O(N^2)$	$O(N^2)$
LOP_flowGP	z, f, l	(23)- (27),(28)	$O(N^3)$	$O(N^3)$
LOP_flowGP_aggr	z, f, l	(23)- (27),(29)	$O(N^3)$	$O(N^2)$
LOP_flow2	x, z'	(31), (32)	$O(N^3)$	$O(N^3)$

5 Computational experiments and results

Numerical experiments were proceeded using a workstation with a dual processor Intel Xeon E5-2650 v2@2.60GHz, for 16 cores and 32 threads in total. Cplex version 20.1 was used to solve LPs and ILPs. Cplex was called using OPL modeling language and OPL script. LocalSolver in its version 10.5 was used as a heuristic solver benchmark to compare primal solutions when optimal solutions are not proven. The maximal time limit for Cplex and LocalSolver was set to one hour, Cplex was used with its default parameters. For reuse and reproducibility, code and generated instances are available online at `https://github.com/ndupin/linearOrdering`.

5.1 Data generation and characteristics

It was necessary to generate specific instances for this study. As mentioned by [1, 13], instance characteristics are crucial in the resolution difficulty. In many

social choice applications and datasets, N is small, exact resolution with formulation LOP_ref is almost instantaneous. For the biological application, N is very large but median of permutations among similar permutations is easier than general instances. In the extreme case where $w_{i,j}$ coefficients encode a permutation (median of 1-permutation, trivial problem), trivial bounds UB give the optimal value, and LP relaxations of every ILP formulation give the integer optimal solution. For this numerical study, as in [14], quality of polyhedral descriptions are analyzed on the implications on the quality of LP relaxation using diversified directions of the objective function. Three generators were used for this study:

- `aleaUniform` (denoted aUnif): $w_{i,j}$ for $i \neq j$ are randomly generated with a uniform law in $[\![0, 100]\!]$.
- `aleaSum100` (denoted aSum): uniform generation in $[\![0, 100]\!]$ such that $w_{i,j} + w_{j,i} = 100$: for $i < j$ $w_{i,j}$ is randomly generated in $[\![0, 100]\!]$ and $w_{j,i}$ is then set to $w_{j,i} = 100 - w_{i,j}$.
- `aleaShuffle` (denoted aShuf): $\max(N/2, 20)$ random permutations are generated (with Python function shuffle), $w_{i,j}$ are then computed using Kendall-τ distance and Kemeny ranking, as in [1–3].

A fourth generator was coded, as in `aleaShuffle`, but generating small perturbations around a random permutation. Actually, the results were very similar for ILP formulations to the 1-median trivial instances. Real-life structured instances for median of permutations are much easier than random instances. The generators allow to analyze the impact of structured instances.

Number of items N was generated with values $N \in \{20, 30, 40, 50, 100\}$. For $N \in \{20, 30, 40\}$, the Best Known Solution (BKS) are optimal solutions proven by Cplex. For $N \in \{50, 100\}$, LocalSolver always provides the BKS. There is also no counter-example where LocalSolver does not find a proven optimal solution in one hour, we note that LocalSolver is also very efficient in short time limits. For each generator and value of N, 30 instances are generated and results are given in average for each group of 30 similar instances, with the denomination XX$-N$ where XX $\in$ {aUnif,aSum,aShuf}. Lower and upper bounds $v(i)$ on instance i are compared with gaps to BKS, denoted BKS(i):

$$gap = \frac{\mid v(i) - BKS(i) \mid}{BKS(i)} \tag{34}$$

5.2 Comparing LP relaxations

To analyze the quality of polyhedral descriptions recalled in Table 2, Table 3 presents gaps of LP relaxations of ILP formulations for LOP and the naive upper bound (21). Table 4 presents the computation time for LP relaxations, to highlight the impact of the number of variables and constraints recalled in Table 2. These tables illustrate the difficulty of instances, aShuf are easy instances with good naive upper bounds and LP relaxations. Datasets aSum and aUnif induce

Table 3. Comparison of the average gaps to the BKS for the LP relaxations of formulations recalled in Table 2 and the naive upper bound (21)

Instances	(21)	ref/SSB2	GP3	MTZ	flow-GP	flow-GP-agg	flow2
aUnif-20	12,86 %	0,02 %	10,49 %	10,84 %	5,22 %	11,53 %	12,86 %
aUnif-30	14,86 %	0,17 %	13,20 %	13,39 %	7,34 %	13,98 %	14,86 %
aUnif-40	16,98 %	0,60 %	15,68 %	15,78 %	9,22 %	16,16 %	16,98 %
aUnif-50	17,84 %	1,12 %	16,80 %	16,86 %	10,22 %	17,17 %	17,84 %
aUnif-100	21,65 %	3,17 %	21,10 %	21,11 %	-	21,25 %	21,65 %
aSum-20	19,00 %	0,05 %	15,56 %	16,13 %	7,26 %	17,23 %	19,00 %
aSum-30	21,96 %	0,31 %	19,53 %	19,84 %	10,25 %	20,45 %	21,96 %
aSum-40	24,40 %	1,18 %	22,52 %	22,70 %	12,53 %	23,21 %	24,40 %
aSum-50	26,24 %	2,25 %	24,71 %	24,83 %	14,16 %	25,23 %	26,24 %
aSum-100	31,44 %	4,97 %	30,64 %	30,65 %	-	30,84 %	31,44 %
aShuf-30	1,93 %	0,00 %	1,39 %	1,42 %	0,29 %	1,89 %	1,93 %
aShuf-40	1,44 %	0,00 %	1,18 %	1,18 %	0,42 %	1,42 %	1,44 %
aShuf-50	1,41 %	0,00 %	1,18 %	1,18 %	0,44 %	1,40 %	1,41 %
aShuf-100	1,80 %	0,02 %	1,65 %	1,55 %	-	1,80 %	1,80 %

more difficulties with worse continuous bounds, and aSum is even more difficult than aUnif.

Contrary to ATSP where GP2, GP3, SSB2 are not redundant [14], (3) induces much better LP relaxations for LOP than (14) and (15). Adding (14) and (15) in ILP formulations with (3) or (12) does not induce any difference in the quality of LP relaxation. It explains why in Table 2, we remove constraints of type (3) to compare quality of LP relaxations. An explanation is the different nature of LOP and ATSP problems because of different objective functions: if polyhedrons defined by constraints are identical, objective functions with weighted sums in x or y changes the projection on the space of interest.

Flow formulation flow-GP improves significantly the quality of LP relaxation of GP3, as for the ATSP, but it is still significantly worse than SSB formulations. Computation time of LP relaxation is much higher with flow-GP, computations were stopped in one hour without termination for $N = 100$. With aggregation (29) instead of (28), LP relaxation is computed quickly, but the quality of LP relaxation is dramatically decreased, the continuous bounds are close to the naive upper bounds (21). MTZ adaptation has the quickest LP relaxation, but the continuous bounds are close to the ones of GP3. Last flow formulation always provides exactly the naive upper bounds (21), constraints (33) do not tighten flow-GP formulation, this result differ from [15].

LP relaxation of LOP_ref is of an excellent quality, which illustrates polyhedral results and proven facets from [11]. In Table 2, LOP_ref and SSB2 formulations have the same values: except on three instances, LP relaxation are the same (with a tolerance to numerical errors on the last digit). On instance number 27 in aUnif-20 and instances number 17 and 29 in aSum-20, SSB2 improves the reference formulation around 0.01%, making a difference of one unit in the integer ceil rounding of the continuous relaxation. With additional experiments,

Table 4. Comparison of the average time (in seconds) to compute LP relaxations for ILP formulations recalled in Table 2

Instances	ref	SSB2	GP3	MTZ	flow-GP	flow-GP-agg	flow2
aUnif-20	0,04	0,14	0,32	0,00	1,21	0,06	0,07
aUnif-30	0,28	0,75	1,08	0,01	7,62	0,18	0,25
aUnif-40	0,49	2,27	3,58	0,03	38,60	0,53	0,83
aUnif-50	0,95	5,59	10,55	0,12	173,49	1,36	2,32
aUnif-100	26,63	278	839	1,04	-	18,24	54,61
aSum-20	0,04	0,17	0,33	0,00	1,20	0,06	0,07
aSum-30	0,29	0,77	1,08	0,01	7,48	0,19	0,25
aSum-40	0,50	2,12	3,43	0,03	37,10	0,55	0,83
aSum-50	0,95	5,65	10,74	0,12	168,24	1,40	2,27
aSum-100	27	282,46	819	1,75	-	17,40	55,63
aShuf-30	0,06	0,24	1,04	0,01	6,18	0,18	0,27
aShuf-40	0,16	0,84	3,73	0,04	26,86	0,49	0,74
aShuf-50	0,33	2,17	11,63	0,13	98,88	1,32	2,13
aShuf-100	17,7	353,5	1360	1,76	-	16,45	51,34

the difference is only due to (3) instead of (12), no difference was observe adding only (13). These results proves that LOP_SSB2 is in theory strictly tighter than LOP_ref, but with small and rare improvements.

5.3 Comparing Branch&Bound convergences

This section aims to compare the impact of modeling LOP with LOP_ref and LOP_SSB2, in the Branch&Bound (B&B) convergence. Table 5 analyzes the impact of Cplex cuts and heuristics at the root node, before branching in the B&B tree. If LOP_SSB2 improves slightly LP relaxation quality, the open question is to determine if additional variables and constraints help modern ILP solvers detecting other structures for cut generation, as in [4]. For LOP, computations at the root node of B&B tree are much slower with SSB2, coherently with the higher number of variables, but the efficiency of cuts and primal heuristics is significantly worse with the heavier SSB2 formulation. Having a larger ILP model, slower matrix operations for generation of cutting planes are needed by Cplex, and this stop earlier cuts that would have been generated using LOP_ref formulation, the size of ILP matrix is crucial here. Note also that Table 5 shows that few improvement of LP relaxation is provided at the root node of B&B tree, cuts are not very efficient to improve the LP relaxation, which was of a good quality. These elements explain the difference in the B&B convergence in one hour allowing branching, LOP_ref formulation is largely superior. For some instances with $N = 40$ or $N = 50$, LOP_ref can converge in ten minutes whereas a significant gap between lower and upper bounds remains after one hour for LOP_SSB2. This definitively validates the LOP_ref formulation as baseline ILP model for [2].

Table 5. Comparison of Lower Bounds (LB) and Upper Bounds (UB) of formulations LOP_ref and LOP_SSB2 after Cplex cuts and heuristics at the root node (i.e. before branching). Common UB with the LP relaxation are also provided for comparison.

	LP	UB	LB	time	UB	LB	time
	ref,SSB2	ref			SSB2		
aUnif-20	0,02%	0,00%	0,00 %	0,1	0,00 %	0,00 %	0,4
aUnif-30	0,17%	0,00%	0,00 %	1,5	0,09 %	0,50 %	14
aUnif-40	0,60%	0,44%	0,22 %	30,5	0,52 %	4,58 %	159
aUnif-50	1,12%	0,96%	0,82 %	212,9	0,98 %	5,27 %	1336
aUnif-100	3,17%	3,07%	5,49 %	3600	3,15 %	7,37 %	3600
aSum-20	0,05%	0,00%	0,00 %	0,12	0,00 %	0,00 %	0,55
aSum-30	0,31%	0,02%	0,02 %	2,0	0,16 %	0,79 %	20
aSum-40	1,18%	0,75%	0,32 %	64	0,95 %	5,08 %	296
aSum-50	2,25%	1,82%	0,95 %	297	1,95 %	6,72 %	989
aSum-100	4,97%	4,82%	6,87 %	3600	4,95 %	9,62 %	3600
aShuf-30	0,00%	0,00%	0,00 %	0,14	0,00 %	0,00 %	0,86
aShuf-40	0,00%	0,00%	0,00 %	0,37	0,00 %	0,00 %	2,9
aShuf-50	0,00%	0,00%	0,00 %	0,90	0,00 %	0,00 %	8
aShuf-100	0,02%	0,02%	0,02 %	477	0,02 %	6,02 %	3395

5.4 Variable Fixing heuristics

The excellent quality of the LP relaxation with LOP_ref formulation allows to use continuous solutions of LP relaxation to design primal heuristics as in [7]. Variable Fixing (VF) denotes here a heuristic reduction of the search space based on the LP relaxation, to set integer values to variables in the ILP resolution. One may use a VF preprocessing for variables with an integer value in the continuous relaxation, expecting that these integer decisions are good. Generally, it makes a difference to apply VF preprocessing on zeros and ones in the LP relaxation, as in [7]. There are in general many possibilities of VF preprocessing, considering also specific rules to select a subset of variable to fix [7].

For LOP, imposing $x_{i,j} = 1$ implies fixation $x_{j,i} = 0$ with constraints (2). Note also that constraints (3) may induce having continuous solution with variables $x_{i,j} = x_{j,k} = x_{k,i} = 2/3$, so that rounding to ones variables lower that 2/3 induce direct infeasibility on the corresponding (3) constraint. This property does not hold rounding to ones variables that are superior to 0.7 Hence, two VF strategies were implemented, on one hand fixing the integer value, and on the other hand considering the threshold for rounding to 0.8. Actually, there were slight differences for these two strategies. Experiments were also done using the quick MTZ relaxation for the LP relaxation, this was significantly degrading the performance of the VF heuristic.

Table 6 compares the gap to BKS and computation time using the VF preprocessing to LOP_ref formulation. For small and easy instances where LOP_ref gives optimal solutions, the degradation of the objective function is small with the VF heuristic, speeding up significantly the computation time. For the largest instances with $N = 100$, VF matheuristic is significantly better that the exact

resolution, illustrating the difficulty of the ILP solver to find good primal solutions with its primal heuristics. The primal solutions of matheuristic are in this case also significantly worse than the ones of LocalSolver, the VF speed up is not sufficient to reach an advanced phase of the B&B convergence.

Table 6. Comparison of gaps to BKS and computation time of Cplex in ILP solving using the reference formulation, without and with Variable Fixing (VF) preprocessing on integer values in the LP relaxation of the reference formulation. BKS are optimums for $N \leqslant 40$, for $N \geqslant 50$ BKS were given by LocalSolver

Instances	LB ref	time (sec)	LB ref + VF	time (sec)
aUnif-20	0,00 %	0,1	0,01 %	0,04
aUnif-30	0,00 %	1,5	0,04 %	0,62
aUnif-40	0,00 %	30,5	0,17 %	13
aUnif-100	5,49 %	3600	2,36 %	3600
aSum-20	0,00 %	0,13	0,05 %	0,06
aSum-30	0,00 %	2,1	0,09 %	0,77
aSum-40	0,00 %	63,5	0,43 %	12,7
aSum-100	6,87 %	3600	3,14 %	3600
aShuf-30	0,00 %	0,13	0,00 %	0,03
aShuf-40	0,00 %	0,37	0,00 %	0,08
aShuf-50	0,00 %	0,92	0,00 %	0,12
aShuf-100	0,00 %	1820	0,00 %	35,7

6 Conclusions and perspectives

If the reference ILP formulation seemed to be improvable using ATSP results, only a slightly tighter ILP formulation is obtained after this reformulation work. Analyzing the ILP convergence with a modern ILP solver shows that the LP relaxation is of an excellent quality with the reference formulation, but is fewly improved after. Also, primal heuristics are not efficient on the problem, a basic VF matheuristic improves significantly the primal solutions for difficult instances. Furthermore, this paper illustrates the graduated difficulty of instances, structured instances from the biological application as median of permutations are easier that random instances of LOP.

These results offer perspectives for the biological application also with the extension with ties [1]. Matheuristics can be used in this context, combined to specific reduction space operators related to the easier median of permutation instances [1, 13]. Perspectives are also to combine matheuristics and local search approaches which are efficient for the problem, as shown by LocalSolver benchmark on this study, and also by [8].

References

1. P. Andrieu, B. Brancotte, L. Bulteau, S. Cohen-Boulakia, A. Denise, A. Pierrot, and S. Vialette. Efficient, robust and effective rank aggregation for massive biological datasets. *Future Generation Computer Systems*, 2021.
2. B. Brancotte, B. Yang, G. Blin, S. Cohen Boulakia, A. Denise, and S. Hamel. Rank aggregation with ties: Experiments and analysis. *Proc. of the VLDB Endowment (PVLDB)*, 8(11):2051, Aug 2015.
3. S. Cohen-Boulakia, A. Denise, and S. Hamel. Using medians to generate consensus rankings for biological data. In *International Conference on Scientific and Statistical Database Management*, pages 73–90. Springer, 2011.
4. N. Dupin. Tighter MIP formulations for the discretised unit commitment problem with min-stop ramping constraints. *EURO Journal on Computational Optimization*, 5(1):149–176, 2017.
5. N. Dupin, R. Parize, and E. Talbi. Matheuristics and Column Generation for a Basic Technician Routing Problem. *Algorithms*, 14(11):313, 2021.
6. N. Dupin and E. Talbi. Machine learning-guided dual heuristics and new lower bounds for the refueling and maintenance planning problem of nuclear power plants. *Algorithms*, 13(8):185, 2020.
7. N. Dupin and E. Talbi. Parallel matheuristics for the discrete unit commitment problem with min-stop ramping constraints. *International Transactions in Operational Research*, 27(1):219–244, 2020.
8. C. Garcia, D. Pérez-Brito, V. Campos, and R. Martí. Variable neighborhood search for the linear ordering problem. *Computers & OR*, 33(12):3549–3565, 2006.
9. L. Gouveia and J. Pires. The asymmetric travelling salesman problem and a reformulation of the miller–tucker–zemlin constraints. *Euro J of Oper Res*, 112(1):134–146, 1999.
10. M. Grötschel, M. Jünger, and G. Reinelt. A cutting plane algorithm for the linear ordering problem. *Operations research*, 32(6):1195–1220, 1984.
11. M. Grötschel, M. Jünger, and G. Reinelt. Facets of the linear ordering polytope. *Mathematical programming*, 33(1):43–60, 1985.
12. C. Miller, A. Tucker, and R. Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM*, 7(4):326–329, 1960.
13. R. Milosz and S. Hamel. Space reduction constraints for the median of permutations problem. *Discrete Applied Mathematics*, 280:201–213, 2020.
14. T. Öncan, I. Altınel, and G. Laporte. A comparative analysis of several asymmetric traveling salesman problem formulations. *Computers & OR*, 36(3):637–654, 2009.
15. F. Peschiera, R. Dell, J. Royset, A. Haït, N. Dupin, and O. Battaïa. A novel solution approach with ML-based pseudo-cuts for the Flight and Maintenance Planning problem. *OR Spectrum*, 43(3):635–664, 2021.
16. S. Sarin, H. Sherali, and A. Bhootra. New tighter polynomial length formulations for the asymmetric traveling salesman problem with and without precedence constraints. *Operations research letters*, 33(1):62–70, 2005.
17. E. Talbi. Combining metaheuristics with mathematical programming, constraint programming and machine learning. *Annals of OR*, 240(1):171–215, 2016.
18. E. Talbi. Machine learning into metaheuristics: A survey and taxonomy. *ACM Computing Surveys (CSUR)*, 54(6):1–32, 2021.

Combination of Optimization and Machine Learning for Health-care problems: An exploratory Study in hospital sector

Oumayma Bahri and Lionel Amodeo

University of Technology of Troyes - UTT, 10000 Troyes, France.
oumayma.bahri@utt.fr lionel.amodeo@utt.fr

Abstract. The aim of this study is to analyze the literature gap between two different fields: Machine Learning ML and Optimization OPT in the health-Care sector. Previously, very few efforts have been made to examine the interaction between both fields in this sector. To this end, this paper first examines the existing approaches combining the two axes, then presents an overview of solutions proposed to solve health-care problems. Finally, an exploratory study is proposed to show the necessity and importance of sustainable and intelligent systems based on ML & OPT for hospital logistic problems.

Keywords: Optimization · Machine Learning · Health-Care · Logistic · Sustainable Transportation

1 Introduction

The increasing amount of data being gathered in health-care systems and the competitive context of our modern medicine, are leading research to new emerging challenges and directions. Not only is high health-care cost a challenge, many other important challenges have arisen such as the patient's treatment effectiveness and the care-workers quality of work life.

In order to address such complex issues, we observe today more and more research works which deal first with the integrative aspect (throughout the data collection and analysis until the problem modeling) using techniques derived from Artificial Intelligence (AI) and Machine Learning (ML); then which cover the paradigm of Optimization (OPT) to find the best possible outcomes.

Indeed, ML techniques have advanced rapidly and been successfully applied to deal with the increasingly available healthcare data. For instance,researchers believe using ML will reduce the high complexity of disease diagnosis and treatment outcome risk prediction. This complexity stems often from multiple data sources in the health-care environment (medical records, patient surveys and comments, genomic information, administrative databases, etc.); or for example real-time predictions in the event of anomaly or incident detection in the database. Besides, all these data need to be exact and very accurate to be practically taken into account for treatment. This is why ML techniques are

promising and have great potential to provide accurate predictions or discover new knowledge in healthcare.

On the other hand, research and investment in OPT have rapidly expanded in all the fields over the last decade. In fact, OPT is a family of algorithms that allocate resources while minimizing costs or maximizing benefits in the presence of diverse constraints. In the last decade, OPT is becoming more and more powerful to solve NP-hard health-care problems. For instance, an OPT method can be used to reduce scheduling errors by allocating efficiently nurses to specific surgical cases, to maximize time used in the operating room, to minimize the transportation costs of hospital operations, etc.

While the two domains having more evolved independently from each other, their classical methods have become increasingly inadequate with the recent data explosion. Then, to deal with the exponential increase of data volume and complexity, industry and research communities have paid a lot of attention to the synergy and interplay between ML and OPT. In the health care industry in particular, the interaction of two domains have been very little studied . This raises two questions: Why and when an OPT and ML intersection is necessary for solving decision-making problems? What are the impacts of such intersection on health care organizations ?

Motivated by the originality of such special topic, our main idea behind this work is to analyze the intersection between ML and OPT in a generic context, to further analyze the state-of-the-art regarding health-care problems and to present an our current exploratory study in this sector.

2 OPT and ML: difference and relationship

Over many years, ML and OPT fields have provided great theoretical insights, offered many successful methods and found practical applications in all areas of science. In fact, ML analysts prefer simpler algorithms that work in reasonable computational time for specific classes of problems. Thus, ML research advances and develops rapidly, which has made a lot of works applied in various popular fields such as image recognition, recommender systems and anomaly detection, natural language processing, etc. In turn, OPT researchers often address more complex (or NP-hard) problems by deriving the core resolution process and using specific optimizers to solve them. OPT methods and approaches have attracted much attention in almost all decision-making fields.

Today, in order to deal with the exponential increase of data volume and complexity, researchers have paid a lot of attention to the synergy and interplay between ML and OPT. Indeed, the two scientific disciplines are deeply interwoven: On one hand, OPT lies at the heart of ML in the sense that most ML problems, once formulated, can be solved as OPT problems. On the other hand, OPT concepts and methods equip ML researchers with tools for training large families of models. Besides, modern OPT algorithms are using ML theories and techniques to improve their efficiency. Recently, numerous research works have been focused on the developments of new techniques and tools inspired at the

same time by the ML principles and OPT concepts. In this context, we propose to give some classification and summary of the most popular studies, which can offer guidance and inspiration to contribute in both domains. Thus, we first suggest to distinguish two major types of research directions:

1. OPT for ML: The first question is how the combination with OPT can help ML researchers to rapidly develop new tools for more complex families of learning models? The second question is how to use OPT methods to solve specific ML problems? For instance, how to yield optimal estimations or predictions based on large or dynamic collected data?

2. ML for OPT: The question here is how to improve the OPT algorithmic structure, process and strategies by means of ML techniques? An interesting challenge involves performance measurements problems namely, how to use ML models to efficiently report and assess the results quality of a particular OPT algorithm?

In the following paragraphs, we seek to examine a brief state-of-the-art of existing approaches along the two axes.

2.1 OPT for ML

This section presents the recent advances in ML community, especially the studies that use OPT principles or methods to exploit novel ML branches. A wide range of works aimed at extending well-known OPT methods to create novel mixed learning models and paradigms [28]. For instance, [30] discussed the extensive use of mathematical programming methods in ML. In [9], convex optimization methods are applied for nonlinear kernel approximation or classification problems. In [16], constraint-based OPT methods are developed for incorporating domain knowledge into graphical learning models. In [20], parametric or hybrid OPT methods are used to find the optimal solution for simulation-based problems. Moreover, many researchers have discussed the role of some analytical methods of optimization in very popular fields like neural network (NN), reinforcement learning, meta learning, etc. They have shown that the development of optimization algorithms in specific ML fields can be inspiring to perform more informative learning [29]. For example, learning the parameters of complex NNs is one of the most well studied problems in the field of ML. [3] [18] have proposed an adaptive gradient-based methods for online NN learning. Unfortunately,the gradient descent scheme can result in poor learning training and performance in the case of NNs that have multiple hidden layers (i.e. deep networks). Thus, recent works have focused on the combination of new layer-wise training methods with stochastic gradient-based algorithms (i.e. which are first-order optimization approach) [10]. The results of such a combination lead to new general high-order optimizers which can efficiently learn deep models without any need for pre-training [26] [21]. Some problems are emerging when applying these adaptive optimizers. For example, the learning rate can be oscillating in the later training

stage, which may lead to non-converging problem. Other popular methods that have a significative influence on various ML fields are: optimization-based meta-learning methods, Adam optimization for image super resolution, trust-region optimization for deep reinforcement learning, etc [29].

2.2 ML for OPT

In turn, ML has highly motivated advances in the OPT community by allowing the develop of new effective methods. This original research axe presents modern algorithms including small changes in their underlying core process that enable high computing power. Then, in order to promote the development of OPT algorithms, a series of effective learning models were put forward, which have improved their performance and efficiency [14][32].

From the perspective of integrating the ML techniques into optimization methods, we suggest to divide research interests into three categories according to the chosen step integration [2]:

i) Before the model mathematical definition, ML techniques can be used to learn for example uncertain and missing data in order to incorporate adequate constraints into the model.
ii) During the selection and optimization process in order to enforce the optimality conditions and to converge to optimal solution(s) more efficiently.
iii) In the process of algorithms validation by paying attention to the characteristics of their parameters.

Many of the papers blend these different categories and novel methods have been successfully designed and applied to NP-hard problems [8] [33]. Recently, evolutionary machine learning (EML) took the interest of many researchers [7] since they have shown performance in many difficult problems. [14] published a survey about the use of statistics and machine learning for the distributed optimization. [15] reviewed existing literature on the combination of metaheuristics with machine learning methods and then introduces the concept of learn-heuristics, a novel type of hybrid algorithms. Recently, [32] proposed a novel taxonomy of data-driven meta-heuristics.

Several questions remain in this research track, particularly in the case of NP-difficult OPT problems. In the health field which is currently booming, combination of OPT and ML could help to produce more meaningful data and results for complex issues. The next section presents an overview of studied health-care problems in the literature and shows the impact of innovative solutions in this sector.

3 Health-care problems: A literature review

In the literature, previous research has concluded that major problems in the health-care sector are organizational and are related to patient safety, waiting

times and integration. [24] discussed for example the old modes of data collection/storage and the existing working practices to identify problems or needs in hospitals. Many other researchers [13] [6] proposed innovative solutions according to the organizational challenges. The findings of these studies are interesting in the sense of organizational conditions and requirements necessary for delivering health care. From a social viewpoint, there is a greater patient satisfaction.

Thereafter, many researchers [23] [5] have focused on the health-care logistics that have an important role in optimizing the cost and quality of public health. For instance, hospital logistic is considered as a complex optimization problem characterized by a diversity of activities, services, products, and a lot of internal and external information flows. Otherwise, logistic costs may affect more significantly the hospital expenditure.

With regard to the economic aspect, some studies [12] have attempted to reduce the operational costs by adopting new optimal strategies for medical staff who spent frequently time on indirect care activities often related to logistics. Other studies [25] have tried to analyse the competencies that are essential for effective management of logistics by applying a classification approach.

Most researches have concentrated on the transportation issue in order to deal with the complexity of logistic flows. In fact, high transportation costs can directly impact the logistics costs. In the literature, many different health-care routing problems have been treated [11] [4] [17]. More precisely, the aim of such OPT problems is to cope with various care demands, including different types of services or medical products, in different designed areas. Many constraints have been also considered such as the number of vehicles used, the occupancy degree, the number of pickup and delivery points, etc.

Although transportation is one of the most important sectors, it is often cited as barriers to health-care access. Such barriers may lead to missed or delayed care treatments, a lack of medication use, and consequently poorer management of chronic illness and heath outcomes. [31] have discussed and measured the impact that transportation interventions have on chronic disease care. More recently, authors paid attention to the environmental impact of transportation processes and logistics flows in health-care industry [27]. A great number of studies have proposed sustainable health-care design by including extended knowledge of operations and supply chain management [22], the automation of internal logistics to improve efficiency [19]. Today, intensive collaborative research and automobile manufacturers focus on improving and increasing investments in the innovative transportation of zero-emission vehicles in all health-care sectors.

To this end, the aim of our exploratory study is to analyze the impact of sustainable transportation on complex health-care problems and to identify new interesting research directions in this sector.

4 Exploratory Study

The problem we are addressing is first inspired by the daily challenges and logistical activities that health-care systems face, in particular, in hospital sector. One

of the major challenges is to deal with the complexity of managing a very diverse flow of materials from the hospital stocks to different care units and patients. Secondly, aware of the importance of economic, social and environmental challenges, we focus our attention on the issue of sustainable transport operations using automated and/or electric vehicles. Thanks to the undeniable benefits of this type of vehicle (safety, autonomy, time saving, etc.), its deployment in the health-care and hospital sector becomes essential.

Hence, the majority of existing works relating to sustainable routing problems are often dedicated to single-objective problems and with few management and environmental constraints. Comparing to theses works, we propose to resolve a problem variant reflecting more the complexity occurring in practice. We recently published a paper that addresses a problem of this nature [1].

Then, our motivation behind this exploratory study is to develop a sustainable decision-making system for hospital logistics while allowing a high quality care without exhausting limited natural resources. Then, the basic idea is to combine two problem categories: Hospital logistic management problems and Electric and automated routing problems. More precisely, this study arose from the question of fleet dimensioning for daily route planning at the hospital complex of Troyes (France). The hospital logistic network is composed of the daily transport of food trolley, linens (dirty and clean), pharmaceutical and medical products. In this context, a fleet of heterogeneous vehicles is used by a group of drivers every day to pick up these different equipments and products from one location and then to deliver them to another location. The fleet is heterogeneous in terms of capacity, fixed cost and fuel mileage. Besides, some services exploit a set of automated guided vehicles (AGV) to ensure automatic transport by trolleys of foodstuffs and products (meals, linen, pharmaceutical products, etc).

Main objectives for such problems are minimizing the total cost in terms of distance/time, minimizing the number of vehicles or maximizing the number of served care-services or patients. Many constraints can also be considered such as the number of drivers or vehicles used, the occupancy degree, the number of pickup and delivery points, the time of break or time of service, etc. Other issues can be related to electric vehicles like the recharging and discharging strategies.

Therefore, to address the complexity and diversity of this topic, important questions focus on:

- How to integrate all the data and constraints related to hospital issues and electric vehicle or automated technologies ?
- How to process real-time data from multiple sources which can be uncertain or with a lot of variability ?
- How to develop an optimization model considering the best compromises between conflicting objectives (in terms of cost, quality of service, autonomy, etc.) ?

Therefore, the main scientific purpose will be to develop an intelligent decision-making system for hospital logistics. We believe using ML and OPT techniques informed by all these data will improve the efficiency and inform better decision making.

The first phase will be to address the predictive and integrative aspects of data using techniques derived from ML, through the collection and analysis of real-time data, their interpretation and processing. Once this phase is completed, the objective is to design a mathematical programming model integrating the objectives and the different constraints for this type of problem (NP-hard). The next phase will be to develop a decision support system that covers the optimization and resolution of such problems in order to determine the best possible solution(s). Several questions remain on this subject, namely the choice of OPT and ML methods to be used, the level of integration of learning into the optimization process. Finally, this study opens the door to many interesting issues and perspectives in this active research area.

References

1. Bahri O., Talbi E.-G., Amodeo L.: Use of Electric Vehicles in Home Health Care Routing Problems : Analysis of a Multi-objective Approach under Uncertainty, 16th IFAC Symposium on Control in Transportation Systems CTS'21, Lille France, (2021)
2. Bahri O., Amodeo L.: Combination of Machine Learning and Optimization: A Survey of Recent Methods for Large Complex Problems, International Conference on Optimization and Learning OLA'21, Catania Italy (2021)
3. Amari, S. I., Park, H. and Fukumizu, K.: Adaptive method of realizing natural gradient learning for multilayer perceptrons. Neural computation, 12(6), 1399-1409, (2000)
4. Augusto, V., and Xie, X.: Redesigning pharmacy delivery processes of a health care complex. Health care management science, 12(2), pp. 166-178 (2009)
5. Ageron, B., Benzidia, S., and Bourlakis, M.: Healthcare Logistics and Supply Chain–Issues and Future Challenges, Supply Chain Forum: An International Journal 19 (1), pp. 1–3 (2018)
6. Azarm-Daigle, M., Kuziemsky, C. and Peyton, L.: A review of cross organizational healthcare data sharing, Procedia Computer Science, 63, pp. 425–432 (2015)
7. Al-Sahaf, H., Bi, Y., Chen, Q., et al.: A survey on evolutionary machine learning. Journal of the Royal Society of New Zealand, 49(2), 205-228, (2019)
8. Bennett, K. P. and Parrado-Hernandez, E.: The interplay of optimization and machine learning research. The Journal of Machine Learning Research, 7, 1265-1281, (2006)
9. Bubeck, S.: Theory of convex Optimization for Machine Learning. arXiv preprint arXiv. 15, 1405-4980, (2014)
10. Bengio, Y., Lamblin, P., Popovici, D. and Larochelle, H. Greedy layer-wise training of deep networks. Advances in neural information processing systems, 19, 153, (2007)
11. Beaudry, A., Laporte, G., Melo, T., Nickel, S.: Planning patient transports in hospitals: Proceedings of the Operational Research Applied to Health Services, Saint Etienne, pp. 15–20 (2007)
12. Belanger, V., Beaulieu, S. Landry, and P. Morales.: Where to Locate Medical Supplies in Nursing Units: An Exploratory Study. Supply Chain Forum: An International Journal, 19 (1), pp. 81–89 (2018)
13. Boyle, P. J., DuBose, E. R., Ellingson, S. J., Guinn, D. E., and McCurdy, D. B.: Organizational ethics in health care: Principles, cases, and practical solutions. John Wiley & Sons, 10 (2001)

14. Boyd, S., Parikh, N. and Chu, E.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Now Publishers Inc, (2011)
15. Calvet, L., de Armas, J., Masip, D. and Juan, A. A.: Learnheuristics: hybridizing metaheuristics with machine learning for optimization with dynamic inputs. Open Mathematics, 15(1), 261-280, (2017)
16. Ciravegna, G., Giannini, F., et al.: A Constraint-based Approach to Learning and Explanation. In Proceedings of the AAAI Conference on Artificial Intelligence, 34(4), 3658-3665, (2020)
17. Di Mascolo, M., Martinez, C., Espinouse, M.L.: Routing and scheduling in home health care: A literature survey and bibliometric analysis. Comput. Ind. 158, (2021)
18. Duchi, J., Hazan, E. and Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. Journal of machine learning research, 12(7), (2011)
19. Granlund, A. and Wiktorsson, M.: Automation in healthcare internal logistics: a case study on practice and potential. International Journal of Innovation and Technology Management, 10 (03) (2013)
20. Gray, G. A., Fowler, K. and Grifn, J. D.: Hybrid optimization schemes for simulation-based problems. Procedia Computer Science, 1(1), 1349-1357, (2010)
21. Hu, J., Jiang, B., et al.: Structured quasi-Newton methods for optimization with orthogonality constraints. SIAM Journal on Scientific Computing, 41(4), 2239-2269, (2019)
22. Kumar, A. and Rahman, S.,: RFID-enabled process reengineering of closed-loop supply chains in the healthcare industry of Singapore. Journal of Cleaner Production 85, pp. 382–394 (2014)
23. Landry, S. and Philippe, R.: How logistics can service healthcare. In Supply Chain Forum: An International Journal, 5(2), pp. 24–30 (2004)
24. Lee, T. H. and Mongan, J. J.: Chaos and organization in health care. MIT Press (2012)
25. Pillay, R.: Managerial Competencies of Hospital Managers in South Africa: A Survey of Managers in the Public and Private Sectors. Human Resources for Health, 6 (1), pp. 4 (2008)
26. Martens, J.: Deep learning via hessian-free optimization. In International Conference on Machine Learning, 27, 735-742, (2010)
27. Scaburi, A., Ferreira, J. C., and Steiner, M. T. A.: Sustainable Logistics: A Case Study of Vehicle Routing with Environmental Considerations, In International Business, Trade and Institutional Sustainability, Springer, Cham, pp. 765–779 (2020)
28. Sra, S., Nowozin, S. and Wright, S. J.: Optimization for machine learning. Mit Press, (2012)
29. Sun, S., Cao, Z., Zhu, H. and Zhao, J.: A survey of optimization methods from a machine learning perspective. IEEE transactions on cybernetics, 50(8), 3668-3681, (2019)
30. Scheinberg, K., Peng, J., et al.: Mathematical Programming in Machine Learning and Data Mining, (2007)
31. Solomon, E. M., Wing, H., Steiner, J. F., and Gottlieb, L. M.: Impact of transportation interventions on health care outcomes: A systematic review. Medical care, 58(4), pp. 384-391 (2020)
32. Talbi, E-G.: Machine Learning into Metaheuristics: A survey and taxonomy of data-driven metaheuristics . ACM Comput. Surv., hal-02745295. (2020)
33. Weichert, D., Link, P., Stoll, A., et al.: A review of machine learning for the optimization of production processes. The International Journal of Advanced Manufacturing Technology, 104(5), 1889-1902, (2019)

Mathematical modeling and optimal selection of renewable energy community based on different criteria

Hamza GRIBISS[1], MohammadMohsen AGHELINEJAD[1], and Farouk YALAOUI[1]

Laboratory of Computer Science and Digital Society (LIST3N)[1], University of Technology of Troyes, France
(hamza.gribiss, mohsen.aghelinejad, farouk.yalaoui)@utt.fr

Keywords : Energy community, MCDM, Renewable energy, self-consumption, Mathematical modeling.

1 Introduction

The use of fossil energy resources affects global warming. To tackle climate change, most energy consumers (e.g. industry, residential, transportation) are encouraged to play an active role by reducing their demands or producing energy. In this context, energy communities have discovered that more can be done in this regard by acting collectively [1]. Energy communities were created as local initiatives to accelerate the energy transition while providing economic, environmental, technical, and social benefits to their members. Today, more than 3500 energy communities have been constructed in the European Union [2].

Many researchers around the world addressed this concept deal with the study of energy self-consumption in energy communities. Few of these, tackled the comparison between the different configurations of the energy self-consumption based on different criteria [3, 4]. In this study, our objective is to propose different configurations and make a comparison.

The Multi-Criteria Decision Making (MCDM) methods have been applied to different types of energy problems during the past years. The advantage of these methods is that they allow the evaluation of multiple criteria, even contradictory sometimes. These methods have been applied to different domains, such as project selection and implementation of the installation of energy production. For example, Salameh et al [5] use the Preference by Similarity to Ideal Solution (TOPSIS) technique to choose from nine hybrid renewable energy solutions to meet power and hydrogen demands in a Saudi Arabian metropolis. Haaren and Fthenakis [6] used the MCDM method for selecting the wind farm site using spatial data and multiple criteria in New York.

This study uses three MCDM methods to compare different configurations of energy self-consumption in the energy community based on different criteria such as economic, environmental, technical, and social.

2 Problem definition

In this study, 16 different configurations for photovoltaic self-consumption in energy communities were compared according to the considered criteria. The objective is to find the configuration that gives better performance to the members of the energy community. This section presents these different configurations and criteria.

2.1 Configurations of self-consumption in the energy communities

An energy community typically consists of multiple independent energy consumers, each trying to maximize their own benefit. There are several configurations of energy self-consumption in energy communities that contain industrial companies. In this study, 16 different configurations divided into 5 categories were presented. The first category represents the case where each factory has its own photovoltaic production. The second category is the case where all factories have a shared photovoltaic production. The third category is a combination of categories 1 and 2. The fourth category and the last category are respectively the same cases as the first category and the third category, but with the addition of the energy exchange between the factories. For all these categories, the option of adding individual and collective storage has been studied. In total, 16

configurations were selected and modeled in order to find the one that gives the best results for the economic, environmental, technological, and social criteria. Figure 1 represents the schema of the configuration that includes individual and collective energy self-consumption, and individual and collective storage. In this configuration, the total demand of each factory j is satisfied by the energy sources available in the shared production, the factory j, and the grid at period t. In our previous work [7], we described the modeling and data generation details of this configuration. For example, we use the European Commission's Photovoltaic Geographic Information System (PVGIS) to calculate the energy produced by photovoltaic installations.

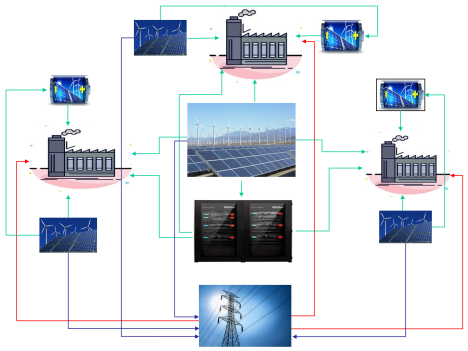


Fig. 1. Schema of global configuration for a case of three factories [7]

2.2 Criteria selection

This subsection aims to identify and classify the indicators used in this study to evaluate the economic, environmental, technical, and social criteria of energy communities.

In the design of energy communities, the economic impact is the most important factor. Economic goals are typically represented by a cost function that includes capital, operating, and maintenance costs over the lifetime of the energy system. The economic criterion used in this research is the one shown in equation 1. It enables the calculation of net present value in order to assess the profitability of a project investment over a given time period. With $R_1 = R_2 = ... = R_n$ is the annual cash flow at year n, N is the project lifetime, r is the discount rate, and IC_0 is the initial investment cost.

$$NPV = \sum_{n=1}^{N} \frac{R_n}{(1+r)^n} - IC_0 \tag{1}$$

The environmental criterion chosen in this study measures the CO_2 emission reductions. It utilizes the EI^{grid} $(KgCO2/KWh)$ emission factor for energy from the grid (E^{grid}) and the EI^{RE} $(KgCO2/KWh)$ emission factor for electricity produced by photovoltaic panels $(E^{RE}$). Equation 2 presents this criterion.

$$EI = EI^{grid} * E^{grid} + EI^{RE} * E^{RE} \tag{2}$$

To evaluate the technical criteria, the self-sufficiency ratio is used. It determines the proportion of an energy community's demand $(D_{total,EC})$ which is satisfied by renewable production $(E_{RE,EC})$. This rate of energy self-sufficiency in an energy community is calculated using Equation 3.

$$SSR = \frac{E_{RE,EC}}{D_{total,EC}} \tag{3}$$

For the social criterion, the number of new job opportunities created by the realization of an energy community was considered. This criterion may be a deciding element in REC investment acceptance. Equation 4 represents this criterion.

$$J_{op} = \frac{new\ jobs}{installed\ RE\ power} \tag{4}$$

3 Methodology (MCDM methods)

To compare the 16 configurations of energy self-consumption in energy communities, three MCDM method are used. The first method is Weighted Sum (WS), which is one of the simplest MCDM methods and the best-known [8]. The idea of this method is to calculate the overall performance of an alternative as a sum of the products of the normalized performance scores and the criteria weights. The best alternative is that has the first score. The second method is Weighted product (WP). It is similar to the WS, the main difference is that instead of addition in the main mathematical operation, there is now a multiplication. The third method is Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS), which is a practical and valuable technique for ranking and selecting some possible alternatives by measuring Euclidean distances [5]. It is a simple ranking method in its design and application.

4 Results and Conclusion

Each configuration was modeled and solved using 4 objectives : economic, environmental, technical, and social in order to obtain the results necessary to apply the MCDM methods. For the weighting of the criteria, 4 scenarios were used to study the correlation between the different criteria and to see the impact of changing the weights on the final result. The first scenario is the equality between the four criteria. The second, third, and fourth scenarios give importance to the economic, environmental, and social criteria, respectively. The 16 configurations of energy self-consumption are ranked according to the three MCDM methods and the four weight scenarios. It can be concluded that for all scenarios of the weights of the criteria and for three methods, the category that combines individual and collective energy self-consumption gives the best results compared to the other configurations.

This article compares 16 configurations of energy self-consumption in energy communities according to economic, environmental, technical, and social criteria. The models of these configurations were tested and compared with three MCDM methods. The results show that the category of combining individual and collective energy self-consumption is the one that gives the best results according to the different criteria.

References

1. Tineke Van Der Schoor and Bert Scholtens. Power to the people: Local community initiatives and the transition to sustainable energy. *Renewable and sustainable energy reviews*, 43:666–675, 2015.
2. Aura Caramizaru and Andreas Uihlein. *Energy communities: an overview of energy and social innovation*. Publications Office of the European Union, 2020.
3. Hassam ur Rehman, Francesco Reda, Satu Paiho, and Ala Hasan. Towards positive energy communities at high latitudes. *Energy conversion and management*, 196:175–195, 2019.
4. Vladimir Z Gjorgievski, Snezana Cundeva, and George E Georghiou. Social arrangements, technical designs and impacts of energy communities: A review. *Renewable Energy*, 2021.
5. Tareq Salameh, Enas Taha Sayed, Mohammad Ali Abdelkareem, AG Olabi, and Hegazy Rezk. Optimal selection and management of hybrid renewable energy system: Neom city as a case study. *Energy Conversion and Management*, 244:114434, 2021.
6. Klaas Van Alphen, Wilfried GJHM van Sark, and Marko P Hekkert. Renewable energy technologies in the maldives—determining the potential. *Renewable and Sustainable Energy Reviews*, 11(8):1650–1674, 2007.
7. Hamza Gribiss, MohammadMohsen Aghelinejad, and Farouk Yalaoui. Economic and environmental evaluations of pv installations for self-consumption in industrial energy communities. *10th IFAC Conference on Manufacturing Modelling, Management and Control.*, 2022. [Accepted].
8. Dragisa Stanujkic and Edmundas Kazimieras Zavadskas. A modified weighted sum method based on the decision-maker's preferred levels of performances. *Studies in Informatics and Control*, 24(4):461–470, 2015.

Design of a complete supply chain for the textile and clothing industry to improve the economic and environmental performance

EP. Mezatio[1,2,*], MM. Aghelinejad[1,**], L. Amodeo[1,***], and I. FERREIRA[2,+]

[1] University of Technology of Troyes, 12 Rue Marie Curie, 10300 Troyes
eric_papain.mezatio@utt.fr*,mohsen.aghelinejad@utt.fr**, lionel.amodeo@utt.fr***
[2] Institut français du textile et de l'habillement, 270 Rue du Faubourg Croncels, 10000 Troyes
iferreira@ifth.org+

1 Introduction

The new regulations to protect the environment have made manufacturing companies aware of the importance of adopting cleaner supply chain management practices. The apparel industry accounts for 60% of the global textile industry. According to studies, it is responsible for nearly 10% of greenhouse gas emissions worldwide [1]. To our knowledge, there is no article in the literature that deals with a direct and reverse supply chain, addressing economic and environmental issues together. The contributions of this paper are : (1) presentation of a more complete supply chain integrating all the actors of the direct and reverse logistic, with resource saving and reuse of the recycled materials. (2) The proposal of a new mixed integer linear programming model (MILP) integrating economic and environmental performance indicators, and a carbon pricing mechanism. (3) The design of a new benchmark for the textile and clothing industry based on several data from the existing literature.

1.1 Literature review

In the literature, the most recent works address supply chain management problems with a stronger focus on environmental aspects. [2], presented a direct supply chain based on a stochastic mixed integer linear programming model for the manufacture, which is used to assess the effects of life cycle carbon emissions on the supply chain. Although their work deals with the economic and environmental performance of the supply chain, they do not include the aspect of reverse logistics, which is an effective way to improve the uncontrolled consumption of virgin resources. In [3], the authors have proposed a mixed integer mathematical model to improve resource reuse and recovery in direct and reverse logistics. But in their conceptions of the forward and reverse supply chain, they consider separately the costs generated by the production process and the recycling process. Although they integrated aspects of forward and reverse logistics, their work is more focused on improving economic performance with low environmental implications. [4] proposed a mathematical model and a particle swarm optimization (PSO) algorithm for route planning in a forward and reverse logistics network for a fresh produce transport company, minimizing the carbon footprint. Their method is efficient on economic and environmental aspects, but does not take into account all the actors in the supply chain, and does not take into account the carbon life cycle. Thus, in the field of the textile and clothing industry, [1] proposes a mixed integer mathematical model to evaluate the economic, social and environmental contribution of recycling on textile waste management, but without integrating the reverse logistics aspect and carbon emissions.

Although all these works focus on better supply chain management, they do not sufficiently integrate environmental and recycling aspects. Moreover, each paper deals separately with carbon emissions, recycling and the coupling between direct and reverse logistics. However, the coupled integration of all aspects has a strong impact on the decision-making process related to supply chain management.

To this end, the integration of all levels and actors of the direct and reverse supply chain could improve the economic and environmental performance of the whole chain.

Given the scarcity of work in the literature dealing completely with direct and reverse logistics, and to our knowledge, there is no article that deals with a direct and reverse supply chain in

textile and clothing industry, addressing economic and environmental issues together. Hence, the importance of our current work, which has three objectives: (1) The presentation of a more complete direct and reverse logistics structure integrating all actors. (2) The proposal of a new mathematical optimization model integrating economic and environmental performance indicators, incorporating carbon life cycle analysis and a carbon pricing mechanism. (3) The design of a new benchmark for the textile and clothing industry based on several data from the existing literature for the evaluation of the different methods proposed.

2 Problem statement and mathematical modelling

The addressed problem consider the following assumptions : (1) All demands are deterministic and satisfied, (2) The variation of the carbon price is defined under several scenarios, (3) The capacity of all suppliers, manufacturers, warehouses, etc. is limited, (4) The model will decide on the selection of suppliers, subcontractors, the opening sites, warehouses, etc. (5) Model will also decide on the amounts of raw materials to be ordered, based on the carbon footprint and the amount of recycled second-hand raw materials. Figure 1 show the representation of considered supply chain.

2.1 Model formulation

To deal with this problem, a new mathematical model incorporating economic and environmental performance indicators is presented. In this model, the objective function found the optimal solution that minimizes the overall cost presented in expression (1).

$$Min \;\; Z = PC + TC + MC + RC + WC \tag{1}$$

- PC : represent the procurement and CO_2 emission cost for the production of raw materials.
- TC : represent the transportation and CO_2 emission cost induced by the transport activity during the supply.
- MC : represent the manufacturing, the opening and CO_2 emissions cost during the production process.
- RC : represent all costs generated by the processes of sorting, recycling and the costs of CO_2 emitted in the recycling process.
- WC : is for the variable storage cost for each product in all the different warehouses.

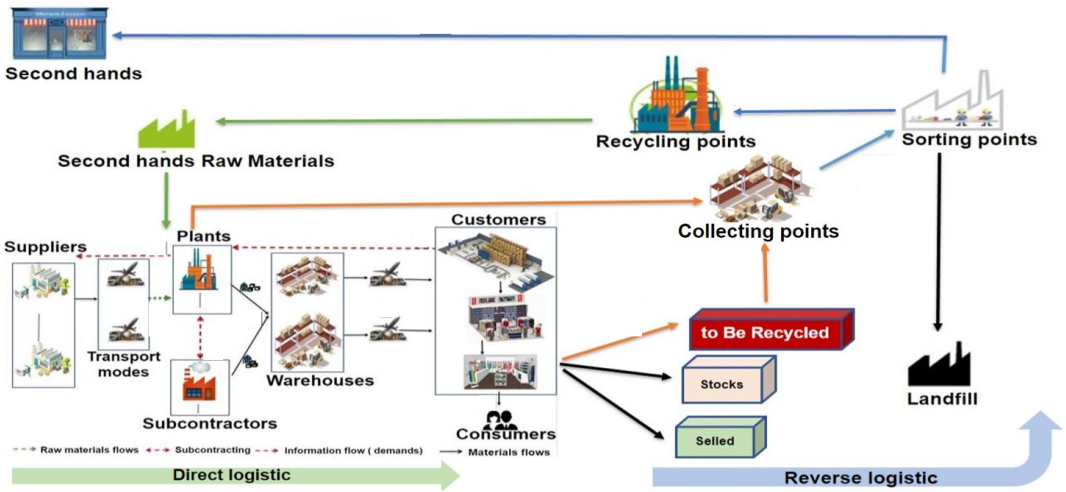


Fig. 1. Direct and Reverse logistics

2.2 Experimentation

The supply chain network for the experiment, consider 3 period (T=3) and includes 3 raw material suppliers (S=3), 2 production sites (F=2), 2 warehouses (W=2), 3 customers (N=3), 1 collection, sorting, and recycling center (L=1), 6 raw materials (R=6), 6 different products (P=6), 2 subcontractors (U=2) and 1 second-hand store (V=1). The demands of retailers are between 500 and 2000 per product units.

For solving the mathematical models, we used the Cplex 12.0 solver, with a Core i5 PC, frequency 2.3 GHz.

Table 1. Percentage increase in overall cost VS reduction in carbon emissions

Scenarios	Sc_0	Sc_1	Sc_2	Sc_3	Sc_4	Sc_6	Sc_7
$Carbon\_prices(euro/ton_CO_2)$	0	30	40	50	60	70	80
$Overall\_cost(\%)$	-	+3.4	+6.8	+9.1	+12.5	+15.4	+17.81
$Carbon\_Emission(\%)$	-	-23.6	-26.5	-29.7	-33.8	-35.2	-38.9

Table 1, present the impact of changes in the carbon price on the overall supply chain cost, in percentage. In table 1, the carbon price have a highly correlated variation in supply chain costs for each carbon price scenario (about 3.5% per variation).

The table shows a direct impact of carbon pricing on the reduction of carbon emissions for each scenario on the supply chain ($Carbon_Emission(\%)$). This reduction can be justified by the new choice of low-carbon, low-cost suppliers, subcontractors, and transportation modes. It is also justified by the use of second-hand raw materials from recycling, which have a carbon life cycle half as polluting as virgin material.

For example, for a change in $Carbon_prices$ between 0 and 30 euro/ton of CO_2 emission, there is a 23.6% decrease in emissions and a 3.4% increase in overall cost. Similarly, for a carbon price of 80, the overall cost increases by 3.6% compared to the carbon cost of 70; while for the same price variation, there is a carbon reduction of 3.7%.

3 Conclusion and discussion

This article has addressed the problem of direct and reverse supply chain in the textile and clothing industry, considering both economic and environmental issues. Firstly, this work presented a closed-loop reverse logistics structure. Secondly, a new mixed integer linear programming model integrating economic and environmental performance indicators is developed. Finally, a new benchmark for the textile and apparel industry based on Figure 1 will be designed to analyze the model.

The aim of this study is to shows that taking into account the environmental aspects and the recycling of textile waste in the direct and reverse supply chain, would strongly influence the behavior of the actors of the supply chain. This will act at two levels: at the supply chain configuration level. The second level concerns the economic impact, by reinjecting the recycled raw material at the production sites in order to reduce the quantity of raw material to be ordered, and the consumption of virgin materials and the environmental impacts.

References

1. Cuc, Sunhilde, and Milorad Vidovic. "Environmental sustainability through clothing recycling." Operations and Supply Chain Management: An International Journal 4.2 (2014): 108-115.
2. Ren, Hongtao, et al. "Incorporation of life cycle emissions and carbon price uncertainty into the supply chain network management of PVC production." Annals of Operations Research 300.2 (2021): 601-620.
3. Fu, Rong, et al. "Closed-loop supply chain network with interaction of forward and reverse logistics." Sustainable Production and Consumption 27 (2021): 737-752.
4. Guo, Jianquan, et al. "Forward and reverse logistics network and route planning under the environment of low-carbon emissions: A case study of Shanghai fresh food E-commerce enterprises." Computers Industrial Engineering 106 (2017): 351-360.

SHAMan: a versatile auto-tuning framework for costly and noisy HPC systems

S. Robert[1], S. Zertal[2], and P. Couvée[1]

[1] Atos BDS RD Data Management
sophie.robert@atos.net
philippe.couvee@atos.net
[2] University of UPSacaly-UVSQ
soraya.zertal@uvsq.fr

1 Introduction

Most of the software of modern computer systems come with many configurable parameters that control the system's behavior and its interaction with the underlying hardware. These parameters are challenging to tune by solely relying on field insight and user expertise, due to huge spaces and complex, non-linear system behavior. Besides, the optimal configuration often depends on the current workload, and parameters must be changed at each environment variations. Consequently, users often have to rely on the default parameters given by the provider, and do not take advantage of the possible performance with a more appropriate parametrization of their tuned system. The more complex these systems are, the more important the tuning becomes, as the components interact with each other in ways that are hard to grasp by the human mind. As architecture becomes more and more service oriented, the number of components per system increases exponentially, along with the number of tunable parameters. This problem is particularly observed within HPC systems with hundreds of devices assembled to build very complex and highly configurable supercomputers. As performance is the major concern in this field, each component must be finely tuned, which is almost impossible to achieve solely through field expertise. An additional constraint is the often noisy setting, because making resources exclusive is expensive and goes against the current highly shared programmming environments. Automatic tuning methods thus must need to take into account this possible interference on the tuned application, which degrades the performance of classical auto-tuning heuristics. Faced with the inability of relying solely on users to take adequate decisions for the parametrization of complex computer systems, new tuning methods have emerged from various computer science communities to automate parameter selection depending on the current workload. Because they do not require any human intervention, these approaches are commonly called *auto-tuning* methods, a term which encompasses a broad range of methods related to the optimization and machine learning field. Throughout the years, they have been successfully applied to a wide range of systems, such as storage systems, database management systems and compilers.

In this paper, we introduce a new Open Source software, called SHAMan (**S**mart **H**PC **A**pplication **Man**ager), which provides an out-of-the-box Web application to perform black-box auto-tuning of custom computer components running on a distributed system, for an application submitted by the user. The framework integrates three state-of-the-art black-box optimization heuristics, as well as resampling-based noise reduction strategies to deal with the interference of shared resources, and pruning strategies to limit the time spent by the optimization process. It is to our knowledge the only generalistic optimization framework specifically tailored to find the optimum parameters of configurable HPC systems, by taking into account their specific constraints.

This paper is organized as follows. We introduce in section 2 the works related to ours, and discuss the improvements provided by SHAMan compared to the state-of-the-art. Section 3 introduces the theoretical context of black-box optimization for noisy and expensive systems. In section 4 we present the different features of SHAMan and its software architecture. In section 5, we present the advantages of using SHAMan on three use-cases by tuning two different I/O accelerators and OpenMPI collectives. Finally, section 6 concludes the paper and gives insights into planned further works.

2 Related works and software

Within the HPC community, auto-tuning has gained a lot of attention for tuning particular HPC application and improve their portability across architectures [13]. Seymour et al. [35] and Kni-

jnenburg et al. [22] provide a comparison of several random-based heuristic searches (Simulated annealing, genetic algorithms ...) that have provided some good results when used for code auto-tuning. Menon et al. [25] use Bayesian Optimization and suggest the framework HiPerBOT to tune application parameters as well as compiler runtime settings. HPC systems energy consumption can also benefit from Bayesian Optimization, as Miyazaki et al. have shown in [27] that an auto-tuner based on a combination of Gaussian Process regression and the Expected Improvement acquisition function has raised their cluster to the Green500 list. The MPI community has also shown the superiority of a hill-climbing black-box algorithm over an exhaustive sampling of the parametric space in [19] and [41].

In terms of auto-tuning frameworks, several have been proposed recently in different domain where optimization is required. The Machine Learning community has proposed several frameworks to find the parameters that return the best prediction scores for a given model and dataset. Among the most popular frameworks, we can cite Optuna [11], which relies on Tree Parzen Estimators to perform the optimization. Autotune [23] is an other framework which supports several black-box optimization techniques. Scikit-Optimize [8] which supports a wide range of surrogate modeling techniques. The SHERPA [20] library provides different optimization algorithms with the possibility to add new ones. It also comes with a back-end database and a small Web Interface for experiment visualization. GPyOpt [12] is another library for users who wish to use Bayesian Optimization. Finally, the framework TPOT [24] relies on genetic algorithms for the optimization of Machine Learning pipelines. Within the MPI community, the two most famous commercical implementations come with their own tuning tool: OPTO [14] is the standard tool used by the Open MPI community for tuning MCA parameters, and similarly mpitune [5] from Intel MPI. These methods only include exhaustive grid search, making these tools slow to use for tuning expensive HPC applications. The main drawbacks identified with these already existing frameworks concern the difficulty of integrating these libraries for purpose other than the ones they were designed for by using them for HPC tuning. It is also difficult to enhance them with other optimization techniques, such as noise reduction strategies. In addition, none of them offer a satisfying Web Interface allowing an easy manipulation of the software.

Faced with the highlighted deficiencies of the existing solutions, we have developed our own framework, and provide these main contributions and features:

(a) **Versatility**: it can handle a wide range of use-cases, and new components can be registered through a generalist configuration file.
(b) **Accessibility**: the optimization engine is accessible through a Web Interface.
(c) **Optimization diversity**: as different heuristics work differently for different systems, our framework provides several state-of-the-art heuristics.
(d) **Easy extention**: the optimization engine uses a plug-in architecture and the development of the heuristic is thus the only development cost.
(e) **Integration within the hpc ecosystem**: the framework relies on the Slurm workload manager [21] to run hpc applications. The microservice architecture enables it to have no concurrent interactions with the cluster and the application itself. It is not intrusive allowing users to launch applications on their own. Also, no privileged rights are required to use the software.
(f) **Customizable target measure**: the optimized target function can be defined on a case-per-case basis to allow the optimization of various metrics.
(g) **Integration of noise reduction strategies**: because of the highly dynamic nature and the complexity of applications and software stacks, running in parallel on shared ressources are subject to many interference, which results in a different performance measure for each run even with the same system's parametrization. Noise reduction strategies are included in the framework to perform well even in the case of strong interference.
(h) **Integration of pruning strategies**: runs with unsuited parametrization are aborted, to speed-up the convergence process.

3 Theoretical background

SHAMan relies on black-box optimization, which consists in treating the tuned system as a black-box, deriving insight only from the relationship between the input and the output parameters, as described by the optimization loop in figure 1.

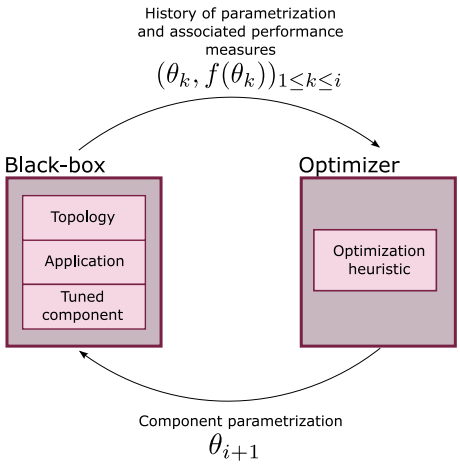


Fig. 1: Schematic representation of the optimization loop, without

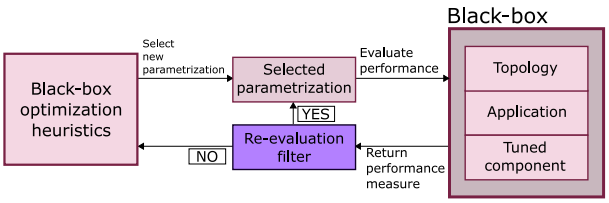


Fig. 2: Schematic representation of the optimization loop, with noise reduction through resampling

3.1 An overview of the optimization loop

Black-box optimization refers to the optimization of a function f with unknown properties in a minimum of evaluations, without making any assumption on the function. The only available information is the history of the black-box function, which consists in the previously evaluated parameters and their corresponding objective value. Given a budget of n iterations, the problem can be transcribed as:

$$Find\{p_i\}_{1 \leq i \leq n} \in P \ s.t. \ \mid min(f(p_i)_{1 \leq i \leq n}) - min(f) \leq \epsilon \mid \quad (1)$$

- f the function to optimize
- P the parameter space
- ϵ a convergence criterion between the found and the estimated minimum

Every black-box optimization process starts with the selection of the initial parameters for the algorithm. An acceptable initialization starting plan should respect two properties [18] [40]: the space's constraints and the non-collapsible property. The space constraints are shaped by the possible values that can be taken by the parameters. The non-collapsible property specifies that no parametrization can have the same value on any dimension. A design plan respecting this constraint is called a *Latin Hypercube Design* (LHD)[18]. The next step consists in a feedback loop, which iteratively selects a parametrization, evaluates the black-box function at this point and selects accordingly the next data point to evaluate. The higher procedure for searching an optimal solution in a parametric space is called an optimization *heuristic*. There are many black-box heuristics, and we have implemented the following set in our optimization engine because of their simplicity of implementation and their proven efficiency for HPC systems' tuning. A detailed motivation and description of our implementation can be found in [30] and [31].

- **Surrogate models**: Surrogate modeling consists in using a regression or interpolation technique over the currently known surface to build a computationally cheap approximation of the black-box function and to then select the most promising data point in terms of performance on this surrogate function by using an acquisition function.
- **Simulated annealing**: the simulated annealing heuristic is a hill-climbing algorithm which can probabilistically accept a solution worse than the current one.
- **Genetic algorithms**: Genetic algorithms consist in selecting a subset of parameters, among the already tested parametrizations, considering the objective value of each parametrization, and then combining them to create a new parametrization.

3.2 Stop criteria

When running the optimization algorithm, the easiest stop criterion is based either on a budget of possible steps (exhaustion based) or on a time-out based on the maximum elapsed time for the

algorithm. Once the iteration budget has been spent, the algorithm stops and returns the best found parametrization. However, while very simple to implement, this criterion can be inefficient, as it has no adaptive quality on the tuned system. Using other stop criterion to speed-up the convergence process, while still providing a good solution is thus essential for tuning expensive systems, and SHAMan integrates two different stop criteria:

Exhaustion based criteria Exhaustion based criteria are criteria based on the number of allowed iterations performed by the heuristic. Once all of the possible iterations have been tried by the algorithm, the algorithm stops and the parametrization which returned the best corresponding performance measure is kept. They are the most popular in the black-box optimization literature [44] because of their simplicity of implementation and the control they give over the optimization process. However, they can be a waste of resource because they can either:

- Stop the algorithm while the maximum optimization potential has not been reached, thus not finding the optimal parametrization. The algorithm either has to be started from scratch or can resume, depending on the practical auto-tuning implementation.
- Keep the algorithm running even though the maximum potential of optimization has already been reached. This is a waste of time and resources, as the algorithm runs aimlessly without providing any improvement.

Exhaustion-based criteria thus do not provide much flexibility in the optimization process and do not have any adaptive qualities to the behavior of the system. Because of this, SHAMan integrates two other criteria based either on the value of the performance function or the value of the tested parameters:

Improvement based criteria They consist in stopping the optimization process if it does not bring any improvement over a given number of iterations. Depending on the target behavior, the improvement can either be measured globally as the average of the evaluated values or locally as the change in optimum values.

- **Best improvement**: Improvement of the best objective function value is below a threshold t for a number of iterations g.
- **Average improvement**: Improvement of the average objective function value is below a threshold t for a number of iterations g.
- **Median improvement**: Improvement of the median objective function value is below a threshold t for a number of iterations g.

Movement based criteria Movement based criteria consider the movement of the parametric grid as a criteria to stop the optimization. Two variations of the criteria are available in our framework:

- **Count based**: The optimization algorithm is stopped once there is less than t different parametrization evaluated over a number of iterations g.
- **Distance based**: The optimization algorithm is stopped once the distance between each parametrization goes below a certain threshold t for a number of iterations g.

3.3 Resampling for noisy systems

Resampling consists in adding a "resampling filter" by using a set logical rule to select which parametrization to reevaluate. A detailed schematic representation of the integration of resampling within the black-box optimization tuning loop is available in figure 2. The general goal of resampling is to reduce the standard deviation of the mean of an objective value in order to augment the knowledge of the impact of the parameter on the performance.

Algorithmically, we define a resampling filter as a function $\mathcal{RF}$ which takes as input an optimization's trajectory already evaluated fitness and corresponding parameters $(\theta_i \in \Theta, F(\theta_i) \in R)_k$, for the optimization trajectory at step k, and outputs a boolean on whether or not the last parametrization should be re-evaluated. This filter can be integrated for both initialization draws and exploitation draws, or only for exploitation ones. We make the latter choice, as we want to keep the initialization draw to test as many parametrization as possible, and if needed, let the algorithm come back to these parametrization for further investigation. Resampling is a trade-off between having a better knowledge of the space and waste some computing times on re-evaluation.

We present here two of the most popular resampling algorithms in order to efficiently reevaluate a parametrization and a more exhaustive description is proposed in [36]. Three noise strategies are available in the framework:

- **Static resampling**: re-evaluates each parametrization for a fixed number of iterations.
- **Standard Error Dynamic Resampling** [17]: re-evaluates the current parametrization until the standard error of the mean falls below a set threshold.
- **An improved noise reduction algorithm**: a complex decision algorithm for re-evaluating the current parametrization [31].

3.4 Pruning of expensive systems

Pruning strategies consist in stopping some runs early because their parametrization is unpromising, compared to already tested parameters. Two pruning strategies are available in the framework:

- **Default based**: It consists in stopping every run that takes longer than the execution time corresponding to the default parametrization.
- **Estimator based**: It consists in stopping every run that takes longer than the value of an estimator computed on previous runs. For example, if the selected estimator is the median, the current run is stopped if its elapsed time takes longer than 50% of the already tested parametrization. This pruning only applies to runs performed after the initialization ones.

4 Software architecture and features

SHAMan (Smart HPC Applications Manager) performs the auto-tuning loop by parametrizing the component, submitting the job through the Slurm workload manager, and getting the corresponding execution time Using the combination of the history (parametrization and execution time) to select the next most appropriate parametrization until the stop criterion is reached.

4.1 Terminology

Throughout this section, we will use the following terms:

- **Component**: the configurable component which optimum parameters must be found.
- **Target value**: the measurement that needs to be optimized.
- **Parametric grid**: the possible parametrization defined as (minimum, maximum, step value).
- **Application**: a program that can be run on the clusters' nodes through Slurm to be tuned.
- **Budget**: the maximum number of evaluations to find the optimum value.
- **Experiment**: A combination of a component, a target value, an application and a parametrized black-box optimization algorithm that will output the best parametrization for the application and the component.

4.2 Optimization and vizualization procedure

The main features of SHAMan are the possibility to:

Declare a new configurable component and register it for later optimization Running the command `shaman-install` with a YAML file describing a component registers it to the application and makes it possible to be optimized. This file must describe how the component is launched and declares its different parameters and how they must be used to parametrize it. After the installation process, the components are available for optimization in the launch menu.

Design and launch an experiment through a Web interface or through a command line interface The main way is to launch the experiment through the Web interface via the menu. The user has to configure the black-box by:

1. Writing an application according to Slurm sbatch format.
2. Selecting the component and the parametric grid through the radio buttons.
3. Configuring the optimization heuristic, chosen freely among available ones. Resampling parametrization, stop criterion and pruning strategies can also be activated.
4. Selecting a maximum number of iterations and the name of the experiment.

The optimization process will begin to run and its information will be available in the exploring section of the Web application.
Another way to use SHAMan is to use it directly through a command line interface. It allows more flexibility of the different features and requires the same information as the Web interface.

Visualize data and results of finished or running experiments After the submission, the evolution of the running experiments can be visualized in real-time. The optimization trajectory is available through a display of the different tested parameters and the corresponding execution time, as well as the improvement brought by the best parametrization. The other metadata of the experiment are also available through side menus. Figure 3 and 4 show the tunes performance without any aggregation, then with it if the noise reduction is enabled.

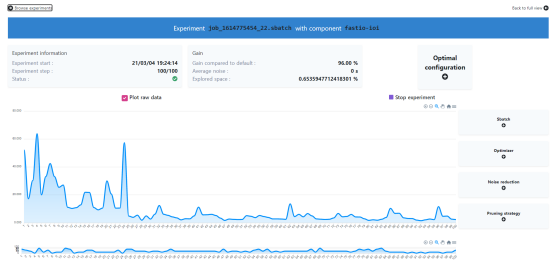


Fig. 3: Visualization of an optimization trajectory

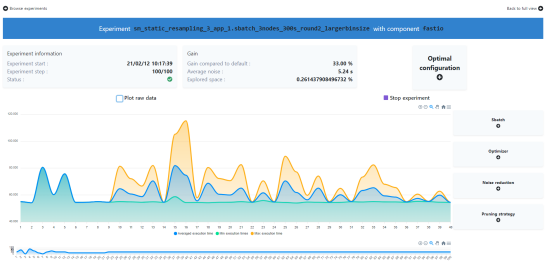


Fig. 4: Visualization of an optimization trajectory when noise reduction is enabled

4.3 Software architecture

The architecture of SHAMan relies on microservices, as can be seen in figure 5 and detailed in [32]. It is composed of several services, which can each be deployed independently:

- An optimization engine which performs the optimization tasks.
- A front-end Web application
- A back-end storage database
- A rest api enabling communications of all the services

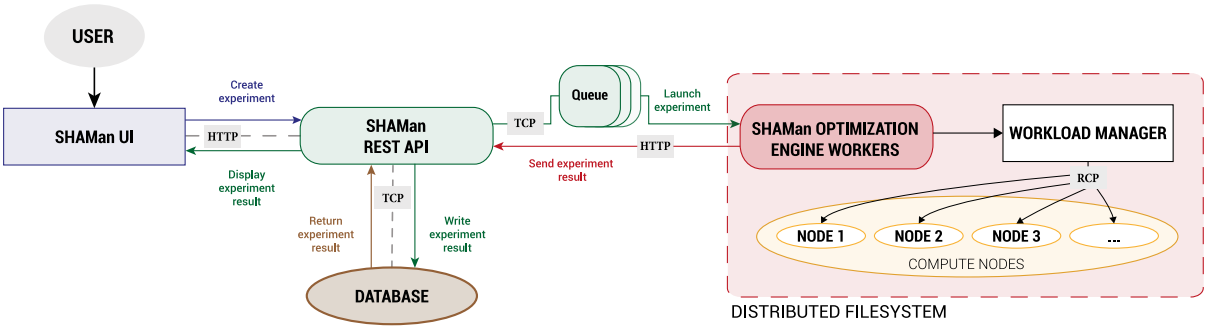


Fig. 5: General architecture of the tuning framework

4.4 Implementation choices

SHAMan uses *Nuxt.js* as a frontend framework. The optimization engine is written in Python. The database relies on the NoSQL database management system *MongoDB*. The message broker system uses Redis [16] as a queuing system, manipulated with the ARQ Python library [1]. The API is developed in Python, using the FastAPI framework. The framework is fully tested, can be fully deployed as a stack of Docker containers [26]. The code is available on Github [9] and thoroughly documented [3].

5 Use-cases and results

In some of our previous works, we have shown the efficiency of SHAMan on two different use-cases belonging to I/O accelerators: a smart prefetching strategy and a burst buffer [33] [34]. To further prove the versatility of SHAMan, we tackle another difficult to tune HPC component: MPI

collectives. We begin this section by summarizing the main results of our previous experiments, and then introduce the new results provided by our experiment on MPI collectives.

5.1 I/O accelerators

I/O accelerators are software or hardware components which aim is to reduce the increasing performance gap between compute nodes and storage nodes, which can slow down I/O intensive applications. Indeed, on large supercomputers, the many compute nodes performing reads or writes can stress the storage bay and make the application wait while it performs its I/O, generating *I/O bottlenecks*. This is especially true for HPC applications that periodically save their current state (by performing *checkpoints*) which causes many writes during a short timeframe. The link between the compute and the storage node can become saturated, which slows down the application. To mitigate these problems, several I/O accelerators have been developed over the years, and we focused specifically with SHAMan on tuning two commercial implementations of I/O accelerators: a pure software one called smart read optimizer [10] and a mix of software and hardware one called smart burst buffer [2]. Because these I/O accelerators come with many parameters, they are difficult to tune, and operate very differently which make them good use-cases to demonstrate the versatility of SHAMan and black-box optimization.

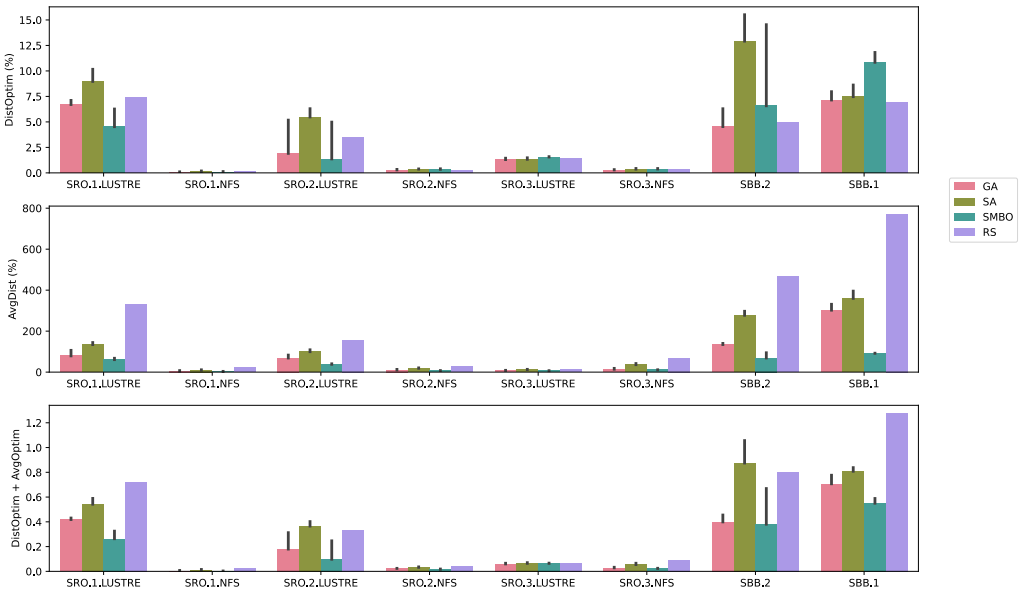


Fig. 6: Best values for DistOptim, AvgDist and the sum of both for every heuristic

For both I/O accelerators, we performed a comparative study of the impact of each black-box optimization heuristic with SHAMan and showed that surrogate models offer the best trade-off between optimization quality and stability of the trajectory, and outperforms by far a random sampler.

As displayed in figure 6, we show that with a distance to the true minimum inferior to 4% for every application, our auto-tuner exhibits good convergence properties. As we have found convergence rate inferior to 40 steps for reaching 5% of the optimal value, we have also demonstrated that the auto-tuner can operate in a sparse production environment for expensive systems.

5.2 Tuning MPI collectives

MPI is a standardized and portable message-passing standard designed to function on parallel computing architectures through many implementations as Open MPI [6] and MPICH [4]. Its

collective operations provide a standardized interface for performing data movements within groups of processes and come with several tunable parameters. The optimal configuration greatly depends on the size of the transmitted message [29], as well as the architecture and the topology of the target platform [43], and the default parametrization is not adapted for a wide range of cases. In particular, the Open MPI [6] implementation features a modular architecture and the selection of modules along with their parametrization is achieved through MCA parameters, which can be provided using either configuration files, command-line arguments or environment variables. This implementation is the one we will be focusing on, by considering the subset of parameters related to the `coll_tuned` component that allow to dynamically set: (1) **the algorithm**; (2) **fan-in/fan-out**; (3) the **segment size**. The main reason for choosing these parameters is that they have been confirmed as having the most impact in several previous studies [38].

The importance of tuning MPI collectives The importance of this tuning challenge is well known across the MPI community and several studies confirm and further develop the results of our own study: the optimal collective parametrization depends on many factors, such as the physical topology of the system, the number of processes involved, the sizes of the message, as well as the location of the root node [29][39]. An especially thorough analysis of the performance gap between the default parametrization and the optimum parametrization found by exhaustive search for the MPICH implementation is available in [42], and the importance of choosing the right algorithm to perform the collective operations is emphasized in [28]. Using parametrization adapted to the message size and the collective is thus necessary to maximize system's performance. The easiest solution for tuning is to rely on brute force, *i.e.* testing every single possible parametrization and running the benchmark with this parametrization, but this can cause the tuning process to go up to several days in time. Brute force is thus impractical on a production system, as it requires too many computing resources and user time.
All these reasons show the relevance of using black-box optimization through SHAMan: finding the optimal configuration of MPI collective is crucial for the performance of the system as the default parametrization is unsuitable for many communication problems, but exhaustive search is a very impractical way of finding it.

Experiment plan For the purpose of demonstrating the efficiency of SHAMan in the case of MPI tuning, we have selected a subset of 4 blocking collectives to tune amongst the most used ones in HPC applications [38] [15], and to cover all communication patterns (one-to-all, all-to-one, all-to-all): (1) **Broadcast**, (2) **Gather**, (3) **Reduce** and (4) **Allreduce**.

The tuning is performed using the OSU MPI microbenchmark suite [7], which provides tests for every collective operation. For each of the tuned collective and each tested size, we use the corresponding benchmark in the suite. To ensure stability and reduce the noise when collecting execution times, the OSU benchmark was parameterized to perform 200 warmup runs before performing the actual test. The tuned message size range from 4KB to 1MB, with a multiplicative step of 2. Two hardware configurations are selected using the 12 nodes of a cluster, to emulate two of the most common types of process placements encountered in HPC applications: (1) **One MPI process per node**, for a total of 12 MPI processes, as is typical of hybrid applications relying on MPI for inter-node communications and on OpenMP for their implicit, intra-node communications ; (2) **One MPI process per core**, for a total of 576 MPI processes, 48 MPI processes per node, which is typical of pure, MPI-only applications which rely on the MPI library for all their communications (inter-node and intra-node alike). This results in a total of 160 SHAMan optimization experiments (4 collectives, 20 sizes and 2 different topologies). The performance metric for tuning is the time elapsed by the benchmark for the selected size of operation, as output by the OSU benchmark.

To provide a thorough analysis of the advantage of black-box optimization compared to exhaustive search, the reference execution time is first computed using the default parametrization which is run 100 times to account for possible noise in the collected execution time. An exhaustive sampling of the parametric space is then performed to select the parametrization with the minimal execution time as the optimal one, which acts as the ground truth. This ground truth is also run 100 times for noise mitigation.

SHAMan parametrization The tuning is then performed with SHAMan, using Bayesian Optimization specifically configured for MPI. The initialization plan is composed of 10 parametrization. The selected maximum number of iterations is set to 150 and the stop criterion is improvement

based: we choose to stop the optimization process if the best execution time over the last 15 iterations is less than 1% better than the currently found minimum. The best parametrization found by the optimization process is considered to be the best parametrization found by SHAMan and is also run one hundred times to account for noise.

Main performance gains

Improvement compared to default The first important result is the gain brought by using the auto-tuner rather than the default parametrization, which is represented in figure 7. Over all experiments, we find an average improvement of 48.4% (52.8% in median), using the best parametrization found with Bayesian Optimization. We find an average improvement of 38.42% (29% in median) for experiments with one mpi process per node and of 58.9% (65.3% in median) when using one mpi process per core, highlighting the efficiency of tuning the Open MPI parametrization instead of simply relying on the default parametrization.

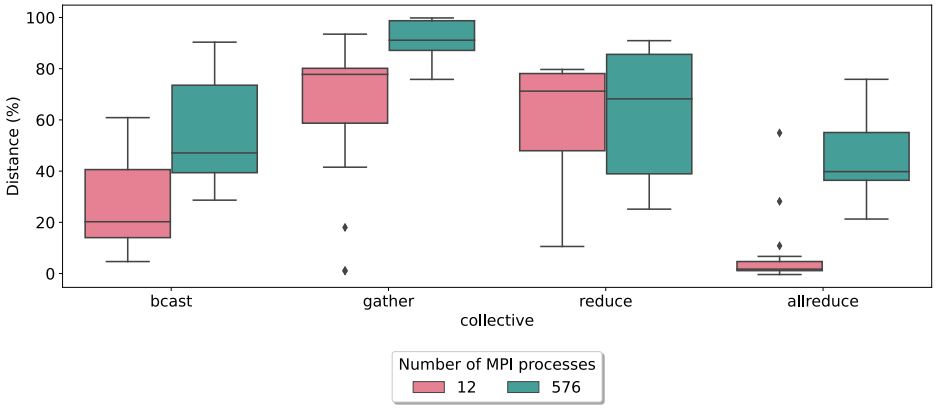


Fig. 7: Performance gain with Bayesian Optimization compared to the default parametrization

The time gain brought by SHAMan varies depending on the tuned collective, as the default parametrization is more adapted than others for some collectives. It is the case for the *allreduce* collective when running one MPI process per node, where the optimum parametrization provides a median improvement of 0.9% (1.8% on average). Other collectives have a default parametrization that is not adapted at all. It is for example the case of the gather collective with one MPI process per core, where we see an improvement of 91% in median and on average. The improvement compared to the default parametrization is strongly dependent on each evaluated parameter: message size, number of processes per node or collectives and is difficult to predict. This highlights the importance of tuning each configuration to get the best performance, and the need for an auto-tuning method that can be used on every architecture.

Tuning quality compared to exhaustive sampling The median difference in elapsed time, along with the noise measurement, between the best parametrization found by SHAMan and the optimal parametrization found by exhaustive search is represented in table 1. Over all optimization experiments, the average distance between the optimum and the result returned by Bayesian Optimization is of 5.71 microseconds (0.04 in median) for an average noise of respectively 2.03 microseconds in mean and 0.05 microseconds in median. This means that in median, the difference between using the best parametrization of our tuner compared to the true best parametrization is imperceptible from the noise. When looking at the relative difference between the optimum and the results from Bayesian Optimization, we find an average distance of 6% (0.7% in median) between the two.

When looking at the different collectives and topologies, we find the difference between the two optimal parametrizations to be inferior to the measured noise for all collectives except for allreduce with 576 MPI processes. When looking at each optimization problem separately, we find that for 105 optimization problems out of 160, the distance of the performance returned by Bayesian Optimization to the optimum is below the measured noise of the system. For the problems where the difference between the results returned by the tuner and the optimum cannot be explained by noise, we find a quite low average difference of 1.90 microseconds (0.18 in median).

Table 1: Median difference in elapsed time and noise between best parametrization found by Bayesian Optimization and optimal parametrization

Collective	# of MPI processes	Relative difference (%)	ΔT (μs)	Noise (μs)
Allreduce	**12**	0.51	0.43	0.04
	576	15.28	1.05	3.81
Broadcast	**12**	4.79	0.32	0.26
	576	2.35	0.32	0.18
Gather	**12**	0.74	0.04	0.01
	576	0.00	0.02	0.00
Reduce	**12**	0.00	0.06	0.00
	576	0.00	0.03	0.00

The noise difference between collectives is explained by multiple factors. Gather and reduce show low noise due to their simple communication pattern (all-to-one). On the opposite, the allreduce collective involves much more intertwined messages, which explains its higher noise and noise sensitivity. Broadcast's higher noise is explained by the best performing algorithm found (k-nomial tree) which, according to Subramoni et al. in [37], introduces some noise due the imbalanced communication pattern.

Tuning speed compared to exhaustive sampling The elapsed time required to reach the optimum for the two tuning solutions and for each of the collectives and configurations is reported in table 2.

Table 2: Time to solution for each heuristic and each collective

Collective	# of MPI processes	Exhaustive search (minutes)	SHAMan (minutes)	Gain (%)
Allreduce	**12**	52.07	4.48	**91.40**
	576	452.55	46.77	**89.66**
Bcast	**12**	744.10	23.65	96.82
	576	5097.53	133.25	**97.39**
Gather	**12**	23.45	3.42	**85.42**
	576	1040.07	77.41	**92.56**
Reduce	**12**	87.50	5.24	**94.01**
	576	550.80	61.58	**88.82**

With a time gain of more than 85% for each collective, we see the benefit of using guided search heuristics with SHAMan to explore the parametric space instead of testing every parametrization with exhaustive sampling. The time required to run all the 160 optimization experiments ranges from a total of 8048 minutes (approximately 134 hours) using brute force to 355 minutes (approximately 6 hours) using Bayesian Optimization, resulting in a total speed-up of 95%. The speed-up is relatively uniform across each collective and each topology.

Overall experiments, we demonstrate that using Bayesian Optimization, we reach 94% of the average potential improvement, for a speed-up in tuning time of 95% on the overall tuning phase. Compared to default Open MPI parametrization, this leads to an average improvement of 48.4% in collective operation performance. This confirms the accuracy of our solution for optimization and makes it a satisfactory alternative to exhaustive search, especially when considering the strong improvement it brings when compared to the default parametrization. This study thus confirms the versatility of SHAMan and black-box optimization for the tuning of a wide range of parametrizable components, and shows that the scope of our work can be extended to many of tunable components within the HPC ecosystem.

6 Conclusion

In this paper, we suggest an OpenSource auto-tuning framework, called SHAMan (**S**mart **H**PC **A**pplication **M**anager) for tuning noisy and expensive systems, which addresses some of the gaps in the tuning frameworks already present in the literature. While some of our previous works already showed the strong performance of SHAMan on I/O accelerators, we added another use-case to further prove its universality, by tuning MPI collectives. When performing the optimization

of four MPI collective communication operations, on two different hardware topologies and for 20 different message sizes, we demonstrate that using Bayesian Optimization, we reach 94% of the average potential improvement, for a speed-up in tuning time of 95% on the overall tuning phase. Compared to default Open MPI parametrization, this leads to an average improvement of 48.4% in collective operation performance. In the near future, we intend to consider additional parameters than the topology and the message size to refine our optimization. Also, we identified additional systems to tune beyond these three cases, such as the MSR parameters of the SAP HANA database which will further extend SHAMan's scope. Regarding the improvement of the optimization methods, we plan on investigating the behavior of the tuner when using pruning strategies. Indeed, these pruning strategies cut off some runs and prevent us from measuring the true performance corresponding to this parametrization. We intend to apply survival analysis to deal with this "censored" data in order to speed up even more SHAMan's convergence speed.

-–

References

1. ARQ. `https://arq-docs.helpmanual.io/`.
2. Atos boosts hpc application efficiency with its new flash accelerator solution. `https://atos.net/en/2019/product-news_2019_02_07/atos-boosts-hpc-application-efficiency-new-flash-accelerator-solution`.
3. Documentation of the SHAMan application. `https://shaman-app.readthedocs.io/`.
4. MPICH: a high performance and widely portable implementation of the Message Passing Interface (MPI) standard. `https://www.mpich.org/`.
5. mpitune. `https://software.intel.com/content/www/us/en/develop/documentation/mpi-developer-reference-linux/top/command-reference/mpitune.html`.
6. Open MPI: Open Source High Performance Computing. `https://www.open-mpi.org/`.
7. OSU micro-benchmarks. `http://mvapich.cse.ohio-state.edu/benchmarks/`.
8. Scikit-optimize. `https://github.com/scikit-optimize/`.
9. The SHAMan application. `https://github.com/bds-ailab/shaman`.
10. Tools to improve your efficiency. `https://atos.net/wp-content/uploads/2018/07/CT_J1103_180616_RY_F_TOOLSTOIMPR_WEB.pdf`.
11. Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, pages 2623–2631, 2019.
12. The GPyOpt authors. GPyOpt: A bayesian optimization framework in python. `http://github.com/SheffieldML/GPyOpt`, 2016.
13. P. Balaprakash, J. Dongarra, T. Gamblin, M. Hall, J. K. Hollingsworth, B. Norris, and R. Vuduc. Autotuning in high-performance computing applications. *Proceedings of the IEEE*, 106(11):2068–2083, 2018.
14. Mohamad Chaarawi, Jeffrey M. Squyres, Edgar Gabriel, and Saber Feki. A tool for optimizing runtime parameters of Open MPI. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, pages 210–217, 2008.
15. Sudheer Chunduri, Scott Parker, P. Balaji, K. Harms, and K. Kumaran. Characterization of MPI usage on a production supercomputer. In *SC'18: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 386–400, 2018.
16. Maxwell Dayvson Da Silva and Hugo Lopes Tavares. *Redis Essentials*. Packt Publishing, 2015.
17. A. Di Pietro, L. While, and L. Barone. Applying evolutionary algorithms to problems with noisy, time-consuming fitness functions. In *Proceedings of the 2004 Congress on Evolutionary Computation*, volume 2, pages 1254–1261, 2004.
18. K. T. Fang, R. Li, and A. Sudjianto. *Design and Modeling for Computer Experiments (Computer Science & Data Analysis)*. Chapman & Hall/CRC, 2005.
19. Ahmad Faraj and Xin Yuan. Automatic generation and tuning of mpi collective communication routines. In *Proceedings of the 19th Annual International Conference on Supercomputing*, pages 393 – 402, 2005.
20. Lars Hertel, Julian Collado, Peter Sadowski, Jordan Ott, and Pierre Baldi. Sherpa: Robust hyperparameter optimization for machine learning. In *SoftwareX*, volume 12, 2020.
21. M. Jette, A. Yoo, and M. Grondona. Slurm: Simple linux utility for resource management. In *Lecture notes in computer science*, 2003.
22. P. Knijnenburg, T. Kisuki, and M. O'Boyle. Combined selection of tile sizes and unroll factors using iterative compilation. *The Journal of Supercomputing*, 24:43–67, 2003.
23. Patrick Koch, Oleg Golovidov, Steven Gardner, Brett Wujek, Joshua Griffin, and Yan Xu. Autotune. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

24. Trang T Le, Weixuan Fu, and Jason H Moore. Scaling tree-based automated machine learning to biomedical big data with a feature set selector. In *Bioinformatics*, volume 36, pages 250–256, 2020.
25. Harshitha Menon, Abhinav Bhatele, and Todd Gamblin. Auto-tuning parameter choices in hpc applications using bayesian optimization. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 831 – 840, 2020.
26. Dirk Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, (239), 2014.
27. T. Miyazaki, I. Sato, and N. Shimizu. Bayesian optimization of hpc systems for energy efficiency. In *High Performance Computing*, pages 44–62, 2018.
28. Rajesh Nishtala and Katherine A. Yelick. Optimizing collective communication on multicores. In *Proceedings of the First USENIX Conference on Hot Topics in Parallelism*, 2009.
29. Jelena Pjesivac-Grbovic, Thara Angskun, George Bosilca, Graham Fagg, Edgar Gabriel, and Jack Dongarra. Performance analysis of MPI collective operations. In *Cluster Computing*, 2005.
30. S. Robert, S. Zertal, and G. Goret. Auto-tuning of io accelerators using black-box optimization. In *Proceedings of the International Conference on High Performance Computing Simulation (HPCS)*, 2019.
31. Sophie Robert. *Auto-tuning of computer systems using bmack-box optimization: an application to the case of I/O accelerators*. PhD thesis, University of UPSaclay, 11 2021.
32. Sophie Robert, Soraya Zertal, and Philippe Couvee. Shaman: A flexible framework for auto-tuning HPC systems. In *Modelling, Analysis, and Simulation of Computer and Telecommunication Systems - 28th International Symposium, MASCOTS 2020, Nice, France, November 17-19, 2020, Revised Selected Papers*, volume 12527 of *Lecture Notes in Computer Science*, pages 147–158, 2020.
33. Sophie Robert, Soraya Zertal, and Philippe Couvee. Shaman: A flexible framework for auto-tuning hpc systems. In *Revised selected papers of the 28th International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 147–158, 2021.
34. Sophie Robert, Soraya Zertal, Grégory Vaumourin, and Philippe Couvée. A comparative study of black-box optimization heuristics for online tuning of high performance computing i/o accelerators. *Concurrency and Computation: Practice and Experience*, 2021.
35. K. Seymour, H. You, and J. Dongarra. A comparison of search heuristics for empirical code optimization. In *2008 IEEE International Conference on Cluster Computing*, pages 421–429, 2008.
36. Florian Siegmund, A. Ng, and K. Deb. A comparative study of dynamic resampling strategies for guided evolutionary multi-objective optimization. In *2013 IEEE Congress on Evolutionary Computation*, pages 1826–1835, 2013.
37. Hari Subramoni, Krishna Kandalla, Jérôme Vienne, Sayantan Sur, William Barth, Karen Tomko, Robert Mclay, Karl Schulz, and D.K. Panda. Design and evaluation of network topology-/speed- aware broadcast algorithms for infiniband clusters. In *Proceedings of the IEEE International Conference on Cluster Computing (ICCC)*, pages 317–325, 2011.
38. Rajeev Thakur, Rolf Rabenseifner, and William Gropp. Optimization of collective communication operations in MPICH. In *International Journal of High Performance Computing Application*, volume 19, pages 49–66, 2005.
39. Bibo Tu, Ming Zou, Jianfeng Zhan, Xiaofang Zhao, and Jianping Fan. Multi-core aware optimization for mpi collectives. In *Proceedings - IEEE International Conference on Cluster Computing, ICCC*, pages 322–325, 2008.
40. Y. Hamadi V. K. Ky, C. D'Ambrosio and L. Liberti. Surrogate-based methods for black-box optimization. *International Transactions in Operational Research*, (24), 2016.
41. S.S. Vadhiyar, G.E. Fagg, and J. Dongarra. Automatically tuned collective communications. In *SC '00: Proceedings of the 2000 ACM/IEEE Conference on Supercomputing*, 2000.
42. S.S. Vadhiyar, G.E. Fagg, and J. Dongarra. Automatically tuned collective communications. In *SC '00: Proceedings of the 2000 ACM/IEEE Conference on Supercomputing*, 2000.
43. W. Zheng, J. Fang, C. Juan, F. Wu, X. Pan, H. Wang, X. Sun, Y. Yuan, M. Xie, C. Huang, T. Tang, and Z. Wang. Auto-tuning MPI collective operations on large-scale parallel systems. In *IEEE 21st International Conference on High Performance Computing and Communications*, pages 670–677, 2019.
44. Karin Zielinski, D. Peters, and R. Laur. Stopping criteria for single-objective optimization. 2005.

Evolutionary-Based Co-Optimization of DNN and Hardware Configurations on Edge GPU

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. The ever-increasing complexity of both Deep Neural Networks (DNN) and hardware accelerators has made the co-optimization of these domains extremely complex. Previous works typically focus on optimizing DNNs given a fixed hardware configuration or optimizing a specific hardware architecture given a fixed DNN model. Recently, the importance of the joint exploration of the two spaces draw more and more attention. Our work targets the co-optimization of DNN and hardware configurations on edge GPU accelerator. We investigate the importance of the joint exploration of DNN and edge GPU configurations. We propose an evolutionary-based co-optimization strategy for DNN by considering three metrics: DNN accuracy, execution latency, and power consumption. By combining the two search spaces, we have observed that we can explore more solutions and obtain a better tradeoff between DNN accuracy and hardware efficiency. Experimental results show that the co-optimization outperforms the optimization of DNN for fixed hardware configuration with up to 53% hardware efficiency gains for the same accuracy and latency.

1 Introduction and related works

Deep Neural Networks (DNN) and hardware accelerators are both leading forces for the observed progress in Machine Learning (ML). However, DNNs are becoming more and more complex and resource-demanding. Therefore, they need careful optimization to achieve the best tradeoff between accuracy and hardware efficiency. To meet this challenge, Hardware-aware Neural Architecture Search (HW-NAS) has been proposed [1] in which DNN hardware efficiency is considered during the exploration process. Nevertheless, hardware efficiency depends not only on the DNN architecture but also on the hardware configuration [2–4]. Most existing works on HW-NAS fall into the optimization of DNN without considering the reconfigurability of the hardware accelerator. As discussed in [5], this approach is sub-optimal because the HW-NAS search space is narrower when considering only a fixed hardware configuration. Thus, by considering the hardware design space, it is possible to find tailor-made DNNs for each hardware configuration and vice-versa. The joint exploration of both search spaces is referred to as *the co-optimization* in this paper.

Recent works [2, 6–13] have tackled the co-optimization problem where DNN architectures are optimized jointly with hardware configurations. Thereby, DNN-HW pairs are generated during the exploration process. However, as pointed out by [14], this strategy incurs a huge search time given the complexity of the joint search space. Therefore, another co-exploration strategy has been proposed by [15–17], in which separate optimization algorithms optimize DNN and hardware. The results can be then communicated between the two optimization algorithms to adjust the exploration process at some points. However, although this strategy solves the drawback of the first joint approach, the sub-optimality of the final results remains its critical issue. The works mentioned above can also be classified according to the following factors [5]: DNN search space and targeted hardware accelerator, exploration algorithm, objective functions, and fitness evaluation strategy. Nevertheless, only a few works have attempted to consider the co-optimization problem for GPU-based hardware platforms. Recent edge GPUs allow the reconfigurability of different hardware parameters such as processing units and operating frequencies. The impact of these parameters on DNN performance has been well discussed and analyzed in [18, 19]. Moreover, recent works shed light on the impact of these parameters when varying other parameters of the DNN. For instance, the authors in [3] adjust both the configuration of the GPU operating frequencies and the batch size of the DNN to maximize the inference hardware efficiency.

Our paper is structured as follows. In section 2 we present and explain the motivation of our work. In section 3 we first formulate our multi-objective co-optimization problem. Then, we

describe and explain our optimization approach. Section 4, presents our experimental setup and results. Then we discuss our obtained results and compare them to other approaches and state-of-the-art solutions. Finally, the conclusion will be given in section 5.

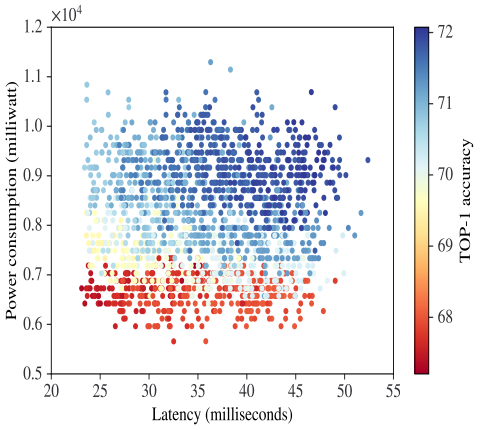


Fig. 1: The results of performing an HW-NAS under fixed edge GPU's hardware configuration.

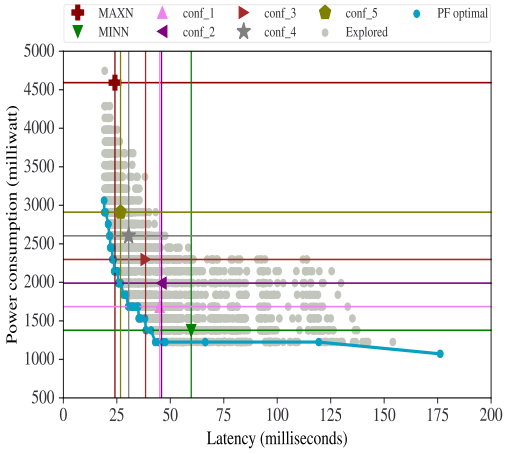


Fig. 2: The results of optimizing edge GPU's hardware configuration for a fixed DNN model.

2 Motivation

Figure 1 shows that different DNN models give different tradeoffs between accuracy and hardware efficiency (i.e., latency and power consumption) under a fixed hardware configuration. This figure gives the results of a Hardware-aware Neural Architectural Search (HW-NAS) [20] that we have performed under a fixed hardware configuration. As hardware platform, we used the NVIDIA Jetson AGX edge GPU. Each point represents a DNN model in the search space. The x-axis and y-axis represent the measured latency and power consumption, respectively, on the edge GPU. The color of the points indicates the TOP-1 accuracy of the DNN. Figure 1 shows that different tradeoffs are obtained between accuracy and hardware efficiency for each explored DNN model.

Figure 2 illustrates that the hardware efficiency of a single DNN varies when varying the hardware configuration. This figure gives the results of an exhaustive exploration of hardware configurations for a fixed DNN model, EfficientNet-B0 [21] in this case. To showcase the impact of the hardware configuration on the overall hardware efficiency of the DNN, we compare the obtained results with the predefined default hardware configurations proposed by NVIDIA. In this figure MAXN (resp. MINN) is the NVIDIA Jetson AGX configuration with the highest (resp. the smallest) allowed clock frequency. MAXN (resp. MINN) in general maximizes (resp. minimises) the processing speed at the cost of a high (resp. low) power consumption. The other configurations (i.e., from conf_1 to conf_5) are proposed to achieve a tradeoff between MAXN and MINN [22]. Remarkably, from the optimal Pareto front (marked in blue), the exhaustive exploration identifies hardware configurations that completely dominate all NVIDIA's predefined default configurations. It's important to mention that the Pareto front does not contain any configuration of the NVIDIA's predefined configurations. Furthermore, the dominant solutions in the Pareto front improve upon the default configurations of NVIDIA (i.e., MAXN and MINN) by 57% and 40% for power consumption and latency, respectively This result shows the necessity to explore the space of hardware configurations, to enhance the hardware efficiency of the DNN.

From these two figures we can conclude that, the performances of a DNN model are determined by the DNN architecture and the HW configuration. However, understanding the impact of these factors is not straightforward. For instance, the DNN with the highest computational complexity is not necessarily the most accurate model in the DNN search space [23]. In addition, the correlation between the performance metrics is also complex. For instance, minimizing latency incurs a maximization of power consumption and vice versa [24].

3 Proposed Approach

3.1 Problem formulation

Our DNN-HW co-optimization problem can be formulated as a multi-objective problem. As exploring the whole HW space and the whole DNN space is time-consuming and costly in terms of development efforts, we need a tool for a rapid DNN/HW co-design space exploration. Furthermore, while accelerating the co-design space exploration, the tool must adequately provide good approximation of the Pareto front. In this paper, we focus on solutions depicting the highest DNN accuracy and hardware efficiency. The term hardware efficiency refers to the trade-off between latency and power consumption. The mathematical formulation of our problem is as follows:

$$MOP: \begin{cases} \min F(dnn, hc) = [(Error_{dataset}(dnn), Latency_{HW_{ACC}}(dnn, hc), Power_{HW_{ACC}}(dnn, hc)]^T \\ \text{s.t.} \quad (dnn, hc) \in (DNN \times HWConf) \end{cases} \tag{1}$$

Where *dnn* represents a DNN model defined by the DNN decision variables detailed in table 1. *hc* represents a hardware configuration defined by the hardware decision variables listed in table 1. DNN and $HWConf$ are the decision spaces of DNNs and hardware configurations, respectively, detailed in table 1. Finally, F is the objective vector to optimize by minimizing the DNN error (i.e., maximizing accuracy) on *dataset*, DNN latency and power consumption on the hardware accelerator HW_{ACC}. We note that the *Error* is measured by calculating the TOP-1 error rate, which is the percentage of images from *dataset* for which the correct label is not the class label predicted by the DNN. *Latency* is the execution time (in milliseconds) of *dnn* on the hardware accelerator HW_{ACC}. Finally, *Power* is the average power consumption (in milliwatt) observed when executing *dnn* on the hardware accelerator HW_{ACC}.

3.2 Optimization Methodology

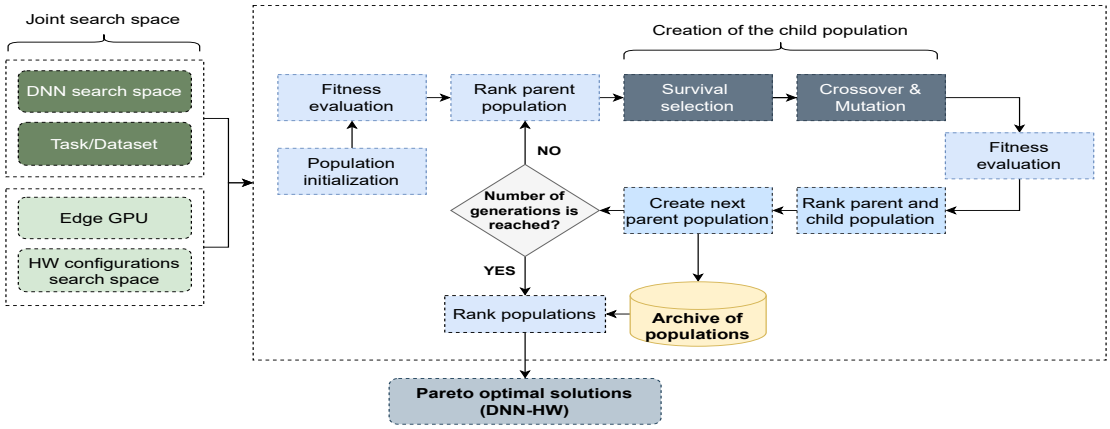


Fig. 3: Overview on the proposed co-optimization approach based on NSGA-II

To solve the above problem, we propose an evolutionary-based co-optimization strategy, where we search for both the optimal DNN architecture and hardware configuration. The search is done by exploring DNN and HW search spaces. Figure 3 details the proposed co-optimization approach. Our methodology includes three main components:

– **Joint search space:** We extend the search space of the HW-NAS by including the hardware configurations. Furthermore, by definition, the joint search space can be generalized to any DNN, task, dataset, and hardware accelerator. Thus, these four factors are considered as inputs in our co-optimization framework. In this paper, we use the joint search space detailed in table 1. We note that all the considered decision variables are discrete.

– **Optimization algorithm:** We choose NSGA-II [25] as an evolutionary algorithm to explore the joint search space. NSGA-II is a widely used algorithm for NAS problems in general, and HW-NAS in particular [26]. Moreover, it typically provides a fast and efficient convergence by searching a wide range of solutions. These two abilities is due to its selection strategy based on non-dominated sorting and crowding distance, which allow for both convergence and diversity of solutions. In this paper, the parameters used for NSGA-II are detailed in table 2. We first initialize the population using the LHS method (Latin HyperCube Sampling). Then, next populations are generated from: 1) selecting the best solutions using the non-dominated sorting algorithm of NSGA-II and 2) applying mutation and crossover on these best solutions to create the offspring population. We choose a high crossover probability of 80% to increase the reproducibility of good candidate solutions. However, we decrease the probability of mutation to 30% to prevent the risk of losing traces of good candidate solutions. Crossover and mutation are chosen uniformly.
– **Evaluation strategy:** The explored pairs are evaluated regarding the DNN accuracy and hardware efficiency. DNN accuracy is evaluated in two stages: 1) We use a fast evaluation technique to quickly determine the DNN accuracy during the exploration, then after the exploration, 2) we perform a more complete evaluation of the DNN accuracy for the elite solutions. We note that the results of the two evaluation techniques are highly correlated. Furthermore, DNN hardware efficiency is directly measured by executing the DNN on the hardware accelerator under the specified configuration.

4 Evaluation

Table 1: The joint search space of DNN and hardware parameters

Decision variables	DNN search space					Hardware search space		
	Input resolution	Width	Depth	Kernel size	Expand ratio	CPU frequency	GPU frequency	Memory frequency
Values	[192, 288]	[16, 1984]	[1, 8]	[3, 5]	[1, 6]	[0.1, 2.3]	[0.1, 1.4]	[0.2, 2.1]
Cardinality	4	16	8	2	4	29	14	9

Table 2: NSGA-II parameters

Parameter	Value
Number of generations	50
Population size	100
Population initialization	LHS
Mutation, probability	Uniform, 30%
Crossover, probability	Uniform, 80%

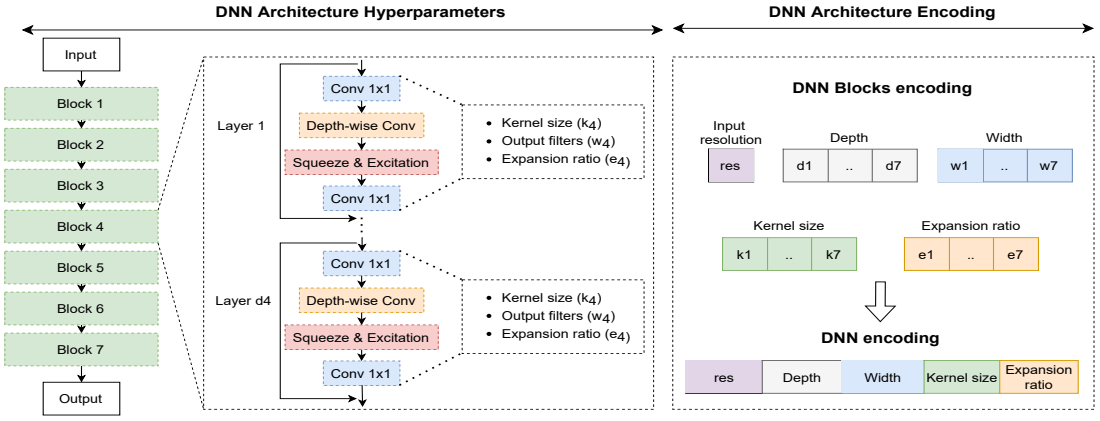


Fig. 4: **DNN search space encoding:** A candidate DNN architecture is real-encoded using a single vector that comprises five sub-vectors depicting: input resolution, depth, width, kernel size, and expansion ratio of each block

4.1 Experimental Setup

With the essential concepts described above, our co-optimization problem instance has the following inputs:

– *DNN search space:* We use the same search space provided by [20, 27]. The search space contains 10^{11} DNN architectures, as detailed in table 1 and figure 4. The authors in [20] provide a prediction model for accuracy assessment, as a fast evaluation tool, and a pretrained supernet, as a complete evaluation tool. However, the second strategy is time-consuming as the sampled DNN needs to be calibrated on the entire training dataset (ImageNet). Thus, we used a fast

evaluation strategy during the exploration then performed a complete evaluation using the pretrained supernet for the elite solutions.

– *Dataset:* We choose to explore the joint search space for a state-of-art dataset such as ImageNet. Thus, the DNN accuracy are calculated on the ImageNet [28] validation dataset. All images are preprocessed using data augmentation techniques such as whitening, upsampling, random cropping, and random horizontal flipping, before feeding them to the DNN.
– *Hardware search space:* We choose the NVIDIA Jetson AGX Xavier GPU as as a hardware accelerator[29]. NVIDIA Jetson GPU accelerators allow the reconfigurability of different hardware parameters such as the number of operating cores. It also allows to have different operating clock frequency in the cores, GPU, and memory units. The chosen values of these parameters depend on the application requirements. For our case, we only vary the operating frequencies as detailed in table 1

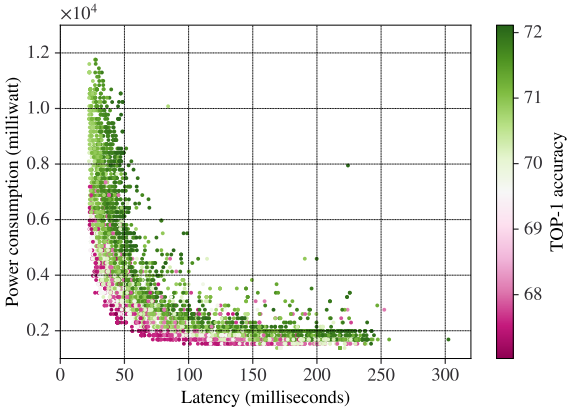


Fig. 5: Co-exploration results on the joint search space

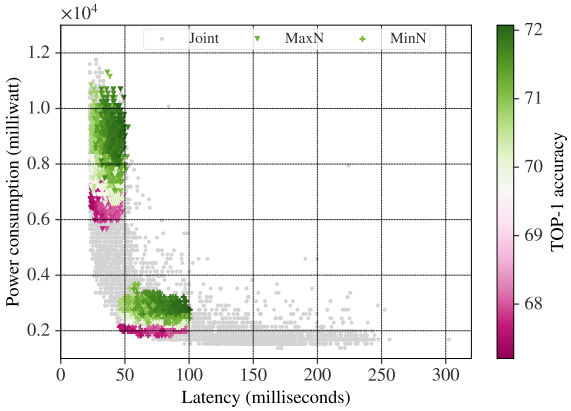


Fig. 6: Results of the three exploration approaches

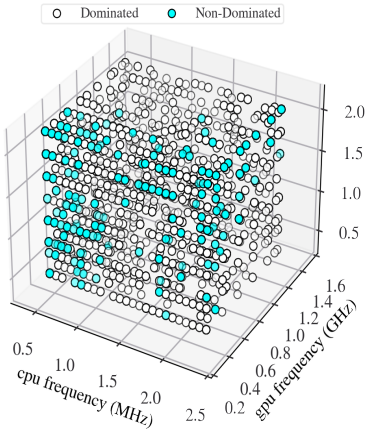


Fig. 7: Explored hardware configurations

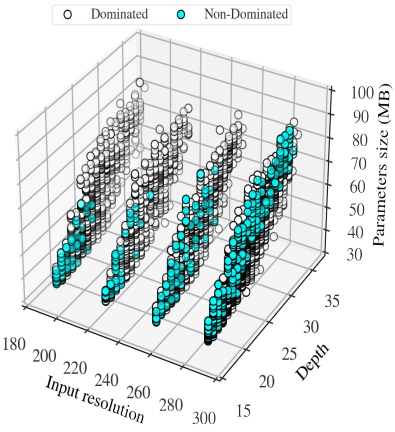


Fig. 8: Explored DNN models

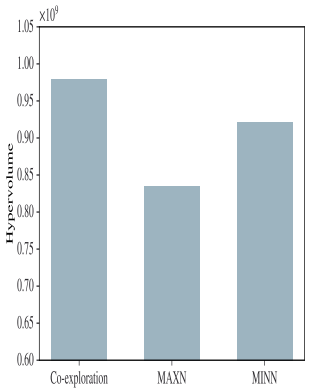


Fig. 9: Hypervolume results of the three optimization approaches

4.2 Experimental Results

In this section we will discuss the obtained results from two main perspectives:

– *Efficiency of the co-exploration:* To underline the co-exploration's importance, we compare the results obtained when co-exploring the joint search space and when performing a typical HW-NAS under fixed hardware configurations. We choose two default configurations proposed by the hardware manufacturer, NVIDIA in our case: MAXN and MINN. Figure 5 gives the results of the co-exploration, where figure 6 depicts a comparison between the results of the three approaches (i.e., joint, MAXN, and MINN), marked with different points shapes. After

analyzing the two figures, we can clearly see that the region explored in the joint space is much larger than the regions explored when fixing the hardware configuration to MAXN or MINN. Furthermore, the explored regions when fixing the hardware configuration are included in the co-exploration. Indeed, the joint search space allows for exploring much larger solutions and hence different tradeoffs between DNN accuracy and hardware efficiency. The obtained hypervolume results presented in table 2 confirm this observation. The obtained hypervolume from the co-exploration is larger than the hypervolumes of the HW-NAS under MAXN and MINN. Furthermore, we give figures 7 and 8 to show the diversity of the explored solutions. The white points correspond to all the explored solutions, whereas the solutions of the Pareto set are marked in blue. In figure 8, we show the explored hardware decision space. From this figure, we can observe that the Pareto optimal solutions are diverse and well distributed. This confirms our earlier observation that no a priori knowledge can be used to choose the best-suited hardware configuration without actual exploration and evaluation. Figure 7 gives the characteristics of the explored DNN models in terms of input resolution, depth (i.e., number of layers), and size of trainable parameters (in Mega-Bytes). Similarly, the Pareto optimal solutions are well distributed and diverse. This also supports the importance of the exploration as we can assume a priori which DNN model will be Pareto optimal without actual evaluation of its performance.

– *Optimality of the obtained results:* To further investigate the efficiency of the co-exploration, we select top pairs of (DNN, hardware configuration) from the Pareto front and compare them to SOTA DNN models under the widely used default configuration proposed by NVIDIA, MAXN. Table 1 details the obtained results. Our co-optimization approach was able to identify better solutions in terms of accuracy and hardware efficiency. We can notice power gains of up to 53% under the same latency constraints. Furthermore, we observe an accuracy improvement of up to 0.5% on the ImageNet dataset.

Table 3: Performance of the baseline models proposed by AttentiveNAS [20] compared to our top solutions of (DNN,hw-conf) obtained from the Pareto front approximation of the co-optimization

DNN, hw_conf	TOP-1 Acc (%)	Latency (ms)	Power consumption (mw)
AttentiveNAS-A2, MAXN	78.8	29.91	6575
AttentiveNAS-A3, MAXN	79.1	33.51	6575
AttentiveNAS-A4, MAXN	79.8	32.67	7033
AttentiveNAS-A5, AMXN	80.1	35.66	6881
Ours-B0, hc0	**79.0**	**28.85**	**4744**
Ours-B1, hc1	**79.6**	**30.82**	**4591**
Ours-B2, hc2	**79.9**	**33.03**	**4591**
Ours-B3, hc3	**80.2**	**34.10**	**6118**

5 Conclusion

In this paper, we investigated the importance of the joint exploration of DNN and hardware configurations for edge GPU accelerators. We propose a co-optimization approach based on an evolutionary algorithm (NSGA-II) to explore these two search scpaces. We aim was to minimize three objective functions: DNN TOP-1 error, latency, and power consumption. Experimental results on the Jetson AGX Xavier demonstrated the efficiency of the co-optimization compared to typical HW-NAS under fixed hardware configurations. Moreover, the top pairs found by our co-optimization are more energy-efficient with up to 53% gains than solutions found by state-of-the-art models under the same accuracy and latency constraints. As future work, we plan to enhance our co-optimization strategy by proposing more efficient selection and recombination operators for the optimization algorithm. We also aim to investigate more hardware configurations and DNN benchmarks to showcase the importance of co-optimization.

References

1. Hadjer Benmeziane, Kaoutar El Maghraoui, Hamza Ouarnoughi, Smaïl Niar, Martin Wistuba, and Naigang Wang. A comprehensive survey on hardware-aware neural architecture search. *CoRR*, abs/2101.09336, 2021.
2. Cong Hao, Xiaofan Zhang, Yuhong Li, Sitao Huang, Jinjun Xiong, Kyle Rupnow, Wen-mei Hwu, and Deming Chen. Fpga/DNN co-design: An efficient design methodology for 1ot intelligence on the edge. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2019.

3. Seyed Morteza Nabavinejad, Sherief Reda, and Masoumeh Ebrahimi. Coordinated batching and DVFS for DNN inference on gpu accelerators. *IEEE Transactions on Parallel and Distributed Systems*, 2022.
4. Lei Yang, Zheyu Yan, Meng Li, Hyoukjun Kwon, Liangzhen Lai, Tushar Krishna, Vikas Chandra, Weiwen Jiang, and Yiyu Shi. Co-exploration of neural architectures and heterogeneous ASIC accelerator designs targeting multiple tasks. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.
5. Weiwen Jiang, Lei Yang, Edwin Hsing-Mean Sha, Qingfeng Zhuge, Shouzhen Gu, Sakyasingha Dasgupta, Yiyu Shi, and Jingtong Hu. Hardware/software co-exploration of neural architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(12):4805–4815, 2020.
6. Yujun Lin, Driss Hafdi, Kuan Wang, Zhijian Liu, and Song Han. Neural-hardware architecture search. *NeurIPS WS*, 2019.
7. Yuhong Li, Cong Hao, Xiaofan Zhang, Xinheng Liu, Yao Chen, Jinjun Xiong, Wen-mei Hwu, and Deming Chen. Edd: Efficient differentiable dnn architecture and implementation co-search for embedded ai solutions. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.
8. Weiwen Jiang, Lei Yang, Sakyasingha Dasgupta, Jingtong Hu, and Yiyu Shi. Standing on the shoulders of giants: Hardware and neural architecture co-search with hot start. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):4154–4165, 2020.
9. Weiwei Chen, Ying Wang, Shuang Yang, Chen Liu, and Lei Zhang. You only search once: A fast automation framework for single-stage DNN/accelerator co-design. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1283–1286. IEEE, 2020.
10. Kanghyun Choi, Deokki Hong, Hojae Yoon, Joonsang Yu, Youngsok Kim, and Jinho Lee. Dance: Differentiable accelerator/network co-exploration. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 337–342. IEEE, 2021.
11. Yun Liang, Liqiang Lu, Yicheng Jin, Jiaming Xie, Ruirui Huang, Jiansong Zhang, and Wei Lin. An efficient hardware design for accelerating sparse CNNs with NAS-based models. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.
12. Yanqi Zhou, Xuanyi Dong, Berkin Akin, Mingxing Tan, Daiyi Peng, Tianjian Meng, Amir Yazdanbakhsh, Da Huang, Ravi Narayanaswami, and James Laudon. Rethinking co-design of neural architectures and hardware accelerators. *arXiv preprint arXiv:2102.08619*, 2021.
13. Michal Pinos, Vojtech Mrazek, and Lukas Sekanina. Evolutionary neural architecture search supporting approximate multipliers. In *European Conference on Genetic Programming (Part of EvoStar)*, pages 82–97. Springer, 2021.
14. Lukas Sekanina. Neural architecture search and hardware accelerator co-search: A survey. *IEEE Access*, 9:151337–151362, 2021.
15. Qing Lu, Weiwen Jiang, Xiaowei Xu, Yiyu Shi, and Jingtong Hu. On neural architecture search for resource-constrained hardware platforms. *arXiv preprint arXiv:1911.00105*, 2019.
16. Mohamed S Abdelfattah, Łukasz Dudziak, Thomas Chau, Royson Lee, Hyeji Kim, and Nicholas D Lane. Best of both worlds: Automl codesign of a cnn and its hardware accelerator. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.
17. Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.
18. 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.
19. Siqin Liu and Avinash Karanth. Dynamic voltage and frequency scaling to improve energy-efficiency of hardware accelerators. In *2021 IEEE 28th International Conference on High Performance Computing, Data, and Analytics (HiPC)*, pages 232–241. IEEE, 2021.
20. Dilin Wang, Meng Li, Chengyue Gong, and Vikas Chandra. Attentivenas: Improving neural architecture search via attentive sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6418–6427, 2021.
21. Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.
22. Jetson developer kits and modules. https://docs.nvidia.com/jetson/l4t/. Accessed: 2021-05-01.
23. Simone Bianco, Remi Cadene, Luigi Celona, and Paolo Napoletano. Benchmark analysis of representative deep neural network architectures. *IEEE Access*, 6:64270–64277, 2018.
24. 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.
25. Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and Tanaka Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *International conference on parallel problem solving from nature*, pages 849–858. Springer, 2000.

26. Yuqiao Liu, Yanan Sun, Bing Xue, Mengjie Zhang, Gary G Yen, and Kay Chen Tan. A survey on evolutionary neural architecture search. *IEEE transactions on neural networks and learning systems*, 2021.
27. Dilin Wang, Chengyue Gong, Meng Li, Qiang Liu, and Vikas Chandra. Alphanet: Improved training of supernets with alpha-divergence. In *International Conference on Machine Learning*, pages 10760–10771. PMLR, 2021.
28. 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.
29. Jetson AGX xavier developer kit. https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit. Accessed: 2021-02-01.

A Framework of Hyper-Heuristics based on Q-Learning

G. Duflo[1], G. Danoy[1,2], E-G. Talbi[3,2], and P. Bouvry[1,2]

[1] SnT, University of Luxembourg
{gabriel.duflo,gregoire.danoy,pascal.bouvry}@uni.lu
[2] FSTM/DCS, University of Luxembourg
[3] University of Lille, CNRS/CRIStAL, Inria Lille
el-ghazali.talbi@univ-lille.fr

1 Introduction

Hyper-heuristics consist in a high-level algorithm performing a search process in a space of low-level heuristics for a given optimisation problem [1]. A hyper-heuristic is composed of a domain independent high-level strategy, typically relying on machine learning, that searches, combines or generates low-level problem-specific heuristics. A given hyper-heuristic is thus designed for a specific problem due to its low-level component. This work proposes to leverage this limitation thanks to a novel framework of generative hyper-heuristics, Q-Learning Hyper-Heuristic Framework (QLHHF). The latter permits to use a class of optimisation problems as an input and it will generate a heuristic for the given problem after the learning process. QLHHF is applicable to any problem that can be reduced into a graph-based one, i.e. a solution can be represented by a sequence of nodes from that graph. A large class of optimisation problems can thus be addressed, including multi-objective and multi-agent ones, while only requiring the definition of limited number of parameters. It is thus possible to use QLHHF for solving problem ranging from benchmarks (e.g., travelling salesman or job-shop scheduling problems) to real-world problems (e.g. drone swarms mobility management).

The proposed QLHHF framework uses Q-Learning (QL) as high-level algorithm. This comes with two challenges due to the nature of the class of optimisation problems which the framework can tackle. The first challenge is using Reinforcement Learning (RL) in a multi-agent context (MARL). This work deals with a fully cooperative setting, i.e. agents share the policy and the policy function [5]. In addition to this, RL is used here to generate heuristics for potential multi-objective problems (MORL). In this area, two main techniques exist. The first and most used way is to define multiple policies and to scalarise them [4]. Another way is to learn "the entire Pareto front of deterministic non-stationary policies" [3].

2 Proposed Hyper-Heuristic Framework

This section presents the proposed Q-Learning Hyper-Heuristic Framework (QLHHF). In order to work with any optimisation problem, the latter must be reduced to a Graph-based Multi-Agent Multi-Objective Optimisation Problem (GMAMOOP). As an example, a use case is represented as a GMAMOOP in section 3. The remainder of this section first presents the format of low-level heuristics, i.e. the problem-specific part of the algorithm, and then describes the high-level QL algorithm.

2.1 Low-level heuristics

A low-level heuristic takes as an input an instance of the wanted problem and returns a feasible solution. An instance of GMAMOOP considers two graphs: a graph for the environment of the problem, i.e. a solution must be expressed as a sequence of its nodes; a graph representing the communication between agents evolving over the environment graph. A solution of a GMAMOOP instance is thus a sequence of couple agent/node which means that an agent has added a certain node in the solution. For reasons of consistency, every GMAMOOP objective must be minimised and define two types of constraints. The hard constraints prevent agents to put any node in the solution (only nodes which do not violate the constraints). The soft constraints are considered as objective and do not restrict any choice of node but they are expected not to be violated after the

learning process. Given a solution S, the vector $f(S)$ thus contains every objective value (for both the objectives and the soft constraints).

As a low-level heuristic, given a GMAMOOP instance, every agent performs the same process asynchronously, as shown in Algorithm 1. At each iteration, agents add a node in the solution until the latter is feasible. The latter choice is made according to a scoring function F evaluating nodes from a subset $X \subset N$. The methods *terminal*, *get_nodes* and *time* are specific to the given GMAMOOP. These methods must thus be redefined according the problem to which is applied QLHHF. Algorithm 1 is a template of low-level heuristic since a heuristic is represented by a certain definition of F. The purpose of the high-level algorithm (here QL) is to find the best definition of that scoring function F.

Algorithm 1: Template of low-level heuristics

input : Instance $I = (G_e, G_c)$
output : Solution S

$S \leftarrow \emptyset$
foreach *Agent* $a \in A$ **do** `// asynchronously`
  $t \leftarrow 0$
  while $\neg terminal(S)$ **do** `// while` S `is not feasible`
    $X \leftarrow get\_nodes(S, a)$ `// get the nodes to evaluate for agent` a
    $n \leftarrow \arg\max_{n' \in X} F(a, n')$
    $S \leftarrow S \cup \{(a, n)_t\}$
    $t \leftarrow t + time(S, a, n)$ `// add the time for agent` a `to go to node` n
  end
end
return S

2.2 High-level algorithm

A QL algorithm is chosen to generate a heuristic for the wanted GMAMOOP, i.e. to find a definition of F in Algorithm 1. In the RL context, a state is depicted as a GMAMOOP solution. An action for RL is a couple agent/node which is an item of a solution. The latter thus corresponds to a history of actions chosen. For the policy, a function Q_Θ is evaluating an action, i.e. a couple agent/node, from a certain state, i.e. the current solution. Each agent then choose the node which maximises Q_Θ. Its computation is made thanks to a Graph Neural Network (GNN) parameterised by Θ which is used in [2] and is based on a state variable depending on the given couple agent/node (see section 3 to see an example of implementation).

When an action is made by an agent, i.e. when a couple agent/node is added in a certain solution, the reward corresponds to the difference of objective values between solutions before and after the action. Since a GMAMOOP may be multi-objective, one action may receive several rewards (one per objective and soft constraint). In order to update the policy, a single reward must be used. At each action made by an agent, the vectorial reward r received is therefore reduced to a scalar reward R by summing a convex combination of objective rewards and each constraint rewards. Each constraint reward is penalised by a non-negative weight so that a violated constraint inducing a negative reward will lower the global reward R.

$$R = w^\top r_\mathcal{O} + \lambda^\top r_\mathcal{C} = \sum_{o \in \mathcal{O}} w_o \cdot r_o + \sum_{c \in \mathcal{C}} \lambda_c \cdot r_c \in \mathbb{R} \tag{1}$$

with $\mathcal{O}$ and $\mathcal{C}$ the sets of objectives and soft constraints respectively, $\sum_{o \in \mathcal{O}} w_o = 1$ and $\lambda_c \geq 0$ $\forall c \in \mathcal{C}$. The reward is then stored in a memory along with the action chosen and the solutions before and after the action as a tuple. At each iteration, a mini-batch is selected from that memory so that each tuple makes it possible to compute a prediction of the evaluation made by Q_Θ and the targeted evaluation depending on the reward computed and an estimation of the future reward. The value of Θ is thus updated with a stochastic gradient descent step to minimise the squared loss, where the loss is the difference between the predicted and the targeted evaluations. Secondly, for each tuple from the mini-batch, the vector of rewards corresponding to soft constraints, i.e. $r_\mathcal{C}$, is used to update λ. The idea is that if a constraint c has been violated by the action chosen, the value of λ_c increases, and vice versa.

3 Use Case

As a use case for the framework, a variant of the Capacitated Vehicle Routing Problem (CVRP) has been implemented. Vehicles move on a map by following roads (which correspond to the environment graph). Several missions are present and consist in moving a package of from a certain point to another one, i.e. from one node of the environment graph to another one. Each mission also specifies a time to accomplish it. Table 1 presents the implementation of that problem for QLHHF.

Table 1. Implementation of a version of CVRP for QLHHF

Parameter	**Value**
$terminal(S)$	`true` if every mission is accomplished in solution S, `false` otherwise
$get\_nodes(S, a)$	starting nodes of missions not taken yet in solution S and ending nodes of missions started by agent a
$time(S, a, n)$	length of the shortest path to node n from the current position of agent a in solution S
$state(S, a, n)$	time of missions starting at node n in solution S time of missions ending at node n and started by agent a in solution S current capacity of agent a in solution S
$f(S)$	$\mathcal{O}$: current time in solution S sum of delays in solution S for every mission $\mathcal{C}$: maximal capacity in solution S among every agent

Some initial experiments have been conducted with this implementation and have shown good convergence results. Moreover, with some parameterisation specific to the framework (e.g., learning rate and discount factor), heuristics generated by QLHHF were able respect the soft constraints.

4 Conclusion

In this work, a novel hyper-heuristic framework based on QL has been presented (QLHHF). The latter is designed to tackle any graph-based optimisation problems, referred to as Graph-based Multi-Agent Multi-Objective Optimisation Problems (GMAMOOP). Several implementations have been experimented and a version of the CVRP has been presented here. The results demonstrate the ability of QLHHF to generate heuristics with a good convergence, notably by decreasing the loss. No guarantee of performance have however been shown so far.

Future work will consist in conducting more experiments and statistical tests on the implementation of CVRP, including comparison with existing methods. More problems are also planned to be implemented.

References

1. Edmund K Burke, Michel Gendreau, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu. Hyper-heuristics: a survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724, December 2013.
2. Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning Combinatorial Optimization Algorithms over Graphs. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
3. Kristof Van Moffaert and Ann Nowé. Multi-Objective Reinforcement Learning using Sets of Pareto Dominating Policies. *Journal of Machine Learning Research*, 15(107):3663–3692, 2014.
4. Kristof Van Moffaert, Madalina M. Drugan, and Ann Nowe. Scalarized multi-objective reinforcement learning: Novel design techniques. In *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 191–199, Singapore, Singapore, April 2013. IEEE.
5. Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms. In *Handbook of Reinforcement Learning and Control*, Studies in Systems, Decision and Control, pages 321–384. Springer International Publishing, Cham, 2021.

A genetic algorithm to optimize the layout of micro hydro-powerplants using cubic splines

Tapia A.[1] and Gutiérrez Reina, D.[2]

[1] Universidad Loyola (Spain)
atapia@uloyola.es
[2] Universidad de Sevilla (Spain)
dgreina@us.es

1 Introduction

The use of Micro Hydropower plants (MHPP) constitutes one of the most efficient solutions for the problem of energy access in remote rural areas, especially in the context of developing countries [1]. These small installations are capable of exploiting the potential energy of a natural water flow to generate electricity with a minimal environmental impact and a simple equipment. Nevertheless, the context of poverty of these emplacements is usually conditioned by deficient design strategies, generally based on personal experience of local technicians and the use of hand rules. For this reason, the development of efficient and robust design strategies can play an important role in the application of these technologies to combat energy poverty.

An MHPP basically consists of extracting a fraction of the water flow from a natural course and conducting it downhill through a long pipe, (penstock) at the end of which the water interacts with a generation unit and transforms its kinetic energy into electrical energy. It is relevant to note that the water is returned to its natural course, and thus the environmental impact is almost zero. The capabilities of an MHPP rely in a correct use of the terrain, reaching the maximum height difference with the shortest pipe length (as pipe friction lowers the efficiency). When the cost of the equipment is low, the cost of the deployment can become relevant with respect to the overall cost, and thus the civil works involved become variables of interest. This, together with the arbitrary nature of the terrain and the river profile, increases the complexity of the problem and motivates the use of numerical and heuristic optimization approaches.

2 Related work

The problem of optimally designing MHPPs has been extensively discussed in the literature. nevertheless, the complexity of the problem is such that it has been traditionally addressed from particular perspectives, such as dealing with the lack of water in dry seasons [2], minimizing the environmental impact of the plants [3], or improve the viability of these plants through the use of water distribution systems [4].

The particular problem of optimizing the layout of an MHPP has been typically addressed from an analytical approach through several simplifications, such as 2D simplifications of the river profile [5, 6], discretizations of the domain [7] or considering a constant river profile slope [8]. As a significant improvement with respect to previous approaches, this work proposes the optimization of the layout considering a continuous, 3D formulation of the problem, capable of dealing with an arbitrary terrain and river profile, using a Genetic Algorithm (GA).

3 Problem modeling

To model the layout of an MHPP, the terrain height is described through a continuous function $z = f(x, y)$, that can be built as the interpolation of experimental topographic data points. The river can then be defined as a continuous parametrized curve $\gamma(s)$ that is contained in the surface $f(x, y)$. Finally, the MHPP layout is modeled as a parametrized continuous curve Γ that connects two points (these are the water extraction and the turbine emplacements) from the curve γ, as shown in Fig. 1. For a robust formulation, the curve γ has been defined as the Cubic Hermite

spline that interpolates a set of spatial points, named nodes. Thus, any possible solution can be expressed by a set of n nodes in the form of $(x_i, y_i, \Delta z)$ (being Δz the relative height of the node to the terrain), as long as the first and last belong to γ. On the basis of the terrain surface $f(x, y)$ and the curves γ and Γ, the performance of the resulting MHPP (power, water flow, cost, etc.) can be determined using an appropriate model, for which the one proposed in [9] has been employed.

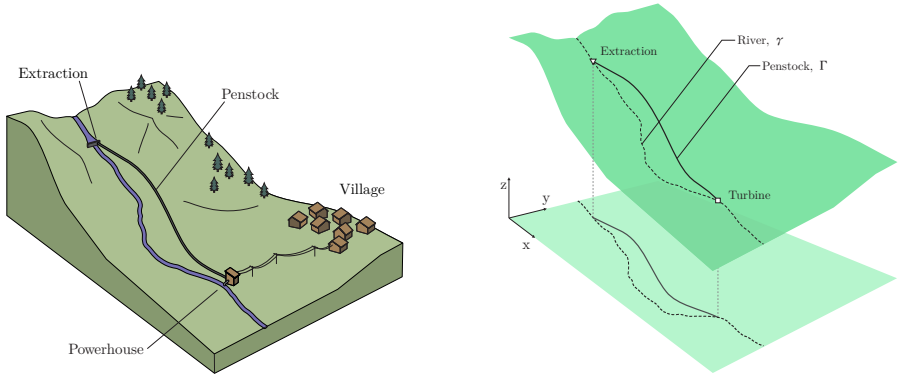


Fig. 1: Basic scheme of an MHPP (left) and its model through 3D spatial curves (right)

The cost of the installation is calculated in accordance with the model proposed in [7], and thus includes the cost of the equipment and the cost of the civil works involved in its deployment. As the generation equipment is sized for the objective power generation level, only the penstock is considered in the equipment. The costs involved in the deployment of the penstock are based on the costs of the excavations and the installation of supports required.

4 Methodology and results

A $\mu + \lambda$ Genetic Algorithm has been employed to address the optimization problem developed, using the coordinates of the nodes of the layout Γ as genes. Given the complexity of generating feasible individuals with a purely random generation scheme, tailored operators have been carefully designed and implemented for the initial population generation, mutation and crossover operations. A greedy search has been applied to determinate the optimal value of the different hyper-parameters. The most important of these, the crossover and mutation probabilities, have demonstrated to provide better results at 0.3 and 0.7, respectively. The algorithm has been tested on an example case study based on designing a 7 kW Pelton installation to supply a remote community in Honduras, where an aerial topographic survey provided the height map and the river profile. The different parameters have been chosen in accordance with the emplacement and local market costs. The main variables relative to the optimal solution have been summarized in Table 1. Also, the optimal layout has been represented on the terrain in Fig. 2.

Table 1: Performance of the optimal solution

Parameter	Value
Power generated (W)	7137.09
Water flow rate (L/s)	13.18
Gross height (m)	101.22
Penstock length (m)	689.10
Pipe diameter (cm)	12
Cost of the civil works (c.u.)	2122.22
Cost of the pipe (c.u.)	24214.47
Total cost (c.u.)	26336.69

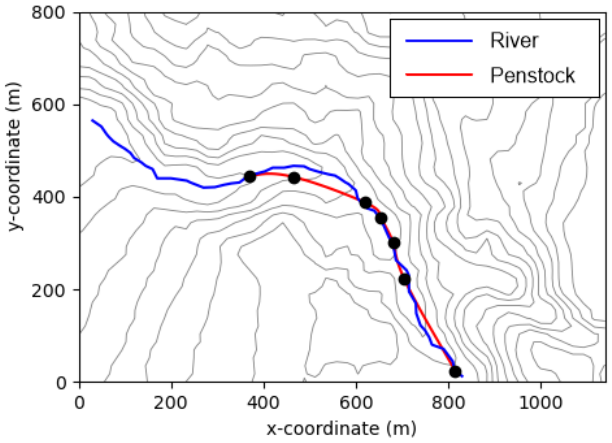


Fig. 2: Layout of the optimal solution

4.1 Conclusion

This work proposes the optimization of an MHPP using a continuous formulation of the problem from a three-dimensional approach. The problem is formulated as the minimization of the cost of the plant with a minimum power generation constraint. The problem not only considers the cost of the equipment, but also the cost of the civil works involved in its deployment, in such a way that the strongly dependence of the path of the penstock through the terrain and the labor involved in terrain excavations and installation of supports is considered.

A GA has been developed to solve the optimization problem, for which tailored initial population generation, mutation and crossover operators have been designed, given the complexity of the constraints involved. The algorithm has been applied to a illustrative case study based on a real case scenario in a remote, small community in Honduras. The real topography of the terrain and the river profile have been determined through an aerial topographic survey, and an optimal layout for the MHPP has been precisely determined. The obtained solution permits the generation of 7.13 kW, using a flow rate of 13.18 L/s, and having a total cost of 26.336 c.u. The analysis of the solution obtained demonstrates how the algorithm builds a layout cutting through rough terrain, thus demonstrating the benefits of using the 3D approach with respect to traditional 2D simplified approaches in the literature.

References

1. Thiago BA Couto and Julian D Olden. Global proliferation of small hydropower plants–science and policy. *Frontiers in Ecology and the Environment*, 16(2):91–100, 2018.
2. Alban Kuriqi, António N Pinheiro, Alvaro Sordo-Ward, and Luis Garrote. Flow regime aspects in determining environmental flows and maximising energy production at run-of-river hydropower plants. *Applied Energy*, 256:113980, 2019.
3. W. Apichonnabutr and A. Tiwary. Trade-offs between economic and environmental performance of an autonomous hybrid energy system using micro hydro. *Applied Energy*, 226:891 – 904, 2018.
4. A. Berrada, Z. Bouhssine, and A. Arechkik. Optimisation and economic modeling of micro hydropower plant integrated in water distribution system. *Journal of Cleaner Production*, 232:877 – 887, 2019.
5. A. Tapia, P. Millán, and F. Gómez-Estern. Integer programming to optimize micro-hydro power plants for generic river profiles. *Renewable Energy*, 126:905 – 914, 2018.
6. A. Tapia, D. G. Reina, and P. Millán. An evolutionary computational approach for designing micro hydro power plants. *Energies*, 12(5):878, 2019.
7. A Tapia, AR del Nozal, DG Reina, and P Millán. Three-dimensional optimization of penstock layouts for micro-hydropower plants using genetic algorithms. *Applied Energy*, 301:117499, 2021.
8. K. V. Alexander and E. P. Giddens. Optimum penstocks for low head microhydro schemes. *Renewable Energy*, 33(3):507–519, 2008.
9. A Tapia, DG Reina, and P Millán. Optimized micro-hydro power plants layout design using messy genetic algorithms. *Expert Systems with Applications*, page 113539, 2020.

Fractal Decomposition based Algorithm for Dynamic camera alignment optimization problem

Arcadi Llanza[1,2], Nadiya Shvai[1] and Amir Nakib[2]

[1] Vinci Autoroutes, Cyclope.ai, Paris, France
[2] Université Paris-Est Créteil, Laboratoire LISSI, Vitry sur Seine, France

1 Introduction

Image Alignment (IA) has become an important task to be taken into account in real-world applications. IA is the process of overlaying images to be able to further analyze them. It is a crucial step because many systems rely on obtaining the correct formatted data to take a posteriory decision. This concept comes from a more generic one called image registration. IA was initially introduced by *Lukas, Kanade et al.* in [1].

In this work, we tackle the Dynamic Optimization Problem (DOP) of IA in a real-world application using a Dynamic Optimization Algorithm (DOA) called Fractal Decomposition Algorithm (FDA), introduced in [2] by *Nakib et al.*. We used FDA to perform IA on CCTV camera feed from a tunnel. As the camera viewpoint can change by multiple reasons such as wind, maintenance, etc. the alignment is required to guarantee the correct functioning of video-based traffic security system.

The rest of this paper is organized as follows. First, Section 2 introduces the problem formulation. Then, Section 3 recalls the FDA foundations. Afterwards, 4 introduces the conducted experiments and explains the obtained results. Finally, in Section 5 the conclusions are presented.

2 Problem Definition

In this real-world application, we aim to solve the dynamic optimization problem of aligning a camera image that has been shifting over time (see Figure 1).

Fig. 1. Left: Images obtained from the camera at time $t-1$. **Right:** Images obtained from the camera at time t. A shift in the camera can be observed including translation and rotation distortions.

To do so, we optimize the matrix H (see Equation (1)) containing 8 Degrees of Freedom (DoF) that correspond to image geometrical transformation (such as translation, rotation, skewing).

$$H = \begin{pmatrix} DoF_{11} & DoF_{12} & DoF_{13} \\ DoF_{21} & DoF_{22} & DoF_{23} \\ DoF_{31} & DoF_{32} & 1 \end{pmatrix} \tag{1}$$

The direct and inverse transformation are given in Equation (2).

$$\begin{aligned} I_1 \cdot H &= I_2 \\ I_1 &= I_2 \cdot H^{-1} \end{aligned} \tag{2}$$

Therefore, given two images I_1 and I_2 that have been captured using the same camera (from the same position, just changing its viewpoint), and that have a view overlap, it should exist a matrix H that let us transform the content of one image into the other one. Specifically, the values in H: DoF_{11}, DoF_{12}, DoF_{21} and DoF_{22} estimate the rotation. Parameters DoF_{13}, DoF_{23} estimate the translation. Finally, DoF_{31}, DoF_{32} estimate the skewing effect. Parameter H_{33} is used for the scale magnitude. Given the problem setup it can be fixed to a constant value of 1.

To perform estimation of the H matrix parameters, a set of point have been initially detected and described in both images. These points were then pairwise matched based on the descriptors similarity (see Figure 2 for an illustration). We formulate the IA as a dynamic optimization problem

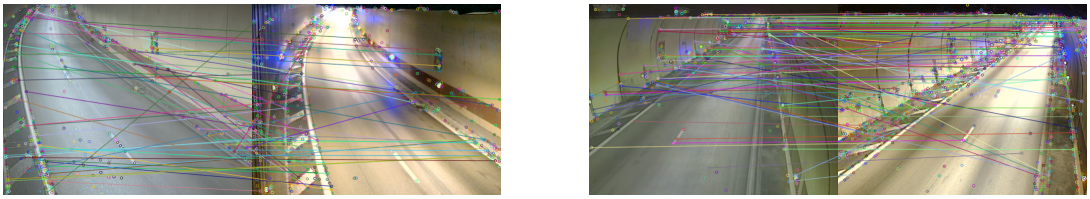


Fig. 2. Sets of images containing the detected and matched keypoints.

by considering the loss function $\mathcal{L}_{loss}$ that provides the (partial) sum of distances between pairs of matched keypoints after apply the transformation H on one point from the pair (see Equation (3)).

$$\begin{aligned} error_k &= d_{L_1}\left(keypoint^k_{I_2}, keypoint^k_{I_1} \cdot H\right) \\ \mathcal{L}_{loss} &= \sum_{error_k < P_i(errors)} error_k \end{aligned} \tag{3}$$

More specifically, we use L_1 distance between pairs of points. When calculating the loss function we take into account lower the distances up to a certain i^{th} percentile P_i. For the experiments we found empirically the value of $i = 80$ to perform well. We use the percentile to smooth the optimization function (*e.g.* filter non-properly matched keypoints).

3 Methodology

To solve the stated problem, we propose to use FDA to deal with this dynamic environment. More specifically, FDA uses a fractal decomposition structure based on hyperspheres (see Figure 3) to explore the search space, and an Intensive Local Search (ILS) method to exploit the promising regions. It was shown in [3] that this approach is exceptionally beneficial in high dimensional space

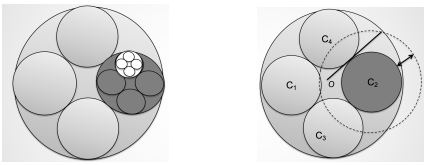


Fig. 3. Left: 4-level decomposition using the hypersphere fractal structure. **Right:** Representation of the hypersphere fractal decomposition with inflation coefficient to ensure that the search space is fully covered given a particular dimension.

problems. Herein, our goal is to demonstrate that FDA can also provide accurate results in a low dimension dynamic optimization problems and can be successfully applied to solve a real-world task.

4 Results

The Figure 4 illustrates the optimization process. The red dashed line shows the best fitness error in the current period, and blue line gives the current fitness error obtained at a given evaluation.

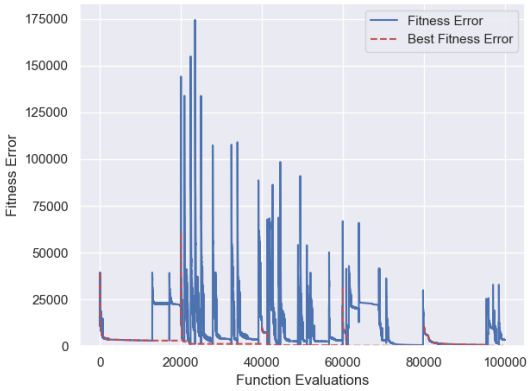


Fig. 4. Graph providing the dynamic optimization experiments. In blue the fitness error (obtained from the loss function) can be seen. In red the best fitness error obtained so far during this period of time is shown. Particularly, it can be observed that a camera has moved over time 5 times (red peaks).

In Figure 5 a pair of blended images is given in the same coordinate space showing that FDA has provided accurate results.

Fig. 5. Sets of blended images represented in the same space showing an accurate match.

5 Conclusions

In this paper, we presented a real-life use case of applying FDA. The obtained results demonstrate the efficiency of the method for solving the problem of image alignment. For the future work, we plan to extend FDA to deal with higher dimensional real-world problems (*e.g.* computer vision) to further validate the FDA's capabilities.

References

1. B. D. Lucas, T. Kanade *et al.*, "An iterative image registration technique with an application to stereo vision." Vancouver, 1981.
2. A. Nakib, S. Ouchraa, N. Shvai, L. Souquet, and E.-G. Talbi, "Deterministic metaheuristic based on fractal decomposition for large-scale optimization," *Applied Soft Computing*, vol. 61, pp. 468–485, 2017.
3. L. Souquet, N. Shvai, A. Llanza, and A. Nakib, "Hyperparameters optimization for neural network training using fractal decomposition-based algorithm," in *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2020, pp. 1–6.

Deep Transformers Optimization for Time Series Forecasting*

J. Keisler[1,3], E-G. Talbi[2], S. Claudel[3], and G. Cabriel[3]

[1] Centre de Recherche en Informatique, Signal et Automatique de Lille
julie.keisler@inria.fr
[2] University of Lille, INRIA, France
el-ghazali.talbi@univ-lille.fr
[3] EDF R&D, Palaiseau, France
sandra.claudel@edf.fr, gilles.cabriel@edf.fr

Keywords: Transformers · Variable-Size Mixed Optimization Problem · Time Series Forecasting.

1 Introduction

The Transformer [15] is a recent Deep Learning model originally created as a seq-to-seq model for machine translation. It achieved state-of-the-art results on this problem. Thus many research has been made to apply this architecture on various fields such as computer vision [1–3, 7, 10, 11], time series forecasting [8, 16, 17] or signal processing [5]. Transformers rely on self-attention to model dependencies in sequences regardless of the distances between input elements positions. Self-attention flexibility makes it relevant to model complex temporal correlations with both long- and short-term dependencies.

Multiple variants of the original architecture (a.k.a. X-formers) [9] have emerged in the literature. The Google Brain Team created a meta-architecture for Transformers, the Evolved Transformer, to perform Neural Architecture Search with an application to NLP [12]. We propose in this paper a search space with fewer parameters to represent relevant Transformers architectures for time-series forecasting. It will then be possible to use any meta-heuristics to optimize this meta-model.

2 Background

The Vanilla Transformer: The vanilla Transformer [15] is a seq-to-seq model created in 2017 for machine translation. It is composed of an encoder and a decoder, each of them is a stack of N identical blocks. Each block is a sequence of a self-attention module followed by a Feed-Forward Layer. The block is completed by Normalization layers to prevent vanishing gradient and residual connections. The encoder blocks can be summarized as:

$$X_1 = NormalizationLayer(Attention(X) + X)$$
$$X_2 = NormalizationLayer(FFN(X_1) + X_1)$$

Where X is the block input. The Attention mechanism is based on a Query-Key-Value (QKV) model. Given $Q \in \mathbb{R}^{N \times D_Q}$, $K \in \mathbb{R}^{M \times D_K}$ and $V \in \mathbb{R}^{M \times D_V}$ being the matrices packing together the sets of queries, keys and values, the self-attention can be defined as:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{D_K}})V$$

The Transformer does not directly use self-attention but multi-head attention instead. The original queries, keys and values are projected into smaller sub-spaces using the projection matrices W^Q, W^K, W^V.

$$MultiheadAttention(Q, K, V) = concat(H_1, ..., H_h)W^O$$
$$\text{where } H_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

Two types of attention are used within the Transformer model. First, the self-attention, where the keys, queries and values are identical: $Attention(X, X, X)$. This attention is used in the encoder and at the beginning of the decoder. Then, the cross-attention uses decoder input as queries and encoder output as keys and values: $Attention(X_d, X_e, X_e)$.

* Supported by EDF.

3 Methods

Multiple Transformers architectures deriving from the Vanilla Transformer have emerged for time series forecasting. We created a meta-architecture able to generalize the four best promising models found in literature: the Vanilla Transformer [16], the Informer [17], the LogSparse Transformer [8] and the SpaceTimeFormer [4]. The main differences between those architectures remain in the input representation, the operations in the encoder/decoder cells and the type of attention used. Fig 1 represents a general representation of this meta-architecture.

Data Embedding: We define the input data as $\mathcal{X} = [\mathcal{X}^{t_0}, ..., \mathcal{X}^{t_N}]$, where $\mathcal{X}^t = [x_1^t, ..., x_M^t] \in \mathbb{R}^M$. The features composing $\mathcal{X}^t$ can be grouped into $\mathcal{X}^t_{Continuous}$ containing the real time series (for example a weather or price indicator) at time t and $\mathcal{X}^t_{Categorical}$ with the integer encoding time series (for example time context values). As the input is not necessarily sequential the absolute position in the sequence of each item $\mathcal{X}^t_{Pos} \in \mathbb{R}$ is also fed to the model. Given those three sub-sequences, the input representation (or data embedding) can be defined as a function $Emb : \mathbb{R}^M \longrightarrow \mathbb{R}^{d_{model}}$ such that:

$$\mathcal{X}^t_{emb} = PositionEncoding(\mathcal{X}^t_{Pos}) + RealEmbedding(\mathcal{X}^t_{Continuous}) + IntEmbedding(\mathcal{X}^t_{Categorical})$$

Cells: The four models share a similar global architecture: an encoder and a decoder made of stacked identical cells. For the encoder, those cells are composed of three blocks. We define a block as a sequence between a layer (attention, feed-forward, convolution, etc) and an assembling strategy (add & norm, concatenation & pooling, etc). The blocks characteristics are set by the layer type. The three main layer types are Attention, Feed-Forward and Convolution. The Encoder cells are composed of three blocks where the first one is necessarily a self-attention block. The Decoder cells are composed of five blocks where the first one is a self-attention block, the third one a cross-attention block and the last one a feed-forward block, as represented Fig 1.

Attention: The attention layer in the attention block can be customized. We generalized the attention formulation via the lens of kernel [14]:

$$\forall x_q \in \mathcal{P}(Q, K) : Attention(x_q, \mathcal{M}(x_q, K)) = \sum_{x_k \in \mathcal{M}(x_q, K)} \frac{\mathcal{K}(x_q x_k^T)}{\sum_{x'_k \in \mathcal{M}(x_q, K)} \mathcal{K}(x_q x_k'^T)} x_k \tag{1}$$

Where $\mathcal{P}(Q, K) \subset Q$ corresponds to the query-prototyping. It selects the relevant queries for the attention computation. $\mathcal{K}(x_q x_k^T)$ is the kernel function which computes the attention scores. $\mathcal{M}(x_q, K) \subset K$ is a sparsity mask that limits the keys and values to which the query has access. $\mathcal{M}$ can be represented as a matrix $L_Q \times L_K$ filled with 0 and 1. Each row corresponds to a query and each column to a key. If a box is set to 1, then the query associated with the box row has access to the corresponding column key.

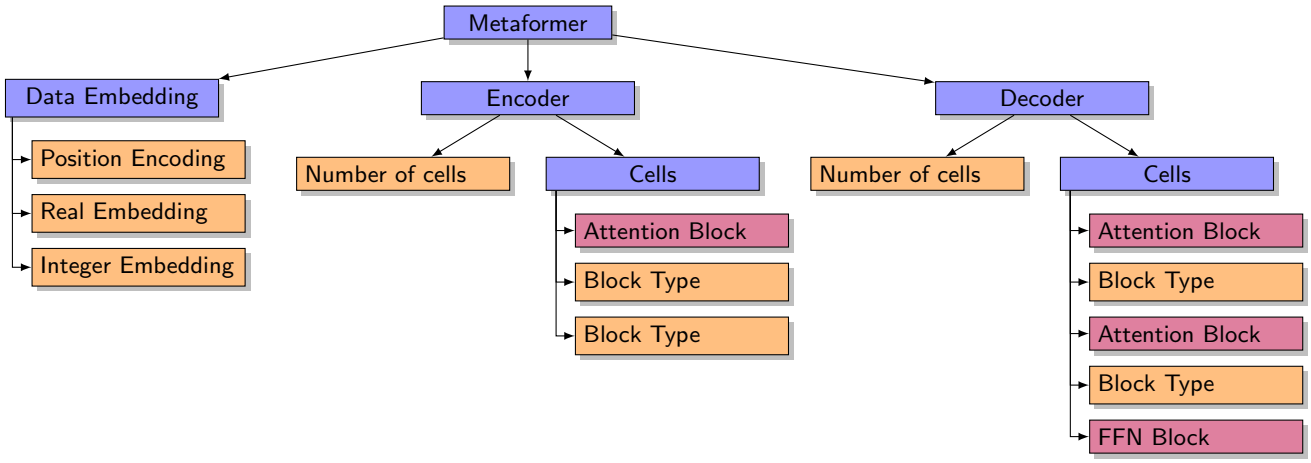


Fig. 1. Metaformer: Meta-Transformer Architecture for time series forecasting

Search Space: The presentation of a solution is expressed as: [dimension model, learning rate, normalization position] + [position embedding, value embedding, temporal embedding, dropout rate after embedding] + [number of encoder cells, encoder normalization layer, dropout rate] + encoder self-attention block parameters + encoder block 2 parameters + encoder block 3 parameters + [number of decoder cells, encoder normalization layer, dropout rate] + decoder self-attention block parameters + decoder block 2 parameters + decoder cross-attention block parameters + decoder block 4 parameters + decoder feed-forward block parameters]. The blocks have a variable number of parameters: [query prototyping, mask, convolution] if attention block, [hidden dimension, activation function] if feed forward block and [convolution type, kernel size, activation function, pooling size] if convolution block or [] if none block. The number of parameters η verifies:

$$\eta \leq \underbrace{3}_{\substack{\text{General}\\\text{parameters}}} + \underbrace{4}_{\substack{\text{Data}\\\text{Embedding}}} + \underbrace{3}_{\substack{\text{General}\\\text{Encoder}}} + \underbrace{3}_{\substack{\text{Attention}\\\text{Encoder}}} + \underbrace{8}_{\substack{\text{2 Blocks}\\\text{Encoder}}} + \underbrace{3}_{\substack{\text{General}\\\text{Decoder}}} + \underbrace{6}_{\substack{\text{2 Attention}\\\text{Decoder}}} + \underbrace{2}_{\substack{\text{FFN}\\\text{Decoder}}} + \underbrace{8}_{\substack{\text{2 Blocks}\\\text{Decoder}}} = 40$$

Optimization: We are in the case of a Mixed-Variable Optimization Problem (MVOP), with mixed parameters types: continuous, categorical, discrete and a variable number of parameters. Our objective function is the model error on the test dataset, which requires training the model. Because of the expensive objective function and the variable size of the solution representation, we simplified our representation by using a cell-based strategy [13]. We defined several typical blocks with fixed parameters inspired by the ones found in the literature. In this search space, any meta-heuristic can be used.

4 Experiments

Datasets:
RTE: With the deregulation of the electricity market, the equilibrium between supply and demand gained importance for the french TSO: Réseau de Transport d'Electricité (RTE). As electricity cannot be stored on a large scale, RTE needs the most accurate half-hourly consumption forecasts to ensure grid stability. The national load consumption is an open-source dataset that can be found online (https://opendata.reseaux-energies.fr/pages/accueil/). Load data is highly fickle and depends on exogenous data such as weather indicators: temperature, wind speed or cloudiness and calendar events: month, weekday, holidays, time change, etc. The weather data used for this experiment is not open-data but equivalents can be found online. We worked on a dataset from 2014 to 2020. We used the years from 2014 to 2017 as the training set, 2018 as the validation set and 2019 as the testing set (to prevent the Covid-19 lockdown effect).
ETT: This dataset is used in the Informer architecture paper [17]. ETT states for Electricity Transformer Temperature. We chose the ETTh1 dataset containing two years of data at a 1-hour level.

The implemented search space can be found in Table 1. We chose Simulated Annealing as a metaheuristic to test our meta-architecture. We performed four optimizations, each starting with one of the four architectures found in literature: Vanilla Transformer, Informer, LogSparseTransformer and SpaceTimeFormer. The results can be found Table 2 for the ETT dataset and Table 3 for the RTE dataset. On the ETT dataset, we managed to reduce by 30% the error and by 50% for the RTE dataset using our meta-architecture. The masks in the attention layers have a significant impact on the result. Unlike convolutional or recurrent architectures, the Transformer only makes a few assumptions about the data structure: time-invariant, locality, etc [9]. Therefore, the Transformer needs lots of data to perform. On our small datasets, the mask can give extra information on the data structure. The band mask, for instance, which only gives access to the closest keys, offers often better results. We believe giving even more flexibility to the model for the mask optimization could improve the results.

5 Conclusion

The goal of this project was to propose a meta-architecture for transformers applied to time series forecasting. Our starting point was four architectures applied to time series forecasting found in the literature. We managed to build a search space generalizing those architectures. This search space corresponds to a Mixed-Variable Optimization Problem where any meta-heuristics can be applied. We used two datasets to show that our meta-architecture achieved better results than the original architectures when optimized using a Simulated Annealing. We managed to reduce the error by 50%.

Table 1. Selected parameters and bounds

Name	Type	Bounds
Model Dimension	Discrete	[100, 600]
Learning Rate	Continuous	[0.0001, 0.5]
Normalization Position	Categorical	[pre, post]
Embedding	Categorical	[sinus, convolution, linear, temporal, t2v [6], identity, none]
Dropout	Continuous	[0.1, 1]
Cells Number	Discrete	[1, 8]
Normalization Layer	Categorical	[BatchNorm, LayerNorm]
Query Prototyping	Categorical	[None, Prob]
Mask	Categorical	[full, triu, global, band, dilated, star, random, bigbird, log]
Convolution	Boolean	[true, false]
Hidden dimension	Discrete	[2, 200]
Activation function	Categorical	[swish, relu, leaky relu, sigmoid, softmax, gelu, elu, none]
Convolution type	Categorical	[causal, down, separable]
Kernel size	Discrete	[1,96]
Pooling size	Discrete	[1,96]

Table 2. Experiments results for the ETT dataset. The first column contains the results using the original architectures and the second one the results using the MetaFormer starting from the corresponding architecture.

Architecture	Original		Optimized	
	MSE	MAE	MSE	MAE
Vanilla Transformer	0.05	0.185	0.0039	0.0505
Informer	0.0129	0.0906	0.0047	0.0618
LogSparse Transformer	0.0174	0.109	0.0114	0.0811
SpaceTimeFormer	0.0269	0.139	0.0167	0.111

Table 3. Experiments results for the RTE dataset. The first column contains the results using the original architectures and the second one the results using the MetaFormer starting from the corresponding architecture.

Architecture	Original		Optimized	
	RMSE	MAPE	RMSE	MAPE
Vanilla Transformer	6210	9.72%	3725	4.81%
Informer	6400	9.73%	3601	4.39%
LogSparse Transformer	5990	8.94%	3712	4.75%
SpaceTimeFormer	7490	11.76%	5882	9.08%

References

1. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging Properties in Self-Supervised Vision Transformers. arXiv:2104.14294 [cs] (May 2021), http://arxiv.org/abs/2104.14294, arXiv: 2104.14294
2. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. arXiv:2010.11929 [cs] (Jun 2021), http://arxiv.org/abs/2010.11929, arXiv: 2010.11929
3. Fan, H., Xiong, B., Mangalam, K., Li, Y., Yan, Z., Malik, J., Feichtenhofer, C.: Multiscale Vision Transformers. arXiv:2104.11227 [cs] (Apr 2021), http://arxiv.org/abs/2104.11227, arXiv: 2104.11227
4. Grigsby, J., Wang, Z., Qi, Y.: Long-Range Transformers for Dynamic Spatiotemporal Forecasting. arXiv:2109.12218 [cs, stat] (Sep 2021), http://arxiv.org/abs/2109.12218, arXiv: 2109.12218
5. Huang, C.Z.A., Vaswani, A., Uszkoreit, J., Shazeer, N., Simon, I., Hawthorne, C., Dai, A.M., Hoffman, M.D., Dinculescu, M., Eck, D.: Music Transformer. arXiv:1809.04281 [cs, eess, stat] (Dec 2018), http://arxiv.org/abs/1809.04281, arXiv: 1809.04281
6. Kazemi, S.M., Goel, R., Eghbali, S., Ramanan, J., Sahota, J., Thakur, S., Wu, S., Smyth, C., Poupart, P., Brubaker, M.: Time2Vec: Learning a Vector Representation of Time. arXiv:1907.05321 [cs] (Jul 2019), http://arxiv.org/abs/1907.05321, arXiv: 1907.05321
7. Khan, S., Naseer, M., Hayat, M., Zamir, S.W., Khan, F.S., Shah, M.: Transformers in Vision: A Survey. arXiv:2101.01169 [cs] (Oct 2021), http://arxiv.org/abs/2101.01169, arXiv: 2101.01169
8. Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.X., Yan, X.: Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting. arXiv:1907.00235 [cs, stat] (Jan 2020), http://arxiv.org/abs/1907.00235, arXiv: 1907.00235
9. Lin, T., Wang, Y., Liu, X., Qiu, X.: A Survey of Transformers. arXiv:2106.04554 [cs] (Jun 2021), http://arxiv.org/abs/2106.04554, arXiv: 2106.04554
10. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. arXiv:2103.14030 [cs] (Aug 2021), http://arxiv.org/abs/2103.14030, arXiv: 2103.14030
11. Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, , Shazeer, N., Ku, A., Tran, D.: Image Transformer. arXiv:1802.05751 [cs] (Jun 2018), http://arxiv.org/abs/1802.05751, arXiv: 1802.05751
12. So, D.R., Liang, C., Le, Q.V.: The Evolved Transformer. arXiv:1901.11117 [cs, stat] (May 2019), http://arxiv.org/abs/1901.11117, arXiv: 1901.11117
13. Talbi, E.G.: Optimization of deep neural networks: a survey and unified taxonomy. ACM Comput. Surv. **00**(00), 37
14. Tsai, Y.H.H., Bai, S., Yamada, M., Morency, L.P., Salakhutdinov, R.: Transformer Dissection: A Unified Understanding of Transformer's Attention via the Lens of Kernel. arXiv:1908.11775 [cs, stat] (Nov 2019), http://arxiv.org/abs/1908.11775, arXiv: 1908.11775
15. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, , Polosukhin, I.: Attention is All you Need p. 11
16. Wu, N., Green, B., Ben, X., O'Banion, S.: Deep Transformer Models for Time Series Forecasting: The Influenza Prevalence Case. arXiv:2001.08317 [cs, stat] (Jan 2020), http://arxiv.org/abs/2001.08317, arXiv: 2001.08317
17. Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., Zhang, W.: Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. arXiv:2012.07436 [cs] (Mar 2021), http://arxiv.org/abs/2012.07436, arXiv: 2012.07436

Fractal decomposition: A divide and conquer approach for global optimization

T. Firmin[1] and E-G. Talbi[2]

[1] Centre de Recherche en Informatique, Signal et Automatique de Lille, France
thomas.firmin@univ-lille.fr
[2] University of Lille & INRIA, France
el-ghazali.talbi@univ-lille.fr

1 Introduction

Optimizing a non linear, non convex, derivative free, or black-box objective function in a high dimensional and continuous search space is a complex task. Commonly we consider a minimization problem for an objective function $f : \mathcal{X} \subset \mathbb{R}^n \to \mathbb{R}$:

$$\hat{x} = \underset{x \in \mathcal{X}}{\operatorname{argmin}} f(x) \tag{1}$$

With, $\hat{x}$ the global optima, f the objective function, and $\mathcal{X}$ a compact set made of inequalities (e.g. upper and lower bounds of decision variables).

In this paper we are interested in a particular class of optimization algorithms, which could be grouped as *branch and reduce* or *divide and conquer* methods for hierarchical search space partitioning. DIviding RECTangle (DIRECT) [1, 12] is one of the most popular algorithm belonging to this class. It is based on the Lipschitzian assumption of the objective function, which allows to compute lower and upper bounds of subrectangles dividing the search space. Furthermore Deterministic/Simultaneous Optimistic Optimization (DOO/SOO) [3, 4] can be seen as a generalisation of DIRECT. Other decomposition-based algorithms are using a different paradigm, such as FRACTOP [5] which uses hypercubes, or Fractal Decomposition Algorithm (FDA) [6] using hyperspheres. All those algorithms are sampling in a high dimensional space by building smaller subspaces.

Along these lines, our goal is to unify all previous approaches by describing and building a common framework based on fractal decomposition, thus facilitating the development of new algorithms based on decision space partitioning using exploration or exploitation metaheuristics, and finding new ways to balance diversification and intensification [7]. Such generalisation will also help to find unified methods to parallelize those strategies in a distributed environment. *Fractal decomposition* paradigm is based on the recursive and auto-similarity properties of fractals in an Euclidian space. In addition, to overcome the *proof-of-concept* implementation, our goal is to build a Python framework called *Zellij* using a modular programming approach. Thus we can develop independent and distinct modules, which, when assembled allows to easily develop various and new *Fractal decomposition* algorithms.

2 Generalized fractal partitioning

We previously described some *space partitioning* algorithms from *branch and reduce* and *divide and conquer* traditional approaches. Our main goal is to unify those approaches, thus allowing to generalize this family of algorithms. This will also open new possibilities to design new decomposition-based algorithms. Here, our contribution is to deepen some aspect of the abstraction, to show that this problem concerns many research communities. To sum up, *fractal decomposition* algorithms can be described by the following concepts:

– **Geometrical fractals objects**: selecting a particular hypervolume has a major impact on the partitioning. Those impacts can be measured by six main criteria: building complexity, memory complexity, tree building complexity, space coverage, tiling regularity, and overlapping (**Fig.**2). The most complex is the Voronoï diagram. (**Fig.**2e,2f) Indeed we cannot build the exact diagram in high dimensions [13, 14]. Therefore, we have to use stochastic methods to build an approximation, such as SpokeDart with hyperplane sampling [13].

- **Tree search algorithm**: the partition of the search space can be stored in a *rooted partition tree*. The root is the initial search space, and nodes are subspaces. The choice of a good algorithm is crucial to efficiently explore and exploit the partition. By introducing pruning strategies, one could also tackle the memory size problem. Popular algorithms are *Best First Search*, *Beam Search*, or *Epsilon Greedy Search*.
- **Sampling strategy**: Sampling in high dimensional spaces is complex. Moreover, one does not sample in exactly the same way in a hypercube or in a hypersphere. In DIRECT or SOO [1, 3], only the center of the hyperrectangle is used. Our appraoch tends to overcome this, by considering deterministic or stochastic, and unique or multiple points methods to sample inside a hypervolume.
  - **Exploration**: The exploration could be done in a passive (MCMC sampling [8], low discrepency sequences [9]...) or active way (metaheuristics [10], surrogate model [11]). The challenge here is to find sampling strategies and adapt them inside complex volumes (e.g. polytopes).
  - **Exploitation**: When a fractal reaches maximum depth of the partition tree, an exploitation algorithm is applied in this fractal. This intensification phase is not constrained inside a subspace. The only bounds will be ones from the initial decision space, so that the exploitation can move freely toward a local or global optimum. Here we could use local search such as Simulated annealing, or Evolutionary algorithm such as CMA-ES. [10]
- **Scoring**: Determining if a subspace is promising or not is essential. Indeed, in FDA the balance between exploration and exploitation of the tree, is made by a scoring method. We can use *best computed score*, *mean*, *median*, *distance to the best* [6], *belief* [5]...

The *Zellij* workflow is described in **Fig.**1. We can see in green all independent modules, this allows to freely develop and combine independent tree search algorithms, hypervolumes, scoring heuristics, metaheuristics and sampling methods. We can for example reproduce DIRECT algorithm with, a *trisection* as the geometrical object, *potentially optimal rectangle* for the tree search algorithm, a subspace will be scored with the best and worst solutions found along longest dimensions, for the exploration strategy: *rectangles center*, and no exploitation. For FRACTOP, *Hypercubes*, *Best First Search*, *Belief* scoring, *Genetic Algorithm* for exploration and *Simulated annealing* for exploitation. Further modules should be developed to encapsulate precedent ones and add parallelization methods to Zellij.

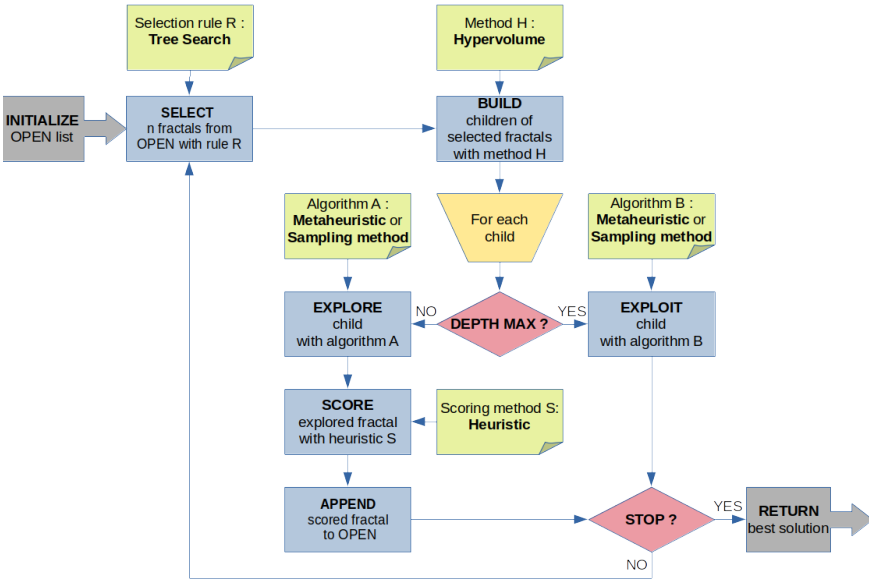
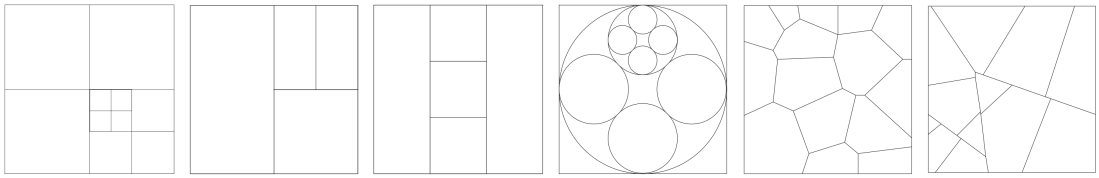


Fig. 1: Zellij workflow



(a) (b) (c) (d) (e) (f)

Fig. 2: Various examples of fractals and their tree building complexity, (a) hypercubes $\mathcal{O}(2^d)$, (b) bisections $\mathcal{O}(2)$, (c) trisections $\mathcal{O}(3)$, (d) hyperspheres $\mathcal{O}(2d)$, (e) dynamic Voronoï $\mathcal{O}(s)$, (f) fixed Voronoï $\mathcal{O}(s)$, s is the number of centroids at each iteration, d is the number of dimensions

3 Conclusion

We described a generalization of *divide and conquer algorithms* using a *Fractal decomposition* paradigm. A Python package named *Zellij* has been developed. Thanks to this framework we are able to model various approaches such as DIRECT, SOO, FRACTOP, FDA and much more. Furthermore, the modular programming standard used in *Zellij*, allows to quickly prototyping new algorithms.

Future works will focus on experimentation using *Zellij*. We will compare, on standard benchmark functions, previous *divide and conquer* algorithms with state of the art algorithms, such as CMA-ES or Bayesian Optimization. This will be an opportunity to study the effects of *curse of dimensionality* on such methods, and maybe identify a way to take advantage of *blessing of dimensionality*. We will try to find, new fractal objects (simplices [15]), ways to sample inside convex polytopes, scoring methods, and tree search algorithms. In a long term perspective, we want to tackle the parallelization issue by making it as modular as the current paradigm of *Zellij*. Finally, we want to use *Zellij* to solve complex optimization problems such as the HyperParameter Optimization (HPO) problem in deep neural networks, which will require to adapt our approach to variable space and mixed variables on expensive optimization problems.

References

1. Jones, D. R., Perttunen, C. D. & Stuckman, B. E. Lipschitzian optimization without the Lipschitz constant. J Optim Theory Appl 79, 157–181 (1993).
2. Liu, H., Xu, S., Wang, X., Wu, J. & Song, Y. A global optimization algorithm for simulation-based problems via the extended DIRECT scheme. Engineering Optimization 47, 1441–1458 (2015).
3. Munos, R. Optimistic Optimization of a Deterministic Function without the Knowledge of its Smoothness. 9.
4. Valko, M., Carpentier, A. & Munos, R. Stochastic Simultaneous Optimistic Optimization. 9.
5. Demirhan, M., Özdamar, L., Helvacıoğlu, L. & Birbil, Ş. I. [No title found]. Journal of Global Optimization 14, 415–436 (1999).
6. Nakib, A., Ouchraa, S., Shvai, N., Souquet, L. & Talbi, E.-G. Deterministic metaheuristic based on fractal decomposition for large-scale optimization. Applied Soft Computing 61, 468–485 (2017).
7. Morales-Castañeda, B., Zaldívar, D., Cuevas, E., Fausto, F. & Rodríguez, A. A better balance in metaheuristic algorithms: Does it exist? Swarm and Evolutionary Computation 54, 100671 (2020).
8. Betancourt, M. A Conceptual Introduction to Hamiltonian Monte Carlo. arXiv:1701.02434 [stat] (2018).
9. Drmota, M. & Tichy, R. F. Sequences, discrepancies, and applications. (Springer, 1997).
10. Talbi, E.-G. Metaheuristics: from design to implementation. (John Wiley & Sons, 2009).
11. Frazier, P. I. A Tutorial on Bayesian Optimization. arXiv:1807.02811 [cs, math, stat] (2018).
12. Jones, D. R. & Martins, J. R. R. A. The DIRECT algorithm: 25 years Later. J Glob Optim 79, 521–566 (2021).
13. Rushdi, A. A., Swiler, L. P., Phipps, E. T., D'Elia, M. & Ebeida, M. S. VPS: Voronoi Piecewise Surrogate models for high-dimensional data fitting. Int. J. UncertaintyQuantification 7, 1–21 (2017).
14. Khachiyan, L., Boros, E., Borys, K., Elbassioni, K. & Gurvich, V. Generating All Vertices of a Polyhedron Is Hard. Discrete Comput Geom 39, 174–190 (2008).
15. Žilinskas, J. B, Branch and bound with simplicial partitions for global optimization. Mathematical Modelling and Analysis 13, 145–159 (2008).

Neural Order-First Split-Second Algorithm for the Capacitated Vehicle Routing Problem

Ali Yaddaden[1], Sébastien Harispe[1⋆], and Michel Vasquez[1]

EuroMov Digital Health in Motion, Univ Montpellier, IMT Mines Ales, Ales, France
{firstname.lastname}@mines-ales.fr

Abstract. Modern machine learning, including deep learning models and reinforcement learning techniques, have proven effective for solving difficult combinatorial optimization problems without relying on handcrafted heuristics. In this work, we present NOFSS, a Neural Order-First Split-Second deep reinforcement learning approach for the Capacity Constrained Vehicle Routing Problem (CVRP). NOFSS consists of a hybridization between a deep neural network model and a dynamic programming shortest path algorithm (Split). Our results, based on intensive experiments with several neural network model architectures, show that such a two-step hybridization enables learning of implicit algorithms (i.e. policies) producing competitive solutions for the CVRP.

Keywords. Neural Combinatorial Optimization Capacitated Vehicle Routing Problem Order-first Split-second Deep Reinforcement Learning

1 Introduction

Modern machine learning, including deep learning models and reinforcement learning techniques, have proven effective for solving difficult combinatorial optimization problems without relying on handcrafted heuristics [1]. The framework known as Neural Combinatorial Optimization (NCO), which proposes to solve combinatorial optimization problems using recent neural networks architectures, is in this context widely studied for routing problems such as the traveling salesman problem (TSP) [2–5] and the capacitated vehicle routing problem (CVRP) [6, 5].

Current NCO approaches implement a construction-based strategy. For the CVRP, such approaches build (i.e. construct) candidate solutions step by step, by selecting at each time step either to visit a client or to go back to the depot to refill, until each client is served. The action to perform at each construction step is chosen based on a probability distribution that will be estimated by a deep neural network, either using supervised or reinforcement learning. This discrete probability distribution defines the probability that an extension of the partial solution under construction, considering each available choices (unsatisfied clients and depot), will lead to the optimal solution. Considering such construction-based NCO approaches, solving the CVRP is therefore reframed as a learning goal aiming to obtain a good estimate of the probability distribution, such as step decisions based on this estimate minimize solution costs.

Using such an approach, the models handle both clients routing and returns to depot. In this context, choices of when to return to the depot are critical. Indeed, more returns to the depot can *de facto* lead to candidate solutions with a number of tours[1] greater than the optimal one. This will result in models failing to efficiently learn interesting resolution strategies, i.e. routing *policies*, due to poor quality candidate solutions, and/or large computational costs inducing prohibitive learning process (millions of learning steps). Handcrafted heuristics and metaheuristics may nevertheless be used to handle return to depot by using an exact tour splitting algorithm - solving a shortest path problem in an auxiliary graph that represents the clients' visit order [7, 8]. Inspired by this problem decomposition, this paper presents NOFSS, Neural Order-First Split-Second, a novel two-step learning-based approach proposing to:

1. Learn how to order clients into a giant tour, using a deep neural network.
2. Optimally split the giant tour into a feasible solution using an exact split algorithm.

⋆ This work used HPC resources of IDRIS (allocation 2022-AD011011309R2) made by GENCI.

[1] A tour is the ordering of clients the vehicle will visit before returning back to the depot. The optimal number of tours will therefore depend on client's demands and vehicle capacity.

NOFSS is a generic approach that will be introduced and tested in the context of CVRP, even if it may be used for a larger class of routing problems. NOFSS relies on a deep neural network that learns a giant tour policy and a dynamic programming algorithm, called Split [8]. Split modifies the giant tour into a feasible solution with respect to vehicle capacity and clients demands. It acts as an oracle that provides feedback on the quality (the total travelled distance) of the giant tour generated from our neural network. This makes it possible to train the NOFSS model through REINFORCE algorithm.

Alongside NOFSS introduction, we present an extensive comparison of various NOFSS and NCO models with state-of-the-art CVRP (meta)heuristics. Results show that, by exploring the search space of giant tours, NOFSS allows to implicitly learn competitive routing policies. [2]

The paper is organized as follows: Section 2 formally introduces the CVRP and notations; Section 3 introduces related work focusing on approaches based on machine learning; Section 4 presents NOFSS; Section 5 presents the experimental protocol as well as results. Discussions and perspectives conclude the paper.

2 Problem Statement

The Capacitated Vehicle Routing Problem (CVRP) is one of the basic types of routing problems where information associated with the clients, the depot and the vehicles are deterministic and known in advance. We consider a set of n clients dispatched on the Euclidean plan and a single depot. In the depot, there is a fleet of homogeneous vehicles with identical transport capacity C. We associate to the clients their coordinates (x_i, y_i) and their demands of goods to deliver $0 \leq d_i \leq C$ $(i \in \{1, ..., n\})$. We associate to the depot its coordinates (x_0, y_0). The demands cannot be split, meaning that a vehicle must satisfy the demand at once. The objective is to minimize the total travelled distance when serving all the clients.

The problem can also be formulated using graph theory [9]. We consider a complete graph $G(V, E)$, where $V = \{0, ..., n\}$ is the vertex set (the vertex 0 represents the depot) and $E = \{(u, v) \in V \times V, u \neq v\}$ is the edge set. We associate with each edge a cost defined as the distance between two vertices. We can represent it as a cost matrix D where $D_{uv} = \sqrt{(x_u - x_v)^2 + (y_u - y_v)^2}$, $(u, v) \in E$. The goal is in this case to find simple circuits called tours such that all clients are served without transgressing the vehicles' capacity and the total travelled distance is as minimum as possible.

3 Related Work

3.1 Neural Combinatorial Optimization (NCO) for the CVRP

We refer to the use of end-to-end deep neural network approaches for solving difficult combinatorial optimization as the Neural Combinatorial Optimization (NCO) framework [3]. In this section, we review the use of this framework to learn construction-based policies for routing problems.

Although the use of neural networks for solving combinatorial optimization problems dates back longer than the appearance of modern deep learning architectures [10], their use has faded away in favor of more efficient metaheuristics. The success of deep learning and reinforcement learning has revived the interest in studying deep neural networks for solving this class of problems. More precisely, with the appearance of the sequence-to-sequence type approaches and the attention mechanism. The general framework (Figure 1) considers two neural networks called respectively encoder and decoder, which can be of different types. The encoder generates the *embeddings* of each element of a problem instance (clients and depot). Embeddings can be viewed as an alternative representation of the element in a higher dimension vector space ($\mathbb{R}^d$ with generally $d = 64$ or $d = 128$). This representation is intended to encompass meaningful features that will be used during the decoding phase. The decoder uses the history of the already visited elements (clients or depot) to compute a query vector that summarizes the solution under construction through a single vector. The query along with the embeddings are used to compute a probability distribution of selecting

[2] Our implementation and results will be available on the following repository https://github.com/AYaddaden/NOFSS

the next element via an attention module. To do so, the attention module confronts the query $q \in \mathbb{R}^d$ to the elements embeddings $e_i \in \mathbb{R}^d$ in order to give attention scores s_i either via a scaled dot-product (i.e. $s_i = \frac{q \cdot e_i^T}{\sqrt{d}}$) or via an additive attention defined as $s_i = v^T \cdot tanh(W_q \cdot q + W_e \cdot e_i)$ with $W_q, W_e \in \mathbb{R}^{d \times d}$, $v \in \mathbb{R}^d$ being learnable parameters. The scores s_i will be converted into a probability distribution by a softmax function[3].

Pointer Networks [2] was the seminal work that considered training LSTM-based encoder and decoder along with an additive attention module via supervised learning on a dataset of TSP instances. The approach successfully solved instances of sizes between 10 and 50 cities. It was next improved by using a policy-based reinforcement learning algorithm for training, namely REINFORCE with critic baseline, thus avoiding the need of a supervision, i.e. to have ground truth optimal solutions for the TSP dataset's instances [3]. Reinforcement learning proved to be more effective for training models on instances of size between 20 and 100 cities, thus achieving better results than Pointer Networks.

Nazari et al. [6] applied the NCO approach to CVRP. Their model considered 1D convolutions instead of an LSTM encoder in order not to bias the model on the inputs' order – LSTM are indeed better suited for modeling sequences where input's order matters. Comparison with classic CVRP algorithms (Clarke and Wright *savings* heuristic and the Sweep algorithm) shows that the deep neural network model performs better on training and test instances' sizes ranging from 10 to 100 clients. It appears also, that the choices of the encoder and decoder are of extreme importance in order to improve the learned policy. The Attention Model (AM) improves the results on the TSP and the CVRP by introducing a model entirely based on the attention mechanism [5]. It uses a Transformer encoder and computes the query vector using a Multi-head attention [11]. The Transformer encoder allows taking into account the graph structure of the TSP and the CVRP in the same way Graph Neural Networks do, thus giving a better representation of the instances. Also, they introduce a new baseline for the REINFORCE algorithm; a greedy rollout baseline that is a copy of AM that gets updated less often.

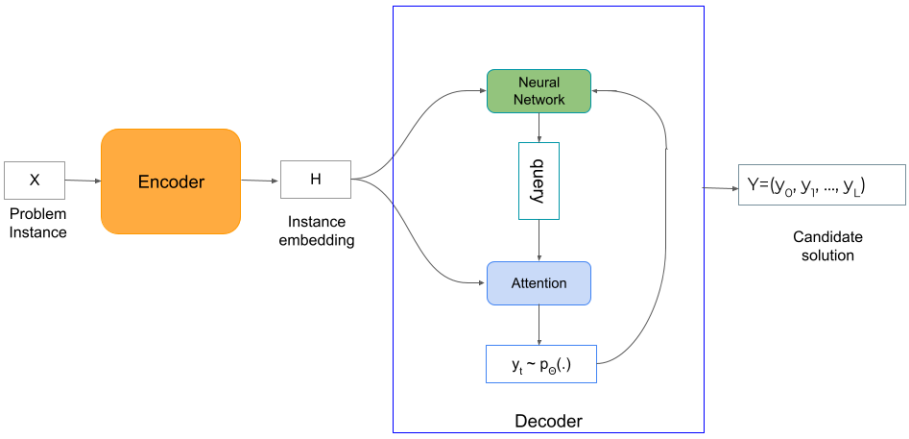


Fig. 1. The general encoder-decoder framework used to solve routing problems. The encoder takes as input a problem instance X and outputs an alternative representation H in an embedding space. The decoder iteratively constructs the candidate solution Y by adding a client or a depot y_t at each step t until all clients are visited.

3.2 Two-step algorithms for the vehicle routing problem

Classical two-step construction approaches for solving the CVRP involve (i) partitioning the clients into feasible clusters with regard to vehicle capacity and (ii) ordering them into routes of minimum length. Based on how the two operations are orchestrated, we can distinguish two types of two-step algorithms: Cluster-first Route-second and Order-first Split-Second.

In Cluster-first Route-second algorithms, the clients are first grouped together following the vehicle capacity constraint, then a traveling salesman problem is solved for each cluster using an

[3] $softmax(s_i) = \frac{exp(s_i)}{\sum_{j=1}^{K} exp(s_j)}$

exact solver or heuristics. The Sweep algorithm is the most common algorithm of this type [12]. Feasible clusters are constructed by considering the polar angle between the clients and the depot, then for each cluster a TSP is solved. An extension of this algorithm called the petal algorithm considers generating several routes and selects the final routes of the solution by solving a set partitioning problem [13]. Another work considers obtaining the clusters by solving a generalized assignment problem [14]. One major drawback of this approach is that it is not computationally efficient due to the clustering algorithms [15].

On the other hand, Order-first Split-second algorithms consider first ordering the customers into a sequence called a giant tour, to then, decompose it into a set of feasible tours considering the vehicle capacity. Traveling salesman problem heuristics are used to get giant tours, and the CVRP tours can be obtained optimally from the giant tours by solving a shortest path problem, as we will detail later. The first documented approach of this type generates the giant tour by random permutation of clients' visit order, followed by a 2-opt improvement, and then builds the routes using Floyd's algorithm [7]. Prins proposed the first genetic algorithm for the CVRP that relies on the Order-first Split-second approach, which was competitive with the best metaheuristic at that time (Tabu Search) [8]. In their approach, the authors proposed a representation of the chromosomes as giant tours and introduced the *Split* procedure based on an auxiliary acyclic graph generated on top of a giant tour. Bellman's algorithm is used in order to extract the feasible routes. HGS, today's state of the art metaheuristic for the CVRP, also uses a giant tour representation and the Split algorithm [16].

The Order-first Split-second approach is appealing. A recent review of this approach surveys more than 70 research papers that build heuristics and metaheuristics to successfully solve vehicle routing problems [17]. Computationally, it is less expensive to build a giant tour and then to split it than building clusters of clients. Also, the search space is reduced to the space of giant tours instead of the direct solution representations with depot placement. As highlighted in the survey, this search space reduction does not make the optimal solution unattainable, since there is an *optimal* giant tour which corresponds to the optimal solution. In addition, for a given giant tour, only its optimal split is retained. This ensures to prevent too many poor quality solutions from appearing often.

3.3 Graph Neural Networks

Since CVRP instances can be modelled as a graph, it is interesting to use neural networks that takes advantage of this structure. This makes Graph Neural Networks (GNNs) an ideal choice to compute a representation of an instance that captures useful information for the resolution process. We define a GNN by stacking K GNN blocks. Each block k relies on message passing in order to compute the node embeddings h_u^k, $\forall u \in V$. This mechanism can be viewed as a differentiable function that computes node embeddings as follows: $h_u^k = F(h_u^{k-1}, \{h_v^{k-1}\}_{v\in\mathcal{N}(u)}, \{e(u,v)\}_{v\in\mathcal{N}(u)})$, with $\mathcal{N}(u)$ being the set of the neighbor nodes of a node $u \in V$ and $\{e(u,v)\}_{v\in\mathcal{N}(u)}$ the set of edges that link the node u to its neighbors $v \in \mathcal{N}(u)$. We use the instance features as an initial input of the first GNN block. The function F itself relies on two mechanisms: neighborhood message aggregation and node embedding update, defined as:

$$m_u^{(k)} = \textsc{Aggregate}\big(\{h_v^{(k-1)}\}_{v\in\mathcal{N}(u)}, \{e(u,v)\}_{v\in\mathcal{N}(u)}\big)$$
$$h_u^{(k)} = \textsc{Update}(h_u^{(k-1)}, m_u^{(k)})$$

Aggregation can either be the mean, the maximum or the sum of neighbors' node embeddings. It can also be a weighted sum with weights computed using an attention mechanism [18]. It can take into consideration the edge weights of the neighboring nodes $e(u,v)$. The update function is a deep neural network that computes a new node embedding by using the message from the aggregation and the node embedding from the preceding block. Graph neural network models differ depending on the choice of the Aggregate and the Update functions.

We can distinguish two families of GNNs: spectral and spatial GNNs. Spectral GNNs rely on spectral graph representations based on graph signal processing theory, such as GCN [19]. Spatial GNNs, such as GAT [18], exploit the graph topology. Refer to Zhou et al. for a GNN review [20].

In the next section, we describe how we use the *Split* algorithm along with the NCO framework to train GNN models for solving the CVRP.

4 The Neural Order-first Split-second algorithm

As mentioned in the previous section, actual NCO construction-based policies for the CVRP produce a sequence by routing the clients and choosing when to return to the depot iteratively until all clients are served. These policies may lead to more returns to depot than necessary and produce poor quality solutions. For example, a policy can decide to refill in the depot after serving each client even if the vehicle capacity allows for serving more than one client at once. Learning from poor quality solutions can slow down and hamper the learning process and produce suboptimal policies. Instead of this, we propose to let the deep neural network build an indirect solution representation via the construction of the giant tour and to delay the routes construction to the Split algorithm. Thus, our neural network implicitly learns to solve vehicle routing problem instances by exploring the space of giant tours. Alternatively, we can view the neural network's output as a permutation of the clients' visit order, which is close to what is done in works for the TSP [3, 4]. This also simplifies the masking procedure used to avoid the appearance of a client twice in the solution. Another advantage of this approach is that the neural network can learn different policies depending on the variant of the vehicle routing problem (e.g. Capacitated VRP, VRP with Time Windows) without additional adaptation. The Split algorithm will handle the additional constraints, and the neural network learns the policy accordingly. Unlink other learning-based construction approaches that build a solution in a variable number of steps due to the return to the depot to refill, our neural network builds the giant tour in a fixed number of steps equal to the number of clients in the instance. Algorithm 1 presents the general approach that will be detailed afterwards.

For a given instance X of the CVRP, our neural network defines a stochastic policy that outputs the probability of generating a giant tour as a sequence Y. Using the probability chain rule, and with θ the parameters of the neural network, this policy is defined as follows:

$$P_\theta(Y|X) = \prod_{t=0}^{n-1} p_\theta(y_t|y_0, ..., y_{t-1}, X)$$

After sampling a sequence Y from P_θ, Y is then transformed into feasible routes using the Split algorithm with regard to the vehicle's capacity constraint. The Split algorithm can be viewed as an oracle that evaluates the goodness of a giant tour by returning the associated solution's total travelled distance. This evaluation makes it possible to train our deep neural network via reinforcement learning. We define the loss as the expected tour lengths of the Y sequences evaluated by the Split algorithm, i.e. $\mathcal{L}(\theta) = \mathbb{E}_{X\sim\mathcal{D}, Y\sim P_\theta(.|X)}\big[\text{Split}(Y, X)\big]$. The objective is to find the best parameters θ that will output good quality sequences Y that would result on short tour lengths. For this, we rely on AdamW as a gradient descent optimizer during training. In order to compute the gradient of the loss, we use REINFORCE with Rollout baseline [5]:

$$\nabla_\theta \mathcal{L}(\theta) = \mathbb{E}_{X\sim\mathcal{D}, Y\sim P_\theta(.|X)}\Big[\Big(\text{Split}(Y, X) - b(X)\Big)\nabla_\theta \log P_\theta(Y|X)\Big]$$

The gradient $\nabla_\theta \mathcal{L}(\theta)$ is approximated using Monte Carlo sampling over a batch of B i.i.d CVRP instances as follows:

$$\nabla_\theta \mathcal{L}(\theta) \approx \frac{1}{B}\sum_{i=1}^{B}\Big[\Big(\text{Split}(Y_i, X_i) - b(X_i)\Big)\nabla_\theta \log P_\theta(Y_i|X_i)\Big]$$

The baseline $b(X)$ is used to reduce the gradient variance, leading to an acceleration of the learning process. We use the greedy rollout baseline $b(X) = \text{Split}(Y^{BL}, X)$ which is an evaluation of the optimal Split of the giant tour Y^{BL} resulting from a copy of the learning neural network with parameters θ^{BL} that acts greedily, i.e. it chooses the next client with the highest probability of appearance at each time step. This baseline proved to be more efficient than actor-critic or REINFORCE with an exponential moving average baseline [5]. During validation, if the performance of θ is significantly better than that of θ^{BL} according to a t-test ($\alpha = 5\%$), the baseline is updated with the parameters of P_θ, i.e. θ^{BL} is set to θ.

4.1 Instance features

For each instance X, we define the nodes and edges features as follows:

Algorithm 1.1: NOFSS REINFORCE with Rollout Baseline

Inputs: θ, Number of epochs E, batch size B, number of instances K, number of clients n, vehicle capacity C, t-test threshold α
$T \leftarrow \frac{K}{B}$
$\theta^{BL} \leftarrow \theta$
for $e \leftarrow 1$ *to* E **do** `// train for` E `epochs`
  for $t \leftarrow 1$ *to* T **do** `// loop over the` T `instance batches`
    `// Get a batch of` B `CVRP instances with` n `clients`
    $X_i \leftarrow \text{getInstance}(n, C), \quad \forall i \in \{1, ..., B\}$
    `// Sample a giant tour according to the learning policy` P_θ
    $Y_i \leftarrow \text{SampleGiantTour}(X_i, P_\theta), \quad \forall i \in \{1, ..., B\}$
    `// Generate a giant tour greedily according to the policy` $P_{\theta^{BL}}$
    $Y_i^{BL} \leftarrow \text{GreedyGiantTour}(X_i, P_{\theta^{BL}}), \quad \forall i \in \{1, ..., B\}$
    `// Evaluate giant tours total travel cost`
    $L_i \leftarrow \text{Split}(X_i, Y_i, C) \quad \forall i \in \{1, ..., B\}$
    $L_i^{BL} \leftarrow \text{Split}(X_i, Y_i^{BL}, C) \quad \forall i \in \{1, ..., B\}$
    `// Compute the loss and update the neural network parameters`
    $\nabla_\theta \mathcal{L} \leftarrow \frac{1}{B} \sum_{i=1}^{B} (L_i - L_i^{BL}) \nabla_\theta \log P_\theta(Y_i|X_i)$
    $\theta \leftarrow AdamW(\theta, \nabla_\theta \mathcal{L})$
  end
  if *t-test*$(P_\theta, P_{\theta^{BL}}) < \alpha$ **then**
    $\theta^{BL} \leftarrow \theta$
  end
end

Node features. Each node $u \in V$ is represented as a quadruplet $(x_u, y_u, \hat{d}_u, a_u)$ where (x_u, y_u) are the node coordinates sampled from a uniform distribution $\mathcal{U}([0,1] \times [0,1])$, $\hat{d}_u = d_u/C \in [0,1]$ is the normalized demand and $a_u = atan((y_u - y_0)/(x_u - x_0)) \in]-\pi/2, \pi/2[$ is the polar angle between the node u and the depot node 0.

Edge features. For each edge $(u, v) \in E$, we define the edge features as the Euclidean distance between the nodes u and v (i.e. $d(u, v) \coloneqq \|u - v\|, \forall (u, v) \in E$). The distance between two nodes in the instance is an interesting feature in the case of vehicle routing problems, since it is information that characterizes the problem well, and it appears in the objective function.

4.2 NOFSS Encoding-Decoding architectures

The NOFSS approach is agnostic to the choice of the encoding and decoding model architectures. Thus, we propose to train various encoder-decoder models that rely on different graph neural networks (GNNs) and a GRU recurrent cell for decoding. The decoded sequence is passed to the Split algorithm in order to retrieve a candidate solution for the instance (Figure 2).

Encoding. We experiment three GNN Encoders for our approach: GCN (a spectral GNN), GAT (a spatial GNN) and TransformerConv (a spatial GNN) [21]. Each encoder have K similar blocks. The GNN outputs an embedding for each node (clients and depot) $h_u^{(K)} \in \mathbb{R}^d, \forall u \in V$ and a graph representation computed using an average pooling $\bar{h} = 1/|V| \sum_{u \in V} h_u^{(K)}$. Finally, to distinguish the clients embeddings from the depot embedding $h_0^{(K)}$, we pass them into a feedforward layer $h_u = W_c \cdot h_u^{(K)} + b_c, \forall u \in V - \{0\}$, with $W_c \in \mathbb{R}^{d \times d}, b_c \in \mathbb{R}^d$ being respectively the weights and the bias of the layer.

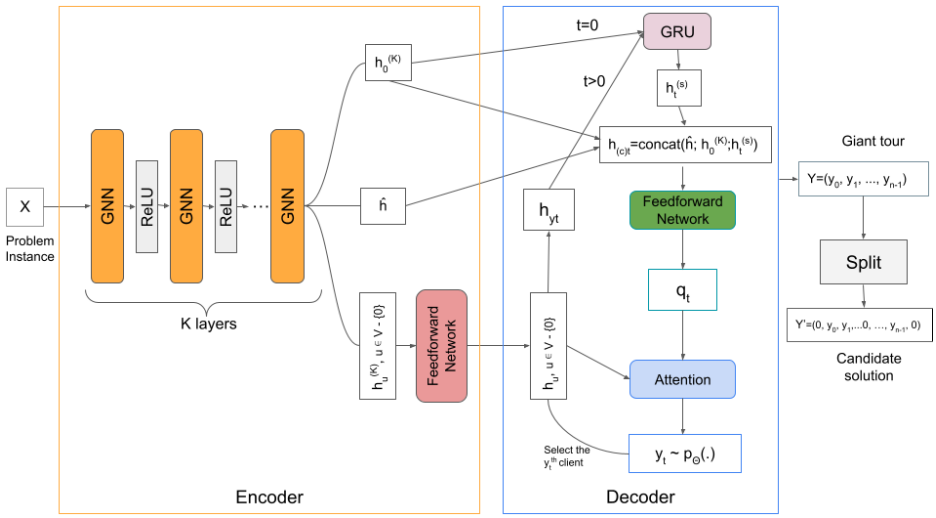


Fig. 2. Our proposed NOFSS model for solving CVRP instances.

Neighborhood definition. As highlighted in Section 2, we can define a CVRP instance as a complete graph. We define the neighborhood $\mathcal{N}(u)$ of a client node $u \in V - \{0\}$ as the κ nearest nodes in terms of Euclidean distance and the depot 0, since it is important for the client's representation to be aware of the depot's existence (i.e. $\mathcal{N}(u) = \{v_1, v_2, ..., v_\kappa \in V; \|v_1 - u\| \leq \|v_2 - u\| \leq ... \leq \|v_\kappa - u\|\} \cup \{0\}$). For the depot, we consider that it is connected to every client. An example of an instance neighborhood definition is depicted in Figure 3. The central node (red square) represents the depot, while the other nodes (blue circles) represent the clients. An edge exists between nodes u and v if $v \in \mathcal{N}(u)$. The number of nearest neighbors κ is determined per instance. We set it to be the average number of clients per route as if they were uniformly distributed on the routes, i.e. $\kappa = \frac{n}{m}$ with n being the number of clients and m being the lower bound of the number of routes. m is determined as the sum of all clients' demands divided by the vehicle's capacity rounded to the next integer ($m = \left\lceil \frac{\sum_{i=1}^{n} d_i}{C} \right\rceil$). The advantage of such a definition of κ is that it takes into account the characteristics of the instance in terms of the number of clients, their demands, and the capacity of the vehicles instead of selecting an arbitrary number of neighbors.

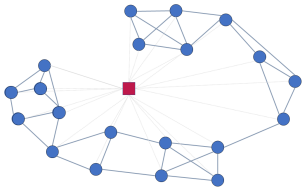


Fig. 3. CVRP instance with relationships between neighboring nodes (central square node is the depot).

Decoding. Since we are decoding a sequence of clients' order, we use a GRU recurrent cell [22]. GRU is relevant as it enables capturing the sequence representation while taking into account the order of its elements. It takes as input the previously selected client representation at step $t-1$ concatenated with the depot representation $h_0^{(K)}$ and incorporates it in the global representation

of the partial giant tour. At $t = 0$, we only use the depot representation $h_0^{(K)}$ as input to the GRU.

$$h_t^{(s)} = \begin{cases} \text{GRU}(h_0^{(K)}), & t = 0 \\ \text{GRU}([h_{y_{t-1}}; h_0^{(K)}]), & t > 0 \end{cases}$$

The graph embedding $\hat{h}$, the depot embedding $h_0^{(K)}$ and the sequence embedding $h_t^{(s)}$ are then concatenated together to form a context vector $h_c \in \mathbb{R}^{3d}$. The context vector is then passed to a feedforward layer made of two linear layers with $ReLU$ activation function in between to output a query vector $q_t \in \mathbb{R}^d$ i.e. $q = W_2 \cdot ReLU(W_1 \cdot h_c + b_1) + b_2$ with $W_2 \in \mathbb{R}^{d \times 3d}$, $W_2 \in \mathbb{R}^{d \times d}$, $b_1, b_2 \in \mathbb{R}^d$ being the parameters of the feedforward layer.

To compute the probability of selecting the next client $p_\theta(y_t | y_0, ..., y_{t-1}, X)$, we compute attention scores s_u ($\forall u \in V - \{0\}$) using a scaled dot-product with a masking mechanism in order to avoid selecting the same client twice. These scores are then clipped within $[-10, 10]$ using $tanh$ [5].

$$s_u = \begin{cases} c \cdot tanh\left(\dfrac{q_t h_u^\top}{\sqrt{d}}\right), & u \neq y_{t'} \quad t' < t, c = 10 \\ -\infty & \text{otherwise} \end{cases}$$

The attention scores are converted into a probability distribution using the softmax function $p_i = p_\theta(y_t = i | y_0, ..., y_{t-1}, X) = softmax(s_i)$ By setting the value of the attention score to $-\infty$, we can perform the masking of already visited clients. Thus, when passed to the softmax function, its associated probability will be 0.

The Split procedure. The algorithm works on the basis of the giant tour output by the neural network augmented with the depot, i.e $\mathcal{Y} = (y_0, y_1, ..., y_n)$ with $y_0 = 0$ being the depot. Using the giant tour, we define an auxiliary graph $H(V^H, E^H)$ with $|V^H| = n + 1$. The nodes in V^H indicate the depot (either for return or departure). The edge set indicates all possible sub-sequences that starts from y_i to y_j $(y_i, y_{i+1}, ..., y_j)$ that do not transgress the vehicle's capacity constraint. We formulate it as follows: $E^H = \{(i, j) \in V^H \times V^H; \quad i < j, \quad \sum_{k=i+1}^{j} d_{y_k} \leq C\}$. The edges are weighted as follows: for an edge $(i, j) \in E^H$ we associate the total travelled distance starting from the depot to the client y_{i+1}, visiting the tour $(y_{i+1}, ..., y_j)$ and going back to the depot from y_j:

$$D^H = \{d_{ij} = dist(0, y_{i+1}) + \sum_{\substack{k=i+1 \\ j-i>1}}^{j-1} dist(y_k, y_{k+1}) + dist(y_j, 0), \quad \forall (i, j) \in E^H\}$$

This gives us a direct acyclic graph where we solve a shortest path problem using Bellman's algorithm. The associated shortest path cost represents the best solution length (total travelled distance) for the CVRP instance with regard to the given giant tour.

5 Experiments

Data generation. We follow the data generation protocol of Nazari et al. [6] to consider 3 types of CVRP instances with number of clients $n = 20$, 50 and 100. For each problem size, we have generated $100k$ instances for training, and two sets of $10k$ instances for validation and test. Clients and depot locations are generated from a uniform distribution $\mathcal{U}(\{[0, 1] \times [0, 1]\})$. The clients' demands are also uniformly drawn from the interval $[1, 9]$. Vehicles' capacities are set to 30, 40 and 50 respectively for $n = 20$, 50, 100.

Hyperparameters. We use an embedding dimension $d = 128$ and a uniform parameter initialization for our deep neural networks $\mathcal{U}(-1/\sqrt{d}, 1/\sqrt{d})$ and set the learning rate to $\eta = 10^{-3}$. The models are trained with a time limit of 100 hours and batch size $B = 128$ on a single NVIDIA V100 GPU with 16 GB of VRAM. For each encoder type, we use $K = 3$ GNN blocks. Implementations use PyTorch and PyTorch Geometric for graph neural networks [23] (Python), while the Split algorithm is implemented in C.

Baselines. We use HGS[4] [16] as baseline as it is one of the state of the art metaheuristics for the CVRP. We also use classical CVRP heuristics[5]: (i) RFCS [7] as a two-step order-first split-second heuristic, (ii) Sweep [12] as a two-step cluster-first route-second approach, and (iii) Nearest Neighbor heuristic as a single-step construction approach [24]. We also trained the model with TransformerConv encoder in an end-to-end manner for depot and clients choice (Full-learning). We first note that NOFSS models are faster to train, completing $E = 1000$ of learning epochs in the 100 hours time budget, while the Full-learning models perform 1000, 500 and 200 training epochs for instance sizes of 20, 50 and 100 respectively. For the exploitation of the learned policies, we use a greedy decoding which considers the highest probability at each decoding step and a sampling strategy which samples 1280 candidate solutions for each test instance from the probability distributions given by the models. Table 1 reports the results of each approach on the test specifying: average solution lengths (obj.), the average gap (in percentage) to the best average solution lengths and the running time (in seconds) to output a candidate solution for a single instance.

Table 1. NOFSS vs. other algorithms. FL for Full-Learning; exploitation, greedy (G), sampling (S).

Method	$n = 20$			$n = 50$			$n = 100$		
	obj.	gap (%)	time (s)	obj.	gap (%)	time (s)	obj.	gap (%)	time (s)
HGS	6.13	0.00	0.003	10.34	0.00	0.09	15.57	0.00	0.69
RFCS	6.30	2.76	0.02	10.90	5.39	0.57	16.62	6.73	7.53
Sweep	7.55	23.16	0.01	15.60	50.93	0.06	28.56	83.37	0.23
Nearest neighbor	7.39	20.57	0.0004	12.63	22.19	0.001	18.95	21.68	0.01
NOFSS-GCN (G)	6.83	11.41	0.0008	12.31	19.05	0.003	19.41	24.66	0.007
NOFSS-GAT (G)	6.59	7.50	0.006	11.74	13.53	0.02	18.34	17.80	0.05
NOFSS-Transformer (G)	6.50	6.03	0.006	11.57	11.89	0.02	18.13	16.44	0.06
FL-Transformer (G)	6.49	5.87	0.006	11.34	9.67	0.02	17.69	13.61	0.06
NOFSS-Transformer (S)	6.24	1.79	1.37	11.03	6.67	1.56	17.45	12.07	2.43
FL-Transformer (S)	6.18	0.81	2.09	10.79	4.35	2.35	17.32	11.23	8.29

5.1 Comparison with a Full-learning setting

Figure 4 presents the evolution of the average solution length per epoch during training and validation on CVRP instances with 20 clients (left) and 50 clients (right). During training, candidate solutions are sampled from the model and their total lengths are averaged over the training set. Let us note that the models' parameters are updated each time a batch is processed via gradient descent, thus the performance of the models changes every batch during training, while validation is performed using the model resulting from the processing of the last batch in the training set, which is theoretically the best model achieved at the end of the epoch. Also, in validation, we use a greedy decoding instead of sampling. The evolution of the average solution lengths shows that the NOFSS model is able to learn an implicit policy for solving the CVRP by learning to output an indirect representation of the solution. On instances with 20 clients, we can observe that during training, the NOFSS model achieves better average solution lengths than the Full-learning model. On validation, we observe the same trend as in training, but starting from the 600^{th} epoch, the Full-learning model slightly outperforms the NOFSS model. The equivalent performance of the two models is confirmed on the test set with average solution lengths of 6.50 and 6.49 on greedy decoding for NOFSS and Full-learning respectively with similar execution times. On sampling decoding, similar performances are observed, with 0.9 % difference in performance between the two models, but with an advantage in execution time in favor of NOFSS. On CVRP with 50 clients, we observe that NOFSS has a better jump start performance on training and a better final performance for the Full-learning model. We observe 2 % difference in performance for greedy and sampling decoding on the test set. We also note similar sampling times for the two types of models in greedy decoding, while NOFSS being 52 %, 50% and 241% faster in sampling respectively for $n = 20$, 50 and 100.

[4] https://github.com/vidalt/HGS-CVRP
[5] https://github.com/yorak/VeRyPy

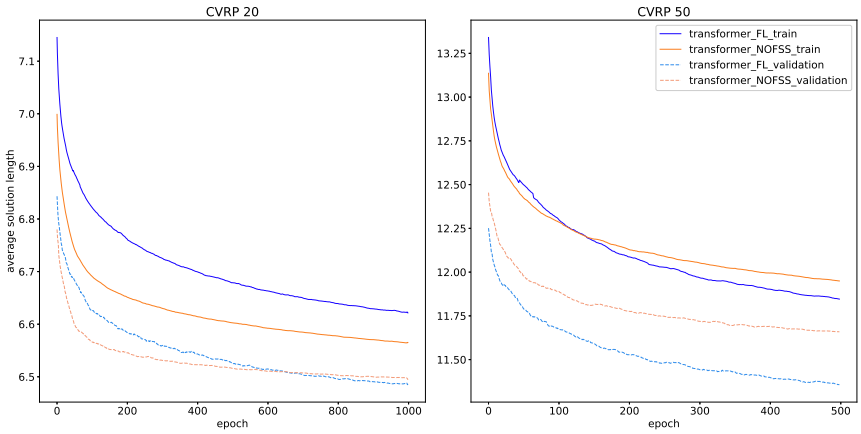


Fig. 4. Learning curves in training and validation for Full-learning (blue) and NOFSS models (orange) on CVRP instances with 20 (CVRP20) and 50 clients (CVRP50); lower is better.

5.2 Comparison to handcrafted heuristics

When compared to handcrafted heuristics, we can observe from Table 1 that either with greedy or sampling exploitation, NOFSS models outperform the Sweep and Nearest neighbor algorithms. NOFSS model seems to output better solution lengths, on average, than RFCS on CVRP with 20 clients when using the sampling strategy but seems to fail scaling to CVRP with 50 and 100 clients. Let us note that while RFCS and NOFSS belong to the same type of two-step strategy, there is a difference in the two approaches in that RFCS explicitly solves a Traveling Salesman Problem, while NOFSS directly evaluates the giant tour using the Split algorithm. The difference in average solution lengths may suggest that NOFSS learned policy is different from a policy that learns to solve a Traveling Salesman Problem.

5.3 Influence of the type of encoder

We investigate the influence of the choice of GNN encoder on models' performance. Figure 5 shows the evolution of the average solutions lengths per epoch in training and validation phases for the 3 types of GNN encoders: GCN, GAT and TransformerConv on CVRP with 20 and 50 clients. We observe the same trends for both training and validation phases, with TransformerConv having the best convergence, followed by GAT encoder and finally by GCN encoder. The instances' representation plays an important role in the resolution process, because a good representation leads to the exploitation of meaningful features and, thus, gives a better solution. The choice of the encoder seems to be a critical part of the model's architecture. It appears from these results that spatial GNNs better perform than spectral GNNs in our evaluation setting. Exploiting the graph topology in the spatial domain seems to benefit more in the context of vehicle routing problems than exploiting the graph structure in the spectral domain. While TransformerConv and GAT are both spatial GNNs, it seems that the way they exploit the node and edges information has an impact on the overall performance of the models.

5.4 On models generalization

We propose to study the generalization of the models trained on a set of instances with a specific size to instances of different size. For this, we evaluate the different test sets on instances of different sizes. For example, we evaluate the NOFSS Transformer model trained on CVRP with 20 clients instances (Transformer-20) on instances with 20, 50 and 100 clients. Table 2 sums up our results. We report the average solution lengths for both greedy and sampling exploitation strategies. For greedy decoding, we report the results for the models trained on the different instance sizes while for sampling, we focus on the model trained on instances sizes which seems more promising based on our findings on the greedy decoding. We observe that for the Transformer-20, the NOFSS model

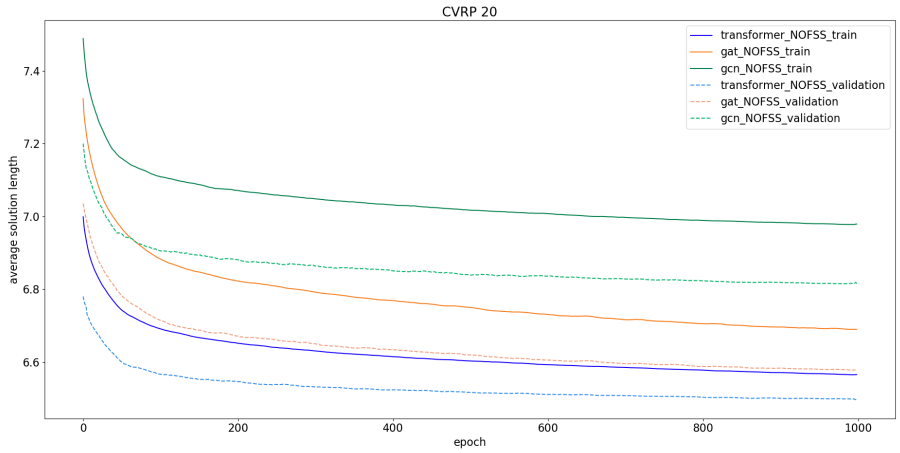


Fig. 5. Comparison of Graph Neural Network encoders on models' performance (training and validation).

has a better generalization property than the Full-learning model, with performance similar for $n = 20$ and $n = 100$ and better for $n = 50$. Since training models on instances with 20 clients is faster, it is relevant to identify that the NOFSS model is a better choice.

For Transformer-50 and Transformer-100, it appears that, for $n = 20$ NOFSS models have better performances than their Full-learning counterparts while staying competitive for $n = 50$ and $n = 100$. An interesting result observed on Transformer-50 is its good generalization to CVRP instances with 100 clients, as it appears that it achieves better performance than the models trained on instances with 100 clients. This may suggest that relevant invariants that are beyond the instance size are learned while training on instances with 50 clients. We push further our investigations on Transformer-50 by analyzing its performance with a sampling exploitation strategy. While for the instances with 20 clients, the models stay competitive with the ones trained on that size, they achieve the best performances on the sets with instances with 50 and 100 clients. Transformer-50 appears to be a good trade-off between learning speed (it is faster to train than Transformer-100) and performance.

Table 2. Comparison of average solution lengths achieved by the NOFSS and Full-learning models on different instance sizes of the test set.

Trained model	NOFSS (G)			Full-learning (G)		
	20	50	100	20	50	100
Transformer-20	6.50	11.62	18.34	6.49	12.01	18.33
Transformer-50	6.64	11.57	17.97	6.76	11.34	17.52
Transformer-100	6.94	11.79	18.13	6.98	11.65	17.69
	NOFSS (S)			Full-learning (S)		
	20	50	100	20	50	100
Transformer-50	6.31	11.03	17.40	6.25	10.79	17.22

6 Conclusion

In this work, we proposed NOFSS, a two-step algorithm hybridizing a deep neural network model and an exact tour splitting procedure for the Capacitated Vehicle Routing Problem. To the best of our knowledge, this is the first model that proposes a hybridization between a deep neural network and a dynamic programming algorithm to successfully learn an implicit policy based on giant tour generation to solve the CVRP. We conducted extensive experiments on the proposed models with various Graph Neural Network encoders and compared them against classic CVRP heuristics and an end-to-end Full-learning model. Our results show that NOFSS is very competitive, even if it

currently does not surpass end-to-end full-learning approaches. NOFSS is however faster than end-to-end approaches in both training and evaluation. It also shows good generalization properties when trained on instances with a specific size and applied to solve instances of different sizes. The NOFSS model is easier to implement than an end-to-end learning-based policy and does not rely on sophisticated handcrafted search strategies to find good quality solutions.

Future work should investigate more on the generalization of the method to instances of bigger sizes. Also, while we tested only greedy and sampling strategies for exploiting the trained models, other relevant strategies may be interesting such as beam search, or using bigger sample sizes than the one we used since NOFSS has a faster execution time. The solution given by NOFSS can also be a good warm start for further improvement by local search algorithms. Finally, since our approach is generic, it would be interesting to evaluate it on other problems, such as the Vehicle Routing Problem with Time Windows.

References

1. Bengio, Y., Lodi, A., Prouvost, A.: Machine learning for combinatorial optimization: a methodological tour d'horizon. European Journal of Operational Research **290** (2021) 405–421
2. Vinyals, O., Fortunato, M., Jaitly, N.: Pointer networks. arXiv:1506.03134 (2015)
3. Bello, I., Pham, H., Le, Q.V., Norouzi, M., Bengio, S.: Neural combinatorial optimization with reinforcement learning. arXiv:1611.09940 (2016)
4. Deudon, M., Cournut, P., Lacoste, A., Adulyasak, Y., Rousseau, L.M.: Learning heuristics for the tsp by policy gradient. In: International conference on the integration of constraint programming, artificial intelligence, and operations research, Springer (2018) 170–181
5. Kool, W., Van Hoof, H., Welling, M.: Attention, learn to solve routing problems! arXiv:1803.08475 (2018)
6. Nazari, M., Oroojlooy, A., Snyder, L.V., Takáč, M.: Reinforcement learning for solving the vehicle routing problem. arXiv:1802.04240 (2018)
7. Beasley, J.E.: Route first—cluster second methods for vehicle routing. Omega **11** (1983) 403–408
8. Prins, C.: A simple and effective evolutionary algorithm for the vehicle routing problem. Computers & operations research **31** (2004) 1985–2002
9. Toth, P., Vigo, D.: The vehicle routing problem. SIAM (2002)
10. Smith, K.A.: Neural networks for combinatorial optimization: a review of more than a decade of research. INFORMS Journal on Computing **11** (1999) 15–34
11. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. (2017) 5998–6008
12. Gillett, B.E., Miller, L.R.: A heuristic algorithm for the vehicle-dispatch problem. Operations research **22** (1974) 340–349
13. Ryan, D.M., Hjorring, C., Glover, F.: Extensions of the petal method for vehicle routeing. Journal of the Operational Research Society **44** (1993) 289–296
14. Fisher, M.L., Jaikumar, R.: A generalized assignment heuristic for vehicle routing. Networks **11** (1981) 109–124
15. Hiquebran, D., Alfa, A., Shapiro, J., Gittoes, D.: A revised simulated annealing and cluster-first route-second algorithm applied to the vehicle routing problem. Engineering Optimization **22** (1993) 77–107
16. Vidal, T.: Hybrid genetic search for the cvrp: Open-source implementation and swap* neighborhood. Computers & Operations Research **140** (2022) 105643
17. Prins, C., Lacomme, P., Prodhon, C.: Order-first split-second methods for vehicle routing problems: A review. Transportation Research Part C: Emerging Technologies **40** (2014) 179–200
18. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv:1710.10903 (2017)
19. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv:1609.02907 (2016)
20. Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., Sun, M.: Graph neural networks: A review of methods and applications. AI Open **1** (2020) 57–81
21. Shi, Y., Huang, Z., Feng, S., Zhong, H., Wang, W., Sun, Y.: Masked label prediction: Unified message passing model for semi-supervised classification. arXiv:2009.03509 (2020)
22. Cho, K., Van Merriënboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: Encoder-decoder approaches. preprint arXiv:1409.1259 (2014)
23. Fey, M., Lenssen, J.E.: Fast graph representation learning with PyTorch Geometric. In: ICLR Workshop on Representation Learning on Graphs and Manifolds. (2019)
24. Rasku, J., Kärkkäinen, T., Musliu, N.: Meta-survey and implementations of classical capacitated vehicle routing heuristics with reproduced results. Toward Automatic Customization of Vehicle Routing Systems (2019)

Solving the Configuration Space Search Problem

C. D'Ambrosio[1], V. Guerrero[2], G. Iommazzo[1], and R. Spencer Trindade[1]

[1] LIX - CNRS,
École Polytechnique,
Institut Polytechnique de Paris,
91120, Palaiseau, France
{dambrosio,giommazz,rst}@lix.polytechnique.fr
[2] Department of Statistics
Universidad Carlos III de Madrid
C/ Madrid, 126
Getafe, Spain
vanesa.guerrero@uc3m.es

1 Introduction and Motivation

We are motivated by the works [8–10], whereby the authors propose to use machine learning and mathematical optimization to decide how to configure a given algorithm. The practical context that motivated the works is the following: we are given a problem Π to be solved with an algorithm A. Different instances of the problem are solved multiple times a day. The instances differ from one another only with respect to a subset of the input data. The performance p of algorithm A depends on the specific instance, which might be from easy to hard to solve, based on how A is configured. The configuration parameters are represented by the vector c and the possible configurations are contained in the set C_A, thus $c \in C_A$. The vector $\pi \in \Pi$ describes the features characterizing a given problem instance. Formally, we wish to find a method that solves the following problem:

$$\min_{c \in C_A} \bar{p}(\pi, c)\,,$$

where π is given and $\bar{p}$ is an approximation of an algorithm performance function $p(\pi, c)$, e.g., the CPU time needed by the algorithm to terminate. Note that neither function p nor function $\bar{p}$ are known.

The methodology proposed in [8–10] has the following characteristics: i) it is instance-based, i.e., the optimal algorithmic configuration depends on the instance features π; ii) it is composed of two phases. In the first phase, it deploys a machine learning paradigm to learn the approximation function $\bar{p}(\pi, c)$ from historical data. In Iommazzo [8], Support Vector Regression (SVR), Decision Trees, and Logistic Regression are used to this end. The second phase is the operational one: upon arrival of a new problem instance π', the Configuration Space Search Problem (CSSP), asking to optimize $\bar{p}$ over the set C_A, is solved to determine the best parameter configuration for running the algorithm A on π'. When SVR is used in the first phase, the CSSP can be written as follows:

$$\min_{\bar{c} \in C_A} \textstyle\sum_{i \in S} \alpha_i^* \exp^{-\gamma \|(\pi_i, c_i) - (\pi', \bar{c})\|_2^2}\,, \tag{1}$$

where S contains the training set indices and $\alpha_i^*, \pi_i, c_i, \gamma$ are found by training. Problem (1) is a nonconvex Mixed Integer Non Linear Programming (MINLP) problem. In [8] the author uses a convex MINLP solver, Bonmin [4], to solve it. Note that Bonmin can only find heuristic solutions for nonconvex MINLPs.

This work presents alternative methods for solving the CSSP, which are capable of achieving a better compromise between CPU time and solution quality.

2 Preliminary Computational Results

We test several commercial or open-source MINLP solvers to treat problem (1). In particular, we compared the performances of Baron [12, 11] and Scip [2, 3], both of which aim to find a global optimum of nonconvex MINLPs, with those of Bonmin and Knitro [1], which provide a heuristic

solution of nonconvex MINLPs such as (1). Moreover, we run the SC-MINLP algorithm, introduced in [5, 6] and whose faster implementation is described in [7]. SC-MINLP is an iterative algorithm which, at each iteration, solves a lower-bounding and an upper-bounding problem. When we reach an iteration in which the two bounds coincides, then we obtained a global solution of the MINLP at hand. Notably, since SC-MINLP deals with separable nonconvexities, we must first reformulate problem (1) so that it falls in this problem class. Namely,

$$\min \textstyle\sum_{i \in S} w_i \tag{2}$$

$$w_i \geq \alpha_i^* \exp^{-\gamma(\sum_{k \in K}(\pi_{ik} - \bar{\pi}_k)^2 + z_i)} \qquad \forall i \in S \tag{3}$$

$$z_i = \sum_{j \in J} (c_{ij} - \bar{c}_j)^2 \qquad \forall i \in S \tag{4}$$

$$\bar{c} \in C_A. \tag{5}$$

The two sets of nonlinear constraints, namely (3) and (4), contain univariate nonlinear functions; their variables are z_{ij} and $\bar{c}_j$. Note that when $\bar{c}_j$ is binary, constraints (4) can be written linearly as $\bar{c}_j^2 = \bar{c}_j$.

To test the different methods mentioned above, we selected 187 hard instances from [8]. Further, since we would like the CSSP to be solved quickly, we set a time limit of 500 seconds to run the solvers.

In Tables 1 and 2, we present a summary of the preliminary results obtained. Firstly, we remark that Bonmin is unable to find a feasible solution for 6 instances, so all the statistics concerning this solver are computed over 181 instances only. Secondly, we point out that we decided not to show the Baron's results in the tables. In fact, the solver stopped and declared the problem infeasible for 175 instances, and it reached the prescribed time limit 3 times; only on the remaining 9 instances it converged to a global optimum within the assigned time limit.

In Table 1, we examine the quality of the best primal (feasible) solutions found by the considered solvers, measured by the accompanying upper bounds. Notably, each table entry reports the number of times the solver in the row wins against the solver in the column, i.e., manages to provide a tighter (namely, lower) upper bound. Note that, when the values contained in two symmetric entries do not sum up to the total number of instances is because of the ties.

	Bonmin	Knitro	Scip	SC-MINLP
Bonmin	-	15/181	121/181	169/181
Knitro	63/181	-	145/187	184/187
Scip	64/181	39/187	-	170/187
SC-MINLP	18/181	3/187	17/187	-

Table 1. Upper Bounds: winners

The clear winner is Knitro, obtaining 184 wins on SC-MINLP, 63 wins on Bonmin and 145 wins on Scip. Bonmin ranks second, with 15 wins on Knitro, 121 wins on Scip, and 169 wins on SC-MINLP. This is not surprising, as Knitro and Bonmin are heuristic solvers for non-convex MINLPs and are hence designed to find good feasible solutions, whereas exact solvers such as Scip seek to find certified global optima. We observe that Bonmin and Knitro find the same solution 109 times, Bonmin and Scip twice, Knitro and Scip 3 times.

Furthermore, we remark that the CPU time depends on the different approaches to the problem. In fact, Bonmin reports the shortest average solution time (11.54 seconds) and Knitro is the second fastest solver (16.82 seconds, on average), while Scip always hits the allotted time limit (500 seconds), and SC-MINLP takes 295 seconds, on average, to solve the instances.

In Table 2, we show the lower bounds achieved by the solvers. We only consider Scip and SC-MINLP, as Baron fails most of the time. Bonmin and Knitro are not considered because they are heuristic methods for nonconvex MINLPs, thus they cannot provide a valid lower bound.

From the table, we gather that, on average, the lower bounds found by Scip are tighter (higher) than the ones provided by SC-MINLP. However, Scip always hits the time limit (# t.l. in the table), while SC-MINLP only does so on 42 instances (we recall that the average CPU time was

	Scip	SC-MINLP
Average	-32.028	-34.074
# t.l.	187	42
best LB	160	27

Table 2. Lower Bounds

500 against 295 seconds). This happens because, in these preliminary computational results, we run SC-MINLP only for the first iteration, for lack of time; thus, the relaxation used to find a lower bound is not refined.

The row "best LB" reports the number of times the solver provides a better lower bound than the other solver in the table. Scip outperforms SC-MINLP 160 times over 187. Since there is no tie, SC-MINLP wins 27 times; out of these, SC-MINLP hits the time limit 13 times. Moreover, whenever SC-MINLP manages to find a better lower bound than Scip, its CPU time exceeds 200 seconds. Thus, we argue that there is room for improvement: refining the relaxation used by SC-MINLP to find the lower bound could largely improve the results of Table 2.

We conclude by observing that, in the specific case of CSSP for SVR, the problem could be solved by plenty of other methods. However, we considered only MINLP solvers because they are general-purpose algorithmic frameworks, that could be deployed to solve CSSPs arising from the use of several other machine learning paradigms in the first phase of the method proposed in [8–10].

3 Conclusions

We performed a computational study focusing on the Configuration Space Search Problem introduced in [8–10]. The problem is a nonconvex MINLP. Exact and heuristic solvers were tested, each showing advantages and drawbacks. In particular, heuristics are fast in finding good quality solutions, while exact methods provide lower bounds. As perspectives, we think that hybrid and tailored methods would be interesting to develop for speeding up the solution process.

References

1. Byrd, R. H., Nocedal, J., Waltz, R.A., KNITRO: An integrated package for nonlinear optimization, In G. di Pillo and M. Roma, editors, Large-Scale Nonlinear Optimization, pages 35–59. Springer, 2006.
2. Bestuzheva, K., Mathieu Besançon, M., Chen, W.-K., Chmiela, A., Donkiewicz, T., van Doornmalen, J., et al. The SCIP Optimization Suite 8.0. Technical report, Optimization Online, December 2021.
3. Bestuzheva, K., Mathieu Besançon, M., Chen, W.-K., Chmiela, A., Donkiewicz, T., van Doornmalen, J., et al. The SCIP Optimization Suite 8.0. ZIB-Report 21-41, Zuse Institute Berlin, December 2021.
4. Bonami, P., Biegler, L.T., Conn, A.R., Cornuejols, G., Grossmann, I.E., Laird, C.D., Lee, J., Lodi, A., Margot, F., Sawaya, N., Waechter, A., An Algorithmic Framework for Convex Mixed Integer Nonlinear Programs. Discrete Optimization. 5(2):186-204, 2008.
5. D'Ambrosio, C., Lee, J., Wächter, A.: A global-optimization algorithm for mixed-integer nonlinear programs having separable non-convexity. In: Algorithms—ESA 2009, LNCS, vol. 5757, pp. 107–118. Springer, Berlin (2009).
6. D'Ambrosio, C., Lee, J., Wächter, A.: An algorithmic framework for MINLP with separable non-convexity. In: J. Lee, S. Leyffer (eds.) Mixed Integer Nonlinear Programming, The IMA Volumes in Mathematics and its Applications, vol. 154, pp. 315–347. Springer New York (2012).
7. D'Ambrosio, C., Frangioni, A., Gentile, C.: Strengthening the sequential convex MINLP technique by perspective reformulations. Optimization Letters 13 (4), 673-684 (2019).
8. Iommazzo, G.: Algorithmic Configuration by Learning and Optimization. Ph.D. Thesis, École Polytechnique, France (2021).
9. Iommazzo, G., D'Ambrosio, C., Frangioni, A., Liberti, L.: Learning to Configure Mathematical Programming Solvers by Mathematical Programming. LION 2020: 377-389
10. Iommazzo, G., D'Ambrosio, C., Frangioni, A., Liberti, L.: A Learning-Based Mathematical Programming Formulation for the Automatic Configuration of Optimization Solvers. LOD (1) 2020: 700-712
11. Sahinidis, N. V., BARON 21.1. 13: Global Optimization of Mixed-Integer Nonlinear Programs, User's manual, 2021.
12. Tawarmalani, M. and N. V. Sahinidis, A polyhedral branch-and-cut approach to global optimization, Mathematical Programming, 103(2), 225-249, 2005.

Categorical-Continuous Bayesian optimization applied to chemical reactions

Theo Rabut[1], Hamamache Kheddouci[1], and Thomas Galeandro-Diamant[2]

[1] Université de Lyon, Université Lyon 1, LIRIS UMR CNRS 5205, F-69621, Lyon, France
[2] ChemIntelligence, Lyon, France

Abstract. Chemical reaction optimization is a challenging field for the industry. Its purpose is to experimentally find reaction parameters (e.g. temperature, concentration, pressure) that maximize or minimize a set of objectives (e.g. yield or selectivity of the chemical reaction). These experiments are often expensive and long (up to several days), making the use of modern optimization methods more and more attractive for chemistry scientists.
Recently, Bayesian optimization has been showed to outperform human decision-making for the optimization of chemical reactions [14]. It is well-suited for chemical reaction optimization problems, for which the evaluation is expensive and noisy.
In this paper we address the problem of chemical reaction optimization with continuous and categorical variables. The presence of categorical variables in an optimization problem often increases its difficulty and decreases the performances of the optimization algorithms.
We propose a Bayesian optimization method with the use of a covariance function initially proposed by Ru *et al.* in the COCABO method [12] and specifically designed for categorical and continuous variables. Also, we experimentally compare different methods to optimize the acquisition function. We establish their performances based on the optimization of two simulated chemical reactions involving categorical and continuous reaction parameters.
We show that the proposed Bayesian optimization algorithm finds optimal reaction parameters in fewer experiments than state of the art algorithms on our simulations.

Keywords: Mixed bayesian optimization · chemical reaction optimization · categorical variables

1 Introduction

Every chemical reaction is optimized before being industrialized. The goal is to find, by carrying out experiments, input parameters (e.g. temperature, pressure, residence time, etc.) that yield optimal values for a set of objectives (e.g maximize the yield, minimize the production of an impurity, etc.).

The pursuit of high-performance optimization methods is driven by the high cost of chemical experiments. The performances of optimization methods applied

to chemical reactions are measured against the quality of the solution (i.e. how close the solution is to the optimization objectives) and how many experiments are needed to find this solution.

One-Variable-At-a-Time (OVAT) and Design of Experiments (DoE) [1, 16] methods are the most used approaches to optimize chemical reactions. The OVAT method iterates by performing experiments and modifying only one parameter at a time. DoE methods consist in planning a series of experiments following a design matrix, running these experiments and building a statistical model (usually linear or polynomial) with the resulting dataset. An optimum is then computed from the model. OVAT and DoE methods tend to need a large number of experiments to be effective. In addition, OVAT can be very slow (because only one variable is changed at a time) and can get stuck in local optima. Simplex-based methods are also sometimes used to optimize chemical reactions [9, 19]. They consists of building a simplex in the search space, then evaluating the objective function at each of the vertices of the simplex and iteratively displacing one vertex at a time following heuristics. Simplex-based methods tend to be easily stuck in local optima [18].

Zhou *et al.* [21] proposed a deep reinforcement learning (DRL) based method to optimize chemical reactions. The authors combined DRL and pre-training to be able to start working with very small amounts of data. This leads to satisfactory results on problems containing only continuous variables but hasn't been tested with categorical variables (without descriptors).

Bayesian optimization (BO) is a powerful approach to optimize problems for which the evaluations are expensive and noisy. It has shown a variety of successful applications [13]. BO concepts are described in figure 1. First, an initialisation is done with a small number of experiments. Then, a surrogate model (e.g. Gaussian process) is trained using these experiments. An acquisition function, that balances the predicted improvement (exploitation strategy) and the uncertainty of the predictions (exploration strategy), is applied to the model. An optimization algorithm is applied to find the maximum of this acquisition function. The set of parameters that gives this maximal value for the acquisition function determines the next experiment (chemical reaction) to run. This experiment is run, its result is added to the dataset, and the algorithm starts a new iteration. The algorithm stops when the objectives are attained or when the experiments budget is spent.

Categorical variables are often present in the optimization of chemical reactions [11]. We can cite as an example the choice of a catalyst or additives, the choice of the solvent or the order of addition of the reactants. Categorical variables have two important particularities. The first one is the non-continuity constraint, since categorical variables are not defined on a continuous space. The second one is the non-ordinality constraint: they can only be compared with the equality operator. For example, with a categorical variable representing a choice between three solvents $water, ethanol, toluene$ asserting that $water > toluene$ is meaningless.

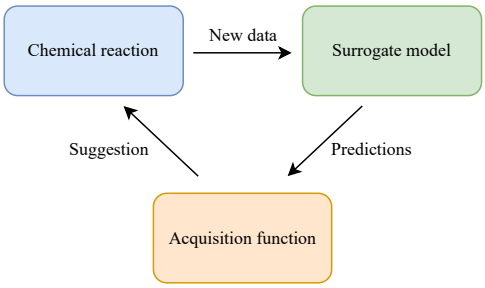


Fig. 1: Simplified Bayesian optimization algorithm applied to chemical reactions.

Mixed-variable optimization can be handled with one-hot encoding: a categorical variable with n categories is encoded as a vector of n corresponding bits, with all bits being equal to 0 except the bit corresponding to the selected category, that is equal to 1. However, in the BO algorithm, treating one-hot dimensions as continuous without any supplementary treatment misleads the acquisition function optimizer and often results in a sub-optimal solution. Indeed, the experiment proposed by the acquisition function optimizer is a real-valued vector and has to be decoded to the closest category. Hence, most of the time, there will be a gap between the experiment suggested by the acquisition function optimizer and the experiment that will actually be performed, leading to a mediocre optimization performance.

The work presented by Garrido-Merchán *et al.* [3] brings an improvement to the basic one-hot encoding approach. During the optimization of the acquisition function, real-valued encoded vectors are transformed to the nearest one-hot vectors before being used as inputs of the model. It follows that the acquisition function optimizer considers real-valued vectors as having the same acquisition values as the associated transformed vectors. Thus, the acquisition optimizer suggests an experiment that can be performed as is, which ensures the convergence to optimal solutions.

Häse *et al.* [4] have developed an augmented Bayesian optimization algorithm called Gryffin that uses a Bayesian neural network as surrogate model. It estimates kernel densities, based on previously evaluated experiments, that are used to approximate the objective function. Griffyn is able to use expert knowledge (descriptors) to guide the optimization, which drastically improves the performances of their method. Its "naive" version doesn't use descriptors, which enabled us to use it in our benchmarks.

COCABO [12] is a Bayesian optimization method designed for mixed-variable optimization. At each iteration, COCABO first selects categories with a multi-armed bandit algorithm and then separately optimizes the numerical variables (after modelling them using a mixed covariance function).

Random forests inherently handle categorical variables and can be used as surrogate model in the Bayesian optimization algorithm [5]. A ready-to-use implementation of this algorithm is provided in a package called SMAC [7].

In this study, we aim at improving the performances (i.e. reducing the number of experiments necessary to reach an optimum) of the Bayesian optimization method for the optimization of chemical reactions with continuous and categorical variables. Our approach is based on Gaussian processes as surrogate models with the COCABO covariance function [12]. We propose different techniques for the optimization of the acquisition function. Next, we compare the different acquisition function optimizer on the optimization of simulated chemical reactions. And finally, we compare our optimization algorithm (using the COCABO covariance function and the highest-performing acquisition function optimizer) with other state-of-the-art algorithms.

2 Problem definition

Our work is applied to problems with a form given by:

$$\text{Minimize } f(\mathbf{z}) \text{ with the smallest possible number of evaluations} \quad (1)$$

where :

- $\mathbf{z} = (\mathbf{x}, \mathbf{h})$
- $\mathbf{x} = x_0, .., x_n$ and $x_i \in [A_i, B_i]$ with $A_i, B_i \in \mathbb{R}$
- $\mathbf{h} = h_0, ..., h_n$ and $h_i \in C_i$ with C_i denotes the categorical space of the ith categorical variable.

This work is restricted to single objective optimization. Moreover, only continuous and categorical variables are used.

The "No-Free Lunch Theorem" [17] stipulates that the performances of every optimization methods are equal when averaged on all possible problems. It implies that in order to increase the performances on a specific optimization problem (e.g. chemical reaction optimization), we must evaluate the optimization method on similar problems without any regards on the performances of unrelated ones. The underlying functions of chemical reactions have some particularities: they are smooth and have few local optima [8, 15]. So, in order to be specific to the chemical reaction optimization problem, we measure the performances of our approach using chemical reaction simulators. We have built these chemical reaction simulators by training machine learning models with publicly available chemical reaction data (see table 1). This benchmarking strategy was initially introduced by Felton *et al.* [2] for measuring performances on chemical reactions with continuous and categorical variables. It allows us to establish optimization performances on chemical reactions without having to run experiments in a chemistry lab.

Table 1: Details of the data used to train the simulators

Reaction type	Number of experiments	Source
Pd-catalysed direct arylation	1728	[14]
Suzuki-Miyaura cross-coupling	4 cases of 96	[11, 2]

3 Propositions

In a first part, we describe the surrogate model including the COCABO kernel and its hyperparameters. In a second part, we present different approaches for the optimization of the acquisition function.

3.1 Gaussian process kernel

We use Gaussian processes (GP) to approximate the underlying functions of chemical reactions. It is the most commonly used model since it can inherently predict both a value and an associated uncertainty. Gaussian processes are mainly defined by their covariance function. Since the underlying functions of chemical reactions are smooth, we use a smooth covariance function, Matérn$_{5/2}$ [10], for the continuous dimensions.

The smoothness of the GP on continuous variables is kept with the use of the one-hot encoding. However, the Euclidian distance used for the calculation of the Matérn$_{5/2}$ kernel is based on all dimensions (continuous and encoded). We believe that, in order to catpure complex relationships between categorical and continuous variables, the covariance function should use the Euclidian distance only on continuous variables and incorporate categorical knowledge later in its calculation. The COCABO method [12] uses such a covariance function (see equation 2). It combines two sub-functions: one for continuous variables, K_{cont}, and one for categorical variables, K_{cat}.

$$\begin{aligned} K(\mathbf{z}, \mathbf{z}') = (1-\lambda) \times (K_{cont}(\mathbf{x}, \mathbf{x}') \times K_{cat}(\mathbf{h}, \mathbf{h}')) \\ + \lambda \times (K_{cont}(\mathbf{x}, \mathbf{x}') + K_{cat}(\mathbf{h}, \mathbf{h}')) \end{aligned} \tag{2}$$

where :

- $\mathbf{z} = (\mathbf{x}, \mathbf{h})$
- $\mathbf{x}$ is the set of continuous variables
- $\mathbf{h}$ is the set of categorical variables

K_{cont} is the Matérn$_{5/2}$ function. It is a standard covariance function for smooth Gaussian processes regressions with continuous inputs. K_{cat}, the kernel for categorical inputs (see equation 3), measures similarity between categorical vectors with the equality operator (which is the only permitted operation for categorical variables).

$$K_{cat}(\mathbf{h}, \mathbf{h}') = \sigma \times \frac{1}{D} \sum_{1}^{D} \alpha(h_d, h'_d) \tag{3}$$

where:

- $\alpha(a, b)$ equals 1 if $a = b$ and 0 if $a \neq b$
- D is the number of categorical variables
- σ is the variance hyperparameter.

The proposition made by Ru *et al.* in COCABO [12] revolves around the hyperparameter λ, which is a trade-off between the two terms of the equation 2: the sum and the product of K_{cont} and K_{cat}. Both of these terms capture different relationships between continuous and categorical variables. The sum of the two sub-kernels produces a learning of a single trend on the continuous variables and shift this trend depending on the categories whereas the product is able to produce a learning of complex relationships with highly different trends depending on the categories. The sum is especially necessary when the amount of training data is low (beginning of the optimization) because the product is able to capture knowledge only if the evaluations have categories in common. For example, if two evaluations have the same continuous features but different categorical ones, the product will be equal to 0 which prevent the model to learn even on continuous variables. Nonetheless, the product is essential because, as the optimization goes on, more evaluations are added to the training dataset and a single trend with a simple shift will not be sufficient to model the complexity offered by the data. In other words the sum alone will not be able to capture all the knowledge available to guide the optimization. With the hyperparameter λ, the authors ensure that the relationships that can be captured either by the sum or by the product are taken into account into the covariance $K(\mathbf{z}, \mathbf{z}')$, because λ is tuned during the fitting of the Gaussian process.

In order to avoid underfitting/overfitting the data while training the Gaussian process (tuning its hyperparameters to minimize its negative log marginal likelihood [10]), we confined hyperparameter values within a range. σ_K, $\sigma_{K_{cont}}$ and $\sigma_{K_{cat}}$ were bounded in $[10^{-2}, 20]$ while the lengthscale parameter of K_{cont} and λ were respectively bounded in $[10^{-2}, 20]$ and $[0.1, 0.9]$. We used the L-BFGS optimizer to tune the GP hyperparameters.

3.2 Acquisition function optimization

We chose to use the Expected Improvement (EI) acquisition function because it has shown good results on diverse applications and has a strong theoretical support [20]. The equation of Expected Improvement is given by:

$$\mathrm{EI}(\mathbf{x}) = \mathbb{E}[\max(f(\mathbf{x}) - f(\mathbf{x}^+), 0)] \tag{4}$$

with $f(\mathbf{x}^+)$ the value of the evaluation that have yielded the best result so far. The analytical form of EI is the following:

$$\mathrm{EI}(\mathbf{x}) = \begin{cases} \sigma(\mathbf{x}) Z \Phi(Z) + \sigma(\mathbf{x}) \phi(Z) & \text{if } \sigma(\mathbf{x}) \neq 0 \\ 0 & \text{if } \sigma(\mathbf{x}) = 0 \end{cases} \tag{5}$$

where

$$Z = \frac{\mu(\mathbf{x}) - f(\mathbf{x}^+) - \xi}{\sigma(\mathbf{x})} \tag{6}$$

$\Phi(Z)$ and $\phi(Z)$ denotes respectively the cumulative distribution function (CDF) and the probability density function (PDF) of the variable Z. Z denotes the predicted improvement divided by the standard deviation (uncertainty) and the parameter ξ determines the weight of the exploration strategy in the equation. This analytical form of EI is cheap to evaluate and can be optimized without sparing on the number of evaluations. Therefore, we propose several approaches for the optimization of the acquisition function with mixed variables.

The first approach (denoted as L-BFGS-OHE) involves a one-hot encoding of the categorical variables and a multi-started gradient descent for the optimization of the acquisition function. However, since the COCABO model do not accept one-hot vectors, one-hot dimensions are systematically decoded before any predictions. In other words, predictions are asked for by the acquisition function optimizer with encoded inputs but they are decoded before they pass through the model. The multi-started gradient descent is performed as follows: 1000 configurations are randomly drawn and the 5 configurations with the highest acquisition function value are kept and a gradient descent (L-BFGS) is performed on each of these 5 configurations.

We also propose an approach based on a "brute-force" optimization of the categorical space and a multi-started gradient descent on the continuous space (see Algorithm 1). First, all the combinations of the categorical parameters are constructed. Then, for each combination, a multi-started gradient descent (previously described) is performed on the continuous parameters. Finally, after determining the maximal acquisition values for each categorical combination, the configuration with the highest acquisition value is suggested as the next experiment. This algorithm reduces the difficulty of the optimization of the acquisition function because instead of dealing with different types of variables (or with supplementary dimensions from the encoding), the acquisition optimizer only works on the continuous dimensions. Still, it can be heavy in terms of computational cost if the number of categories and categorical variables is large.

Algorithm 1 Categorical brute-force and multi-started gradient descent

1: Construct all categorical combinations
2: Multi-started gradient descent optimization of continuous parameters for each combination
3: Choose as suggestion the configuration (continuous and categorical) with the highest acquisition

Lastly, we implemented an evolutionary algorithm based on ant colonies (ACO) that can handle categorical variables [6]. In our experiments, we used the colony hyperparameters proposed by the authors without any restart allowed.

This algorithm is a multi-agent method inspired by the behaviour of ants. An ant represent an evaluation at a given set of parameters. At each generation, each ant randomly moves towards previously evaluated points with good results (exploitation strategy). The presence of multiple ants in the colony and the randomness of their movements enable the mandatory exploration of the search space. It allows the ants to not only moves around promising areas but also randomly explore areas that may have not been explored so far.

4 Results

For each optimization algorithm, we performed 25 runs of 55 experiments each. At each run, we randomly drew 5 initial evaluations and, for a fairness purpose, these 5 evaluations were used to initialize all the optimizers.

First, we compare the performances of different acquisition optimization techniques (using the COCABO kernel and the EI acquisition function).

The figures 2a and 2b compare the performances of the acquisition optimizers. They correspond respectively to a simulation of the direct arylation reaction which contains 3 categorical variables and 2 continuous ones, and to a simulation of the Suzuki-Miyaura reaction (case 1) which contains 1 categorical variables and 3 continuous ones.

In both cases, ACO performs poorly compared to the two other methods. Still, its performances are closer to the two other optimizers than the random strategy so it will be the subject of further work to exploit the potential of the ACO method.

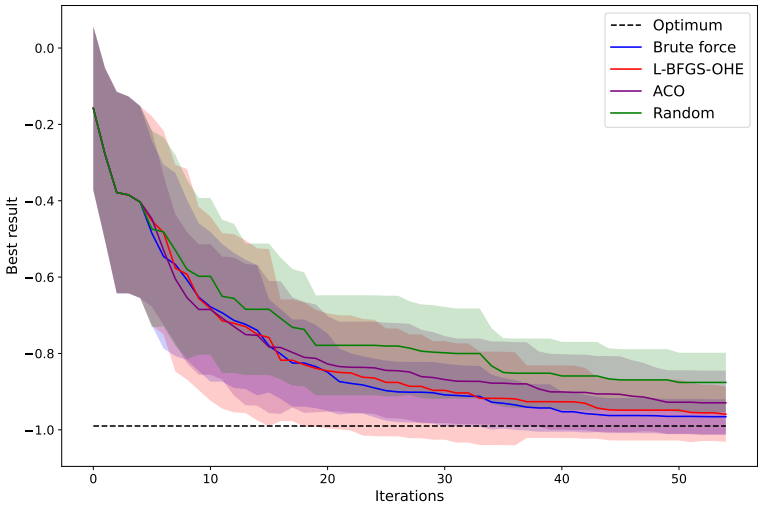
In the figure 2a, the brute-force and the L-BFGS algorithm with one-hot encoded categorical variables give similar results but in the figure 2b, brute-force performs slightly better. Overall, the brute-force approach offers the best performances with the steepest average convergence rate and the lowest standard deviation (filled area).

As consequence of the results presented above, we chose the brute-force approach to be the acquisition function optimizer in the rest of our study.

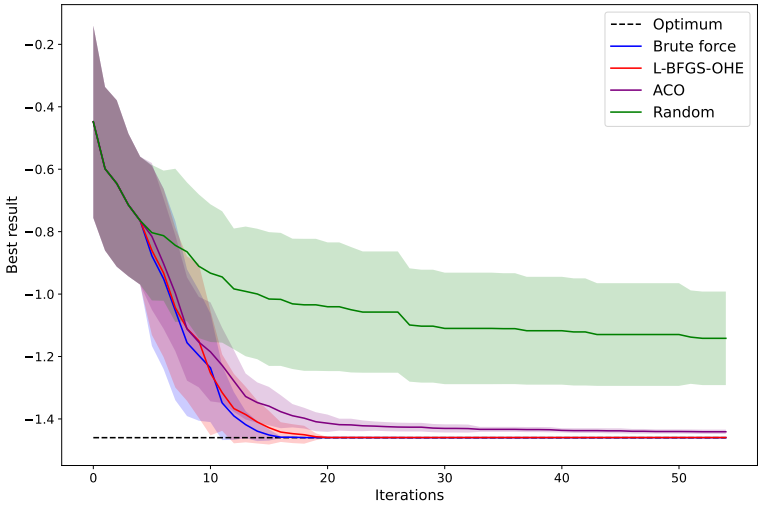
The next results present a comparison between our method (composed by the COCABO kernel, Expected Improvement and the brute-force optimizer), Gryffin [4], COCABO [12], SMAC [5], and the work of Garrido-Merchán *et al.* [3].

We used the "naive" version of Gryffin in its authors' implementation. We used COCABO in its authors' implementation with its default settings and a starting $\lambda = 0.5$. SMAC denotes an optimization algorithm based on Random Forest [5] and the Expected Improvement acquisition function. We used an implementation proposed by Lindauer *et al.* [7].

In Figures 3a and 3b, the Bayesian optimization with the COCABO kernel and the categorical brute-force optimization of the acquisition function gives the best results: it generally converges faster to the optimum than other methods.

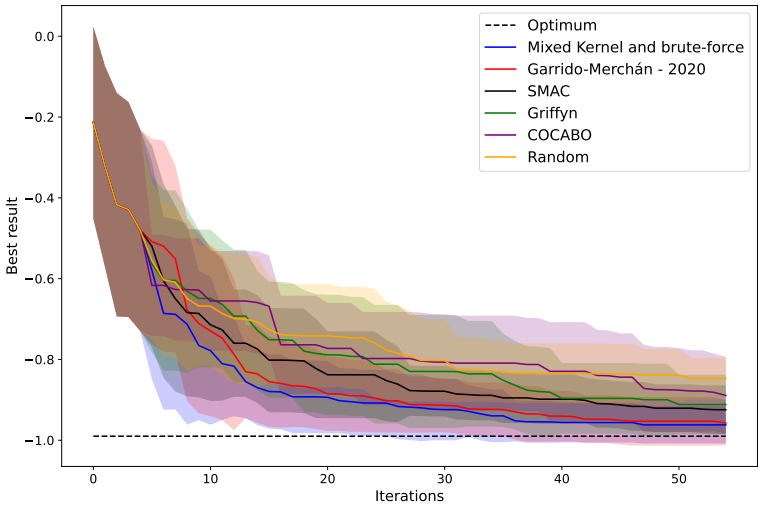


(a) Benchmark function: Pd-catalysed direct arylation simulation

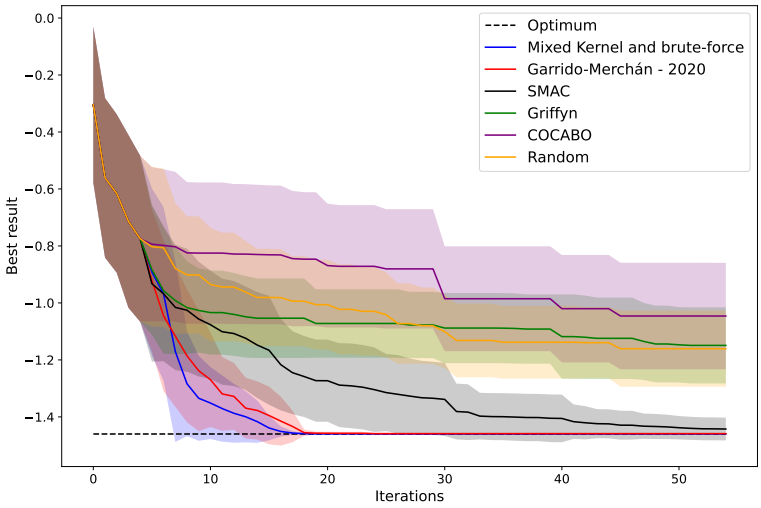


(b) Benchmark function: Suzuki-Miyaura simulation

Fig. 2: Best score evolution on simulations with the use of different acquisition function optimizers (brute-force, ACO, L-BFGS-OHE). A random optimization strategy of each chemical reaction simulation is given.



(a) Benchmark function: Pd-catalysed direct arylation simulation



(b) Benchmark function: Suzuki-Miyaura simulation (case 1)

Fig. 3: Best score evolution on simulations with different optimization methods

SMAC's performances show us that it can handle categorical variables but, overall, it performs poorly compared to the two Bayesian optimization with Gaussian process as surrogate model.

The COCABO method fails to optimize the Suzuki-Miyaura simulation. The multi-armed bandit (MAB) part of the COCABO method is designed for multiple categorical variables with multiple categories and the Suzuki-Miyaura simulation has only one categorical variable.

Our algorithm ("Mixed kernel and brute-force") performs slightly better than the work of Garrido-Merchán *et al.*. The main difference between the two methods is the use of different covariance functions. The COCABO kernel is able to capture more complex relationships than a standard $\text{Matérn}_{5/2}$ function on a one-hot encoded space.

5 Conclusion

This paper presents a method for the optimization of chemical reactions with mixed variables (continuous and categorical).

We expose a Bayesian optimization algorithm based on a Gaussian process with a covariance function specifically designed for continuous and categorical variables [12]. Also, we evaluate different methods for the optimization of the acquisition function and show that a brute-force approach associated to a multi-started gradient descent performs best. This approach performs globally better than other state-of-the-art methods [4, 3, 12] on two simulated chemical reactions with categorical and continuous inputs.

We are working on further increasing the quality of the model by modifying the covariance function. Also, in order to fully establish the performance of the presented method, later works will imply experimental validation in chemistry labs.

Acknowledgement: This work was supported by the R&D Booster SMAPI project 2020 of the Auvergne-Rhône-Alpes Region.

References

1. Carlson, R., Carlson, J.E.: Design and optimization in organic synthesis. Elsevier (2005)
2. Felton, K., Rittig, J., Lapkin, A.: Summit: Benchmarking machine learning methods for reaction optimisation (2020)
3. Garrido-Merchán, E.C., Hernández-Lobato, D.: Dealing with categorical and integer-valued variables in bayesian optimization with gaussian processes. Neurocomputing **380**, 20–35 (2020)
4. Häse, F., Aldeghi, M., Hickman, R.J., Roch, L.M., Aspuru-Guzik, A.: Gryffin: An algorithm for bayesian optimization of categorical variables informed by expert knowledge. Applied Physics Reviews **8**(3), 031406 (2021)

5. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: International conference on learning and intelligent optimization. pp. 507–523. Springer (2011)
6. Liao, T., Socha, K., de Oca, M.A.M., Stützle, T., Dorigo, M.: Ant colony optimization for mixed-variable optimization problems. IEEE Transactions on Evolutionary Computation **18**(4), 503–518 (2013)
7. Lindauer, M., Eggensperger, K., Feurer, M., Biedenkapp, A., Deng, D., Benjamins, C., Ruhkopf, T., Sass, R., Hutter, F.: Smac3: A versatile bayesian optimization package for hyperparameter optimization. In: ArXiv: 2109.09831 (2021), https://arxiv.org/abs/2109.09831
8. Moore, K.W., Pechen, A., Feng, X.J., Dominy, J., Beltrani, V.J., Rabitz, H.: Why is chemical synthesis and property optimization easier than expected? Physical Chemistry Chemical Physics **13**(21), 10048–10070 (2011)
9. Morgan, S.L., Deming, S.N.: Simplex optimization of analytical chemical methods. Analytical chemistry **46**(9), 1170–1181 (1974)
10. Rasmussen, C.E., Nickisch, H.: Gaussian processes for machine learning (gpml) toolbox. The Journal of Machine Learning Research **11**, 3011–3015 (2010)
11. Reizman, B.J., Wang, Y.M., Buchwald, S.L., Jensen, K.F.: Suzuki–miyaura cross-coupling optimization enabled by automated feedback. Reaction chemistry & engineering **1**(6), 658–666 (2016)
12. Ru, B., Alvi, A., Nguyen, V., Osborne, M.A., Roberts, S.: Bayesian optimisation over multiple continuous and categorical inputs. In: International Conference on Machine Learning. pp. 8276–8285. PMLR (2020)
13. Shahriari, B., Swersky, K., Wang, Z., Adams, R.P., De Freitas, N.: Taking the human out of the loop: A review of bayesian optimization. Proceedings of the IEEE **104**(1), 148–175 (2015)
14. Shields, B.J., Stevens, J., Li, J., Parasram, M., Damani, F., Alvarado, J.I.M., Janey, J.M., Adams, R.P., Doyle, A.G.: Bayesian reaction optimization as a tool for chemical synthesis. Nature **590**(7844), 89–96 (2021)
15. Tibbetts, K.M., Feng, X.J., Rabitz, H.: Exploring experimental fitness landscapes for chemical synthesis and property optimization. Physical Chemistry Chemical Physics **19**(6), 4266–4287 (2017)
16. Weissman, S.A., Anderson, N.G.: Design of experiments (doe) and process optimization. a review of recent publications. Organic Process Research & Development **19**(11), 1605–1633 (2015)
17. Wolpert, D.H., Macready, W.G., et al.: No free lunch theorems for search. Tech. rep., Technical Report SFI-TR-95-02-010, Santa Fe Institute (1995)
18. Wright, M.H., et al.: Nelder, mead, and the other simplex method. Documenta Mathematica **7**, 271–276 (2010)
19. Xiong, Q., Jutan, A.: Continuous optimization using a dynamic simplex method. Chemical Engineering Science **58**(16), 3817–3828 (2003)
20. Zhan, D., Xing, H.: Expected improvement for expensive optimization: a review. Journal of Global Optimization **78**(3), 507–544 (2020)
21. Zhou, Z., Li, X., Zare, R.N.: Optimizing chemical reactions with deep reinforcement learning. ACS central science **3**(12), 1337–1344 (2017)

Comparing policies for the stochastic multi-period dual sourcing problem from a supply chain perspective

Youssef Boulaksil*

College of Business and Economics, UAE University, Al Ain, United Arab Emirates

y.boulaksil@uaeu.ac.ae

Younes Hamdouch

College of Business, Zayed University, Dubai, United Arab Emirates

Kilani Ghoudi

College of Business and Economics, UAE University, Al Ain, United Arab Emirates

Jan C. Fransoo

School of Economics and Management, Tilburg University, The Netherlands

***Corresponding Author**

Many inventory models assume that there is only one source or transportation mode to procure, produce, or ship products. However, in practice, inventories can be replenished from more than one source or via different transportation modes. The case of two different sources or transportation modes is referred to as a dual sourcing system, and has become a popular strategy in many supply chains due to offshore production, outsourcing, and different transportation modes. The benefits of offshore sourcing include lower costs and better availability of specific workforce. However, it reduces flexibility and responsiveness in the sense that the replenishment lead times become longer. By contrast, onshore (or domestic) sourcing is faster but incurs higher costs. The resulting dual sourcing setting, that is, the trade-off between utilizing offshore production and an onshore source is the motivation for this study.

We study a supply chain that consists of a buyer and two suppliers. The buyer faces stochastic demand and has two different supply sources for the same product: a slower regular supplier and a more expensive expedited supplier. Such dual sourcing inventory systems have been widely studied in the literature, evaluating what is best for the buyer. As

the decisions of the buyer may adversely affect the costs of the suppliers, and also, potentially, the supply chain profit, we adopt the perspective of the entire supply chain. We compare the performance of two different policies in a multi-period and two-echelon setting: the Dynamic Order Policy (DOP) and the Standing Order Policy (SOP). In the long-run, the DOP policy converges to the Dual-Index Policy (DIP), which is optimal for the buyer in the case of a lead time difference of one period. On the other hand, the SOP policy is appealing from a practical perspective as a fixed (standing) quantity is ordered from the regular supplier in each period. We show that from our supply chain perspective, the DOP policy is not necessarily the better performing policy, and we define the conditions for the preferred policy. We evaluate the policies numerically under various conditions to obtain relevant managerial insights. We find that the preferred policy from a supply chain perspective is mainly the result of a trade-off between responsiveness, flexibility, and cost. Flexibility appears to be valuable if there is a substantial cost difference between the suppliers.

In a single-echelon system, that is, from the buyer's perspective, the DOP policy, which converges to DIP in the long-run, is known to be optimal and will obviously outperform SOP (in case of a lead time difference of one time period). However, from a supply chain perspective, we show that this is not necessarily the case. We define the conditions for the preferred policy and we demonstrate that in several cases, SOP outperforms DIP in a supply chain setting (with a lead time difference of one period), particularly when the wholesale price difference is small, the level of demand uncertainty is high, the unit inventory holding and backorder cost is high, and when the unit profit margin of the expedited supplier is large. This result is not only interesting from a theoretical perspective, but also has significant practical implications. In general, we find that the flexibility offered by the DOP policy is only valuable if there is a substantial cost difference between the suppliers. Otherwise, the SOP policy becomes a legitimate policy alternative, especially given its simple structure.

DEEP REINFORCEMENT LEARNING BASED HOME ENERGY MANAGEMENT SYSTEM (HEMS)

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

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

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

ABSTRACT

Increasing penetration of renewable energy sources (PV, Wind) due to environmental constraints, impose several technical challenges to power system operation. The fluctuating and intermittent nature of wind and solar energy requires constant supply-demand balance for electric grid stability purposes.

Self-consumption is a regulatory framework intended to promote local consumption over export. Thus, self-consumption will raise the profit of PV electricity from grid-connected residential systems and lower the stress on the electricity distribution grid.

This work presents a novel Deep Reinforcement Learning (DRL) Based Energy Management System (EMS) to control a Home Microgrid system powered by renewable energy sources (PV arrays) and equipped with an energy storage system. An optimal energy scheduling is carried out to maximize the benefits of available renewable resources through self-consumption. A DRL approach is used to make optimal decisions and generate the optimal management strategies.

Keywords: Microgrid ; Smart Grid ; Deep Reinforcement Learning ; Self-consumption ; Optimal scheduling ; Optimization.

SELECTED REFERENCES

[1] François-Lavet, V., Taralla, D., Ernst, D., & Fonteneau, R. (2016). Deep reinforcement learning solutions for energy microgrids management. In *European Workshop on Reinforcement Learning (EWRL 2016)*.

[2] Nakabi, T. A., & Toivanen, P. (2021). Deep reinforcement learning for energy management in a microgrid with flexible demand. *Sustainable Energy, Grids and Networks*, *25*, 100413.

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

[4] Luthander, R., Widén, J., Nilsson, D., & Palm, J. (2015). Photovoltaic self-consumption in buildings: A review. *Applied energy*, *142*, 80-94.

[5] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

[6] Li, Y. (2017). Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*.

[7] Powell, W. B. (2007). *Approximate Dynamic Programming: Solving the curses of dimensionality* (Vol. 703). John Wiley & Sons.

Decentralizing and Optimizing Nation-Wide Employee Allocation while Simultaneously Maximizing Employee Satisfaction

Aniqua Tabassum[0000−0003−1727−7252], Ashna Nawar Ahmed[0000−0003−1412−7953], and Subhe Tasnia Alam[0000−0002−0728−4371] Noushin Tabassum[0000−0002−9664−7153] Md. Shahriar Mahbub[0000−0001−8629−3980]

Ahsanullah University of Science and Technology, Dhaka, Bangladesh
{aniqua.tabassum337, ashna.nawar.ahmed, subhe321tasmia, noushintabassum89}@gmail.com, shahriar.cse@aust.edu

Abstract. Despite the studies on human resource allocation, a problem of dissatisfied employees arises in developing and under-developed countries while decentralizing human resources nationwide since rural areas have fewer facilities than urban areas. Randomly allocating employees contributes to employees' dissatisfaction if they are displeased with where they are assigned, leading to an unstable work environment. However, allocating employees solely based on their satisfaction may lead to a centralized solution around the urban cities. Therefore, employee satisfaction and dispersion are the two most essential but opposing factors for employee decentralization in developing countries.
In this study, we have addressed the problem of employee decentralization by proposing a Multi-Objective Optimization approach that maximizes the two conflicting objectives: employee satisfaction (ES) and employee dispersion (ED). A neural network is applied that predicts the ES of an employee allocated in an area/city. Moreover, we have formulated a dispersion function that provides a score based on how well dispersed a specific allocation is. Using a Multi-Objective evolutionary algorithm, we have developed an allocation framework that maximizes these conflicting objectives and finds optimal allocations.

Keywords: Employee Dispersion · Employee Satisfaction · Human Resource Allocation in Developing Countries · Multi-Objective Evolutionary Algorithm

1 Introduction

Non-optimal worker allocation in developing countries is a crucial problem to be solved. There are studies on identifying the problems regarding resource allocation and on finding the optimal solutions [1, 2]. However, these studies do not address the problem of declining employee satisfaction and productivity [3] when the workforce is decentralized throughout developing and underdeveloped countries. In developing countries, such as Bangladesh [4], rural areas are primarily

under-developed as compared to urban areas, motivating the majority of people to move to cities for their livelihood [5]. Consequently, the workforce tends to be centralized around cities [6]. These problems are more observed explicitly among essential workers such as doctors [7], who are often forced to work in rural areas [8]. Most people designated to such areas show acute urgency to move away posthaste [9] owing to extreme dissatisfaction, causing a higher turnover rate and an unstable work environment [10]. On the other hand, the critical problem with allocation solely based on employee satisfaction is that the majority of the professionals would be more interested in working in the more facilitated areas [5]. Thus, the density of worker allocation would be skewed in favor of regions with more amenities, which will result in a vacuum of professionals in underdeveloped regions. Therefore, we need to implement a method to ensure the proper balance between worker dispersion and satisfaction.

To tackle the problem, we first predict employee satisfaction allocated to a particular area/city using Multi-Layer Perceptron (MLP) with nine influencing factors. In addition to this, we formulated a dispersion function that calculates the decentralization score of an allocation[1]. These two objectives are contradictory, especially for developing countries such as Bangladesh; hence, we use Multi-Objective Optimization to simultaneously maximize these objectives.

2 State of the Art

There have been multiple studies on resource allocation, and human resource allocation [1, 2]. The author of [1] has proposed a multi-objective optimization solution aiming at minimizing total cost resulting from resource overallocation, project deadline exceedance, and day-by-day resource fluctuations. In the study [2], the author has shown how the Multi-Objective Particle Swarm Optimization Algorithm can be used for multi-criteria human resource allocation. Another study [11] focuses on minimizing total cost and the total time of logistic relief operation in emergencies.

Several studies can be found that focus on worker and resource distribution in countries such as Nigeria[12] and China[13]. The author of [12] focused on the factors that hinder recruitment and retention of the healthcare workforce, such as insufficient infrastructure, inadequately trained staff, sub-optimal distribution of healthcare workers, mainly in the rural areas, and suggested approaches to improve this situation in Nigeria. The authors of [13] pointed out the ethical flaws in various Government policies leading to the inequality in resource distribution and have proposed countermeasures that optimize resource allocation, such as formulating better policies, strengthening the responsibilities of both governmental and public financial investments, improving the utilization of resources.

Two works are found that focused on the satisfaction of customers[14] and employees[15]. The author of [14] proposed a Satisfaction Function (SF) which is

[1] An allocation refers the assignments of m number of employees to n number of cities.

based on the customers' attitude regarding the products the company is offering. The authors of [15] proposed to make the work environment suitable enough such that the employees can have more decision-making power which will increase sustainable practices and improve employee satisfaction. Another study [16] focused on finding the relationship between employee engagement and job satisfaction. They have also extracted seven job satisfaction factors, such as work culture and fairness at work. In [17], employee satisfaction has been predicted by a machine learning model by analyzing employees' reviews.

To best our knowledge, although there have been several works on human resource allocation, the problem of maximizing human resource dispersion (i.e., decentralization) while also maximizing employee satisfaction in their newly designated area has not been addressed yet. There have been studies regarding identifying job-satisfaction influencing factors inside organizations [17], but none of these studies focuses on identifying the influencing factors of employee satisfaction when they are allocated to new geographical areas.

3 Methodology

To maximize the two conflicting objectives of the problem: employee satisfaction and dispersion, we formulate the problem as a multi-objective optimization problem. We develop a learning model for our satisfaction predictor and formulate the dispersion function. Finally, optimized solutions are found by using a multi-objective evolutionary algorithm. The details of our methodology are presented in this chapter.

3.1 Problem Formulation

Please consider that we want to allocate m number of people to n number of areas. We have to find optimal solutions that maximize the total satisfaction of m people while fulfilling all n cities' demands of employees as much as possible. In order to solve this, we use the following two inputs in our optimization framework:

- **Cities or areas where we want to allocate our employees:** The number of vacancy each of these cities have for a given profession.
- **The people we want to allocate to different cities:** Individual description of each of these people as required by the SF (more description in 3.4).

In order to represent the first input, we considered an array where the indexes represent the cities and the values in the indexes represent the maximum capacity or requirement/demand of the corresponding city has for a specific profession. We call this our **"Capacity Array"**. We demonstrate a figure representing our Capacity Array in Fig. 1, where $a_0 - a_5$ denotes the considered six areas. For example, area number 0, or a_0 has six vacancies, a_1 has seven vacancies, and so on. Similarly, our second input is represented by an array that we call "Employee Array", where each value represents the information of an individual employee.

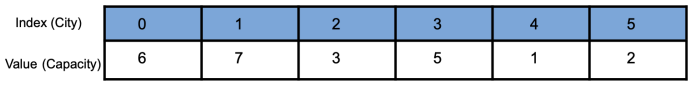


Fig. 1: Capacity Array

3.2 Problem Representation

We represent the problem as an array of size m, where the value of the specific element of the array can be 1 to n. For example, a value of 5 at the index 1 of the array means employee number 5 is assigned to city number 1. An illustration is shown below in Fig. 2.

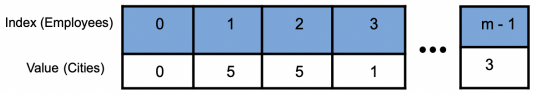


Fig. 2: Output of Optimization Framework

3.3 Overview of the Objectives

In this section, we have briefly described both of our objectives: employee satisfaction and dispersion.

- **Satisfaction:** The first objective is the maximization of employee satisfaction. Satisfaction of one employee refers to a number denoting how satisfied the employee is, on a scale of 1 to 5, when he/she is assigned to a certain area. If we consider that there are m employees, and if we denote individual satisfaction as s_m then for an allocation, the total satisfaction, S is

$$S = \sum_{i=1}^{m} s_m \tag{1}$$

- **Dispersion:** Our second objective is to increase the dispersion of a solution, which refers to allocating as well-spread as possible. A trend in developing countries[18] employees are satisfied only when they are designated to developed regions. Therefore, maximizing satisfaction reduces dispersion.

3.4 Modelling Satisfaction Function

In this section, we will discuss how features for our machine learning model is selected for the satisfaction prediction of an employee. A social experiment is conducted for feature selection and the data collection process. Afterward, machine learning algorithms are used for modeling employee satisfaction.

Feature Selection In order to design the SF, we first needed to explore the factors/features that influence satisfaction. To ensure an unbiased process, we performed a social experiment by randomly choosing 20 unrelated people. We asked them one question: which factors would affect their living satisfaction in an area outside the capital. We completed the experiment without the respondents knowing which parts of their answers we would use in our research to avoid any bias. We present the results below.

- No one would move to any city with high crime rates.
- Seventeen of them would be happier if closer to their families.
- Three of the five married people did not want to move without their spouses.
- Four out of the five married people additionally mentioned that having good schools in their area would be vital to them.

Upon introducing the topic of average house rent of the city, as some cities have higher house rent, all of them agreed to this being a significant issue to consider. Their answers were varied based on gender, age, and occupation. Therefore, nine factors/features are chosen from this social experiment to use in our data collection process: gender, age, occupation, security, house rent, travel time from hometown, schooling, marital status, and spouses' willingness to relocate with their partners.

Creating Questionnaires There being no curated dataset to train our satisfaction predictor, we created our own dataset by circulating questionnaires. We have categorized gender, age, occupation, marital status, and spouses' willingness to relocate with their partners as demographic factors. The rest four factors are scenario-based.

These remaining four scenario-based factors have three levels: low, medium, and high. The respondents have been presented with a scenario by combining the different levels of the four features and asked to provide a satisfaction score.

There are a total of 3^4 or eighty-one possible scenarios. It was impractical to ask to score each respondent all 81 virtual scenarios. Therefore, we had to carefully partition the larger set of eighty-one scenarios into smaller groups, each consisting of three unique scenarios. Thus, $(81/3) = 27$ sets of questionnaires were formed, with each one consisting of identical demographic questions and three unique scenario-based questions. The three scenarios in each set were noticeably different enough, so the respondents could easily differentiate them. A sample set of three virtual scenarios representing three areas is as follows:

- low security, medium schooling facilities, low house rent, medium travel time from hometown
- medium security, low schooling facilities, medium house rent, low travel time from hometown
- high security, high schooling facilities, low house rent, high travel time from hometown

We asked the participants to rate their satisfaction on a scale of 1 to 5 for each scenario.

It was vital to evenly distribute these 27 sets among the respondents as we want to collect respondents' satisfaction for all data points. Therefore, we created a website where we uniformly sampled the scenarios presented to users to ensure balanced distribution. We have collected 855 data points in total.

Necessity of Learning Models/algorithms It is necessary to predict employee satisfaction for our allocation framework. SF modeling problem can be considered as a regression problem. However, satisfaction cannot be represented as a linear combination of its features because it is a complicated and non-linear psychological issue that can vastly vary even within the same demographic. These correlations can be identified by advanced machine learning models. Therefore, We have experimented with various machine learning regression algorithms (presented in section 4.1) to see which one performs best for our case.

3.5 Modelling Dispersion Function

The task of the dispersion function (DF) is to measure the distribution of workers in different areas/cities in a numeric value. If there are four cities and workers are assigned in two cities while there is no allocation in the other two cities, the dispersion value will be lower than the scenario where workers are assigned in all four cities. We also formulated the function so that it has a relation with the number of required personnel in each area. We have denoted the dispersion of an occupation p by $D(p)$, and the function can be represented as follows:

$$D(p) = |R^{(p)}| + \sum_{i=1}^{n} d_i^p \tag{2}$$

Here,

$$R^{(p)} = \{\ i \mid 0 \leq i\ \&\ \ al_i^p \geq minreq_i^p\} \tag{3}$$

$$minreq_i^p = \left\lfloor \frac{t^p * \frac{c_i^p}{\sum_{j=1}^{n} c_j^p} * 100}{100} \right\rfloor \tag{4}$$

$$= \left\lfloor t_p * \frac{c_i^p}{\sum_{j=1}^{n} c_j^p} \right\rfloor \tag{5}$$

$$d_i^p = \begin{cases} \frac{al_i^p}{c_i^p}, & \text{if} \quad al_i^p \leq c_i^p \\ 1 - \frac{(al_i^p - c_i^p)}{c_i^p}, & \text{otherwise} \end{cases} \tag{6}$$

Where,
n = Carnality of the set of areas brought into consideration
$R^{(p)}$ = Set of regions/areas where minimum requirement of allocation of profession p have been met
d_i^p = dispersion of an area i of profession p

$minreq_i^p$ = Minimum number of employees of profession p that needs to be allocated in area i
al_i^p = number of person allocated of profession p in area i
c_i^p = Capacity or requirements of number of profession p in area i
t^p = total number of available employees of profession p

The dispersion of an area d_i is the ratio between the number of allocated persons and the requirements. If the number of allocations met requirements, we would achieve area-wise maximum distribution, which is 1. However, the algorithm may allocate more people than the areas' requirements. Then, we subtract a penalty from the maximum dispersion.

Moreover, in most developing countries, there is a shortage of skilled employees. Therefore, the total number of required people ($\sum_{j=1}^{n} c_j^p$) is higher than the available number of people (t_p). Therefore, we added a secondary mechanism (i.e., $minreq$) to calculate the minimum possible people the algorithm can allocate in an area given that there are t_p number of professionals.

Finally, when calculating total dispersion score, the following two factors are added:

- The number of cities/areas where the minimum number of required employees for that city/area has been allocated
- Summation of all the area-wise dispersion

The theoretical maximum dispersion value can be achieved if for each city, its exact number of $minreq$ professionals are assigned to it.

Sample example In Table 1, we have presented seven cities indexed from zero to six. We have mentioned their required number of employees, and have calculated their minimum requirements, d_i^p and if their demands have been met.

Table 1: Calculating Minimum Requirement, d_i^p and R^p

City No	Capacity	Allocation	minreq	d_i^p	R^p
0	6	4	4	0.66	0
1	10	7	7	0.7	1
2	8	6	6	0.75	2
3	10	7	7	0.7	3
4	3	3	2	1	4
5	4	3	3	0.75	5
6	8	6	3	0.75	6

Therefore, $R^{(p)} = \{0, 1, 2, 3, 4, 5, 6\}$ and $|R^{(p)}| = 7$. Thus, dispersion, $D(p) = |R^{(p)}| + \sum_{i=1}^{n} d_i^p = 7 + 5.316 = 12.316$

4 Experiments and Results

For implementing the SF, we used scikit-learn [19], a python-based machine learning framework. Implementation [2] of multi-objective optimization for finding optimum allocation using SF and DF has been done using jMetalPy [20],

[2] https://github.com/aniquaTabassum/Undergrad-Thesis

a python-based multi-objective meta-heuristic framework. The results of the prediction of SF of different learning algorithms are presented. Different hyper-parameter settings are explored to find the best possible settings. Finally, the results of two optimization algorithms are shown.

4.1 Results of Satisfaction Modelling

This section shows the results of different learning models. We also present some of the hyperparameter settings we have experimented with to find the best setting for the model. In Table 2 we have presented the different methods and their R2 scores [21] on training and cross validation (CV) dataset.

Table 2: R2 Score of Training and Cross Validation (CV) for Different Methods

Name	Train	CV
Linear Regression[22]	40.01%	42.1%
Linear Regression in log-space[23]	42.6%	45.3%
Random Forest[24]	74.8%	34.0%
2-Degree Polynomial Regression[25]	47.5%	48.1%
3-Degree Polynomial Regression[26]	57.8%	0.43%
Multi Layer Perceptron Regression[27]	**51.3%**	**49.7%**

We can see that MLP performs the best R2 score in both training and cross-validation datasets combined. So, MLP Regression is the best-suited method for this problem as it has the highest R2 score. Getting an R2 score higher than 0.5 is hard when the model includes human psychology because each human thinks differently, so it is typically hard to generalize it [28].

Afterward, we do hyperparameter tuning by performing several experiments with different parameter settings for the MLP model. The dataset was standardized and normalized before performing the experiments and running the iterations until convergence. We present the results in Table 3. As per Table 3,

Table 3: R2 Score of Training and Cross Validation

Batch Size	Optimizer	Activation	Layers	Neurons Per Layer	LR	Train	CV
8	**LBFGS**	**ReLU**	**3**	**(100, 8, 8)**	**0.0013**	**51.30%**	**49.74%**
8	LBFGS	tanh	3	(100, 8, 8)	0.0013	45.06%	46.83%
8	Adam	ReLU	3	(100, 8, 8)	0.0013	-80.0%	-35.0%
8	LBFGS	ReLU	3	(100, 8, 8)	0.0003	50.95%	48.89%
16	LBFGS	ReLU	5	(100, 16, 16, 8, 8)	0.0013	49.53%	49.53%
4	LBFGS	ReLU	4	(8, 8, 8, 25)	0.0013	49.9%	45.0%

we chose a 3-layer MLP regressor with ReLU activation function in the hidden layer, LBFGS optimizer, and a learning rate of 0.0013 as our model, as this performs the best on both training and cross-validation datasets.

4.2 Results of Optimization

This section presents a scenario of a doctor allocation problem in Bangladesh. We chose the medical profession for our optimization experiments since they

are categorized as one of the foremost essential workers[7] and are needed in every region of a country. However, our framework works appropriately for other professions as well. Afterward, optimization results of the problem are presented.

Scenario: A Doctor Allocation Problem Thirty-seven doctors filled out our survey, and we have applied our framework to allocate all of them in the six major cities of Bangladesh. Even with just 37 employees being allocated to 6 cities, there are 6^{37} possible allocation combinations, which is impossible to simulate using an exhaustive search. We collect the necessary information (i.e., security, house rent, travel time from hometown, schooling) for the six cities, and the features of the cities are labeled by different markers (i.e., low, medium, and high) based on the collected information. For measuring schooling facilities, the results of the Secondary School Certificate (SSC) exam [29] of the schools in each of the cities for the year 2019 [30] are considered. We took the median house rent [31] for each of these cities. Security is estimated by the total crime rate for each of these cities in 2019 calculated from the data [32]. The maximum time (data taken from Google Map [33]) required to reach one city to another is taken as the travel time between two cities. We prefer travel time over road distance (in KM) because the road distance does not reflect the real situation in Bangladesh as road traffic condition is very poor [3].

Finally, the features of the cities are ranked in low, medium, and high. A feature value fall within bottom 33.33% is "low", a value in between 33.33% and 66.66% is "medium", and a value over 66.66% is considered as "high".

Results: We compare the performance of the two most used multi-objective optimization algorithms (NSGA-II [34], and SPEA2 [35]) on our problem. Two algorithms separately ran fifty times to make a statistically significant comparison. The parameter settings for the experiment are given below in Table 4.

Table 4: Parameter Setting for Optimal Algorithm

Parameter	NSGA-II	SPEA-II
Population Size	100	100
Crossover Type	Single-point	Single-point
Crossover Probability	0.9	0.9
Mutation Type	Random Mutation	Random Mutation
Mutation Probability	0.027[4]	0.027
Max Evaluations	30000	30000

Figure 3 shows the approximated true Pareto-front of the problem. Since the true Pareto-front is unknown, we ran the algorithm 50 times, merged the results, and created an approximation of the true Pareto front. In Figure 3, point p_1 is an

[3] Road distance will perform similarly if the road traffic condition is similar to all over the region.

[4] $mutation\ probability = \frac{1}{number\ of\ decision\ variables} = \frac{1}{37} = 0.027$

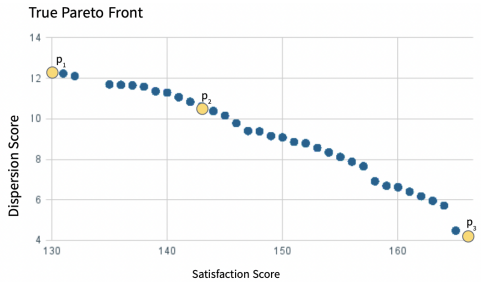
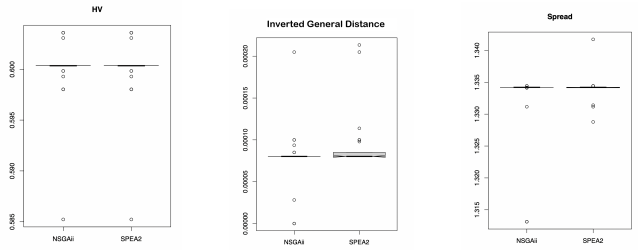


Fig. 3: Approximation of True Pareto-Front

extreme solution with the highest satisfaction value, 166 out of 185, but has the lowest dispersion value, 4.25 out of the theoretical maximum value of 12.31. This solution can be chosen when ES is far more important than ED. Point p_3 has the lowest satisfaction value, 130 out of 166, and the highest possible dispersion score, 12.31. Such a solution is ideal if achieving maximum dispersion is the goal. Point p_2 is a random solution in the Pareto-front. Here, the satisfaction score is 143, and the dispersion score is 10.61.

Boxplots of hypervolume [36, 37], IGD [36, 38] and spread [36, 39] are presented in Figure 4 where we can see that the two algorithms are producing similar results. It can be concluded that either algorithm could be chosen.



(a) HV (b) IGD (c) Spread

Fig. 4: Comparison between NSGA-II and SPEA-II

5 Conclusion

In this study, we have proposed a more practical approach for human resource allocation for developing and under-developed countries, maximizing employee satisfaction and dispersion, essential for stabilizing the workforce problem in rural areas. First, we have identified the satisfaction influencing factors for employees and have gathered a dataset accordingly to create our satisfaction predictor. Then we mathematically formulated employee dispersion to ensure fair distribution of employees. Our proposed dispersion function also considers the shortage of human resources. A multi-objective evolutionary algorithm approach is applied to find the optimal set of solutions, among which any solution can be chosen

based on the situation at hand. On our dataset, the highest satisfaction score achieved by our framework is 166 out of 185 (i.e., 37 doctors can have maximum satisfaction of 5), and the average satisfaction score of 37 employees is 4.49 out of 5.0, which is very high. Even in a solution where the dispersion score is 10.16 out of 12.31 (i.e., maximum dispersion can be calculated theoretically, please see dispersion modeling section in 3.5), the average satisfaction of 37 employees is as high as 3.9. We plan to improve our satisfaction model by feeding it more data in the future. The proposed allocation framework is generalized and can be applied for building an organized and content set of workforce, since decentralizing employees is becoming a necessity for companies as they keep expanding, be that among the cities of a country or worldwide.

References

1. Sofia Kaiafa and Athanasios P. Chassiakos. A genetic algorithm for optimal resource-driven project scheduling. *Procedia Engineering*, 123:260–267, 2015. Selected papers from Creative Construction Conference 2015.
2. Zhengyuan Jia and Lihua Gong. Multi-criteria human resource allocation for optimization problems using multi-objective particle swarm optimization algorithm. In *2008 International Conference on Computer Science and Software Engineering*.
3. Oxford University's Saïd Business School. Happy workers are 13% more productive | University of Oxford, October 2019.
4. UN list of least developed countries, February 2020.
5. International Organization for Migration. Migration: Making the Move from Rural to Urban by Choice, October 2017.
6. Monica Castillo. Rural-urban labour statistics. *20th International Conference of Labour Statisticians*, October 2018.
7. National Center for Immunization and Respiratory Diseases. Interim List of Categories of Essential Workers Mapped to Standardized Industry Codes and Titles | CDC, March 2021.
8. KumKum Dasgupta. 'Young doctors don't want to go to rural areas because they feel ill-equipped', September 2016.
9. Emmanuel Kwame Darkwa, M. Sophia Newman, Mahmud Kawkab, and Mahbub Elahi Chowdhury. A qualitative study of factors influencing retention of doctors and nurses at rural healthcare facilities in Bangladesh. *BMC Health Services Research*, 15(1), 2015.
10. Andesna Nanda, Mochamad Soelton, Sita Luiza, and Eko Tama Putra Saratian. The Effect of Psychological Work Environment and Work Loads on Turnover Interest, Work Stress as an Intervening Variable. *Proceedings of the 4th International Conference on Management, Economics and Business (ICMEB 2019)*, 2020.
11. Deepshikha Sarma, Uttam Kumar Bera, and Amrit Das. A mathematical model for resource allocation in emergency situations with the co-operation of ngos under uncertainty. *Computers & Industrial Engineering*, 137:106000, 2019.
12. Niyi Awofeso. Improving health workforce recruitment and retention in rural and remote regions of Nigeria. *Rural and Remote Health*, 10(1319), 2010.
13. Yiyi Chen, Zhou Yin, and Qiong Xie. Suggestions to ameliorate the inequity in urban/rural allocation of healthcare resources in China. *International journal for equity in health*, 13:34, May 2014.

14. Alexander C. Pereira. A MATHEMATICAL MODEL TO MEASURE CUSTOMER SATISFACTION. *Quality Engineering*, 11(2):281–286, 1998.
15. Simona Vinerean, Iuliana Cetina, and Luigi Dumitrescu. Modeling Employee Satisfaction in Relation to CSR Practices and Attraction and Retention of Top Talent. *Expert Journal of Business and Management*, 1(1):4–14, 2013.
16. Lata Singh. Job satisfaction as a predictor of employee engagement. *Amity Global HRM Review*, 7:20–30, 09 2017.
17. Sumali Conlon, Lakisha Simmons, and Feng Liu. Predicting tech employee job satisfaction using machine learning techniques sumali j. conlon1 lakisha l. simmons2 feng liu3. *INTERNATIONAL JOURNAL OF MANAGEMENT & INFORMATION TECHNOLOGY*, 16:72–88, Jun. 2021.
18. Laura Manzi. Migration from rural areas to cities: challenges and opportunities, September 2020.
19. scikit-learn contributors. scikit-learn Machine Learning in Python, August 2021.
20. jMetalPy contributors. jMetalPy, August 2021.
21. Minitab Blog Editor. Regression Analysis: How Do I Interpret R-squared and Assess the Goodness-of-Fit?, 2013.
22. Yale University. Linear Regression, 1997.
23. K. Benoit. Linear Regression Models with Logarithmic Transformations. 2011.
24. Tony Yiu. Understanding Random Forest - Towards Data Science, May 2021.
25. Abhigyan. Understanding Polynomial Regression!!! - Analytics Vidhya, August 2020.
26. StatsDirect. Polynomial Regression.
27. Wikipedia contributors. Multilayer perceptron, August 2021.
28. Callum Ballard. An Ode To R-Squared, August 2021.
29. Wikipedia contributors. Secondary School Certificate, August 2020.
30. SSC Result 2019, August 2020.
31. Bproperty. Location Based House Rent Search, August 2020.
32. Crime Statistics 2019, 2020.
33. Google Maps. Bangladesh, August 2020.
34. 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.
35. Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. volume 3242, Zürich, January 2001. Eidgenössische Technische Hochschule Zürich (ETH), Institut für Technische Informatik und Kommunikationsnetze (TIK).
36. Charles Audet, Jean Bigeon, Dominique Cartier, Sébastien Le Digabel, and Ludovic Salomon. Performance indicators in multiobjective optimization. *European Journal of Operational Research*, 292(2):397–422, 2021.
37. Anne Auger, Johannes Bader, Dimo Brockhoff, and Eckart Zitzler. Hypervolume-based multiobjective optimization: Theoretical foundations and practical implications. *Theoretical Computer Science*, 425:75–103, 2012.
38. Yanan Sun, Gary G. Yen, and Zhang Yi. IGD Indicator-Based Evolutionary Algorithm for Many-Objective Optimization Problems. *IEEE Transactions on Evolutionary Computation*, 23(2):173–187, 2019.
39. Miqing Li and Jinhua Zheng. Spread Assessment for Evolutionary Multi-Objective Optimization. In Matthias Ehrgott, Carlos M. Fonseca, Xavier Gandibleux, Jin-Kao Hao, and Marc Sevaux, editors, *Evolutionary Multi-Criterion Optimization*, pages 216–230, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

A multi-objective differential evolution approach for optimizing mixtures

G. Ramstein[1], Aman Prasad[2], and F. Tancret[2]

[1] Laboratoire des Sciences du numérique de Nantes, CNRS UMR 6241, University of Nantes, France
gerard.ramstein@univ-nantes.fr
[2] Institut des Matériaux de Nantes, CNRS UMR 6502, University of Nantes, France
aman.prasad@etu.univ-nantes.fr
franck.tancret@univ-nantes.fr

Keywords: Multi-objective optimization, differential evolution, mixture design.

1 Introduction

Mixtures are the subject of many decision-making fields such as chemistry, geology, or economy. In order to obtain a compound product with specific properties, it is necessary to determine the relative quantities of the ingredients used. For instance, the features of an alloy (*e.g.*, strength, corrosion, creep resistance) depend on the proportions of its elements. The issue we address in this paper concerns mixture design using multi-objective optimization. In this context, a solution x is made of d components x_i verifying the following constraints:

$$\sum_{i=1}^{d} x_i = 1 \tag{1}$$

$$x_i^{(L)} \leq x_i \leq x_i^{(U)},\ i = 1, 2, \ldots, d \tag{2}$$

where $x_i^{(L)}$ and $x_i^{(U)}$ are respectively the lower and upper bounds of the variable x_i. In the most general case, the design space is called the unit simplex. Constraint (1) has a drastic consequence on the variables x_i: they cannot be independently controlled. In this study, we formulate a novel variant of differential evolution that takes this specificity into account.

2 A differential evolutionary algorithm for optimizing mixtures

Differential Evolution (DE) is a powerful form of evolutionary computing, notably for real parameter optimization [7]. DE generates an initial population of candidate solutions and creates new individuals by combining existing ones. At each generation, mutation, recombination and selection operators are applied and this process repeats itself until a given termination criterion is satisfied (usually after a given number of generations). Let X be an individual of the population P_G obtained at generation G. Mutation expands the search space by computing a donor vector V from random individuals extracted from P_G. Recombination (also called crossover) incorporates information from X and V to create a new trial vector U. Selection decides if X must be replaced by U at generation $G + 1$. Constraints (1) and (2) lead to several modifications of the standard DE algorithm. Our variant called DES (DE over a Simplex) is mainly based on the fact that Constraint (1) defines a hyperplane $\mathcal{H}$. So any linear combination of solutions lying on $\mathcal{H}$ also verifies Constraint (1). In a second step, Constraint (2) is handled by a dedicated repair method.

2.1 Initialization of the population

A common practice for handling mixtures is to create individuals on the unit hypercube and to project them onto the simplex. More precisely, each generated solution X is projected onto the simplex along the ray that passes through X and the origin. This strategy (referred as RM repair method in section 3) is not recommended as it introduces a bias towards the center of the simplex. A better approach is to apply specific methods for generating uniformly distributed points on a unit simplex, such as the Maximally Sparse Selection method [3].

2.2 Mutation operator

Mutation is the most distinctive operator of DE in relation to other evolutionary algorithms. The mutant (or donor) vector is obtained by randomly selecting three different individuals in the population (X_a, X_b, X_c). A scaled difference of vectors is the basis of several variants of the mutation operator. We adopt the most common strategy, expressed as follows: $V = X_a + F(X_c - X_b)$ where F is an input parameter of the DE algorithm. In our case, the donor has an important property: its vector lies in the same hyperplane $\mathcal{H}$ than the current population (Constraint (1)), but not necessarily within the simplex. As a result, the only required adaptation consists in handling the bounding box defined by Constraint (2). This will be the aim of the repair operator described in section 2.4.

2.3 Crossover operator

This type of operator creates an offspring U from a solution X and a donor vector V. The binomial operator is one of the most used methods in DE [7]. It works as follows: a component of U is taken with probability CR from V, and with probability 1-CR from X, where CR is an input parameter called the crossover rate. The trial vector generated by this operator is likely to be located outside the simplex. Fortunately, there exists a more appropriate operator called the arithmetic crossover (AC [2]). This operator works as follows: $U = X + k(V - X)$ where k is a random value in $[0, 1]$. It is obvious that AC generates a trial vector verifying Constraint (1). However, contrary to the binomial method, AC generates a vector U which contains no component of X. Even if a candidate solution X contains some desirable proportions, they will be more or less altered in the trial vector, except for the case where $k = 0$. Therefore, we propose a more conservative approach inspired from the binomial crossover. As in the original method, the set of component indexes $\mathcal{I} = \{1, 2, \ldots, d\}$ is randomly split into two partitions I and I^c, and a part of the trial vector is assigned from X: $U_i = X_i$, $i \in I$. The remaining components U_j, $j \in I^c$, derived from V, must have a sum equal to $s = 1 - \sum_{i \in I} X_i$. Therefore, U lies on the hyperplane $\mathcal{H}_{X,I}$ defined by the following equation: $\sum_{j \in I^c} U_j = s$. Let V' be the projection of V on $\mathcal{H}_{X,I}$. The trial vector is completed as follows: $U_j = V'_j$, $j \in I^c$.

2.4 Repair operator

Some individuals generated by the crossover and mutation operators may lie outside the bounding box defined by Constraint (2). A common strategy is either to repair or to substitute an unfeasible solution [5]. Many of these methods do not apply in our case, because they treat each component separately. A well adapted bound handling technique is a parent centric operator called the Inverse Parabolic Spread Distribution (IPSD) [6]. This algorithm employs a probability distribution function to bring a solution S back into the search space while utilizing the information of a feasible individual X. By construction, the repaired solution verifies Constraint (2) and lies on the line joining X and S. As this line is included in $\mathcal{H}$, Constraint (1) is also verified.

3 Results

The multiobjective benchmark problem DTLZ2 [4] is based on M objectives of the following form: $f_i(\mathrm{x}) = (1 + g(\mathrm{x_p})).h_i(\mathrm{x_q})$ where $\mathrm{x_p}$ and $\mathrm{x_q}$ form a partition of x. DTLZS2 is an adaptation of DTLZ2 in which the design space is the unit simplex. In our variant, the component x_M can be derived from Constraint (1). Therefore, $\mathrm{x_p}$ now represents the $n - M$ last components of x and $\mathrm{x_q}$ the first $M-1$ components. The g function is expressed as: $g(\mathrm{x_p}) = \sum_{x_i \in \mathrm{x_p}} x_i^2$. The Pareto-optimal solutions of DTLZS2 lie in the same concave region than in the original version. As shown in Fig. 1, the only difference is that the octant is here incomplete because of Constraint (1).

The experiments were performed on three variants of our DES algorithm (RMA, ACA and ABA), all based on a standard Pareto-based DE/rand/1/bin and implemented using the pymoo framework [1]. RMA is a naive algorithm based on a usual solution space (hypercube) and simply repairs the errors using the RM method described in section 2.1. ACA and ABA both have the same methods as a standard DE except for the IPSD repair method and the crossover operator. ACA

used the arithmetic crossover operator and ABA used our dedicated binomial crossover operator. Our test implies a population of 200 individuals in dimension $d = 30$ and the number of generations was set to 100. A series of 21 runs with different random initial populations were conducted and the median of the Inverted Generational Distance (IGD) was computed. Fig. 2 shows the performance measures for various dimensions: $M = 3, 5, 8, 10, 15, 20$. These results indicate that our method outperforms the other ones on this test and that ACA appears to be less efficient than RMA.

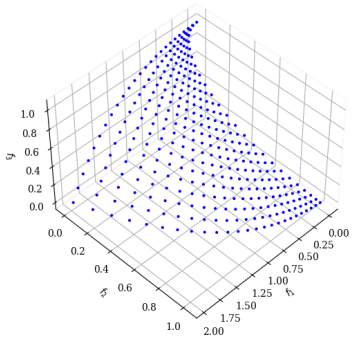


Fig. 1. Pareto Front of DTLZS2 with $M = 3$

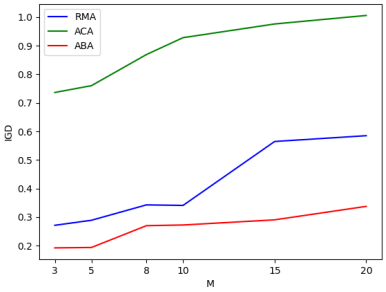


Fig. 2. Performance results

4 Conclusions and Future Work

In this paper, we have introduced DES, an algorithm dedicated to the optimization of mixtures. We have shown that a differential evolution algorithm is well adapted to the simplex search space. Mutation and recombination of existing solutions can be easily performed so that the new candidate solutions remain in the hyperplane including the search space. A general bound handling method can be reused in our particular case to keep solutions inside the bounding box. Our preliminary results show that our algorithm works successfully on a well known test function that has been adapted to the unit simplex. In the next future, the DES algorithm will be used to perform multi-objective optimization of superalloys.

Acknowledgement

This work was supported by the ANR CADHORS project, grant ANR-19-CE08-0017 of the French Agence Nationale de la Recherche (ANR).

References

1. J. Blank and K. Deb. pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020.
2. Swagatam Das and Ponnuthurai Suganthan. Differential evolution: A survey of the state-of-the-art. *IEEE Trans. Evolutionary Computation*, 15:4–31, 01 2011.
3. Kalyan Deb, Sunith Bandaru, and Haitham Seada. *Generating Uniformly Distributed Points on a Unit Simplex for Evolutionary Many-Objective Optimization: 10th International Conference, EMO 2019, East Lansing, MI, USA, March 10-13, 2019, Proceedings*, pages 179–190. 01 2019.
4. Kalyan Deb, L. Thiele, Marco Laumanns, and Eckart Zitzler. Scalable multi-objective optimization test problems. volume 1, pages 825–830, 06 2002.
5. Vinicius Kreischer, Thiago Tavares Magalhães, Helio Barbosa, and Eduardo Krempser. Evaluation of bound constraints handling methods in differential evolution using the cec2017 benchmark. 10 2017.
6. Nikhil Padhye, Kalyan Deb, and Pulkit Mittal. Boundary handling approaches in particle swarm optimization. *Advances in Intelligent Systems and Computing*, 201, 12 2013.
7. K.V. Price, R.N. Storn, and J.A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Natural Computing Series. Springer, 2005.

A Comparison of PSO-based Informative Path Planners for Detecting Pollution Peaks of the Ypacarai Lake with Autonomous Surface Vehicles

M. Jara Ten Kathen[1], D. Gutiérrez Reina[2], and I. Jurado Flores[1]

[1] Loyola Andalucía University, Spain
mcjaratenkathen@al.uloyola.es, ijurado@uloyola.es
[2] University of Seville, Spain
dgutierrezreina@us.es

1 Introduction

The appearance of cyanobacteria in the Ypacarai Lake, Paraguay, is affecting not only the ecosystem but also the public health and the local economy [1]. Several works based on Autonomous Surface Vehicles (ASVs) are being developed for the monitoring of the water quality of the lake [1][2]. The approaches consist of equipping an ASV with a Guidance, Navigation and Control (GNC) system and sensors that measure water quality parameters, such as pH, nitrate, conductivity, among others. The acquired data can be used to create an estimated model of the water quality of the lake.

This paper focuses on the performance comparison of PSO-based informative path planners. Seven algorithms based on PSO are compared. The algorithms are: the Classic PSO; an improved PSO based on Gaussian Process (GP), the Enhanced GP-based PSO [2]; four variants of the Enhanced GP-based PSO; and the PSO based on Epsilon Greedy. The four variants of the Enhanced GP-based PSO are algorithms that come from leaving active a term of the Enhanced GP-based PSO algorithm, they are: the Local Best algorithm, the Global Best algorithm, the Uncertainty algorithm and the Contamination algorithm. The GP is used as a surrogate model. The values of the coefficients of the Classic PSO are set according to [3]. To obtain the optimal values of the coefficients of the Enhanced GP-based PSO algorithm, Bayesian Optimization (BO) is applied. The main contribution of this work is the comparison of PSO-based informative path planners for monitoring high pollution areas using a fleet of autonomous surface vehicles.

2 Related Work

Solving non-linear real-world problems and working with different individuals/agents are some of the advantages of Swarm Intelligence (SI). Among the SI-based algorithms is the PSO. This algorithm contains few hyper-parameters to be set or tuned and the application of this algorithm is straightforward. Because of this, PSO is widely used as a basis for path planners [2] [4]. However, it has the disadvantage of getting stuck at a local optimum. To alleviate this limitation, sensors that measure water quality parameters are used. The problem of finding the optimal path maximizing the obtained information is known as the Informative Path Planner problem [5]. Approaches based on heuristic and approximation techniques such as the GP are used in order to maximize this information [6]. In [6], GP is used as a surrogate model for a path planner based on BO. In the paper, an analysis of kernel functions (constant, Matérn, Radial Basis Function (RBF), among others) and acquisition functions (Expected Improvement (EI), Probability of Improvement (PI), etc.) is presented. Likewise, the authors developed improvements to the classical acquisition functions in order to achieve a better performance of the monitoring system with an ASV. In [7], the authors extend their previous work by implementing a fusion of acquisition functions.

3 Methodology

For this work, the performance of seven PSO-based algorithms, Classic PSO, Enhanced GP-based PSO, PSO based on the Epsilon Greedy technique, and 4 variants of the Enhanced GP-based PSO, were compared. These 4 variants consist of leaving one term of algorithm [2] active.

– **Classic PSO:** The PSO is a meta-heuristic algorithm inspired by the social behavior of a flock of birds [8]. Each element p is called a particle and a set of particles is a swarm. The movement of the particles to find the global maximum is based on their own experience together with the experience of the swarm. The next velocity of the particle $\mathbf{v}_i^{t+1}$, Eq. 1a, is updated with the velocity at that instant $\mathbf{v}_i^t$, the best value of the particle $\mathbf{pbest}_p^t$ and the best value among all particles $\mathbf{gbest}^t$. c_1 and c_2 are the acceleration coefficients that determine the weight of exploitation ($\mathbf{pbest}_p^t$) and exploration ($\mathbf{gbest}^t$). w is the inertia term. r_1 and r_2 are random values in the range [0, 1]. The position is updated with Eq. 1b.

$$\mathbf{v}_p^{t+1} = w\mathbf{v}_p^t + c_1 r_1^t \left[\mathbf{pbest}_p^t - \mathbf{x}_p^t\right] + c_2 r_2^t \left[\mathbf{gbest}^t - \mathbf{x}_p^t\right] \tag{1a}$$

$$\mathbf{x}_p^{t+1} = \mathbf{x}_p^t + \mathbf{v}_p^{t+1} \tag{1b}$$

– **Enhanced GP-based PSO:** This algorithm combines the meta-heristic PSO algorithm and a surrogate model, GP, to alleviate the limitations of the classical PSO [2]. To Eq. 1a, two more terms are added, the maximum uncertainty $max\sigma^t$ and the maximum mean $max\mu^t$ of the surrogate model. This modification is made in order to guide the vehicles to the areas that were not explored (uncertainty) and to exploit the areas with high levels of contamination (mean). The informative path planner uses the samples taken by the sensors and creates an estimated ground truth model, with these data, calculates and selects the positions of the maximum uncertainty **max_un** and maximum contamination **max_con** values in order to update the next ASV speed using the Eq. 2. c_3 and c_4 are the acceleration coefficients that allow to determine the importance of the exploration of unexplored areas and the exploitation of areas with high levels of pollution. r_3 and r_4 are random values in the range [0, 1]. The position is updated with Eq. 1b.

$$\begin{aligned}\mathbf{v}_p^{t+1} = w\mathbf{v}_p^t + c_1 r_1^t[\mathbf{pbest}_p^t - \mathbf{x}_p^t] + c_2 r_2^t[\mathbf{gbest}^t - \mathbf{x}_p^t] + c_3 r_3^t[\mathbf{max\_un}^t - \mathbf{x}_p^t] + \\ c_4 r_4^t[\mathbf{max\_con}^t - \mathbf{x}_p^t]\end{aligned} \tag{2}$$

– **Variants of the Enhanced GP-based PSO:** The following variants of the Enhanced GP-based PSO have been considered for comparison purposes.
  - **Local Best:** This algorithm consists of leaving only the local best term, $\mathbf{pbest}_p^t$, active, apart from the velocity term. Because of this, the trajectory generated for an ASV is only influenced by the experience of the vehicle itself, it focuses on exploiting the zone with the best optimum found by the ASV.
  - **Global Best:** Unlike the Local Best algorithm, the global best focuses on surface exploration, since the term that remains active is the global best, $\mathbf{gbest}^t$. This allows the vehicles to learn from the experience of the fleet.
  - **Uncertainty:** While the Local Best and Global Best algorithms learn from the experience of the vehicles, the Uncertainty and Contamination algorithms use data from the surrogate model. The Uncertainty algorithm focuses on surface exploration using the maximum uncertainty data, **max_un**, from the GP.
  - **Contamination:** The Contamination algorithm takes the maximum contamination term, **max_con**, which would be the maximum mean value of the surrogate model, to guide vehicles to the area of highest contamination. It focuses on the exploitation of such a zone.

– **PSO based on Epsilon Greedy:** This algorithm consists of using the epsilon greedy technique so that the values of the coefficients of Eq. 2 are dynamic. In other words, while the path planner is running, the objective of the algorithm changes in intervals, depending on a condition, between focusing on exploring the surface or exploiting areas with high degree of contamination. The condition that must be met is that the ϵ value is greater or less than a random value val. At the beginning of the algorithm, the value of the ϵ function is 0.95, which means that there is a 95% probability of exploration. Once the ASVs travel a distance $d\epsilon_0$, the value of epsilon decreases $\Delta\epsilon$ at each step (1,000 meters), until the vehicles reach a distance $d\epsilon_f$. At that point, the epsilon value is set to 0.05, where there is 5% probability of exploration and 95% probability of exploitation.

4 Results

For the simulations, 10 different scenarios or ground truths are used, in all simulations the ASVs start from the same starting point. The end criterion of the algorithms is the distance traveled, the

simulation run until one of the vehicles travels 15,000 meters. The metric used for the comparison of the algorithms is the error between the value at the maximum contamination point and the value estimated by the surrogate model at that point. The results shown in Table 1 indicate that the best performances are obtained by the Enhanced GP-based PSO and the Contamination algorithms. Regarding the Enhanced GP-based PSO, this is due to the fact that the BO is used to search for the combination of values of the coefficients with the lowest error in the monitoring of the high contamination area. The tuning results indicate that the coefficients that determine the exploitation of the space (c_1, c_4) should be active with high values, the global best term should have a small weight (c_2) and the maximum uncertainty term should be inactive, $c_3 = 0$. These last two terms allow the exploration of the surface. This combination allows the vehicles to exploit the zones where they are moving and the zone with the highest level of pollution. Because the weight of the global best has a non-zero value, the vehicles also explore the areas through which they travel. However, in the Contamination algorithm, which had the lowest error among the algorithms compared, only the weight of the maximum contamination term has value, the other weights are equal to 0. Because of this, the ASVs go straight and exploit the area where there is the highest level of contamination.

Table 1. Comparison of the error of the seven algorithms

Algorithm	c_1	c_2	c_3	c_4	Error
Local Best	3	0	0	0	0.26254 ± 0.85916
Global Best	0	3	0	0	1.04980 ± 1.44375
Uncertainty	0	0	3	0	0.45391 ± 0.84466
Contamination	**0**	**0**	**0**	**3**	**0.00536 ± 0.01318**
Classic PSO	2	2	0	0	0.58460 ± 1.36174
Enhanced GP-based PSO	3.6845	1.5614	0	3.1262	0.00539 ± 0.01400
Epsilon Greedy	1; 4	4; 1	4; 1	1; 4	0.48916 ± 1.32377

5 Conclusion

In this article, comparisons between PSO-based path planners have been developed. The results showed that the Contamination algorithm performed the best in finding the most contaminated area of the Ypacarai Lake. By a small difference of $3x10^{-5}$, the second best performance was obtained by the Enhanced GP-based PSO, where the values of the acceleration coefficients were obtained by applying Bayesian optimization.

References

1. Arzamendia, M., Gregor, D., Reina, D. G., and Toral, S. L. An evolutionary approach to constrained path planning of an autonomous surface vehicle for maximizing the covered area of Ypacarai Lake. *Soft Computing*, 23(5): 1723-1734, 2019.
2. Kathen, M. J. T., Flores, I. J., Reina, D. G. An Informative Path Planner for a Swarm of ASVs Based on an Enhanced PSO with Gaussian Surrogate Model Components Intended for Water Monitoring Applications. *Electronics*, 10(13): 1605, 2021.
3. Xin, J., Li, S., Sheng, J., Zhang, Y., Cui, Y. Application of improved particle swarm optimization for navigation of unmanned surface vehicles. *Sensors*, 19(14): 3096, 2019.
4. Gul, F., Rahiman, W., Alhady, S. S., Ali, A., Mir, I., Jalil, A. Meta-heuristic approach for solving multi-objective path planning for autonomous guided robot using PSO–GWO optimization algorithm with evolutionary programming. *Journal of Ambient Intelligence and Humanized Computing*, 12(7): 7873-7890, 2021.
5. Popović, M., Vidal-Calleja, T., Hitz, G., Chung, J. J., Sa, I., Siegwart, R., Nieto, J. An informative path planning framework for UAV-based terrain monitoring. *Autonomous Robots*, 44(6): 889-911, 2020.
6. Samaniego, F. P., Reina, D. G., Marín, S. L. T., Arzamendia, M., Gregor, D. O. A bayesian optimization approach for water resources monitoring through an autonomous surface vehicle: The ypacarai lake case study. *IEEE Access*, 9: 9163-9179, 2021
7. Peralta, F., Reina, D. G., Toral, S., Arzamendia, M., Gregor, D. A bayesian optimization approach for multi-function estimation for environmental monitoring using an autonomous surface vehicle: Ypacarai lake case study. *Electronics*, 10(8): 963, 2021.
8. Eberhart, R., Kennedy, J. Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks*, 4: 1942-1948, 1995.

Comparing Parallel Surrogate-based and Surrogate-free Multi-Objective Optimization of COVID-19 vaccines allocation

Guillaume Briffoteaux[1,3], Romain Ragonnet[2], Pierre Tomenko[1], Mohand Mezmaz[1], Nouredine Melab[3] and Daniel Tuyttens[1]

[1] Mathematics and Operational Research Department, University of Mons, Belgium
{guillaume.briffoteaux,pierre.tomenko,mohand.mezmaz,daniel.tuyttens}@umons.ac.be
[2] School of Public Health and Preventive Medicine, Monash University, Australia
romain.ragonnet@monash.edu
[3] University of Lille, Inria, UMR 9189 - CRIStAL, France
nouredine.melab@univ-lille.fr

Abstract. The simulation-based and computationally expensive problem tackled in this paper addresses COVID-19 vaccines allocation in Malaysia. The multi-objective formulation considers simultaneously the total number of deaths, peak hospital occupancy and relaxation of mobility restrictions. Evolutionary algorithms have proven their capability to handle multi-to-many objectives but require a high number of computationally expensive simulations. The available techniques to raise the challenge rely on the joint use of surrogate-assisted optimization and parallel computing to deal with computational expensiveness. On the one hand, the simulation software is imitated by a cheap-to-evaluate surrogate model. On the other hand, multiple candidates are simultaneously assessed *via* multiple processing cores. In this study, we compare the performance of recently proposed surrogate-free and surrogate-based parallel multi-objective algorithms through the application to the COVID-19 vaccine distribution problem.

1 Introduction

In this paper, we address a multi-objective (MO) COVID-19 vaccines allocation problem. We aim to identify vaccines allocation strategies that minimize the total number of deaths and peak hospital occupancy, while maximizing the extent to which mobility restrictions can be relaxed. The onset of the COVID-19 outbreak has been rapidly followed by the development of dedicated simulation software to predict the trajectory of the disease [1, 2]. The availability of such tools enables one to inform authorities by formulating and solving optimization problems. In [3], a SEIR-model (Susceptible, Exposed, Infectious, Recovered) is deployed to simulate COVID-19 impacts. A single-objective (SO) problem is subsequently derived and handled by grid-search to regulate the alleviation of social restrictions. Multiple SO optimizations are carried out independently by

a simplex or a line search algorithm in [4–6] to efficiently allocate doses of vaccines to the age-categories of a population. The number of infections, deaths and hospital admissions are considered as the possible objective. The prioritization rules approved by the government of the studied cohort are integrated as constraints in the linear programming model presented in [7] to minimize mortality. In [8], multiple indicators are combined into a scalar-valued objective function. The MO formulation exhibited in [9] consists in maximizing the geographical diversity and social fairness of the distribution plan. Nevertheless, the MO problem is scalarized into a SO one that is then solved by a simplex algorithm. The approach used by Bubar and colleagues [10] is significantly different to ours, as the authors predefined a set of vaccination strategies and selected the most promising approach among them. In contrast, our continuous optimisation approach automatically designs strategies in a fully flexible way. The optimisation problem solved by McBryde and colleagues [11] is closer to that presented in our work since a similar level of flexibility was allowed to design optimal vaccines allocation plans. However, the authors used a simpler COVID-19 model resulting in significantly shorter simulation times, such that optimisation could be performed using more classical techniques. To the best of our knowledge, it has not been suggested yet to simultaneously minimize the number of deaths, peak hospital occupancy and the degree of mobility restriction through a MO-formulated problem. The fact that we consider the level of restrictions as one of the objectives to minimise represents a novelty compared to the previous works.

Despite the relative computational expensiveness of infectious disease transmission simulators, surrogate-based optimization has been rarely applied to the field. In [12], we harnessed surrogate models to determine the allocation of preventive treatments that minimize the number of deaths caused by tuberculosis in the Philippines. The identification of the regime for tuberculosis antibiotic treatments with lowest time and doses is formulated as a SO problem in [13] and solved by a method relying on a Radial Basis Functions surrogate model. The work presented in [14] deviates from this present study in that it aims to conceive a model prescribing the actions to perform according to a given situation. It is actually more related to artificial neural network hyper-parameters and architecture search. What is called "surrogate" in [14] is actually denominated "simulator" in simulation-based optimization. In this work, we combine machine learning and parallel computing to solve the MO vaccine distribution problem.

This study demonstrates the suitability of parallel surrogate-based multi-objective optimization algorithms on the real-world problem of COVID-19 vaccines allocation. The COVID-19-related problem is detailed in Section 2 and the MO algorithms are exposed in Section 3. Both surrogate-based and surrogate-free parallel MO approaches are applied to the real-world challenge in Section 4 and empirical comparisons are realized. Finally, conclusions are drawn in Section 5 and suggestions for future investigations are outlined.

2 COVID-19 vaccine distribution problem

The vast vaccination programs implemented over the last year or so all around the world achieved dramatic reductions of COVID-19 hospitalizations and deaths [15]. However, access to vaccination remains challenging, especially for low- to middle-income countries that are not able to offer vaccination to all their citizens [16]. The problem we are concerned with consists in optimizing the age-specific vaccines allocation plan to limit the impact of the disease in Malaysia under a capped number of doses. The population is divided into 8 age-categories of 10-years band from 0-9 years old to 70+ years old and the impact is expressed in terms of total number of deaths and peak hospital occupancy.

The simulation is realized in three phases by the AuTuMN software publicly available in `https://github.com/monash-emu/AuTuMN/`. The simulator is calibrated during the first phase with data accumulated from the beginning of the epidemic to the 1st of April 2021. The second phase starts at this latter date and lasts three months during which a daily limited number of doses is shared out among the population. Relaxation of mobility restrictions marks the kickoff of the third phase in the course of which a new distribution plan is applied involving the same number of daily available doses as in phase 2.

Decision variables $x_i \in [0,1]$ for $1 \leqslant i \leqslant 8$ and for $9 \leqslant i \leqslant 16$ represent the proportions of the available doses allocated to the 8 age-categories for phase 2 and phase 3 respectively. Variable $x_{17} \in [0,1]$ expresses the degree of relaxation of mobility restrictions where $x_{17} = 0$ leaves the restrictions unchanged and $x_{17} = 1$ means a return back to the pre-covid *era*. The following convex constraints convey the limitation of the number of doses during phases 2 and 3:

$$\sum_{i=1}^{8} x_i \leqslant 1 \text{ and } \sum_{i=9}^{16} x_i \leqslant 1 \tag{1}$$

The three-objective optimization problem consists in finding $\boldsymbol{x}^*$ such that

$$\boldsymbol{x}^* = \underset{\boldsymbol{x} \in [0,1]^{17} \text{ s.t. } (1)}{\arg\min} (g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), 1 - \boldsymbol{x}_{17}) \tag{2}$$

where $g_1(\boldsymbol{x})$ is the simulated total number of deaths and $g_2(\boldsymbol{x})$ the simulated maximum number of occupied hospital beds during the period.

3 Parallel Multi-Objective Evolutionary Algorithms

3.1 Variation Operators of Evolutionary Algorithms

Evolutionary Algorithms (EAs) are harnessed to deal with the COVID-19-related problem exhibited previously. In EAs, a population of solutions is evolved through cycles of parents selection, reproduction, children evaluation and replacement. EAs are chosen because they have proven their effectiveness on numerous multi-objective real-world problems [17] where the objective functions are black-box

as it is the case in our scenario. The constraint being convex and analytically verifiable, it is thus possible to design specific reproduction operators that directly generate feasible candidates. Assuming that every feasible solution can be reached, this technique has shown to be a reliable one [18].

The specific cross-over operator, called *distrib-X*, considers the two phases and the degree of relaxation independently. For two parents $\boldsymbol{x}$ and $\boldsymbol{y}$, the last decision variable for the two children $\boldsymbol{z}$ and $\boldsymbol{t}$ is set such that $z_{17} = x_{17}$ and $t_{17} = y_{17}$. Regarding the second phase, let I and J be a random partition of $\{1, \dots, 8\}$. For the age categories in I, $\boldsymbol{z}$ receives the proportion of vaccines from $\boldsymbol{x}$ ($z_i = x_i$ for $i \in I$). The remaining proportion of available doses at this step is $r = 1 - \sum_{i \in I} x_i$. For the age categories in J, the remaining proportion of doses is shared out according to the proportion allocated to the corresponding age categories in $\boldsymbol{y}$. In other terms, for $j \in J$, $z_j = \frac{r.y_j}{\sum_{j \in J} y_j}$. A similar treatment is applied to the variables associated to the third phase. The second child $\boldsymbol{t}$ is generated with an analogous procedure, where the roles of the parents are reversed.

The specific mutation operator, denoted *distrib-M*, disturbs a decision variable randomly chosen with uniform probability for $\{1, \dots, 8\}$, $\{9, \dots, 16\}$ and $\{17\}$. The last decision variable is mutated by polynomial mutation [17]. For the remaining ones, two age categories of the same phase are randomly selected and a random amount of doses are transferred from the first category to the second one. Both *distrib-X* and *distrib-M* are inspired by [12].

The *intermediate* and the *2-points* cross-over operators [17] are also considered for the sake of comparison. The *intermediate* strategy combines parents by random weighting average, while the *2-points* operator distributes portions of parents to the children. The portions are defined by two points with the first one separating phase 2 and phase 3 and the second one located between phase 3 and the relaxation decision variable x_{17}.

3.2 Parallel Multi-objective Evolutionary Algorithms

The major challenge in multi-objective optimization is to balance convergence and diversity in the objective space. Convergence is related to the closeness to the Pareto Front (PF) [17]. The PF is the set of the overall best solutions represented in the objective space and the Non-Dominated Fronts (NDFs) are approximations of the PF. Diversity is indicated by an extended coverage of the objective space by the NDFs. Hereafter, we present four algorithms to set this trade-off.

The first algorithm considered in the comparison is the Non-dominated Sorting Genetic Algorithm (NSGA-II) [19]. Firstly, to promote convergence, solutions pertaining to better NDFs are better ranked. Secondly, to favor diversity, solutions composing the same NDF are distinguished by setting the promise as high as the crowding distance is high. The proposed sorting is employed at the selection and the replacement steps of the EA.

The second algorithm reproduced for the experiments is the Reference Vector guided Evolutionary Algorithm (RVEA) proposed in [20] to handle many-

objective problems. A set of reference vectors is introduced in order to decompose the objective space and to enhance diversity. New candidates are attached to their closest reference vector, thus forming sub-populations among which only one candidate is kept at the replacement step. The new angle penalized distance chooses adaptively the candidate to be conserved by favoring convergence at the beginning of the search and diversity at latter stages. It is worth noting that the population size may change during the search in RVEA due to the possibility of empty sub-populations. In cases of degenerated or disconnected PF a high number of sub-populations become empty and the NDF obtained at the end of the search may not be dense enough. In the RVEA* variant, an additional reference vectors set is used to replace the reference vectors that would correspond to empty sub-populations.

In surrogate-based optimization, the additional trade-off between exploitation and exploration is to be specified. Minimizing the predicted objective vectors (POVs) produced by the surrogate boosts exploitation of known promising regions of the search space. Conversely, maximizing the predictive uncertainty enhances exploration of unknown regions.

The third algorithm is the surrogate-based Adaptive Bayesian Multi-Objective Evolutionary Algorithm (AB-MOEA) [21]. The first step of a cycle in AB-MOEA consists in generating new candidates by minimizing the POVs thanks to RVEA. During the second step, the new candidates are re-evaluated by an adaptive function that favors convergence at the beginning and reinforces exploitation as the execution progresses by minimizing the predictive uncertainty delivered by the surrogate. At the third step, q candidates are retained based on an adaptive sampling criterion similar to the reference vector guided replacement of RVEA to promote diversity.

The fourth algorithm is the Surrogate-Assisted Evolutionary Algorithm for Medium Scale Expensive problems (SAEA-ME) [22]. In SAEA-ME, NSGA-II is used to optimize a six-objective acquisition function where the three first objectives are the POVs and the last three objectives are the POVs minus the predictive variances. From the set of proposed candidates, the q ones showing the best hyper-volume improvement considering both the POVs alone and the POVs minus two variances are retained for parallel simulations. SAEA-ME performs well on problems with more than 10 decision variables. The dimensionality reduction feature proposed in [22] is not considered here as it consumes computational budget and can be applied to any method.

A Multi-Task Gaussian Process (MTGP) surrogate model [23] is implemented *via* the GPyTorch library [24] and incorporated into both AB-MOEA and SAEA-ME. Using a MTGP to model multiple objectives has been realized in [25] to control quality in sheet metal forming. In a traditional regression GP [26], a kernel function is specified to model the covariance between the inputs, thus allowing the model to learn the input-output mapping and to return predictions and predictive uncertainties. In the MTGP, inter-task dependencies are also

taken into account in the hope of improving over the case where the tasks are decoupled.

In the present investigation, the tasks are the three objectives and five kernel functions are considered for comparison. The widely used Radial Basis Functions kernel, denoted *rbf* and described in [26], provides very smooth predictors. According to [27], the Matern kernel with hyper-parameter $\nu = 1.5$ or 2.5, called *matern1.5* and *2.5* respectively, is to be preferred to model many physical phenomena. The higher predictive capacity Spectral Mixture kernel proposed in [28] is also raised with 2 and 4 components, denominated *sm2* and *sm4* respectively.

4 Experiments

The computational budget is set to two hours on 18 computing cores, thus allowing 18 simulations to be realized in parallel. The simulation duration varies from one solution to another from 13 to 142 seconds on one computing core. The four competing algorithms are implemented using our pySBO Python tool publicly available at: `https://github.com/GuillaumeBriffoteaux/pySBO`. Ten repetitions of the searches are carried out to ensure statistical robustness of the comparisons. The reference point for hyper-volume calculation is set to an upper bound for each objective $(32.10^6; 32.10^6; 1.5)$.

The surrogate-free approaches NSGA-II, RVEA and RVEA* are equipped with either the *distrib-X*, the *2-points* or the *intermediate* cross-over operator. For NSGA-II, the population size p_s is set to 108 or 162, thus avoiding the idling of the computing cores. For RVEA and RVEA*, we choose $p_s = 105$ or 171 to comply with the constraint imposed by the reference vectors initialization and to keep values close to those imposed for NSGA-II. Ten initial populations composed of 171 simulated solutions are generated to start the algorithms. Each initial population is made at 85% of solutions randomly sampled within the feasible search space and at 15% of candidates picked out on the boundary. When $p_s < 171$, only the best p_s candidates according to the non-dominated sorting defined in [19] are retained. For RVEA and RVEA*, a scaled version of the problem, where the first two objectives are divided by 1000, is also considered to demonstrate the effect of the objectives scales on the behavior of the methods.

The surrogate-based approaches AB-MOEA and SAEA-ME only integrate the *distrib-X* operator and use all the 171 initial samples as initial database. For RVEA in AB-MOEA, $p_s = 105$ and the number of generations is fixed to 20 as recommended in [21], while $p_s = 76$ for NSGA-II in SAEA-ME according to the guidance provided in [22] and the population evolved for 100 generations.

Table 1 shows the ranking of the algorithms according to the final hyper-volumes averaged over the ten repetitions. It can be observed in Table 1 that all the surrogate-based strategies outperform all those without surrogate. In particular, SAEA-ME with the *matern1.5* kernel is the best approach. The MTGP equipped with the *matern1.5* covariance function is preferred in both the SAEA-ME and AB-MOEA frameworks. Regarding the surrogate-free meth-

ods, NSGA-II with the *distrib-X* cross-over mechanism and $p_s = 108$ yields the best averaged hyper-volume. It is worth noticing that the *distrib-X* operator, specifically designed for the problem at hand, is to put forward as it surpasses both the *intermediate* and the *2-points* strategies in all contexts. Among the RVEAs, the best variant is RVEA* with $p_s = 105$ and the *distrib-X* cross-over thus indicating a possibly degenerated or disconnected PF. Indeed, the PF is certainly degenerated as indicates Figure 1 where are plotted the objective vectors from the ten final NDFs obtained by SAEA-ME with the *matern1.5* kernel. When analyzing the influence of objectives scales over the efficiency of RVEA and RVEA*, the conclusions drawn in [20] are confirmed as both algorithms are more appropriate when objectives have similar scales. Indeed, the three objectives lie in $[1655; 13,762]$, $[843; 10,962]$ and $[0; 1]$, respectively. The previous ranges are approximated *a posteriori* based on 250,664 simulations performed in RVEA and RVEA* on the original problem. The necessity to adequately scale the objectives brings a disadvantage to RVEAs as the scaling weights are tedious to define especially in the context of black-box expensive simulations. Another drawback is the constraints on the population size preventing to totally impede the idling of computing cores in all scenarios.

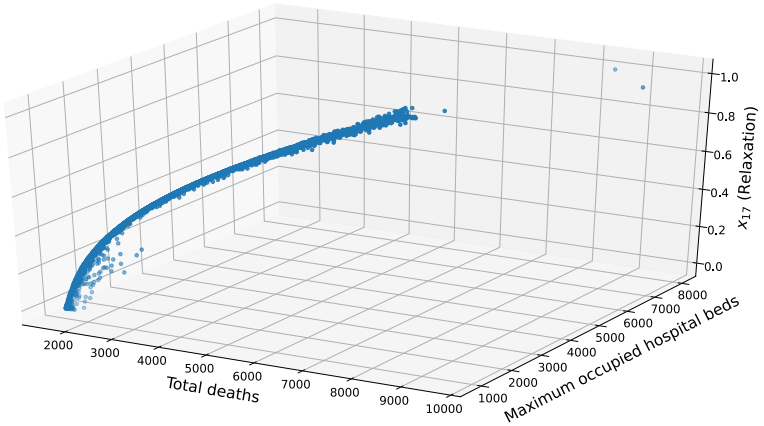


Fig. 1. Best NDFs from the 10 repetitions for SAEA-ME with matern1.5 kernel.

Figure 2 monitors the averaged hyper-volume as the search proceeds for the best strategy per category according to Table 1. The hyper-volume improves sharply at the very beginning of the search for the surrogate-based methods and reaches convergence rapidly (around 300 to 500 simulations). NSGA-II converges much slower but seems not to have converged at the end of the execution. By the right extremities of the curves, it could be expected that the hyper-volume returned by NSGA-II exceeds the one from AB-MOEA for larger numbers of simulations. However, reiterating the experiments for a time budget of four hours has

Algorithm	Cross-over operator	Population size	GP kernel	Objectives scaling	Averaged final Hyper-volume ($\times 10^{10} + 1.535 \times 10^{15}$)
SAEA-ME	**distribX**	**76**	**matern1.5**	-	**80.1800**
SAEA-ME	distribX	76	matern2.5	-	80.1610
SAEA-ME	distribX	76	rbf	-	79.9541
SAEA-ME	distribX	76	sm2	-	79.6701
AB-MOEA	distribX	105	matern1.5	-	79.6200
AB-MOEA	distribX	105	matern2.5	-	79.5879
SAEA-ME	distribX	76	sm4	-	79.5789
AB-MOEA	distribX	105	sm4	-	79.4861
AB-MOEA	distribX	105	sm2	-	79.4841
AB-MOEA	distribX	105	rbf	-	79.4304
NSGA-II	distribX	108	-	-	79.3337
NSGA-II	distribX	162	-	-	79.1876
RVEA*	distribX	105	-	yes	77.2805
RVEA*	distribX	171	-	yes	77.2514
RVEA	distribX	171	-	yes	77.1287
RVEA	distribX	105	-	yes	77.0117
NSGA-II	intermediate	108	-	-	76.9946
NSGA-II	intermediate	162	-	-	76.8320
NSGA-II	2-points	162	-	-	75.6959
NSGA-II	2-points	108	-	-	75.5889
RVEA	distribX	105	-	-	75.5184
RVEA*	intermediate	171	-	yes	75.3816
RVEA*	intermediate	105	-	yes	75.2841
RVEA	intermediate	171	-	yes	75.2006
RVEA	intermediate	105	-	yes	75.1562
RVEA*	distribX	105	-	-	75.1555
RVEA*	distribX	171	-	-	75.1372
RVEA	distribX	171	-	-	75.0563
RVEA*	2-points	171	-	yes	74.9803
RVEA	2-points	171	-	yes	74.9195
RVEA	2-points	105	-	yes	74.7692
RVEA*	2-points	105	-	yes	74.7535
RVEA*	intermediate	105	-	-	74.5607
RVEA	intermediate	105	-	-	74.5585
RVEA	intermediate	171	-	-	74.4959
RVEA*	intermediate	171	-	-	74.4266
RVEA	2-points	171	-	-	74.3694
RVEA	2-points	105	-	-	74.3518
RVEA*	2-points	171	-	-	74.3264
RVEA*	2-points	105	-	-	74.2507

Table 1. Ranking of the surrogate-based and surrogate-free approaches according to the averaged final hyper-volumes over the 10 repetitions.

not allowed to verify this expectation. Figure 2 specifies that the impact of objectives scaling on RVEAs appears from around 300 simulations. In the setting of a capped computational budget, it is important to strongly favor convergence and exploitation at the onset of the search. SAEA-ME and AB-MOEA realizes this by minimizing the POVs at the top beginning of the execution. The difference between the two approaches lies in the incorporation of the predictive uncertainty. In SAEA-ME, a degree of exploration is maintained by maximization of the predictive variance. Conversely, minimization of the predictive uncertainty is involved at latter stages in AB-MOEA. In spite of the convergence-oriented strategy adopted by RVEAs at the early stages of the search, the embedded mechanism set up to handle many objectives is quite heavy and reveals to be unsuitable when the computational budget is restricted. Indeed, in [20] the algorithms are run from 500 to 1,000 generations while 10 to 20 generations are allowed by our computational budget.

Reducing the solving time of moderately expensive optimization problems where the simulation lasts less than five minutes may enable to manage optimization under uncertainty. As the calibration of the simulation tool is uncertain, multiple configurations of its parameters can be considered, resulting in multiple optimization exercises to be executed and thus enabling to gain insight about the variability of the results.

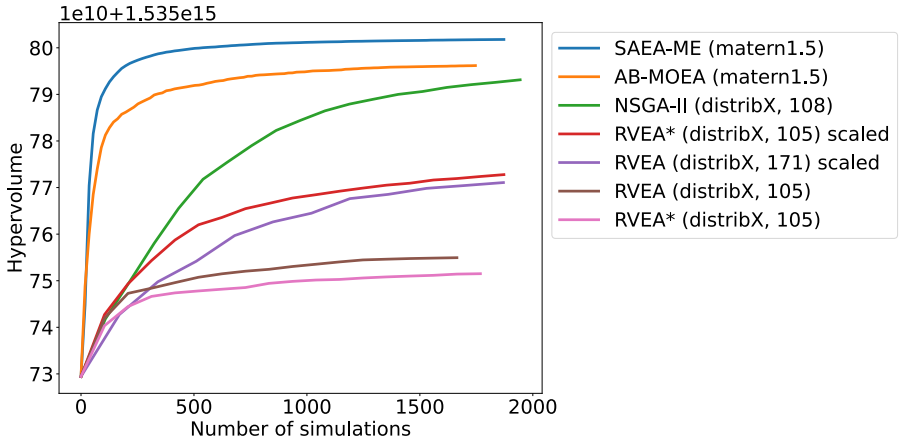


Fig. 2. Averaged hyper-volume according to the number of simulations.

The optimal allocation plan implies providing 70% of the doses to the 10-19 years old age-group and 30% to the 20-29 age-group during phase 2 according to Figure 3. In phase 3, 70% of the doses are assigned to 20-29 years old individuals and 15% to both the 40-49 and 10-19 age-categories. This plan prioritizes the vaccination of younger adults as they are the most transmitting cohort because of their high contact rate in the population [29]. Nevertheless, the present re-

sults have to be considered with caution. Since our experiments date back to the beginning of 2021, few feedback about vaccination efficiency was available. It is assumed here that the vaccine reduces transmission although it might not be the case for the Omicron variant of concern that started to break through the world at the end of 2021. Our results are similar to those presented in [30, 31] for influenza. From Figure 4 where the total number of deaths and the maximum number of occupied hospital beds are displayed with respect to the relaxation variable x_{17}, the alleviation of the physical distancing reveals to trigger an augmentation of the hospital occupancy and deaths.

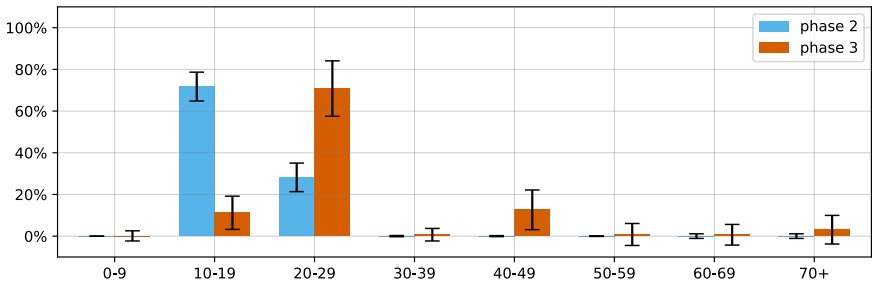


Fig. 3. Vaccines distribution according to age-categories. Averaged solutions from the best final NDFs returned by the 10 repetitions for SAEA-ME with matern1.5 kernel.

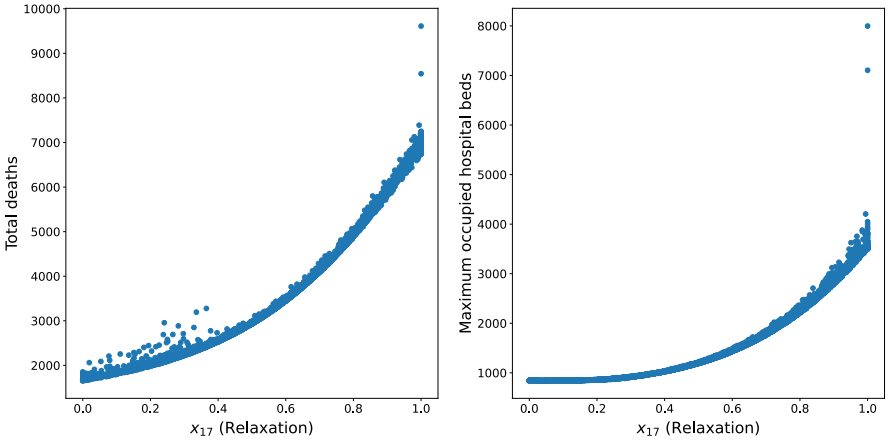


Fig. 4. Total number of deaths and maximum number of occupied hospital beds according to relaxation of the physical distancing x_{17}. Best NDFs from the 10 repetitions for SAEA-ME with matern1.5 kernel.

5 Conclusion

This paper demonstrates the suitability of parallel surrogate-based multi-objective optimization algorithms to handle the moderately computationally expensive COVID-19 vaccines allocation problem for Malaysia. In particular, SAEA-ME provides reliable results in a fast way. As future works, we suggest to benefit from the computational cost reduction of black-box simulation-based problem solving to take the uncertainty around the calibration of the simulator into account.

References

1. S. Chang *et al.* Modelling transmission and control of the covid-19 pandemic in australia. *Nature Communications*, 11,5710, 03 2020.
2. J. M. Trauer, M. J. Lydeamore, G. W. Dalton, D. Pilcher, M. T. Meehan, E. S. McBryde, A. C. Cheng, B. Sutton, and R. Ragonnet. Understanding how victoria, australia gained control of its second covid-19 wave. *Nature Communications*, 12(6266), 2021.
3. D. Duque, D. P. Morton, B. Singh, Z. Du, R. Pasco, and L. A. Meyers. Timing social distancing to avert unmanageable covid-19 hospital surges. *Proceedings of the National Academy of Sciences*, 117(33):19873–19878, 2020.
4. L. Matrajt, J. Eaton, T. Leung, D. Dimitrov, J. T. Schiffer, D. A. Swan, and H. Janes. Optimizing vaccine allocation for covid-19 vaccines: potential role of single-dose vaccination. *Nature Communications*, 12(3449), 2021.
5. L. Matrajt and I. Longini. Optimizing vaccine allocation at different points in time during an epidemic. *PloS one*, 5:e13767, 11 2010.
6. L. Matrajt, J. Eaton, T. Leung, and E. R. Brown. Vaccine optimization for covid-19: Who to vaccinate first? *Science Advances*, 7(6):eabf1374, 2021.
7. C. Buhat *et al.* Using constrained optimization for the allocation of covid-19 vaccines in the philippines. *Applied health economics and health policy*, 19(5):699–708, 2021.
8. S. Han *et al.* Time-varying optimization of covid-19 vaccine prioritization in the context of limited vaccination capacity. *Nature communications*, 12(1):4673, August 2021.
9. H. Anahideh, L. Kang, and N. Nezami. Fair and diverse allocation of scarce resources. *Socio-Economic Planning Sciences*, page 101193, 2021.
10. K. M. Bubar, K. Reinholt, S. M. Kissler, M. Lipsitch, S. Cobey, Y. H. Grad, and D. B. Larremore. Model-informed covid-19 vaccine prioritization strategies by age and serostatus. *Science*, 371(6532):916–921, 2021.
11. E. S. McBryde, M. T. Meehan, J. M. Caldwell, A. I. Adekunle, S. T. Ogunlade, M. A. Kuddus, R. Ragonnet, P. Jayasundara, J. M. Trauer, and R. C. Cope. Modelling direct and herd protection effects of vaccination against the sars-cov-2 delta variant in australia. *Medical Journal of Australia*, 215(9):427–432, 2021.
12. G. Briffoteaux, M. Gobert, R. Ragonnet, J. Gmys, M. Mezmaz, N. Melab, and D. Tuyttens. Parallel surrogate-assisted optimization: Batched bayesian neural network-assisted ga versus q-ego. *Swarm and Evolutionary Computation*, 57:100717, 2020.
13. J. M. Cicchese, E. Pienaar, D. E. Kirschner, and J. J. Linderman. Applying optimization algorithms to tuberculosis antibiotic treatment regimens. *Cellular and Molecular Bioengineering*, 10,6:523–535, December 2017.

14. R. Miikkulainen *et al.* From prediction to prescription: Evolutionary optimization of nonpharmaceutical interventions in the covid-19 pandemic. *IEEE Transactions on Evolutionary Computation*, 25(2):386–401, 2021.
15. T. N. Vilches *et al.* Covid-19 hospitalizations and deaths averted under an accelerated vaccination program in northeastern and southern regions of the usa. *The Lancet Regional Health - Americas*, 6:100147, 2022.
16. M. Sheel, S. McEwen, and S. E. Davies. Brand inequity in access to covid-19 vaccines. *The Lancet Regional Health - Western Pacific*, 18:100366, 2022.
17. E. G. Talbi. *Metaheuristics: From Design to Implementation.* Wiley Series on Parallel and Distributed Computing. Wiley, 2009.
18. Z. Michalewicz, D. Dasgupta, R. G. Le Riche, and M. Schoenauer. Evolutionary algorithms for constrained engineering problems. *Computers and Industrial Engineering*, 30(4):851–870, 1996.
19. 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.
20. R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff. A reference vector guided evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 20(5):773–791, 2016.
21. X. Wang, Y. Jin, S. Schmitt, and M. Olhofer. An adaptive bayesian approach to surrogate-assisted evolutionary multi-objective optimization. *Information Sciences*, 519:317–331, 2020.
22. X. Ruan, K. Li, B. Derbel, and A. Liefooghe. Surrogate assisted evolutionary algorithm for medium scale multi-objective optimisation problems. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, GECCO '20, page 560–568, New York, NY, USA, 2020. Association for Computing Machinery.
23. E. V. Bonilla, K. Chai, and C. Williams. Multi-task gaussian process prediction. In *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2008.
24. J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, 2018.
25. W. Xia, Huan Yang, Xiaoping Liao, and Jiangbang Zeng. A multi-objective optimization method based on gaussian process simultaneous modeling for quality control in sheet metal forming. *The International Journal of Advanced Manufacturing Technology*, 72:1333–1346, 2014.
26. C. E. Rasmussen. *Gaussian processes for machine learning.* MIT Press, 2006.
27. M.L. Stein. *Interpolation of Spatial Data: Some Theory for Kriging.* Springer Series in Statistics. Springer New York, 1999.
28. Andrew Gordon Wilson and Ryan Prescott Adams. Gaussian process kernels for pattern discovery and extrapolation, 2013.
29. K. Prem, A. R. Cook, and M. Jit. Projecting social contact matrices in 152 countries using contact surveys and demographic data. *PLOS Computational Biology*, 13(9):1–21, 09 2017.
30. D. Weycker, J. Edelsberg, M. E. Halloran, I. M. Longini, A. Nizam, V. Ciuryla, and G. Oster. Population-wide benefits of routine vaccination of children against influenza. *Vaccine*, 23(10):1284–1293, 2005.
31. J. Medlock and A. P. Galvani. Optimizing influenza vaccine distribution. *Science*, 325(5948):1705–1708, 2009.

Tuning ForestDisc hyperparameters: A sensitivity analysis

MAISSAE HADDOUCHI[1] and ABDELAZIZ BERRADO[2]

AMIPS team, Ecole Mohammadia d'Ingénieurs (EMI), Mohammed V University in Rabat, Morocco[1,2]
maissaehaddouchi@research.emi.ac.ma
berrado@emi.ac.ma

Abstract. This paper presents and analyzes ForestDisc, a discretization method based on tree ensemble and moment matching optimization. ForestDisc is a supervised and multivariate discretizer that transforms continuous attributes into categorical ones following two steps. At first, ForestDisc extracts for each continuous attribute the ensemble of split points learned while constructing a Random Forest model. It then constructs a reduced set of split points based on moment matching optimization. Previous works showed that ForestDisc enables an excellent performance compared to 22 popular discretizers. This work analyzes ForestDisc performance sensitivity to its tunning parameters and provides some guidelines for users when using the ForestDisc package.

Keywords: Discretization · Optimization · Classification

1 Introduction

Discretization is a key pre-processing step in Machine Learning (ML). It is used to reduce the complexity of the data space and to improve the performance and efficiency of ML tasks [20]. Furthermore, it is a required pre-processing step for several ML algorithms which only support categorical features. The usefulness of discretization in ML, especially prior to classification tasks, has led to the emergence of different discretization approaches. The literature has classified them based on different dimensions, as being supervised versus unsupervised and multivariate versus univariate [19][28][7][24].
Supervised discretization approaches consider the target attribute in the discretization process, which is expected to make them more "knowledgeable" in determining the best splits than their "blind" unsupervised counterparts [24][1]. They use different metrics to minimize the loss of information between the discretized attributes and the target attribute on the one hand, and the number of split points on the other hand [6][26]. Several comparative studies reported better performance for supervised discretization methods compared to unsupervised ones [24][1].
Multivariate discretizers consider all data attributes simultaneously during their

processing, while univariate discretizers consider attributes one at a time. Multivariate discretization is sensitive to the data's correlation structure, unlike unsupervised one [22].
ForestDisc, a supervised and multivariate discretization proceeds in two main steps: First, it extracts the split points generated while learning a Random Forest (RF) model and then builds a set of cut points for each continuous attribute to produce its partition into bins. This second step relies on moment matching optimization to identify a set of cut points that optimally matches the statistical properties of the ensemble of split points generated through the RF model. A previous work [20] has demonstrated that ForestDisc reached an excellent performance compared to 20 other discretizers. The comparative analysis took into consideration different metrics. These metrics included the number of resulting bins per variable and the execution time needed for discretizing it. They also reported the performance of classifiers when discretization is applied before the classification task. This study was performed using 50 benchmark datasets and six well-known classifiers.
ForestDisc is available as an R package: **ForestDisc** [12][13]. The first step in ForestDisc is processed using an RF model with 50 trees. The second step uses the first four moments in the moment matching approach and Nelder-Mead as an optimization algorithm.
In this work, we analyze the sensitivity of ForestDisc performance to the number of trees, the number of moments used, and the optimization algorithm used.
Accordingly, we present in the next section, ForestDisc related work. In section 3, we present ForestDisc sensitivity analysis to its tunning parameters. Finally, we provide in the last section the conclusion.

2 Related work

2.1 ForestDisc algorithm

ForestDisc processes discretization in two stages. In the first stage (Algorithm 1), ForestDisc generates the ensemble of split points learned by an RF model. In the second stage (Algorithm 2), ForestDisc uses this ensemble and returns the set of cut points based on moment matching optimization.
Let ***Data*** represent a dataset. Let ***AttCont*** be the set of its continuous attributes. Let $S = \{S_A\}_{A \in AttCont}$ be the ensemble of splits values that would be learned throught an RF model. Each set S_A corresponds to an attribute A. Let $C = \{C_A\}_{A \in AttCont}$ be the ensemble of cut points that would be learned, where C_A is the set of cut points discretizing the attribute A. Let K_{max} be the maximum value allowed for C_A cardinality.

2.2 ForestDisc tunning parameters

We analyze in this section ForestDisc performance variability depending on the number of trees used in the RF model, the non-linear optimization algorithm

Algorithm 1 ForestDisc Algorithm - Step I

Input: ***Data***
Output: ***S***

Initialization: $S = null$

Fit ***Random Forest*** to ***Data***
Return: $RF = \{T_1, ..., T_{n_{RF}}\}$ ▷ ensemmble of trees of cardinality n_{RF}
for each $A \in AttCont$ **do**
  Initialize: $S_A = null$
  for each *tree* *T* *in* *RF* **do**
    Extract S_A^T ▷ set of split values in tree T
    Update $S_A = S_A \cup S_A^T$
  end for
end for
Return: $S = \{S_A\}_{A \in AttCont}$

Algorithm 2 ForestDisc Algorithm - Step II

Input: ***Data***, ***AttCont***, ***S***, K_{max} ▷ K_{max} set by default to *10*
Output: ***C***, ***DataDisc***

Initialization: $C = null$

▷ We will solve, in the following, the moment matching problem (MMP) with n_m moments
for each $A \in AttCont$ **do**
  for $j \in \{0...n_m\}$ **do**
    Compute S_A moment of order j: $m_A^j = \frac{\sum_{i=1}^{n_A} {S_A}_i^j}{n_A}$ ▷ n_A is S_A cardinality
    Compute S_A weight of order j: $w_A^j = \frac{1}{max(max(S_A), -min(S_A), 1)^{2j}}$
  end for
  for $k \in \{2...K_{max}\}$ **do**
    Solve the moment matching problem:
    Objective function *MMP(A,k):* $min(P_k, X_k) \sum_{j=0}^{n_m} w_A^j \left(m_{X_k}^j - m_A^j\right)^2$
    where $m_{X_k}^j = \sum_{i=1}^{k} p_i {x_i}^j$, and $X_k = \{x_1, ..., x_k\}$ and $P_k = \{p_1, .., p_k\}$ are the decision variables.

    Constraints: $\sum_{i=1}^{k} p_i - 1 = 0$ and $min(S_A) \leq x_i \leq max(S_A)$ and $0 \leq p_i \leq 1$ for *i=1...k*

    Return: X_k^* and P_k^* the solution to *MMP(A,k)*, and Opt_k^* the optimum value
  end for
  Return: $X_A = \{X_k^*\}_{k \in \{2..K_{max}\}}$ and $Opt_A = \{Opt_k^*\}_{k \in \{2..K_{max}\}}$
  Select k_{opt} the value k corresponding to the minimum value in the set Opt_A
  Return: $C_A = X_{k_{opt}}^*$
end for
Return: $C = \{C_A\}_{A \in ContAttr}$

used for solving the MMP (Algorithm 2), and the number of moments considered in MMP. We consider in this work three tunning parameters for ForestDisc.

Number of trees: ForestDisc is based on the decision trees grown by an RF model to build an ensemble of split points partitioning the continuous attributes in a supervised and multivariate way (Algorithm 1). RF is known, in the literature, as an efficient ensemble learning, robust against overfitting, and user-friendly [21]. However, the number of trees is an important tuning parameter that generally influences the performance of an RF model. The optimal number of trees is problem dependant and users generally resort to comparative analysis to assess how the number of trees impacts the RF performance. We will analyze in the following sections how ForestDisc performance varies depending on the number of trees used.

Optimization algorithm: The moment matching problem introduced in Algorithm 2 is a non-linear optimization (NLO) problem. NLO [2] solves optimization problems where the objective function or some of the equality or inequality constraints are non-linear. We compared in previous work [14], the performance of 22 NLO Algorithm on solving the discretization based on MMP. The discretizer used in [14] is a simplified version of ForestDisc. It is a univariate and unsupervised discretization method, mapping each continuous attribute to a categorical attribute based on MMP. The performance of the 22 NLO Algorithms was compared based on multiple measures. The empirical results showed that the Nelder-Mead Simplex Algorithm [23] (Neldermead) achieved the best tradeoff between intrinsic and extrinsic measures [20]. The DIviding RECTangles (locally biased) algorithm [16] (DIRECTL) realized the second-best tradeoff, and the Sequential Least-Squares Quadratic Programming algorithm [17][18] (SLSQP) performed the best matching (optimum value). We compare in the following sections how ForestDisc performance changes regarding these 3 NLO algorithms.

Number of moments The moment matching mapping used in ForestDisc is based on the approach proposed in [46]. Authors in this work have conducted moment matching based on the first four moments to generate a limited number of discrete outcomes. Furthermore, the first four moments are widely known to characterize important statistics of random variables, namely, the mean, the variance, skewness, and kurtosis, which convey practical insights about distributions of random variables. ForestDisc proposed in [20] also uses the four first moments. This being said, it is worthwhile to investigate to what extent changing the number of moments in MMP (Algorithm 2) would impact the performance of the resulting discretization. The following sections investigate the sensitivity of ForestDisc to the number of first moments used in MMP.

3 ForestDisc sensitivity analysis

3.1 Experimental Set Up

In this section, we evaluate the sensitivity of ForestDisc to its tunning parameters: the number of trees, the number of moments, and the optimization algorithm used. We use the following metrics in this comparative analysis.
The first one is the optimum value, which is the solution to the moment matching problem described in Algorithm 2. The second one is the execution time required for discretizing an attribute. The third one is the number of resulting bins per discretized attribute.
The fourth and fifth metrics concern the predictive performance of classifiers pre-processed by discretization. The predictive performance is assessed via accuracy and F1 measures (details about the computation of each of these metrics can be found in [20]). Five well known classifiers are used in this comparative study: Classification And Regression Trees (CART) [4], Random Forest (RF) [3], Tree Boosting (Boosting) [9][5], Optimal weighted K-nearest neighbor classifier (KNNC) [25], and Naive Bayes Classifier (NaiveBayes) [10]. Their respective R functions/packages are: rpart/rpart, randomForest/randomForest, xgboost/xgboost, kknn/kknn, and naiveBayes/e1071. The default parameters were used for each of the aforementioned functions. RF and Boosting were performed using 200 trees.
We also use the Wilcoxon signed-rank test [27] to statistically compare the different results. We adopt the approach used in [11] [20] to perform pairwise comparisons of the metrics considered in this study. The results are summarized by counting the times each method outperforms, ties, and underperforms.
We analyze the different metrics by using benchmark datasets taken from the UCI Machine Learning [8] and Keel data sets repository [15]. Each data set is processed 10 times using Monte Carlo cross-validation procedure (refer to [20] for more details).

3.2 ForestDisc sensitivity to the number of trees and the non-linear optimization algorithm used

We report in Figure 1 the average results on accuracy and F1 score over the five classifiers, the 50 datasets used in [20], and the 10 iterations. These results show that the Neldermed algorithm outperforms the DIRECTL and SLSQP algorithms regardless of the number of trees used. Moreover, increasing the number of trees does not seem to induce an improvement in predictive performance regardless of the NLO algorithm used. The Wilcoxon test results reported in Table 1 confirm this conclusion.

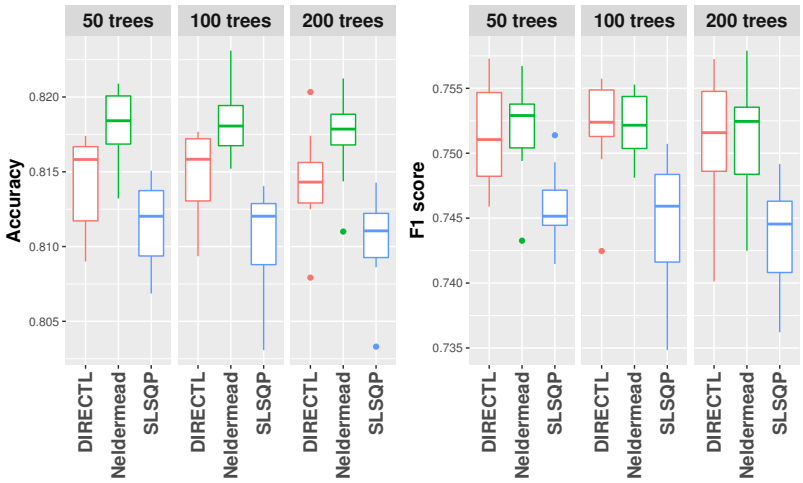


Fig. 1: Variation of the average accuracy and F1 score on the testing sets depending on the number of trees and the NLO algorithms used

Table 1: Wilcoxon signed-rank test scoring in global accuracy and F1 score depending on the number of trees and the NLO algorithms used (on the testing sets)

Accuracy				F1 measure			
Method	Wins	Ties	Losses	Method	Wins	Ties	Losses
Neldermead50	7	1	0	Neldermead50	7	1	0
Neldermead100	7	1	0	Neldermead100	7	1	0
Neldermead200	6	0	2	Neldermead200	5	1	2
DIRECTL50	3	2	3	DIRECTL100	3	3	2
DIRECTL100	3	2	3	DIRECTL50	3	2	3
DIRECTL200	3	2	3	DIRECTL200	3	2	3
SLSQP50	2	0	6	SLSQP50	1	1	6
SLSQP100	0	1	7	SLSQP100	0	2	6
SLSQP200	0	1	7	SLSQP200	0	1	7

Figure 2 displays the average results on the execution time per discretized variable and the number of intervals per variable. Based on these results, the Neldemead algorithm outperforms the other NLO algorithms in terms of execution time regardless of the number of trees used. In addition, the execution time increases as the number of trees increases regardless of the NLO used.
The number of intervals does not seem to vary according to the number of trees. Neldermead algorithm tends to produce the highest number of bins (between 6.5 and 7 intervals), followed by DIRECTL (between 4 and 5), and lastly by

SLSQP (slightly less than 4). The Wilcoxon results reported in Table 2 are in accordance with these conclusions.

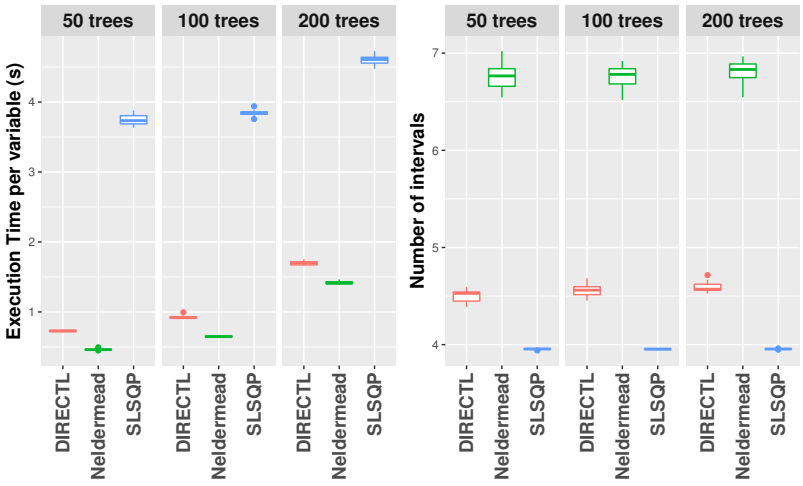


Fig. 2: Variation of the average execution time and number of intervals depending on the number of trees and the NLO algorithms used

Table 2: Wilcoxon signed-rank test scoring in Execution time and number of intervals depending on the number of trees and the NLO algorithms used (on the testing sets)

Execution time				Number of intervals			
Method	Wins	Ties	Losses	Method	Wins	Ties	Losses
Neldermead50	8	0	0	SLSQP50	6	2	0
Neldermead100	7	0	1	SLSQP100	6	2	0
DIRECT50	6	0	2	SLSQP200	6	2	0
DIRECT100	5	0	3	DIRECT50	5	0	3
Neldermead200	4	0	4	DIRECT100	4	0	4
DIRECT200	3	0	5	DIRECT200	3	0	5
SLSQP50	2	0	6	Neldermead50	1	1	6
SLSQP100	1	0	7	Neldermead100	1	1	6
SLSQP200	0	0	8	Neldermead200	0	0	8

3.3 ForestDisc sensitivity to the number of moments used

In this section, we analyze the sensitivity of ForestDisc to the number of first moments used in the MMP. This analysis is performed by varying the number of moments from 2 to 7, on a selection of 15 benchmark data sets used in a previous work [13].

Figure 3 reports the average results on the optimum values (MMP solutions), the execution time per discretized variable, and the number of intervals per variable. These results show that the optimum value is the greatest when 4 moments are used. Nevertheless, the use of 4 moments allows for a very good moment matching since the value of the optimum value remains very small (less than 1E-10 in general). The execution time tends to be slightly higher when 4 or 6 moments are used. Finally, the number of intervals tends to be slightly smaller when 4 or 6 moments are used.

Figures 4 and 5 display the variation of 5 classifiers' accuracy and F1 score depending on the number of moments used. The results are plotted on the training and testing sets. Table 3 reports the Wilcoxon signed test scores computed using accuracy and F1 scores on the testing sets. The results show that there is not a significant difference in predictive performance depending on the number of moments used, except for 7 moments (the worst results for all the classifiers). Using 4 moments seems to be a good alternative for all the classifiers (see Table 3). However, we think that we should use a more extensive analysis to have a robust conclusion about the impact of the number of moments used on the classifiers' predictive performance.

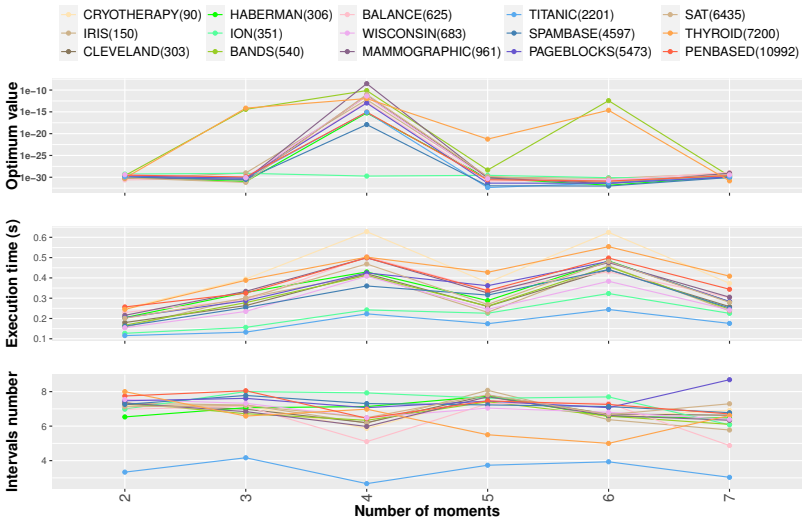


Fig. 3: Discretization performance variation depending on the number of moments (optimum value, execution time, and number of bins)

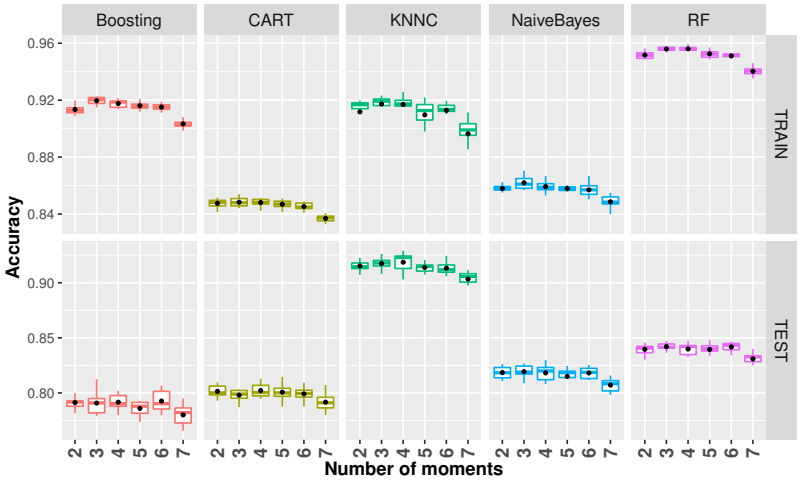


Fig. 4: Classifiers accuracy variation depending on the number of moments (on the training and testing sets). Bold dots show the mean values.

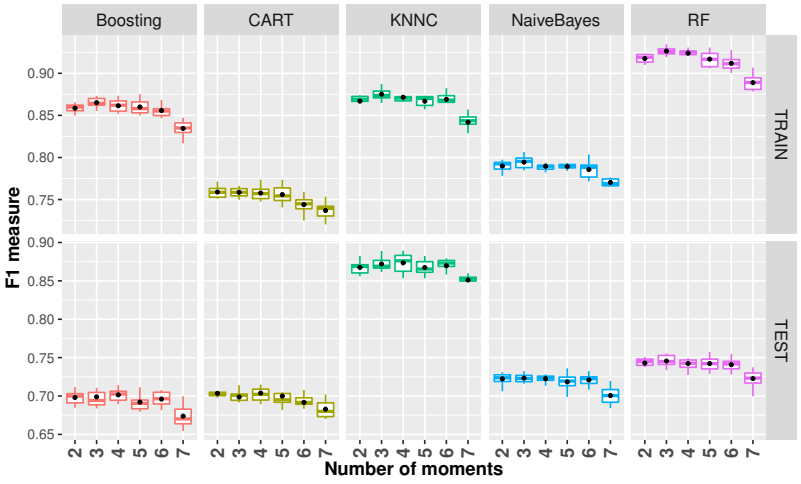


Fig. 5: Classifiers F1 score variation depending on the number of moments (on the training and testing sets). Bold dots show the mean values.

Table 3: Wilcoxon signed-rank test scoring in classifiers accuracy and F1 score depending on the number of moments (on the testing sets)

Accuracy					F1 measure				
Classifier	Moments Nbr	Wins	Ties	Losses	Classifier	Moments Nbr	Wins	Ties	Losses
Boosting	4	2	3	0	Boosting	4	2	3	0
	6	2	3	0		3	1	4	0
	3	1	4	0		6	1	4	0
	2	1	4	0		2	1	4	0
	5	1	2	2		5	1	3	1
	7	0	0	5		7	0	0	5
CART	4	2	3	0	CART	4	3	2	0
	6	1	4	0		2	2	3	0
	5	1	4	0		5	1	4	0
	2	1	4	0		3	1	3	1
	3	1	3	1		6	1	2	2
	7	0	0	5		7	0	0	5
KNNC	4	3	2	0	KNNC	4	1	4	0
	3	3	2	0		3	1	4	0
	2	1	4	0		2	1	4	0
	5	1	2	2		5	1	4	0
	6	1	2	2		6	1	4	0
	7	0	0	5		7	0	0	5
NaiveBayes	3	2	3	0	NaiveBayes	4	1	4	0
	4	1	4	0		3	1	4	0
	6	1	4	0		2	1	4	0
	2	1	4	0		5	1	4	0
	5	1	3	1		6	1	4	0
	7	0	0	5		7	0	0	5
RF	3	2	3	0	RF	3	2	3	0
	4	1	4	0		4	1	4	0
	6	1	4	0		6	1	4	0
	2	1	4	0		2	1	4	0
	5	1	3	1		5	1	3	1
	7	0	0	5		7	0	0	5

4 Conclusion

In this work, we have investigated the sensitivity of the ForestDisc discretizer to its tunning parameters. ForestDisc discretizes continuous attributes in two steps. First, it uses the ensemble of decision trees grown by an RF model to build an ensemble of split points partitioning the continuous attributes. It then relies on moment matching optimization to return a reduced set of cut points for discretizing each continuous attribute. A previous work [20] has demonstrated that ForestDisc reached an excellent performance compared to 20 other discretizers, based on extensive analysis on 50 benchmark datasets and six well-known classifiers.

We have analyzed, in this work, the sensitivity of ForestDisc performance to its tunning parameters by varying the number of trees from 50 to 200 and the number of moments from 2 to 7. We have also used two other alternatives for the optimization algorithm used. This preliminary analysis has shown that using 50 trees, four moments, and the Neldermead algorithm in the ForestDisc framework leads to the best performance. We think, however, that we should, in future work, expand the ranges of the number of trees and the number of mo-

ments to make a robust conclusion about the sensitivity of ForestDisc to these two parameters.

References

1. Agre, G.: On supervised and unsupervised discretization. Cybernetics and Information Technologies (2002)
2. Bazaraa, M.S., Sherali, H.D., Shetty, C.M.: Nonlinear Programming: Theory and Algorithms. Wiley-Interscience, Hoboken, N.J, 3rd ed edn. (2006), oCLC: ocm61478842
3. Breiman, L.: Random Forests. Machine Learning **45**(1), 5–32 (Oct 2001). https://doi.org/10.1023/A:1010933404324
4. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification And Regression Trees . Monterey, CA : Wadsworth & Brooks/Cole Advanced Books & Software, 1984. - 358 p., the wadsworth statistics/probability series edn. (1884)
5. Chen, T., Guestrin, C.: XGBoost: A Scalable Tree Boosting System. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16. pp. 785–794. ACM Press, San Francisco, California, USA (2016). https://doi.org/10.1145/2939672.2939785
6. Ching, J., Wong, A., Chan, K.: Class-dependent discretization for inductive learning from continuous and mixed-mode data. IEEE Transactions on Pattern Analysis and Machine Intelligence **17**(7), 641–651 (Jul 1995). https://doi.org/10.1109/34.391407
7. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and Unsupervised Discretization of Continuous Features. In: Machine Learning Proceedings 1995, pp. 194–202. Elsevier (1995). https://doi.org/10.1016/B978-1-55860-377-6.50032-3
8. Dua, D., Graff, C.: UCI machine learning repository (2017)
9. Friedman, J.H.: Greedy Function Approximation: A Gradient Boosting Machine. Annals of Statistics **29**, 1189–1232 (2000)
10. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. Machine Learning **29**(2), 131–163 (Nov 1997). https://doi.org/10.1023/A:1007465528199
11. Garcia, S., Luengo, J., Sáez, J.A., López, V., Herrera, F.: A Survey of Discretization Techniques: Taxonomy and Empirical Analysis in Supervised Learning. IEEE Transactions on Knowledge and Data Engineering **25**(4), 734–750 (Apr 2013). https://doi.org/10.1109/TKDE.2012.35
12. Haddouchi, M.: ForestDisc: Forest Discretization. R package version 0.1.0. https://CRAN.R-project.org/package=ForestDisc (2020)
13. Haddouchi, M., Berrado, A.: An implementation of a multivariate discretization for supervised learning using Forestdisc pp. 1–6 (Sep 2020). https://doi.org/10.1145/3419604.3419772
14. Haddouchi, M., Berrado, A.: Discretizing continuous attributes for machine learning using nonlinear programming. International Journal of Computer Science and Applications, 2021, 18(1), pp. 26–44 p. 20 (2021)
15. J. Alcalá-Fdez, Fernandez, A., Luengo, J., Derrac, J., García, S., Sánchez, L., Herrera, F.: KEEL Data-Mining Software Tool: Data Set Repository, Integration of algorithms and Experimental analysis Framework. Journal of Multiple-Valued Logic and Soft Computing 17:2-3 pp. 255–287 ((2011))
16. Jones, D.R., Perttunen, C.D., Stuckman, B.E.: Lipschitzian optimization without the Lipschitz constant. Journal of Optimization Theory and Applications **79**(1), 157–181 (Oct 1993). https://doi.org/10.1007/BF00941892

17. Kraft, D.: A Software Package for Sequential Quadratic Programming. Deutsche Forschungs- Und Versuchsanstalt Für Luft- Und Raumfahrt Köln: Forschungsbericht, Wiss. Berichtswesen d. DFVLR (1988)
18. Kraft, D., Munchen, I.: Algorithm 733: TOMP - Fortran modules for optimal control calculations. ACM Trans. Math. Soft pp. 262–281 (1994)
19. Liu, H., Hussain, F., Tan, C.L., Dash, M.: Discretization: An Enabling Technique. Data Mining and Knowledge Discovery **6**, 393–423 (2002)
20. Maissae, H., Abdelaziz, B.: A novel approach for discretizing continuous attributes based on tree ensemble and moment matching optimization. International Journal of Data Science and Analytics (Mar 2022). https://doi.org/10.1007/s41060-022-00316-1, https://doi.org/10.1007/s41060-022-00316-1
21. Maissae Haddouchi, Berrado, A.: A survey of methods and tools used for interpreting Random Forest pp. 1–6 (Oct 2019). https://doi.org/10.1109/ICSSD47982.2019.9002770
22. Mehta, S., Parthasarathy, S., Hui Yang: Toward unsupervised correlation preserving discretization. IEEE Transactions on Knowledge and Data Engineering **17**(9), 1174–1185 (Sep 2005). https://doi.org/10.1109/TKDE.2005.153
23. Nelder, J.A., Mead, R.: A Simplex Method for Function Minimization. Comput. J. **7**, 308–313 (1965). https://doi.org/10.1093/comjnl/7.4.308
24. Ramırez-Gallego, S., Garcıa, S., Martınez-Rego, D., Benıtez, J.M., Herrera, F.: Data Discretization: Taxonomy and Big Data Challenge p. 26
25. Samworth, R.J.: Optimal weighted nearest neighbour classifiers. The Annals of Statistics **40**(5), 2733–2763 (Oct 2012). https://doi.org/10.1214/12-AOS1049
26. Wang, C., Wang, M., She, Z., Cao, L.: CD: A Coupled Discretization Algorithm. In: Tan, P.N., Chawla, S., Ho, C.K., Bailey, J. (eds.) Advances in Knowledge Discovery and Data Mining. pp. 407–418. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30220-6_34
27. Wilcoxon, F.: Individual Comparisons by Ranking Methods. Biometrics Bulletin **1**(6), 80 (Dec 1945). https://doi.org/10.2307/3001968
28. Yang, Y., Webb, G.I., Wu, X.: Discretization methods. In: Maimon, O., Rokach, L. (eds.) Data Mining and Knowledge Discovery Handbook, pp. 101–116. Springer US, Boston, MA (2010). https://doi.org/10.1007/978-0-387-09823-4

Author Index