Proceedings OLA'2024
International Conference on Optimization and Learning

13-15 May 2024

Organizers

RIT Croatia
# Table of contents

JaxDecompiler: Redefining Gradient-Informed Software Design, Pochelu Piersick

Fractal Decomposition Algorithm with Bipartite Binarization, Becerra-Rozas Marcelo [et al.]

Optimization of Hydro Generation and Load Forecasting Based On LST-Net, Du Xingbang [et al.]

HOTS : A containers resource allocation hybrid method using machine learning and optimization., Leclercq Etienne [et al.]

Approaching Single-Episode Survival Reinforcement Learning with Safety-Threshold Q-Learning, Perotto Filipo [et al.]

Tabu Tenure Policy with Deep Reinforcecement Learning, Konovalenko Anna [et al.]

NeuroLGP-SM: A Surrogate-assisted Neuroevolution Approach using Linear Genetic Programming, Stapleton Fergal [et al.]

A Multi-Objective Metaheuristic Applied to the Airborne Wind Energy Farm Layout Problem, Carvalho Da Costa Rui [et al.]

How is the objective function of the Feature Selection problem formulated?, Cisternas-Caneo Felipe [et al.]

Evidence on the regularisation properties of Maximum-Entropy RL, Hos-
Benchmarking the operational efficiency of major container ports using machine learning and bootstrap DEA, Ablanedo-Rosas Jose Humberto


Learning Insertion Patterns to Enhance Operational Efficiency in Large-Scale Dial-a-Ride Systems, Liu Chijia [et al.]

Flexible Routing Strategies Based on Heuristic Approaches for Optimizing Last-Mile Delivery, Ejjanfi Mohamed-Ali [et al.]

Contribution to Forecasting Demand and Occupancy for Smart Lockers in the Context of Last-Mile Delivery, Ejjanfi Mohamed-Ali [et al.]

Parallel Strategies for Iterative Kriging-Based Surrogate Model Optimization: Bayesian Inference in Engineering Simulations, Sunay Yunus Emre [et al.]

Optimizing Object Detection Training Over Diverse Datasets, Dutta Pramit [et al.]

Nature-Inspired Techniques for Combinatorial Reverse Auctions in Electricity Consumption, Jahan Sifat [et al.]

A Bayesian Optimization Approach to Algorithm Parameter Tuning in Constrained Multiobjective Optimization, Cork Jordan [et al.]

Biologically-Inspired Algorithms for Adaptive Non-Player Character Behavior in Video Games, Alizadeh Sina [et al.]

Adapted Q-learning for the Blocking Job Shop Scheduling Problem, Karima Rihane [et al.]

An hybrid approach based on Graph Attention Network for the Team Orienteering Problem, Peña-Arenas Iván Guillermo [et al.]
A Parallel Genetic Algorithm for Qubit Mapping on Noisy Intermediate-Scale Quantum Machines, Rouzé Jérôme [et al.] 212

Machine Learning Applications in Road Safety: A Comprehensive Review, Modni Ilyas [et al.] 224

Patient Visits Forecasting in the Post-Pandemic Era at Emergency Departments, Haxaire Nicolas [et al.] 225

Solving a shareable-setup time prize collection VRP applied to an electrical maintenance sector, Tavares Roberto [et al.] 237

Robust Models for Learning Languages, Jastrzab Tomasz [et al.] 249

Optimisation-based classification tree: A game theoretic approach to group fairness, Liapis Georgios I. [et al.] 263

The Capacitated Vehicle Routing Problem with Pickup and Delivery using Suppliers for single and multiple commodities, Blanchard Damien [et al.] 275

Advancing Road Safety Metrics: Exploring Index construction, Ibtissam El Khalai [et al.] 287

Battery Management Strategies Optimization for Urban Plug-in Hybrid Buses, Ruiz Patricia [et al.] 299

Automatic Software Performance Optimization using Genetic Algorithms, Aragon-Jurado Jose Miguel [et al.] 302

A Multi-objective Clustering Algorithm Integrating Intra-clustering and Inter-clustering Measures, Flamia Azevedo Beatriz [et al.] 305

A Benchmark for Missing Data Imputation Techniques: development perspectives and comparative of performance, Cabrera-Sánchez Juan-Francisco [et al.] 317

Selecting the Most Relevant Objectives Using Data Mining: A Real-World Case Study, Gaspar-Cunha António [et al.] 330
On the Organization Network Analysis, Lalou Mohammed [et al.] 331

GPU Computing in Chapel: Application to Tree-Search Algorithms, Helbecque Guillaume [et al.] 341

S3EA: A Self-Supervised Stacked Ensemble Framework for Robust Anomaly Detection to Reduce False Alarms, M Naidu Sarala [et al.] 344

A Parallel Surrogate Approach for High-Dimensional Constrained Optimization with Discrete Variables, Regis Rommel 356

Feature Selection Based on Membrane Clustering, Sheuly Sharmin Sultana [et al.] 359

Collaborative Smart Agriculture: Leveraging Technology and Community Partnerships to Improve Crop Yields and Sustainability- A Case Study, Weraikat Dua [et al.] 371

Ants’ detection and tracking based on Convolution neural networks: a comparative study, Nakib Amir 373

Through Optimal Vision Transformer architectures based on metaheuristics, Nakib Amir 374

Neural Architecture Tuning: A BO-Powered NAS Tool, Ouertatani Houssem [et al.] 375

Adaptive greedy heuristic algorithm for rubber tyred gantry crane scheduling problem at container terminal, Ben Zina Saber 376

Author Index 387
JaxDecompiler: Redefining Gradient-Informed Software Design

Pierrick Pochelu∗1

1University of Luxembourg [Luxembourg] – Luxembourg

Abstract

Among numerical libraries capable of computing gradient descent optimization, JAX stands out by offering more features, accelerated by an intermediate representation known as Jaxpr language. However, editing the Jaxpr code is not directly possible. This article introduces JaxDecompiler, a tool that transforms any JAX function into an editable Python code, especially useful for editing the JAX function generated by the gradient function. JaxDecompiler simplifies the processes of reverse engineering, understanding, customizing, and interoperability of software developed by JAX. We highlight its capabilities, emphasize its practical applications especially in deep learning and more generally gradient-informed software, and demonstrate that the decompiled code speed performance is similar to the original.
Fractal Decomposition Algorithm with Bipartite Binarization

Marcelo Becerra-Rozas\textsuperscript{1}[0000–0003–0426–0144], Javier Peña\textsuperscript{1}[0000–0003–0440–5630], Thomas Firmin\textsuperscript{2}[0000–0002–5157–8393], Broderick Crawford\textsuperscript{1}[0000–0001–5500–0188], Ricardo Soto\textsuperscript{1}[0000–0002–5755–6929], El-Ghazali Talbi\textsuperscript{2}[0000–0003–4549–1010], and Giovanni Giachetti\textsuperscript{3}[0000–0003–2809–5120]

\textsuperscript{1} Pontificia Universidad Católica de Valparaíso, Valparaíso, Chile
\{marcelo.becerra.r, javier.pena.r\}@mail.pucv.cl
\{broderick.crawford,ricardo.soto\}@pucv.cl
\textsuperscript{2} University of Lille & INRIA, Lille, France
el-ghazali.talbi@univ-lille.fr
\textsuperscript{3} Universidad Andres Bello, Santiago, Chile
giovanni.giachetti@unab.cl

Abstract

In this article and within the framework of our first study, we propose an innovative strategy that involves the utilization of a metaheuristic initially designed to address continuous problems and adapt it successfully to solve binary domain problems. The specific metaheuristic in question is known as the Fractal Decomposition Algorithm (FDA). The adaptation process consists of two crucial steps. First, we transform the values belonging to the continuous domain into a range of values between 0 and 1. Subsequently, through meticulously defined rules in our study, we are able to map this range of values to binary unit values, i.e., 0 and 1. As a result of these adaptations, we have successfully presented an implementation of FDA, and the obtained results, along with the performance of this metaheuristic in solving the Knapsack Problem, demonstrate its effectiveness.

Keywords: Binary optimization · Knapsack Problem · Fractals · Decomposition

1 Introduction

The Knapsack Problem (KP) is a classical and fundamental challenge in the field of combinatorial optimization \cite{13}. It is a maximization problem that involves a finite set of elements \(i \in \{1, 2, ..., n\}\), where each element \(i\) has an associated value \(v_i\), a weight \(w_i\), and a knapsack of maximum capacity \(C\). The objective is to select a subset of elements in a way that maximizes the total value contained in the knapsack without exceeding its capacity.

Mathematically, the problem can be formulated as follows:

- Decision variables: \(x_i \in \{0, 1\}\), where \(x_i = 1\) if element \(i\) is selected, and \(x_i = 0\) otherwise.


- Objective function: Maximize the sum of the values of the selected elements:

\[ \max \sum_{i=1}^{n} v_i \cdot x_i \]

- Constraint: The sum of the weights of the selected elements must not exceed the knapsack capacity:

\[ \sum_{i=1}^{n} w_i \cdot x_i \leq C \]

- Additional constraints: Since \( x_i \) is binary, each element \( i \) can be selected (\( x_i = 1 \)) or not (\( x_i = 0 \)): \( x_i \in \{0, 1\} \) for all \( i \)

In this context, an interesting connection with the metaheuristic known as Fractal Decomposition Algorithm (FDA) is presented. The FDA is a continuous optimization technique with a single solution trajectory, commonly used in problems involving continuous and multimodal domains. However, in this study, its application in the KP will be explored.

Traditionally, the KP has been approached using discrete optimization algorithms. Nonetheless, this work introduces a new perspective by employing the binarized FDA, seeking to adapt this powerful continuous technique to a problem in the discrete domain, such as the binary knapsack problem. This innovative approach aims to harness the advantages of FDA in solving complex problems while addressing the challenge posed by a discrete domain with limitations in searching for optimal solutions.

The main objective of this research is to explore how the FDA metaheuristic can be binarized and successfully applied to the KP. We will explain the binarization process in two steps and how different techniques are used to achieve this goal. Additionally, we will describe the utilization of the "Zellij" tool to facilitate the binarization process and improve the efficiency of the resulting algorithm. It is expected that this investigation will provide new perspectives and efficient solutions to the KP challenge, contributing valuable insights to the field of combinatorial optimization.

The paper organization is as follows: Section 2 introduces the pertinent background in terms of solving the Knapsack Problem, encompassing approaches, metaheuristic techniques, binarization, and an overview of the Fractal Decompose Algorithm (FDA). Subsequently, Section 3 meticulously examines the application of the binarized FDA to the Knapsack Problem. Lastly, Sections 4 and 5 showcase the attained outcomes in problem resolution and outline the corresponding deductions.
2 Related Work

2.1 Metaheuristics: Exploring Alternative Techniques in Optimization

Metaheuristics (MH) are widely employed techniques for solving optimization problems, overcoming the limitations of local searches [3, 16, 4]. These strategies are based on two key components: exploration and exploitation [10]. Exploration involves generating a variety of solutions to explore the search space globally, while exploitation focuses on finding optimal solutions in a local region. Properly balancing these components is essential to achieve global optimization [17]. The selection of the best solutions ensures convergence towards the optimum, while randomization and diversification allow for exploring the entire search space, increasing the diversity of solutions.

MH offer the advantage of generating near-optimal solutions in reduced computational time compared to exact methods and have the flexibility to adapt to various problems compared to heuristic methods. MH are often designed for continuous domains, which reflects their development and classification. There are various approaches to classify MH, as proposed by Heidari et al. [8] and Cuevas et al. [6], categorizing them as trajectory-based and population-based. Another relevant taxonomy, presented in [11], explores the key components of these algorithms, including solution evaluation, parameters, encoding, agent or population initialization, population management, operators, and local search. Some classical examples of such MH include Particle Swarm Optimization [12], Cuckoo Search [18], and Genetic Algorithm [9]. In the literature, population-based MH are more widely used than single-solution MH.

2.2 Binarization: Adapting Algorithms for the Binary Domain

Continuous MH algorithms have been extensively developed and employed in optimization problems within continuous domains. However, when confronted with binary optimization problems, specific adaptations are necessary to handle binary values of 0 and 1. Over time, various authors have proposed several approaches to adapt MH algorithms and achieve effective application in the binary context [3].

2.3 Fractal Decomposition Algorithm

Fractal-based Decomposition Algorithm (FDA) it is a metaheuristic grounded in the concept of fractals and their application to search space partitioning [15]. The fundamental principle of the FDA revolves around the utilization of fractal geometry to subdivide the search space into intricate, self-replicating structures, each representing a unique region for exploration. This distinctive approach is orchestrated through a series of intricately designed steps, as expounded below.
Decomposition: FDA starts by dividing the search region into $2^n$ hyperspheres positioned at a distance from the center as indicated by the equation 1. Let $\vec{C}_j^{(i)}$ be the new center of the hypersphere, $\vec{C}_j^{(0)}$ is the center of the region to be decomposed, $r'$, corresponding to the radius of the new spheres, is defined as $r/\alpha$, where $r$ is the radius of the region to be decomposed.

$$\vec{C}_j^{(i)} = \vec{C}_j^{(0)} + (-1)^i \times ((r - r') \times \vec{e}_j)$$  \hspace{1cm} (1)

Once the hyperspheres that decompose the region are defined, each hypersphere is evaluated based on the equation 2.

$$q = \max \{g_1, g_2, g_c\}$$  \hspace{1cm} (2)

with:

$$g_1 = \frac{\|f(s_1) - f(BSF)\|}{\|s_1 - BSF\|}, g_2 = \frac{\|f(s_2) - f(BSF)\|}{\|s_2 - BSF\|}, g_c = \frac{\|f(s_c) - f(BSF)\|}{\|s_c - BSF\|}$$  \hspace{1cm} (3)

where:

$$s_1 = C_d + \alpha \frac{r}{\sqrt{D}} \times \vec{e}_d, \text{ for } d = 1, 2, 3, ..., D$$ \hspace{1cm} (4)

$$s_2 = C_d - \alpha \frac{r}{\sqrt{D}} \times \vec{e}_d, \text{ for } d = 1, 2, 3, ..., D$$ \hspace{1cm} (5)

With BSF being the best point found up to the moment. Once all the fractals of the level have been identified and evaluated, the process proceeds to expand the next level, until all the fractals have been explored and the fractals of lower levels reach the assigned limit.

Intensive level search: Once the process of search space decomposition cannot be further continued, the exploitation of the best regions is undertaken. This can be executed through various means, including local search algorithms or even metaheuristics. In this study, a specialized local search algorithm designed for continuous problems was employed. In Intensive Level Search (ILS), two candidate solutions are evaluated per dimension. These solutions are positioned on opposite sides of the best current point found by the fractal, at a distance of $w$ from the initial point, as shown in equations 6 and 7.

$$\vec{x}^{s_1} = \vec{x}^* + w \times \vec{e}_i$$  \hspace{1cm} (6)

$$\vec{x}^{s_2} = \vec{x}^* - w \times \vec{e}_i$$  \hspace{1cm} (7)

Subsequently, the values of $s, s_1,$ and $s_2$ are compared to determine the best one based on the problem to solve and proceed with the search in the next dimension. If, after traversing all dimensions, no improved value is found, the value of $w$ is reduced. If the value of $w$ becomes less than a pre-defined threshold, the process concludes, and the most recent outcome of this procedure remains as the best point achieved by the algorithm.
3 bFDA: Binary Fractal Decomposition Algorithm

As the FDA algorithm is primarily designed to address continuous optimization problems, it possesses limitations when confronted with combinatorial problem-solving tasks. In light of this challenge, we introduce a two-step binarization approach (Fig. 2) aimed at enabling the FDA to effectively tackle the problem at hand. In the scope of this research, we adopt the framework proposed by Firmin[7], wherein the FDA is fine-tuned for continuous problem optimization. This adaptation allows the FDA to function seamlessly in continuous problem scenarios. However, when the algorithm necessitates the determination of an optimum within the context of combinatorial challenges, we facilitate the transformation of the solution obtained into its combinatorial equivalent. To achieve this transformation, we initially map the continuous domain values to a normalized range between 0 and 1 using the S-shaped fourth equation (Fig. 1). Subsequently, we employ standard binary operators to derive binary values (See Table 1). With this innovative approach, we ensure that the FDA can maintain its original purpose while concurrently generating solutions tailored for combinatorial problems.

<table>
<thead>
<tr>
<th>Type</th>
<th>Transfer Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>S4 [5,14]</td>
<td>( T(d_j^w) = \frac{1}{1+e^{-d_j^w}} )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Binarization Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>( X_{new}^j = \begin{cases} 1 &amp; \text{if } \text{rand} \leq T(d_j^w) \ 0 &amp; \text{else.} \end{cases} )</td>
</tr>
</tbody>
</table>

Fig. 1: S-Shaped Equation 4
4 Experimental Results

In order to validate the performance of our proposal, we conducted a comparison of bFDA with classical results from the binary Knapsack Problem (KP) literature [1,2]. These instances can be downloaded from: http://artemisa.unicauca.edu.co/~johnyortega/instances_01_KP/. Specifically, we solved 24 instances from this webpage. Finally, the instances can be categorized into 2 groups, and the configurations related to them are detailed in the table 2.

- Low-dimensional.
- Large-dimensional.
  - Uncorrelated instances.
  - Weakly correlated instances.
  - Strongly correlated instances.

Fig. 2: Two-step binarization

The algorithms were developed using Zellij, an open-source framework specialized in the design and reuse of fractal-based decomposition algorithms, and they were executed on the machines provided by Grid5000 using MPI.

Table 3 provides detailed computational results for 24 different instances using our bFDA. To explain the different columns and rows of the table: the first column, titled "Inst.," represents the instance of the problem being solved. The second column, titled "Opt.," displays the known optimal value for each instance. The subsequent columns are organized by group and represent the algorithm used to solve the KP. Within this group, the first column shows the results for
Fractal Decomposition Algorithm with Bipartite Binarization

Table 2: 0-1KP Datasets Descriptions

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Capacity</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>f1_l-d_kp</td>
<td>269</td>
<td>10</td>
</tr>
<tr>
<td>f2_l-d_kp</td>
<td>878</td>
<td>20</td>
</tr>
<tr>
<td>f3_l-d_kp</td>
<td>20</td>
<td>4</td>
</tr>
<tr>
<td>f4_l-d_kp</td>
<td>11</td>
<td>4</td>
</tr>
<tr>
<td>f5_l-d_kp</td>
<td>375</td>
<td>15</td>
</tr>
<tr>
<td>f6_l-d_kp</td>
<td>60</td>
<td>7</td>
</tr>
<tr>
<td>f7_l-d_kp</td>
<td>10,000</td>
<td>23</td>
</tr>
<tr>
<td>f8_l-d_kp</td>
<td>80</td>
<td>5</td>
</tr>
<tr>
<td>f10_l-d_kp</td>
<td>879</td>
<td>20</td>
</tr>
<tr>
<td>Uncorrelated data instances</td>
<td></td>
<td></td>
</tr>
<tr>
<td>knapPI_1</td>
<td>100</td>
<td>1,000</td>
</tr>
<tr>
<td>knapPI_1</td>
<td>200</td>
<td>1,000</td>
</tr>
<tr>
<td>knapPI_1</td>
<td>500</td>
<td>1,000</td>
</tr>
<tr>
<td>knapPI_1</td>
<td>1000</td>
<td>1,000</td>
</tr>
<tr>
<td>Weakly correlated instances</td>
<td></td>
<td></td>
</tr>
<tr>
<td>knapPI_2</td>
<td>100</td>
<td>1,000</td>
</tr>
<tr>
<td>knapPI_2</td>
<td>200</td>
<td>1,000</td>
</tr>
<tr>
<td>knapPI_2</td>
<td>500</td>
<td>1,000</td>
</tr>
<tr>
<td>knapPI_2</td>
<td>1000</td>
<td>1,000</td>
</tr>
<tr>
<td>Strongly correlated instances</td>
<td></td>
<td></td>
</tr>
<tr>
<td>knapPI_3</td>
<td>100</td>
<td>1,000</td>
</tr>
<tr>
<td>knapPI_3</td>
<td>200</td>
<td>1,000</td>
</tr>
<tr>
<td>knapPI_3</td>
<td>500</td>
<td>1,000</td>
</tr>
<tr>
<td>knapPI_3</td>
<td>1000</td>
<td>1,000</td>
</tr>
</tbody>
</table>

Table 3: Comparison Results

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Low dimensional data instances</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f1_l-d_kp_10_269</td>
<td>295</td>
<td>295</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>f2_l-d_kp_20_878</td>
<td>1024</td>
<td>1009</td>
<td>971</td>
<td>1.46</td>
</tr>
<tr>
<td>f3_l-d_kp_4_20</td>
<td>35</td>
<td>35</td>
<td>35</td>
<td>0.00</td>
</tr>
<tr>
<td>f4_l-d_kp_4_11</td>
<td>23</td>
<td>23</td>
<td>23</td>
<td>0.00</td>
</tr>
<tr>
<td>f5_l-d_kp_15_375</td>
<td>481.0694</td>
<td>481.0694</td>
<td>468.8897</td>
<td>0.00</td>
</tr>
<tr>
<td>f6_l-d_kp_10_60</td>
<td>52</td>
<td>52</td>
<td>52</td>
<td>0.00</td>
</tr>
<tr>
<td>f7_l-d_kp_7_50</td>
<td>107</td>
<td>107</td>
<td>107</td>
<td>0.00</td>
</tr>
<tr>
<td>f8_l-d_kp_23_10000</td>
<td>9767</td>
<td>9767</td>
<td>9762</td>
<td>0.00</td>
</tr>
<tr>
<td>f9_l-d_kp_5_80</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>0.00</td>
</tr>
<tr>
<td>f10_l-d_kp_20_879</td>
<td>1025</td>
<td>997</td>
<td>961</td>
<td>2.73</td>
</tr>
<tr>
<td>Uncorrelated instances</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>knapPI_1_100_1000</td>
<td>9147</td>
<td>9147</td>
<td>8957</td>
<td>0.00</td>
</tr>
<tr>
<td>knapPI_1_200_1000</td>
<td>11238</td>
<td>11048</td>
<td>10105</td>
<td>1.69</td>
</tr>
<tr>
<td>knapPI_1_500_1000</td>
<td>28857</td>
<td>28262</td>
<td>26941</td>
<td>2.06</td>
</tr>
<tr>
<td>knapPI_1_1000_1000</td>
<td>54503</td>
<td>52868</td>
<td>49478</td>
<td>3.00</td>
</tr>
<tr>
<td>Weakly correlated instances</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>knapPI_2_100_1000</td>
<td>1514</td>
<td>1496</td>
<td>1459</td>
<td>1.19</td>
</tr>
<tr>
<td>knapPI_2_200_1000</td>
<td>1634</td>
<td>1530</td>
<td>1465</td>
<td>6.36</td>
</tr>
<tr>
<td>knapPI_2_500_1000</td>
<td>4566</td>
<td>4093</td>
<td>3964</td>
<td>10.36</td>
</tr>
<tr>
<td>knapPI_2_1000_1000</td>
<td>9052</td>
<td>7652</td>
<td>7543</td>
<td>15.47</td>
</tr>
<tr>
<td>Strongly correlated instances</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>knapPI_3_100_1000</td>
<td>2397</td>
<td>2163</td>
<td>2098</td>
<td>9.76</td>
</tr>
<tr>
<td>knapPI_3_200_1000</td>
<td>2697</td>
<td>2381</td>
<td>2267</td>
<td>11.72</td>
</tr>
<tr>
<td>knapPI_3_500_1000</td>
<td>7117</td>
<td>6073</td>
<td>5837</td>
<td>14.67</td>
</tr>
<tr>
<td>knapPI_3_1000_1000</td>
<td>14390</td>
<td>11489</td>
<td>11064</td>
<td>20.16</td>
</tr>
</tbody>
</table>

that algorithm in terms of the best solution found ("Best"), the average of all solutions found ("Avg"), and finally, the relative percentage deviation (RPD) calculated using Equation 8. Some cells are highlighted in bold, indicating that the corresponding algorithm found the best solution for that instance.

\[
\text{RPD} = 100 \cdot \frac{(\text{Best} - \text{Opt})}{\text{Opt}}. \tag{8}
\]

The figure 3 presents how fasts the algorithm converges to the results presented.
5 Conclusion

This article introduces an innovative strategy for solving binary domain problems using the Bipartite Binary Fractal Decomposition Algorithm (bFDA). Adapting the Fractal Decomposition Algorithm (FDA) to address binary domain issues is challenging due to the fundamental differences between these types of problems. Nevertheless, an effective solution is proposed, involving the transformation of continuous domain values into a range between 0 and 1, followed by mapping these values onto binary units.

The proposed strategy is grounded in the concept that continuous values can be discretized into binary units through precise, meticulously defined rules. These rules are based on the principle that continuous values can be categorized into two groups: those exceeding a specific threshold and those equal to or less than this threshold. Values above the threshold are assigned a value of 1, while those equal to or below the threshold are assigned a value of 0. This discretization approach is known as the two-step technique, or as we refer to it, bipartite binarization.

To evaluate the effectiveness of the proposed strategy, the FDA was applied to a series of Knapsack Problem instances. Experimental results indicate that as dimensionality increases, the FDA’s ability to reach the known optimum becomes more challenging. This may be attributed to the inherent characteristics of the algorithm.

We hope this research lays the groundwork for future studies related to the Fractal Decomposition Algorithm (FDA) and its application in solving various
other potential problems, such as the Traveling Salesman Problem, quadratic assignment problems, set covering problems, and numerous other classical or real-world challenges. Additionally, there is potential to explore alternative transfer functions from existing literature and different binarization rules. Furthermore, innovation in the form of implementing machine learning or artificial intelligence techniques could be considered to enhance the algorithm’s performance. This research not only addresses current challenges but also paves the way for a myriad of exciting possibilities and potential advancements in the field of optimization and metaheuristics.

Furthermore, it is crucial to emphasize that this work represents a preliminary effort undertaken during a three-month doctoral internship. It marks the beginning of a more ambitious and extensive project. The progress made in this brief period underscores the potential and significance of the presented approach, laying the foundation for future research that will delve deeper and expand upon these initial findings.

6 Acknowledgements

Broderick Crawford and Ricardo Soto are supported by Grant ANID/FONDECYT/REGULAR/1210810.
Broderick Crawford and Ricardo Soto are supported by Grant DI Investigación Asociativa Interdisciplinaria/VINCI/PUCV/039.347/2023
Marcelo Becerra-Rozas is supported by the National Agency for Research and Development (ANID)/Scholarship Program/DOCTORADO NACIONAL/2021-21210740.
Javier Peña is supported by the National Agency for Research and Development (ANID)/Scholarship Program/DOCTORADO NACIONAL/2022-21220879

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

We deeply appreciate the Big Optimization and Ultra-Scale Computing (BONUS) project team at Inria for their invaluable support during the development of this research throughout the doctoral internship.

References


Optimization of Hydro Generation and Load Forecasting Based On LSTNet

Xingbang Du, Enzhi Zhang

1 Hokkaido University, Sapporo, Hokkaido, 060-0808, Japan
duxingbang916@gmail.com
2 Hokkaido University, Sapporo, Hokkaido, 060-0808, Japan
zhangenzhi657@gmail.com

Abstract. Renewable energy generation and power load forecasting are important in both advanced smart grid and sustainable development. Many RNN related methods were used in prediction of power generation time series data but they often fail to capture very long-term correlations in practice due to the vanishing gradient problem. We introduced a modified version of the LSTNet model, which incorporates CNN, LSTM, SKIP-LSTM and Dense components. This model captures both short-term patterns and addresses the issue of vanishing gradients when capturing long-term patterns. We applied this model to predict hydro power and grid load, and by comparing the MAPE, MSE, and MAE metrics, it is evident that the performance is superior to the commonly used LSTM, GRU, and SimpleRNN models in time series analysis of smart grid. The integrated model can be applied to the intelligence of renewable energy grids.

Keywords: LSTNet · time series · Hydro Generation · Load · Smart Grid.

1 Introduction

Energy sustainability and environmental preservation have become worldwide concerns [1]. Renewable energy refers to energy that can be continuously used and will not be exhausted. It can reduce the negative impact on the environment and achieve sustainable development of energy.

However, renewable energy is greatly affected by environmental factors and not stable. To ensure the stability of the grid, smart grid need to be deployed and applied. Smart grid, also called intelligent grid [2], forms an intelligent electricity network working with all connected components to deliver sustainable electricity supplies [3]. It includes management, optimization and control of power systems through information and communication technology. Smart grid-related technologies include renewable energy integration, energy management [4] and optimization, energy storage [5], electric vehicle interaction with the grid, data sharing and interactive connection. In energy management, it’s necessary to establish a prediction model to forecast unstable renewable energy, and electric
load forecasting is also important [6]. After the output of the forecast results, power supply and demand side [7] of the grid are balanced by combining energy storage technology.

In recent years, with the advancement in the field of deep learning, LSTM [8], GRU [9] and related models have been increasingly applied to time series forecasting problems and have shown promising results. Reference [10] applied GRU to wind power forecasting while Reference [11] utilized GRU to predict short-term power Load. CNN-LSTM was used in electricity theft detection in smart grid [12] and Reference [13] applied LSTM to predict the stability of smart grid. LSTM was also used in wind power short-term prediction in Reference [14]. Reference [15] proposed ConvLSTM and applied it to near-term precipitation forecasting. Reference [16] utilized LSTM to predict hydrological time series data such as flow, precipitation, and evaporation. Additionally, in Reference [17], LSTM was employed to predict short-term load forecasting for individual residential households.

The Long- and Short-term Time-series network (LSTNet) model, proposed in Reference [18], combines CNN, RNN, and auto-regressive linear models and incorporates a skip-connection structure with recursion. It effectively captures short-term and long-term features, making it particularly suitable for predicting time series data with strong periodicity. In the context of power grid, both hydro generation and grid load data exhibit noticeable periodic patterns, as shown in Fig. 1. Therefore, considering the actual data characteristics, we have decided to modify and simplify the LSTNet model for application in forecasting hydropower generation and load in smart grids. This will facilitate the management and
Optimization of Hydro Generation and Load Forecasting Based On LSTNet

2 Methodology

Simple RNN, GRU, and LSTM models are commonly used in time series forecasting problems. These models can remember past information and capture relatively long-term dependencies. However, due to the problem of vanishing gradients, GRU or LSTM models often cannot capture very long-term correlations [18]. In this section, we will discuss the details of modified LSTNet model (as shown in Fig. 2) using real data.

![Fig. 2. Modified LSTNet model. The model consists of 4 components: Firstly, the 1D-CNN component is responsible for extracting short-time patterns in the time dimension. Secondy, the outputs of the 1D-CNN component are fed into the LSTM component which captures long-time patterns. The SKIP-LSTM component utilizes historical periodic data to capture very long-time patterns. Finally, we use the Dense layer component to concatenate the outputs of the LSTM and SKIP-LSTM components and output the prediction results.]

2.1 1D CNN Component

The first layer of LSTNet is a non-pooling CNN used to capture patterns in short-term time series data [18]. Since the data for water and electricity load is one-dimensional and 1D CNN is a specialized CNN commonly used for processing one-dimensional data [19], we first use a 1D CNN model to process the data.

The input sequence is denoted as $X = [x_1, x_2, ..., x_n]$, where $x_i$ represents the $i$-th data point in the time series. This convolution includes several convolutional kernels with a height of 1 and uses ReLU as the activation function. Therefore,
the k-th convolutional kernel processes the input data $X$ and produces the output:

$$O_k = \text{RELU}(W_k \ast X + b_k)$$  (1)

where the $O_K$ is the output and $\ast$ is the convolution operation. We use the operation padding equals to same to ensure that the size of each output $O_c$ is the same as the size of the input $X$, meaning that the length of the output $O_c$ is also $n$. The final output size of this convolutional layer is $K_c \ast n$, where $K_c$ is the number of convolutional kernels.

### 2.2 LSTM Component

The output of the 1D-CNN component will serve as the input for the LSTM component.

The original LSTMNet model utilized GRU as the structure for the RNN layer. However, in time series forecasting, the LSTM model with LSTM memory cells can indicate when the network should forget historical information and when to update the memory cells with new input information [20]. Therefore, through the forget gate and input gate in the gate units, LSTM is better equipped to handle the issues of vanishing gradients and exploding gradients compared to GRU. It exhibits greater stability and is more suitable for processing longer time series data. Hence, though GRU is simpler to compute and implement [21], we use the LSTM structure instead of GRU.

The calculation of the hidden layer node’s state at time $t$ can be represented as follows:

$$i_t = \sigma(W_{xf}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i)$$  (2)

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f)$$  (3)

$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$  (4)

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o)$$  (5)

$$h_t = o_t \tanh(c_t)$$  (6)

In the equation, $i$, $f$, $c$, $o$ represent the input gate, forget gate, cell state, and output gate respectively, $b$ represents the corresponding bias term, $W$ represents the weight matrix between the gates, $\sigma$ represents the sigmoid activation function, and $\tanh$ represents the hyperbolic tangent activation function.

### 2.3 SKIP-LSTM Component

The LSTM structure with memory cells is able to effectively capture long-term dependencies in data. However, due to the vanishing gradient problem, LSTM often fails to capture extremely long-term correlations in practice. It is a well-known fact that power generation from renewable sources
exhibit an evident cyclic pattern [22], especially in the case of hydroelectric power generation, daily water and electricity data exhibit clear periodic patterns, as observed in Fig. 1. The electricity load data, influenced by a number of factors, which can be classified as: economic factor, time, day, season, weather and random effects [23], in particular, displays distinct regularity corresponding to human activities and daily routines, as seen in Fig. 1. Therefore, in addition to using the data from a few hours before 16:00 today to predict the hydroelectric power generation and electricity load at that time, it is evident that we can also leverage data from the same time the previous day or even two days ago to make predictions. In the LSTNet model, a recurrent-skip component with skip connections is designed. This regression structure helps the model capture long-term patterns and correlations based on historical data from the same period, thus avoiding the vanishing gradient issue and optimizing the prediction model [18]. The calculation of this skip-LSTM component, which incorporates the skip connection, can be represented as follows:

\[
\begin{align*}
  i_t &= \sigma(W_{xt} x_t + W_{hi} h_{t-p} + W_{ci} c_{t-p} + b_i) \\
  f_t &= \sigma(W_{xf} x_t + W_{hf} h_{t-p} + W_{cf} c_{t-p} + b_f) \\
  c_t &= f_t c_{t-p} + i_t \tanh(W_{xc} x_t + W_{hc} h_{t-p} + b_c) \\
  o_t &= \sigma(W_{xo} x_t + W_{ho} h_{t-p} + W_{co} c_t + b_o) \\
  h_t &= o_t \tanh(c_t)
\end{align*}
\]

In the equation, \(i, f, c, o\) represent the input gate, forget gate, cell state, and output gate respectively. \(b\) represents the corresponding bias term, \(W\) represents the weight matrix between the gates, \(\sigma\) represents the sigmoid activation function, and \(\tanh\) represents the hyperbolic tangent activation function. \(P\) represents a specific and applicable time period for real data.

### 2.4 Dense Layer

We utilize a fully connected layer [24] to merge the output results of the LSTM component and the SKIP-LSTM component. The output of this fully connected layer is as follows:

\[
  h_t^P = W^L h_t^L + \sum_{i=0}^{p-1} W^S_i h_i^S + b
\]

In the equation, the input of this fully connected layer consists of two parts: the \(h_t^L\) represent the hidden state of the LSTM component at time \(t\), \(h_{t-p}^S, h_{t-p+2}^S, ..., h_t^S\) are the total of \(p\) hidden states from time \(t-p+1\) to \(t\) from the SKIP-LSTM layer.

In the original LSTNet model, considering that the scale of input signals varies significantly in some real-world data and the variation is non-periodic,
which has a significant impact on the prediction accuracy of the model, an auto-regressive component is added to solve the local scaling problem and improve the accuracy and robustness of the model. However, in the hydroelectric power generation data and electric load data, the change in scale is not obvious, as shown in Fig. 1 and Fig. 2, so the model we designed this time canceled the AR part.

3 Hydro Generation and Load Forecasting Model

Hydroelectric data and electricity load data are both single-variable time series data consisting of fixed time interval observations. We assume \( X_{ht} \) represents the hydroelectric data and \( X_{lt} \) represents the load data, where \( t \) denotes the time point, and \( x \) represents the power data. We will establish and train two separate uni-variate time series prediction models.

In order to achieve the prediction of hydroelectric power generation, the model takes consecutive \( n \) time steps of hydroelectric data as inputs, and predicts the hydroelectric power generation data at time step \( t+1 \) as the output. Here, \( n \) corresponds to the size of the input layer steps. The same applies to the electricity load data. Therefore, the expressions for the hydroelectric power generation and electricity load models are as follows:

\[
X_{ht}(t+1) = f(X_{ht}(t-n+1), \ldots, X_{ht}(t-1), X_{ht})
\]  
\[13\]

\[
X_{lt}(t+1) = f(X_{lt}(t-n+1), \ldots, X_{lt}(t-1), X_{lt})
\]  
\[14\]

In the equation, \( X_{ht}(t-n+1), \ldots, X_{ht}(t-1) \), and \( X_{ht} \) represent consecutive \( n \) time steps of hydroelectric data, \( X_{lt}(t-n+1), \ldots, X_{lt}(t-1) \), and \( X_{lt} \) represent consecutive \( n \) time steps of load data, while \( X_{ht}(t+1) \) and \( X_{lt}(t+1) \) are the prediction at time step \( t+1 \).

4 Experiment and Result Analysis

In this chapter, we will introduce the publicly available data used in the experiments, the data preprocessing steps, parameter settings, as well as the experimental results and analysis.

4.1 Data Preparation

In this experiment, we used publicly available data from the Bonneville Power Administration (BPA), specifically the hydroelectric power generation data and electricity load data of the BPA control area from January 1, 2020, to June 30, 2020. BPA is a branch of the U.S. government, and manages the electrical operations in the Pacific Northwest region. The hydroelectric power generation and electricity load data we used were recorded every five minutes for a duration of three months. Example data is shown in Tables 1 and Fig. 3. In chronological order, we select the first 80 percent of the data as the training set and the rest as the test set.
Table 1. 20-minutes of hydro generation and power load data

<table>
<thead>
<tr>
<th>DateTime</th>
<th>Hydro Generation (MW)</th>
<th>Power Load (MW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2020/01/01 00:00</td>
<td>4419</td>
<td>5788</td>
</tr>
<tr>
<td>2020/01/01 00:05</td>
<td>4422</td>
<td>5764</td>
</tr>
<tr>
<td>2020/01/01 00:10</td>
<td>4545</td>
<td>5769</td>
</tr>
<tr>
<td>2020/01/01 00:15</td>
<td>4532</td>
<td>5746</td>
</tr>
<tr>
<td>2020/01/01 00:20</td>
<td>4527</td>
<td>5751</td>
</tr>
</tbody>
</table>

Fig. 3. Daily hydraulic power generation and grid load data, with a data point every five minutes, measured in MW. The difference between hydroelectricity generation and load varies at different times of the day, leading to different management actions in the smart grid. When the power generation exceeds the load, excess energy can be stored. On the other hand, the smart grid management system needs to purchase energy from the power market or utilize the stored energy when the load exceeds the generation, in order to prevent power shortages.
As each time point is spaced at every five minutes, we choose a consecutive sequence of 24 time points as the input dataset. This means that we use a two-hour data window as the input to predict the value at the next time point. Additionally, the SKIP-LSTM component requires the determination of a period P to introduce the data from the same time point of the previous cycle as an input. We have chosen P to be one day, meaning that we utilize data from the same time point one day prior as the input to the recurrent-skip component.

Before training the input model, we normalize the input data to reduce prediction errors caused by large differences between input data. The normalization range is set as [0,1], and the transformation formula is:

$$X^* = \frac{X - \text{min}}{\text{max} - \text{min}}$$

(15)

$$X^* = (X - \text{min}) / (\text{max} - \text{min})$$, where max is the maximum value in the sample data and min is the minimum value. X represents the original training data, and X* represents the normalized data. After making predictions, we need to use inverse normalization to obtain meaningful prediction result data.

4.2 Experiment environment

The hardware specifications for this experiment are as follows: CPU: Intel(R) Xeon(R) Silver 4214R with a clock speed of 2.40GHz and 90GB of memory. GPU: 1* RTX 3080 Ti with 12GB of video memory. The experiment platform is based on the Tensorflow framework version 2.9.0, and the experiment programming language is Python 3.8.

4.3 Result and Analysis

We set the experimental parameters as follows and conducted the tests: Look back = 24, which means input time steps equals to 24; Skip = 12, which means period P equals to 1 day, because $24 \times 12 \times 5 \text{ mins} = 24 \text{ hours} = 1 \text{ day}$. The training and testing datasets were divided in an 8:2 ratio. CNN component: filters = 64, the size of the convolutional kernel = 6, and RELU is the activation function. LSTM component and SKIP-LSTM component: units = 32, and tanh is the activation function. The initial learning rate is set to 0.01, and the minimum learning rate is 0.001. The number of epochs is set to 100, and the batch size is 32. The objective function is set as Mean Squared Error (MSE):

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (Y_t - P_t)^2$$

(16)

In the equation, n represents the number of samples, $Y_t$ represents the label data in the test dataset, and $P_t$ represents the prediction results from the model. The
Fig. 4. The variation of the loss function values during the training process of the hydroelectricity prediction model and load prediction model. We utilize Mean Squared Error (MSE) as the objective function to train the models, and the data has been normalized prior to model training.

The predicted results of hydroelectric power and power grid load predictions with the same time period are shown in Fig. 5 and Fig. 6.

Fig. 5. Comparison of predicted results of hydro-power model with actual data in the test set. The predicted results closely align with the actual data.

Then we integrate the results of hydroelectric power generation prediction and grid load prediction to forecast the specific time periods when hydroelectric power can fully supply the regional electricity demand and store excess power, as well as the time periods when the region needs to purchase electricity from the power market to prevent partial power outages caused by insufficient hydroelectric power. The result of integration is shown in Fig. 7.

Table 2 and Table 3 present the comparison between the experimental results of the model in this study (modified LSTNet) and other commonly used models (LSTM, GRU, and simple RNN). The tests were conducted under the same parameters using the same dataset, and all predicted values are subject to reverse normalization then metric calculation.
Fig. 6. Comparison of predicted results of load model with actual data in the test set. The predicted results closely align with the actual data.

Fig. 7. Integration of Hydro and Load Prediction Result, which can be used to assist the intelligent power grid management system to adjust its strategy based on the GAP between the predicted electricity generation and the predicted load, making decisions to either purchase electricity from the market or utilize excess electricity for storage.

Table 2. Comparison of hydro-power prediction between this model and other models. The MSE, MAE, and MAPE metrics of this model’s prediction results are superior to those of the LSTM, SimpleRNN, and GRU models.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>LSTNet</th>
<th>LSTM</th>
<th>SimpleRNN</th>
<th>GRU</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE</td>
<td>5711.55</td>
<td>5857.39</td>
<td>6126.40</td>
<td>5934.09</td>
</tr>
<tr>
<td>MAE</td>
<td>55.29</td>
<td>55.73</td>
<td>57.04</td>
<td>56.28</td>
</tr>
<tr>
<td>MAPE</td>
<td>0.460</td>
<td>0.470</td>
<td>0.480</td>
<td>0.470</td>
</tr>
</tbody>
</table>

Table 3. Comparison of load prediction between this model and other models. The MSE, MAE, and MAPE metrics of this model’s prediction results are superior to those of the LSTM, SimpleRNN, and GRU models.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>LSTNet</th>
<th>LSTM</th>
<th>SimpleRNN</th>
<th>GRU</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE</td>
<td>1761.17</td>
<td>1804.99</td>
<td>1860.00</td>
<td>1813.02</td>
</tr>
<tr>
<td>MAE</td>
<td>28.83</td>
<td>29.28</td>
<td>29.67</td>
<td>29.41</td>
</tr>
<tr>
<td>MAPE</td>
<td>0.486</td>
<td>0.493</td>
<td>0.499</td>
<td>0.496</td>
</tr>
</tbody>
</table>
From the test results, it can be observed that the modified version of the LSTNet model used in this study demonstrates improvements in the MSE, MAE, and MAPE metrics for hydroelectric power prediction and power grid load prediction compared to several commonly used models in the past.

5 Conclusion

In this paper, we introduced a modified version of the LSTNet model, which incorporates 1D-CNN, LSTM, SKIP-LSTM and Dense components. This model captures both short-term patterns in time series data and addresses the issue of vanishing gradients when capturing long-term patterns. We applied this model to the prediction of hydroelectric power and grid load, and from the test results, it is evident that the performance is superior to previous models. In future work, we will combine various forms of renewable energy such as wind and solar power to design and establish prediction models that cover all renewable energy sources. We aim to improve the performance of the models and integrate them with the latest developments in intelligent grid management and decision-making systems, accelerating the substitution of fossil/nuclear energy sources with renewable energy in the power grid.

References

HOTS : A containers resource allocation hybrid method using machine learning and optimization.

E. Leclercq, Jonathan Rivalan
Smile, France
etienne.leclercq@smile.fr
jonathan.rivalan@smile.fr

Abstract. One of the main challenges in cloud computing is resource management, the ability to schedule workloads and services over the infrastructure in the most automated way. By optimizing cloud assignment and resource usage, energy can be saved, production incident can be anticipated and services QoS improved. With the recent years emergence of light virtualisation, known as containerization, the resource allocation problem was brought back, notably to support containers elasticity, hence the dynamic allocation of resource at runtime at a single service scale.

In this paper we show that using an hybrid loop system, which combines unsupervised learning and optimization techniques, our algorithm provides and iteratively improves scheduling solutions to containers resource assignment, enabling capacity planning over dynamic resource loads. Within our benchmarks, these solutions outperform state of the art algorithms, by an average of 6.3%, while providing more expressivity and control over input parameters.

We describe also the implementation of this method, through an open source Python library called HOTS, which allows hybrid optimization for time series based use cases.

1 Introduction

The resource management is one of the main important task in operating Cloud infrastructures. This is why the resource allocation problem has been and is still actively being studied, classically with virtual machines and more recently with containers.

The latter opens for both new perspectives and risks coming to resource allocation: on one hand, a dynamic use of shared resources, enabling elasticity of the overall resource usage, and on the other hand, an erratic resource consumption behavior, putting at risk capacity planning or static resource assignments. Due to this dynamic use of resources, containers assignment problem modeling is a complex task, as their runtime elasticity provides an unknown parameter prior to assignment.

In order to have an optimized containers management platform, while taking into account all the induced constraints, we consider a hybrid machine learning / optimization approach that addresses our resource allocation problem, where unsupervised learning is used to make informed heuristic choices towards scheduling. As, in the production environment, new metrics are created continuously, we developed a method using a loop mechanism, which aims at evaluating current solution with new metrics, and updating this solution if needed. Since in production services are running in real time, the solution must be found quickly, modeling the full problem with all its criteria and constraints cannot be considered. Thus we use clustering information, updated in time using optimization methods, to evaluate the current assignment solution and update it if needed.

After describing our method and its implementation, we will describe its evaluation over generated and real datasets, to provide various evaluation scenarios. Finally, we will discuss the related works before concluding the paper by opening on some future research and work perspectives.

2 Loop description

Before entering the methodology details, we first give some information about the data and the time which are important in understanding our method description.
2 Leclercq and Rivalan

2.1 Preliminary information

During the execution, at any given time $t$, we have a clustering solution and an assignment solution (computed at time $t - \tau$ or at initialization) which we want to update in order to reflect the data evolution (in our case, changes in used resource trends). This section aims to explain how we achieve this update using optimization techniques. First we give relevant information introducing the method, then we describe the optimization models used for both clustering and assignment problems, and how these models are used in time to update the existing solutions.

Data description The method presented here uses time series data, which represents in our case the evolution of resource consumption by micro-services as part of a container based hosting system. An example of these data is displayed in Figure 1, through a trial dataset, the vertical line showing the end of the first analysis timeframe (offline) and the beginning of the retro action loop (online).

![Figure 1: Resource consumption of 10 containers on 12 datapoints, displaying timeframes with $\tau = 3$](image)

**Time management** Each loop $t$ is associated to one time window $T_t$, in which we have the resource consumption data of each individual $c_i$, denoted $D_t$. The duration of one time window is denoted $\Theta$, and this window progress in time at ticks $\tau$, with overlaps. An example of two time windows is in Figure 2.

![Figure 2: Data at $T_2$ and $T_{t+1}$ with $t = 7$, $\Theta = 6$ and tick $\tau = 1$.](image)

**Inputs of loop** At each loop $t$ on timeframe $T_t$, we already have a clustering solution $C_{t-1}$ and an assignment solution $A_{t-1}$, that are generated from the timeframe $t - 1$, or at initialization if the current loop is the first considered.
2.2 Optimization models

In order to evaluate the current solutions with new data $D_t$, both clustering and assignment problems are modeled as Integer Linear Problem (ILP), called respectively $CP$ and $AP$. As in our context new data is incoming continuously, the solution must be found quickly, then we can’t consider modeling the full problems with all criteria and constraints, and these programs are not the exact modeling for these problems. They are actually used, with information of current solutions and considering the continuous relaxations (respectively $CCP$ and $CAP$), for evaluating these solutions according to an adapted objective, using the dual values of added constraints. The notations used in the following sections are given first in Table 1, then the details of the models are given (see Figure 3 and Figure 4 respectively for clustering and assignment models) before explaining how the solutions $C_t$ and $A_t$ can be adjusted.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I$</td>
<td>Number of containers</td>
</tr>
<tr>
<td>$M$</td>
<td>Number of nodes</td>
</tr>
<tr>
<td>$K$</td>
<td>Number of clusters (clustering)</td>
</tr>
<tr>
<td>$T$</td>
<td>Total time</td>
</tr>
<tr>
<td>${o_1, o_2, \ldots, o_I}$</td>
<td>Set of containers</td>
</tr>
<tr>
<td>${n_1, n_2, \ldots, n_M}$</td>
<td>Set of nodes</td>
</tr>
<tr>
<td>${c_1, c_2, \ldots, c_K}$</td>
<td>Set of clusters</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Color</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green</td>
<td>Data</td>
</tr>
<tr>
<td>Blue</td>
<td>Variables</td>
</tr>
<tr>
<td>Red</td>
<td>Parameters</td>
</tr>
</tbody>
</table>

Table 1: Notations used for the optimization models

\[
\min \sum_{(i,j)\in[1..I]} w_{i,j} u_{i,j} \\
\text{s.t.} \sum_{k=1}^{K} y_{i,k} = 1 \quad \forall i \in [1..I] \\
y_{i,k} \leq b_k \quad \forall i \in [1..I], k \in [1..K] \\
\sum_{k=1}^{K} b_k \leq nb\_clusters \\
u_{i,j} = \sum_{k=1}^{K} y_{i,k} y_{j,k} \quad \forall (i,j) \in [1..I] \\
\]

Fig. 3: Clustering model $CP$

\[
\min \sum_{(i,j)\in[1..I]} u_{i,j}v_{i,j} + (1 - u_{i,j})v_{i,j}d_{i,j} \\
\text{s.t.} \sum_{m=1}^{M} x_{i,m} = 1 \quad \forall i \in [1..I] \\
x_{i,m} \leq a_m \quad \forall i \in [1..I], m \in [1..M] \\
\sum_{m=1}^{M} a_m \leq max\_nodes \\
v_{i,j} = \sum_{m=1}^{M} x_{i,m} x_{j,m} \quad \forall (i,j) \in [1..I] \\
\]

Fig. 4: Assignment model $AP$

While the clustering optimization model is already known in the litterature, the novelty in these optimization models is the use of clustering solution to formulate the objective function for the placement problem. The use of these models is also innovative: as mentioned before, we add additional constraints to these optimization models, in order to add current solution information to the models. We call these constraints mustLink constraints, defined by the following equation for the clustering problem:
mustLink \(_C\) : \(u_{i,j} = 1\) \(\forall (i,j)\) belonging to same cluster in \(C_t\)

In the same idea, the mustLink constraint added in the assignment model is as following :

mustLink \(_A\) : \(v_{i,j} = 1\) \(\forall (i,j)\) assigned to the same node in \(A_t\)

### 2.3 Updating solutions

For both clustering and assignment problems, we solve the continuous relaxation of the problems, respectively \(CCP\) and \(CAP\), and compute the values of the dual variables associated with the mustLink constraints (respectively mustLink \(_C\) and mustLink \(_A\) ). These dual variables are named respectively \(\overline{u}_{ijt}\) for the clustering and \(\overline{v}_{ijt}\) for the assignment. We consider these models together, tackling a two-level problem: if the clustering is not modified, the assignment is not evaluated. But if the clustering changes can not improve the assignment solution, these changes are not applied.

The overall loop process is illustrated in Figure 5.

[Diagram of the loop process]

**Fig. 5**: Schema of retro action loop global working

**Updating clustering** Let \(\overline{u}_{ijt-1}\) be the dual variable associated to the constraint 2.2 for previous time period \(t - 1\), and \(\epsilon_C\) a tolerance coefficient. For each constraint 2.2, if \(\overline{u}_{ijt} > \overline{u}_{ijt-1}(1 + \epsilon_C)\), then we consider the containers \(i\) and \(j\) in conflict.

For every container, taken in descending order of conflicts number, we compute the distance with other clusters mean profiles, and assign it to the cluster with the lowest distance. As soon as a container is assigned, all the conflictive constraints with this container are removed, in order not to deal with containers in conflict. If some containers appear only once in conflict constraints, we chose to move the furthest one from the cluster mean profile. This new clustering solution is used to update the data \(u_{i,j}\) in the assignment problem \(AP\).

**Updating assignment** As for the clustering, we compare \(\overline{v}_{ijt}\) values with \(\overline{v}_{ijt-1}\), with a given tolerance coefficient \(\epsilon_A\). For each constraint 2.2, if \(\overline{v}_{ijt} > \overline{v}_{ijt-1}(1 + \epsilon_A)\), then we consider the containers \(i\) and \(j\) in conflict. For every container, taken in descending order of conflicts number, we assign the container on the node for which the total consumption variance on the node is the lowest. If we have containers that appears only once in conflicts constraints, we chose the one that gives, after removing its consumption on its actual node, the lowest total consumption variance on this node.

The constraints mustLink \(_C\) and mustLink \(_A\) are updated with the new solutions and the linear relaxation of the new models \(CP\) and \(AP\) are immediately solved to allow the comparison of the dual variables for next period \(t + 1\).
2.4 Initial solutions

For any loop execution, we start from existing clustering and assignment solutions. The initial solutions for these two problems are given by two heuristics performed on an historized time window. An initial clustering of containers resource consumption (represented by time series) is performed using a classical K-means algorithm [MacQueen, 1967] over the time series. It produces containers groups having similar resource consumption profiles. This information is then used to feed an assignment heuristic, in which the main objective is to co-localize containers with distant profiles, in order to have the most stable total consumption in nodes and the lower number of nodes in use. The heuristic used is based on a greedy heuristic for bin packing problem called First Fit Decreasing [Garey and Johnson, 1981]: the cluster pairs are ordered by decreasing distance between mean profiles, then containers from the two most distant clusters are co-localize, before considering the next two most distant clusters. This step gives two solutions: one clustering $C_0$ and one assignment $A_0$ that will be both used as initial inputs for the loop.

3 HOTS implementation

In order to implement our method described in the previous section, we developed a Python library named HOTS (Hybrid Optimization for Time Series). This library, published as a PyPi package [Smile, 2023b] and available on GitHub [Smile, 2023a], allows the user to obtain a configuration strategy from a metrics dataset, describing in our case containers resources consumption evolution in a given set of servers.

We describe first the library working process before talking about its architecture and modularity.

3.1 Working process

In its current implementation, the HOTS library considers historical data as its informational input. This data is made available through .CSV files representing resources consumption evolution by containers.

When HOTS is launched, the consumption data is divided in two parts (this separation is configurable through a parameter):

- Data for time less than $\text{sep} \text{ time}$ will form the analysis data for the analysis period, whose goal is to obtain the first clustering and placement solutions (see Section 2.4);
- Data for time greater than $\text{sep} \text{ time}$ will form the running period in which we execute the loops (see Section 2).

3.2 Architecture and modularity

Regarding the code implementation, specific work has been done aiming at making this code as modular as possible, in order to support any metrics based usecases.

This modularity enables customization by the user of each step from the method. For example, the first clustering solution is obtained by default using the K-means algorithm, but we also implemented a hierarchical clustering method and a spectral method, and the user can also easily add his own clustering method.

This goal translates through the library implementation and structure, enabling the possibility of implementing new use cases. This last point is illustrated by two examples, described below.

Modularity for solvers All actions (build, solve, update, dual variables use...) around optimization problems are grouped in the file model.py. However, this file is an interface in which the different functions are developed, using in particular the Python Pyomo [Pyomo, 2020] module. With regard to the exact modeling of problems, the user can provide its own file defining all the constraints, variables and objective functions of its problems. By default, a file defining the models as described in Section 2.2 is provided with the library.

Moreover, the interface uses high-level functions to solve optimization problems. These functions call on a solver; in particular to obtain the values of the dual variables used in our method. There
are several solvers, the most common of which is CPLEX. The latter being a proprietary tool, and as we published our library to the open source community, we allow the user to implement, through Pyomo, the solver he wishes to use, by providing his own problem definition and the desired solver. During the development of HOTS, we tested this modularity with the solvers CPLEX and GLPK (which is open source).

**Modularity for use cases solving** The HOTS library was developed for the dynamic placement of containers on an infrastructure. Its development responds above all to the need to implement and evaluate the method described in Section 2. We can imagine that this type of methodology, combining individuals profiling and optimization techniques in a loop system, can be used to answer other problems than ours. This is why the business aspect appearing in the library is also the subject of a modular development: all the actions concerning the containers placement (the heuristic used in the analysis period, the container moves in loops etc ...) are grouped in the file `placement.py`. To use the same methodology but apply it to a different problem, users can easily provide new functions in a module inspired by `placement.py`.

To give an example, we explored the problem of resources level allocated to containers. For this, we have created an additional module named `allocation.py` to first develop use cases and tests, which are then directly used in the main methodology. This exploratory work extends the issue of container placement in a more global vision of an optimized resource allocation.

### 4 Experiments

In this section, we present our method evaluation through several scenarios and compare business and scientific domain related criteria and results with other methods.

#### 4.1 Domain efficiency

As our method is applied on a precise business problem, the containers placement, we first want to evaluate this method through business related criteria and compare it to others well known methods. In Cloud infrastructures, one of the most important criteria is the energy consumption, mainly impacted by the number of used servers. We also believe that the more stable these servers consumption is, easier is the infrastructure management (less consumption peaks, better problems anticipation). We then identified 4 quantitative criteria allowing the comparison:

- \( c_1 = \sum_{n=1}^{N} a_n \), with
  \[
  a_n = \begin{cases} 
  1 & \text{if node } n \text{ is open (containers on it)} \\
  0 & \text{else} 
  \end{cases}
  \]
  \( c_1 \) represents the number of nodes, which we are trying to minimize.

- \( c_2 = \max_{n \in N} \left( \max_{t \in T} \left( \text{cons}_{n,t} \right) - \min_{t \in T} \left( \text{cons}_{n,t} \right) \right) \)
  \( c_2 \) represents the greatest amplitude of consumption among all the nodes. This indicates a level of nodes consumption stability that we want to maximize, so this criterion is minimized.

- \( c_3 = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T} \left( \text{cons}_{n,t} - \overline{\text{cons}_{n,t}} \right)^2 \)
  \( c_3 \) represents the average consumption variance over all nodes, which is another index of consumption stability, thus we want to minimize it also.

- \( c_4 \) is the number of times where the maximum node capacities have been reached. In HOTS, a fixing function has been implemented for all methods, which moves containers if one node has reached its capacity. We want to avoid these overloads which can lead to production issues.

Then we identified 5 different methods to be compared using the previous criteria:

- **initial**: The initial assignment, as supplied in the datasets
− *spread*: A classic spread technique, available by default in schedulers [Docker, 2016a], has also been implemented for comparison. This technique consists in going through the containers one by one, and place them on nodes by going through them iteratively (the first container on the first node, the second container on the second node, and so on).

− *iter-consol*: Another technique has been implemented which uses the analysis period to set the minimum number of needed nodes, and then apply the container spread technique on this number of nodes.

− *heuristic*: Our assignment solution, resulting from the heuristic at the end of the analysis period without executing any loop in the following, is another considered method.

− *loop*: The execution of the loops after the heuristic solution, as described in Section 2.

### 4.2 Clustering evaluation

In our method, we present also a new way to update a clustering in time with new data incoming. We also want to evaluate this streaming clustering method.

The clustering solution evaluation is not trivial, and we often use other business related criteria to perform this evaluation. However, several performance indicators have been introduced, the most used being the *Silhouette coefficient* [Rousseeuw, 1987].

This criteria is compared between the 3 following methods updating the clustering:

− *loop_cluster*: clustering used in our loop, as described in Section 2;

− *kmeans_scratch*: apply the K-means method [MacQueen, 1967] from scratch at each loop;

− *dynamic_kmeans*: result of StreamKM++ algorithm, as described in [Ackermann et al., 2012], one of the most used streaming clustering algorithm.

### 4.3 Datasets

Two real use case datasets are used for this evaluation:

− Internally within Alter way (a SMILE company), data from three nodes is continuously reported. A one-week digest of these data made it possible to obtain an initial dataset with 3 nodes and resource consumption from 354 containers, each one having 973 different datapoints.

− Alibaba publicly provided datasets from a part of their cloud infrastructure [Alibaba, 2018], one of them representing around 3000 nodes for 8 days of data. To limit the benchmarking time over the various methods, we built a subset of 50 nodes from this dataset, having 1350 datapoints for each one of 900 containers.

Moreover, to evaluate our method’s resistance to more disturbed data, as well as to better reflect the diversity of possible consumption data on the infrastructures, we generated new datasets, following different main profiles and different scenarios (new profile, profile disappearance).

### 4.4 Parameters

Amongst the parameters present in the Python module *HOTS*, we assess the impact of two parameters in the results obtained:

− *k*: the number of clusters used for the clustering part. In order to fix it, we perform a priori analysis on the dataset, using the *Gap Statistic* [Tibshirani et al., 2001] on each dataset and run *HOTS* with the return value for *k*.

− *τ*: the size of the considered time window. The impact of this parameter is studied by running *HOTS* with multiple value for *τ*, from 5 to 25 (percentage of entire dataset).

### 4.5 Results

We use different ways to show the results:

− criteria *c₁* and *c₄* are presented in tables, in average of multiple runs (multiple *τ* values);

− criteria *c₂* and *c₃* are presented using performance profiles [Dolan and More, 2002] (showing the probability that one method gives a solution at distance *x* from the best solution for all considered methods)
Domain evaluation For the sake of readability, we show in Table 2 the average values for criteria $c_1$ and $c_4$ for all methods and all datasets. The first domain related criterion $c_1$, representing the number of nodes, is equally improved using the methods $\text{iter-consol}$, the heuristic alone $\text{heur}$ and our $\text{loop}$ method. When looking at the criterion $c_4$, we see that, by using more nodes, the initial (for real datasets, synthetic ones having no initial assignment) and spread methods are less likely to reach maximum node capacities. However, our loop provides an additional property to manage overload risks, thanks to an online placement adjustment which takes into account profiles evolution. That’s why although one cannot win on this criterion $c_4$ compared to the first two methods, we can still keep a very low number of overloads with our loop compared to the other saving nodes methods.

![Performance profiles](image1)

![Performance profiles](image2)

![Performance profiles](image3)

![Performance profiles](image4)

Fig. 6: Comparison of criteria $c_2$ and $c_3$ on both real and synthetic datasets.

In Figure 6 we show the performance profiles considering the stability criteria $c_2$ and $c_3$ and comparing all the methods. We see that, while our method outperforms other saving nodes methods on these criteria, the methods $\text{initial}$ and $\text{spread}$ are still better (especially for real data). Nonetheless, it is difficult to compare stability criteria when methods use different number of nodes, as it is easier to flatten total nodes consumption with more nodes in use. To allow a better comparison between all the methods focusing on stability criteria, we ran the same experiments by specifying a fixed number of nodes to use for all methods. With these new results, shown in Figure 7, we see the prevalence of our method over the other methods. These observations are in line with the overloads criterion $c_4$ shown in previous tables: by improving the consumption stability on nodes, we reduce the overload risk.
Clustering evaluation In addition to the Silhouette coefficient for evaluating the clustering, we computed also the number of changes in either the clustering and the placement solutions during execution. The Table 3 thus show the average values over multiple runs for the Silhouette indicator ($\text{Silh.}$), the number of changes in clustering ($\text{moves}_C$) and the number of changes in placement ($\text{moves}_A$).

We can see that the clustering intrinsic criteria is not dominated by any method. However, we observe through these results, that less containers moves are required using our method rather than using the two others, providing a better stability in resulting placement solutions. This point can be important in a context of the production environment, where moving containers from one node to another can be cost inefficient and errors prone.

Results conclusion With this evaluation, we show an improvement in the performance criteria specific to the business operations, in particular by the energy savings represented by the use of a reduced number of servers, while maintaining load stability on the latter, which reduces the risks of production incidents.

We also showed that our new clustering update method remained consistent from the clustering intrinsic criteria point of view. In addition, the indicators specific to the placement business problem, for which our clustering modeling was designed, show the interest of this updating method: the business criteria are improved with our method and the obtained solutions are more stable in time.

Finally, the datasets presented above are from one week long real containers usage, and our module execution runtime did not exceed 33 minutes, with each loop executed from 30 seconds to 4 minutes. It allows us to be in line with the domain constraints (new metrics created continuously) and to foresee the support for bigger datasets or metrics sets.

5 Related work

In this section, we discuss the existing work, in our use case domain first, then in the combined use of optimization and machine learning.

5.1 Containers scheduling

In the cloud computing domain, the allocation problem is actively studied, especially for VMs technology [Usmani and Singh, 2016]. However, containers technology brings new constraints, such as elasticity [Al-Dhraibi et al., 2017] which leads to potentially erratic and unpredictable consumptions.

Currently, the Docker scheduler [Docker, 2016b] use some naive placement strategies (random, spread, binpack). But this scheduler does not take into account the resources evolution
while placing containers on clusters [Hassen, 2016]. Many works have been done in developing autoscalers [Awada and Barker, 2017, Barna et al., 2017, Fokaefs et al., 2016], but autoscalers are highly application-dependant.

Based on load balancing, the authors in [Sureshkumar and Rajesh, 2017] proposed a Docker container scheduling algorithm to dynamically control the load of each container within a certain threshold in the cluster, so that the load of each container is well balanced. It considers the levels of resources allocated, but not the node assignment as we do.

In [Kaewkasi and Chuenmuneewong, 2017], the authors proposed a container scheduling algorithm based on ant colony optimization algorithm (ACO), the purpose of which is to balance the use of resources so that applications in container clusters will have better performances. More recently, the authors of [sa-, 2020] proposed a hybrid method using Simulated Annealing (SA) and Moth Flame Optimization Algorithm (MFOA). These works take into account reserved resources.

Table 2: Average c1 and c4 values on real and synthetic data.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>c1</th>
<th>c4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alibaba</td>
<td>initial</td>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>spread</td>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>iter-consol</td>
<td>46.8</td>
<td>3.21</td>
</tr>
<tr>
<td></td>
<td>heuristic</td>
<td>46.3</td>
<td>2.73</td>
</tr>
<tr>
<td></td>
<td>loop</td>
<td>46</td>
<td>0.20</td>
</tr>
<tr>
<td>Alter way</td>
<td>initial</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>spread</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>iter-consol</td>
<td>2</td>
<td>2.85</td>
</tr>
<tr>
<td></td>
<td>heuristic</td>
<td>2</td>
<td>1.69</td>
</tr>
<tr>
<td></td>
<td>loop</td>
<td>2</td>
<td>0.15</td>
</tr>
</tbody>
</table>

(a) Real datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>c1</th>
<th>c4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stab_5n</td>
<td>spread</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>iter-consol</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>heuristic</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>loop</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Add_5n</td>
<td>spread</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>iter-consol</td>
<td>5</td>
<td>6.3</td>
</tr>
<tr>
<td></td>
<td>heuristic</td>
<td>5</td>
<td>4.1</td>
</tr>
<tr>
<td></td>
<td>loop</td>
<td>4.9</td>
<td>1.7</td>
</tr>
<tr>
<td>Change_5n</td>
<td>spread</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>iter-consol</td>
<td>5</td>
<td>11.7</td>
</tr>
<tr>
<td></td>
<td>heuristic</td>
<td>5</td>
<td>4.5</td>
</tr>
<tr>
<td></td>
<td>loop</td>
<td>5</td>
<td>1.2</td>
</tr>
<tr>
<td>Stab_10n</td>
<td>spread</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>iter-consol</td>
<td>8.7</td>
<td>3.9</td>
</tr>
<tr>
<td></td>
<td>heuristic</td>
<td>8.7</td>
<td>3.4</td>
</tr>
<tr>
<td></td>
<td>loop</td>
<td>8.7</td>
<td>2.9</td>
</tr>
<tr>
<td>Add_15n</td>
<td>spread</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>iter-consol</td>
<td>15</td>
<td>22.3</td>
</tr>
<tr>
<td></td>
<td>heuristic</td>
<td>15</td>
<td>1.8</td>
</tr>
<tr>
<td></td>
<td>loop</td>
<td>15</td>
<td>1.8</td>
</tr>
<tr>
<td>Change_10n</td>
<td>spread</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>iter-consol</td>
<td>9.6</td>
<td>2.3</td>
</tr>
<tr>
<td></td>
<td>heuristic</td>
<td>9.5</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>loop</td>
<td>9.3</td>
<td>0.2</td>
</tr>
</tbody>
</table>

(b) Synthetic datasets.

Table 3: Clustering evaluation on real and synthetic data.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>Silh. moves</th>
<th>c1</th>
<th>c4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alibaba</td>
<td>loop_cluster</td>
<td>0.769</td>
<td>21.54</td>
<td>12.02</td>
</tr>
<tr>
<td></td>
<td>stream_km</td>
<td>0.771</td>
<td>54.71</td>
<td>21.32</td>
</tr>
<tr>
<td>Alter way</td>
<td>loop_cluster</td>
<td>0.952</td>
<td>18.23</td>
<td>23.03</td>
</tr>
<tr>
<td></td>
<td>stream_km</td>
<td>0.948</td>
<td>45.19</td>
<td>66.42</td>
</tr>
<tr>
<td>Stab_5n</td>
<td>loop_cluster</td>
<td>0.705</td>
<td>11.54</td>
<td>19.10</td>
</tr>
<tr>
<td></td>
<td>stream_km</td>
<td>0.703</td>
<td>19.42</td>
<td>41.70</td>
</tr>
<tr>
<td>Add_5n</td>
<td>loop_cluster</td>
<td>0.778</td>
<td>3.30</td>
<td>3.71</td>
</tr>
<tr>
<td></td>
<td>stream_km</td>
<td>0.758</td>
<td>19.23</td>
<td>21.51</td>
</tr>
<tr>
<td>Change_5n</td>
<td>loop_cluster</td>
<td>0.777</td>
<td>3.50</td>
<td>103.06</td>
</tr>
<tr>
<td></td>
<td>stream_km</td>
<td>0.640</td>
<td>39.50</td>
<td>103.06</td>
</tr>
</tbody>
</table>

(c) Real datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>Silh. moves</th>
<th>c1</th>
<th>c4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stab_5n</td>
<td>loop_cluster</td>
<td>0.918</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>stream_km</td>
<td>0.919</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Add_5n</td>
<td>loop_cluster</td>
<td>0.648</td>
<td>17.21</td>
<td>88.00</td>
</tr>
<tr>
<td></td>
<td>stream_km</td>
<td>0.652</td>
<td>24.17</td>
<td>82.14</td>
</tr>
<tr>
<td>Change_5n</td>
<td>loop_cluster</td>
<td>0.705</td>
<td>11.54</td>
<td>19.10</td>
</tr>
<tr>
<td></td>
<td>stream_km</td>
<td>0.703</td>
<td>19.42</td>
<td>41.70</td>
</tr>
<tr>
<td>Stab_10n</td>
<td>loop_cluster</td>
<td>0.777</td>
<td>3.30</td>
<td>3.71</td>
</tr>
<tr>
<td></td>
<td>stream_km</td>
<td>0.758</td>
<td>19.23</td>
<td>21.51</td>
</tr>
<tr>
<td>Add_15n</td>
<td>loop_cluster</td>
<td>0.569</td>
<td>51.54</td>
<td>325.02</td>
</tr>
<tr>
<td></td>
<td>stream_km</td>
<td>0.577</td>
<td>54.71</td>
<td>307.07</td>
</tr>
<tr>
<td>Change_10n</td>
<td>loop_cluster</td>
<td>0.640</td>
<td>32.81</td>
<td>73.63</td>
</tr>
<tr>
<td></td>
<td>stream_km</td>
<td>0.640</td>
<td>39.50</td>
<td>103.06</td>
</tr>
</tbody>
</table>

(d) Synthetic datasets.
by containers, and set the assignment solution at the beginning of the workload, while our method is able to adapt the solution at runtime, considering changes in resources consumption levels.

5.2 Optimization and machine learning

Moreover, combined use of machine learning and optimization in this context is not yet widespread. In [Galstyan et al., 2004], the authors use reinforcement learning to improve a resource allocation solution in a grid. The clustering problem can be modeled by an optimization problem [Ouali et al., 2016]. In [Weichert et al., 2019], a study of machine learning uses for optimizing production processes shows that it is hard to have correlation between the used data, the machine learning algorithms and the used optimization techniques.

Optimization and machine learning techniques are already widely used in container based architectures, but mainly separately. In addition to optimization algorithms mentioned in previous section, machine learning techniques are used to solve several containers related problems, such as resource usage prediction, anomaly detection, or performance analysis [Zhong et al., 2022].

Our method claims for a strong interaction between the machine learning part (clustering) and the optimization one (assignment problem, ILP and duality), using the one to guide the solution of the other. Indeed, the clustering allows us to have typical profiles that are used to guide the heuristic allocation. Then, clustering and assignment problems are described as Integer Linear Programs (ILP), from which the continuous relaxations are used to get information about how to adapt the clustering and the allocation, when new data is made available.

6 Conclusion and future work

In this paper, we introduced a new method for containers assignment, combining unsupervised learning and optimization in a loop, resulting in adaptative solutions for configuring their execution environment. We compared our loop efficiency with other methods, such as the classic spread method used in current schedulers, by running benchmarks over real and generated datasets, showing that domain related objectives were improved, servers in use diminished and volatility limited. We also presented this method implementation through a Python library named HOTS, available online as an open source PyPI package, with high modularity and customization possibilities, allowing its use for various use cases involving time series data.

This is a work in progress, and further research directions include various aspects of the project.

One is to enrich the business problem solution, by improving the assignment strategy, providing for example better choice for the new node for each moving container. A specific work could also explore the resources level definition problem, providing a complete scheduling coverage, and opening for new considerations within the optimization loop.

The method itself can also be improved, independently from the business view, by considering other ways to adapt the clustering (with incremental clustering [Lin et al., 2004] for example). The time management can also be improved, by considering a dynamic time window \( \tau \) (depending on number of changes on clustering for example), to discover new consumption profiles along new periodicities.

We tested our method on synthetic and real datasets, and are now ready to tackle bigger datasets at faster paces, in order to scale up and confirm our initial results in live contexts.

References


Approaching Single-Episode Survival Reinforcement Learning with Safety-Threshold Q-Learning

Filipo Studzinski Perotto, Melvine Nargeot, and Aymane Ouahbi

ONERA, DTIS, Toulouse, France
filipo.perotto@onera.fr

Abstract. Survival Reinforcement Learning is a specific type of RL problem that is constrained by a risk of ruin. The underlying stochastic sequential decision process with which the agent interacts includes a budget that evolves over time with the received rewards and must remain positive throughout its entire lifetime. The goal is to find a good trade-off between exploration, exploitation, and safety, during a single learning episode, maximizing rewards while managing the available budget to minimize the probability of ruin. Existing approaches do not provide satisfactory solutions to this problem. This paper introduces the safety-threshold heuristic, which is used to extend the standard Q-Learning method. A simulated grid environment was used to evaluate its performance, yielding positive results.

1 Introduction

Reinforcement Learning (RL) and particularly Deep RL algorithms have achieved great success in a wide range of applications in recent years. RL techniques enable an agent to improve its behavior while interacting with an unknown environment, only guided by reward signals. However, the baseline RL methods are not suitable to be applied directly on expensive or safety-critical real-world systems due to the lack of integrity guarantees during the learning process. This difficulty constitutes a barrier to the use of RL techniques in non-simulated environments since the exploratory behavior of these algorithms can potentially lead the system to catastrophic states or cause unsustainable losses. In some cases, even an apparently good initial policy learned from simulation cannot provide enough guarantees that it can be correctly transferred to the real world. In other cases, reliable simulators are simply not available.

Safe RL is the subdomain of RL focused on systems that can learn from experience while being robust to strong disturbances, avoiding dangerous side-effects of trial-and-error and satisfying safety constraints. Survival RL is, in its turn, a particular case of Safe RL in which the agent is constrained by a budget that represents a vital, limited, and expendable resource, but that can be recharged during running with the received rewards. The process terminates in failure if the budget is over, requiring the agent to mitigate that risk of ruin during a single run (i.e. without reset). Most RL methods capitalize on the episodic nature of simulated environments, in which reaching a catastrophic state can be penalized with a significantly bad reward, helping the agent to learn about what should be avoided in subsequent episodes. That assumption is particularly important for model-free methods, which suffer from high sample inefficiency. However, in non-episodic scenarios, the agent must learn to avoid catastrophic states without ever experiencing them.

In a survival RL problem, the agent deals with a risk of ruin while facing a single-life process. The objective is finding a strategy to prevent from being ruined throughout that unique lifetime while still trying to learn an optimal policy with minimal regret. Some practical applications corresponding to Survival RL scenarios can include automated trading agents able to learn while operating on the stock market with a limited bankroll, a drone, rover, or satellite, that must learn to better accomplish their mission, but paying attention to the charge of their batteries. In fact, several real-world problems as well as many bio-inspired mechanisms involve this kind of survival concerns: robots and software agents must learn to improve their performance while managing a finite amount of resources, avoiding ruin in the same way that organisms in the nature act to escape from death.

⋆ This work is supported by a French government grant managed by the National Research Agency under the France 2030 program with the reference ANR-23-DEGR-0001.
The contribution presented in this paper is an original heuristic to deal with survival RL problems, since, to the best of our knowledge, no specific solutions have been addressed in the literature. The insight is to extend standard algorithms by making the agent operates in two different modes, depending on the remaining budget: a survival mode, which is activated when the budget is too low, making the agent behave conservatively in order to recharge its budget, and a normal mode, to which the agent switches back when a comfortable budget level is reached again. The Survival RL problem is formally defined in Section 2. The background concepts related to RL, and more specifically to the classic Q-Learning algorithm, are presented in Section 3. An overview on the Safe RL state-of-the-art is done in Section 4. The contribution of the paper, the Safety-Threshold (ST) strategy, used to modify the Q-Learning method, giving rise to ST-Q-Learning, is explained in Section 5. Preliminary experimental results are described in Section 6, showing that the use of the proposed heuristic increases the survival expectancy of the agent, even reducing the regret due to the maintain of a persistent but cautiousness exploratory behavior. Section 7 concludes the paper and suggests some future work.

2 Definition: the survival RL problem

The Survival RL problem, as defined in this paper, is illustrated in Figure 1 and can be formally stated as $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, P, R, S_0, B_0, h, \gamma\}$, where:

- $\mathcal{S} = \{s_1, \ldots, s_n\}$ is a finite set representing the unstructured space of $n$ discrete states;
- $\mathcal{A} = \{a_1, \ldots, a_m\}$ is a finite set representing the unstructured space of $m$ discrete actions;
- $P = P(s' | s, a)$ is a stationary distribution defining the probability of observing a transition to the next state $s'$ when action $a$ is taken from state $s$;
- $R = P(r | s, a, s')$ is a stationary distribution defining the probability of drawing a reward $r \in \mathbb{R}$ after a transition from state $s$ to $s'$ with action $a$;
- $S_0 = P(s_0)$ is a probability distribution from which the initial state $s_0$ is drawn;
- $B_0 = P(b_0)$ is a probability distribution from which the initial budget $b_0 \in \mathbb{R}^+$ is drawn;
- $h \in \mathbb{N}^+ \cup \{\infty\}$ is the maximal time-horizon of the process;
- $\gamma \in [0, 1]$ is a constant discount factor to evaluate the long-term utility of the rewards.

The process starts at round $t = 0$, in which the agent observes the initial state $s_0 \sim S_0$ and the initial budget $b_0 \sim B_0$. The process evolves round by round. At each successive round $t$, the agent observes the current state $s_t$ and chooses an action $a_t$ to perform, which provokes a state transition to $s_{t+1}$ and returns a reward $r_{t+1}$. The budget changes according to the received rewards, so as:

$$b_{t+1} = b_t + r_{t+1} \quad (1)$$

where $b_t$ is the budget at round $t$ and $r_{t+1}$ the reward received at round $t + 1$. If the budget is depleted during running, the agent is ruined and the process stops. Otherwise, the process stops at the maximal time-horizon, when $t = h$, and in this case $b_h = b_0 + \sum_{t=1}^{h} r_t$. The agent disposes of an initial policy $\pi_0$ that is improved with the experiences by a learning function $\psi$, producing a new policy $\pi_t$ at each time step, after observing $s_t$, $r_t$, and $b_t$. The evolving process is illustrated in Figure 1. An optimal policy $\pi^*$ maximizes the expected $\gamma$-discounted sum of rewards over a given (potentially infinite) time-horizon $h$, for any initial state $s_0$:

$$\forall s_0 \in \mathcal{S}, \forall \pi \in \Pi \quad : \quad \mathbb{E} \left[ \sum_{t=1}^{h} \gamma^t r_t \mid \pi^* \right] \geq \mathbb{E} \left[ \sum_{t=1}^{h} \gamma^t r_t \mid \pi \right]. \quad (2)$$

In an online context, with no previous separated training phase, the overall performance of a RL algorithm is impacted by trial-and-error actions, increasing a total regret, defined as the difference between the sum of actually perceived rewards, and the expected sum of rewards the agent could have earned if following an optimal policy $\pi^*$ from the beginning.
3 Background: RL and Q-Learning

Without considering safety or survival issues, a RL agent must learn from its observations, exploring the environment and exploiting its knowledge to increase its gain, eventually converging to a policy of actions that maximizes the expected discounted sum of future rewards (an optimal policy). Classically, the RL loop is characterized by an interleaved interaction between an agent that perceives a state and executes an action, and an environment that returns a new state and a reward [1]. The environment can be described as a stochastic Markovian Decision Process (MDP) which evolves discretely over time [2, 3, 4].

When the “model” is known (i.e. when the functions $R$ and $P$ are given), an MDP can be exactly solved with polynomial complexity using linear or dynamic programming (DP) techniques [5]. Reinforcement Learning methods are necessary when a model of the world is not available in advance, which is a very common situation in practice. Model-based RL strategies try to estimate the functions $R$ and $P$ from the observed samples, during interaction, then extracting a policy of actions from them [6, 7], for example, using adaptive dynamic programming [8]. In contrast, model-free RL algorithms have been more effective [9] by trying to learn a policy directly from the experience, generally relying on local approximations of the state-action utilities (Q-values), updated after each observation.

The Bellman’s equation [10] defines the value for a policy $\pi$ from state $s$ as:

$$V_{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s'} [P(s' \mid s, \pi(s)) \cdot V_{\pi}(s')] .$$

The value of a given action $a$ executed from state $s$, followed by policy $\pi$, is:

$$Q_{\pi}(s, a) = R(s, a) + \gamma \sum_{s'} [P(s' \mid s, a) \cdot V_{\pi}(s')] .$$

An optimal policy $\pi^\star$ corresponds to:

$$\forall s \in S : \pi^\star(s) = \arg \max_a \left\{ R(s, a) + \gamma \sum_{s'} [P(s' \mid s, a) \cdot V_{\pi^\star}(s')] \right\} .$$

Q-learning [11] is a traditional model-free algorithm that implements an off-policy immediate temporal-difference (TD) strategy for learning a policy on environments with delayed rewards. The estimated Q function is stored in memory as a table in the form $S \times A \rightarrow \mathbb{R}$, where the entries represent the estimated utility of each state-action pair $(s, a)$ in time $t$. The stored Q-table is updated after each transition through a simple value-iteration step, based on the equation 3, using the $\alpha$-weighted average of the old and the new values, where $s$, $a$, and $s'$ are the observed transition, $r$ is the received reward, and $\alpha$ is the learning rate:

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

Fig. 1: In Survival RL, the agent-environment cycle includes a budget.
Since what can be learned depends on the agent’s behavior, a necessary trade-off must be found between exploration (trying different actions) and exploitation (choosing the action with best expected utility based on the current knowledge). In Q-Learning, the ε-greedy technique is used to induce unidirected exploration by introducing some randomness to the decision-making process [1]. At each time step, the agent either executes a random action with probability ε, or follows the current estimated optimal policy, otherwise, i.e. with probability 1 − ε. To avoid a linear expected regret due to a constant exploration, those algorithms can have their parameters regulated dynamically, gradually decreasing ε, while t evolves [12].

A different approach to solve the exploration-exploitation dilemma is the optimism in the face of uncertainty. The idea is to explore potentially good but insufficiently frequented state-action pairs [13] (directed exploration). In tabular methods, like Q-Learning, it can be done by simply initializing the Q-values optimistically, and then following a completely greedy strategy. The more a state-action pair will be tried, the closer its estimated utility will approach its true value. Most exploration-exploitation trade-off strategies found in the literature lean on the upper-confidence bound principle [14, 15, 16, 17].


The baseline RL methods are not suitable to be applied directly on expensive or safety-critical real-world systems due to the absence of integrity guarantees during the learning process [18, 19, 20]. Particularly in the case of complex physically embodied systems, even if the agent is able to previously learn a policy of actions, either through offline RL using a database of experiences, or through online RL into a simulated environment, in the real scenario the agent will eventually face unexpected events, perhaps disastrous. Successful examples like the drone racing agents [21] are extensively trained in simulation, coupled with physically informed constraints, and fine-tuned with real-world data using off-line RL. For that reason, Safe RL is the object of an increasing number of publications [22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40], attracting the attention of the AI community and giving rise to dedicated workshops in prestigious conferences.

Safe RL is concerned with safety during the learning process, and is synonym of “Risk-Averse” RL. Two distinct approaches appear in the literature do deal with risk [40]: (a) modifying the optimality criterion (the classic expected discounted cumulative reward) with a safety factor, or (b) incorporating external knowledge during the exploration process. In the first approach, the agent takes into account the expected variability on the utility of state-action pairs in order to identify (and avoid) less predictable (thus considered risky) actions. The notion of safety corresponds to reward and transition stability. The risk-reward trade-off is then addressed by mean-variance metrics or value-at-risk metrics, based on the lower quantiles of the estimated utility distribution for each state-action pair [28, 31, 38]. However, no notion of budget is considered in that approach, making it incompatible with the survival RL problem proposed here. The risk is understood as variability or unpredictability, and does not account for a risk of ruin.

In the second Safe RL approach, the idea is to use a standard RL mechanism, but monitoring and eventually interfering in decision making whenever needed in order to ensure safety. For example, a shield module can be introduced within the sensorimotor loop of the agent, endowed with previous knowledge to prevent the RL mechanism from choosing unsafe actions [37]. Another strategy is to gradually improve an initial predefined baseline policy, which is assumed to be stable and safe but suboptimal. The idea is to promote a controlled exploration in which the agent deviates from the baseline behavior smoothly [19, 39, 30]. The baseline policy works by delimiting a safe region, which is gradually extended by the learning mechanism. In a similar approach called Conservative RL, the goal is to perform at least as well as an existing baseline policy during the learning process [42].

In another framework called Budgeted MDP [43, 44, 45], the goal is to maximize the rewards constrained by a total budget, but with separated reward and cost functions. The budget is consumed by the costs, which means that it can only decrease at every round, until eventually being depleted. Solving this problem involves searching for a good reward-cost ratio. The same strategy is not feasible in the survival setting due to the fact that there is a single reward function returning positive and negative values in a process that can possibly run infinitely.

In this paper we are interested in a class of problems that we call Survival RL, which cannot be reduced to any of the previously cited Safe RL approaches, and can be seen as the multi-state
version of Survival Multi-Armed Bandits [46, 47, 48]. A similar problem found in the literature is called Single-Life RL [49], but it is more related to transfer learning techniques, where an agent must adapt to a new scenario based on what was previously learned in other scenarios, for example, by shaping rewards to stimulate staying within the known region of the state space.

In the proposed problem, the agent aims to learn an optimal policy constrained by a budget that can be increased or decreased at each round with the received rewards, and which must remain positive all along the process. Since rewards can be positive and negative, the agent can either increase the probability of running the process indefinitely, becoming infinitely rich, or inversely, can increase the probability of ruin, until eventually getting broke. More generally, the underlying Survival RL problem can be seen as a Constrained MDP [50, 28, 51, 52, 53], where the rewards are to be maximized, subject to maintaining a positive budget along the entire process lifetime. Another possible definition is a multi-objective optimization problem, where the rewards are to be maximized, and the probability of ruin is to be minimized.

5 Contribution: Safety-Threshold Q-Learning

In this section, the Safety-Threshold Q-Learning (STQ) method is introduced (Algorithm 1), as the main contribution of the paper. It is designed to tackle survival RL problems using the original Safety-Threshold heuristic. The insight is the following: when the budget is low, the risk of ruin is high, then the agent should be pragmatic by exploiting its current knowledge to recharge the budget; in contrast, when the budget is high, the risk of ruin is low, then the agent can continue with exploratory actions. STQ distinguishes between a high and a low budget based on two hand-tuned hyperparameters, \(w_0\) and \(w_1\).

In addition, STQ extends the classic Q-Learning by storing a second state-action value table, called K-table, with the same dimensions than the Q-table. Both tables are updated in the same way (Eq. 6) and at same time, after every round. The only difference lies on their initialization: the Q-table is initialized with zeros (neutral), i.e. \(\forall s,a : Q_0(s,a) = 0\), while the K-table is initialized with a positive value \(k_0 > 0\), in order to create an optimistic behavior, so as \(\forall s,a : K_0(s,a) = k_0\), where \(k_0\) is also a hand-tuned hyperparameter. In this way, when the actions are chosen based on the K-table, the principle of optimism in the face of uncertainty holds, and the agent will execute a directed exploration, tending to navigate to promising few observed transitions.

In this way, beyond the exploration rate \((\varepsilon)\), the learning rate \((\alpha)\), and the discount factor \((\gamma)\), STQ must be tuned with 3 additional hyperparameters: the safety threshold \((w_0)\), the exploration threshold \((w_1)\), and the initial k-value \((k_0)\). During running, the agent switches between two behaviors: normal mode and survival mode. When the budget falls under the safety threshold \((b_t < w_0)\), the agent enters in survival mode, becoming greedy (i.e. following the best estimated action for the observed states, and canceling any random exploration), choosing the actions based on the “neutral” Q-table, in order to recharge its budget. In contrast, when the budget overpasses the exploration threshold \((b_t > w_0)\), the agent returns to normal mode, free to conduct exploration due to both the undirected approach resulting from the use of \(\varepsilon\)-greedy action choice, and the directed approach resulting from the use of the “optimistic” K-table for decision making. The insight is that the safety threshold helps to keep the agent far from ruin by making it conservative (greedy) in order to prioritize budget recharging by following the estimated best actions according to the current experience.

To ensure a comfortable budget level for exploration, two thresholds are necessary. Once activated, the survival mode is kept until the budget exceeds a superior exploration threshold, then the agent reverts to its normal mode, at least while it does not fall below the safety threshold again. The use of two separated thresholds, like a thermostat, allows to avoid quick inefficient switches between the two modes. For illustrating it, in the experience shown in the Figure 2, a better exploration rate is obtained, for the same number of steps, when the agent is using separated safety and exploration thresholds, than when it is using a unique threshold. The STQ method is described in Algorithm 1.

In classic RL, an intuition involving the exploration-exploitation dilemma is that the agent should have an initial exploratory tendency that is gradually changed by a greedy behavior at the limit when time goes to infinity, and the error on the estimated utilities is supposed to approach zero. In the survival problem, however, that passage should be based on the budget instead of time [48]. For this reason, when an STQ agent has enough budget to explore, it will base its decisions on
Algorithm 1 - Safety Threshold Q-Learning (STQ)

**Input:** \( b_0 \) {the initial budget}, \( s_0 \) {the initial state}, \( w_s \) {the safety threshold}, \( w_k \) {the exploration threshold}, \( k_0 \) {the initial optimistic mean value}, \( \varepsilon \) {the random exploration rate}, \( \gamma \) {the discount factor}, \( \alpha \) {the learning rate}, \( h \) {the maximum time-horizon}.

\[
t \leftarrow 0, \quad s \leftarrow s_0, \quad b \leftarrow b_0
\]

\[
\forall s, a : \quad Q(s, a) \leftarrow 0, \quad K(s, a) \leftarrow k_0
\]

\[
survival \leftarrow \text{false}
\]

**while** \( b > 0 \) **and** \( t \leq h \) **do**

\[
survival \leftarrow \begin{cases} 
\text{true} & \text{if } b_1 < w_k \\
\text{false} & \text{if } b_1 > w_k
\end{cases}
\]

**if** survival **is true** **then**

\[
a_t \leftarrow \arg\max_s Q(s, a) \quad \# \text{greedy Q action}
\]

**else**

\[
a_t \leftarrow \arg\max_s K(s, a) \quad \# \text{greedy K action}
\]

**either with probability** \( \varepsilon \)

\[
a_t \leftarrow \arg\max_s K(s, a) \quad \# \text{greedy K action}
\]

**or with probability** \( 1 - \varepsilon \)

\[
a_t \leftarrow \text{U}(A) \quad \# \text{random action}
\]

**end if**

\[
\text{execute } a_t
\]

\[
\text{observe } s_{t+1}, r_{t+1}
\]

\[
b_{t+1} \leftarrow b_t + r_{t+1}
\]

**update** \( Q(s, a) \) **and** \( K(s, a) \) **using Equation 6**

**learn from experience**

**end while**

The optimistic K-table. However, if it is in survival mode, it will use the neutral Q-table. As the update formula is the same for both tables, they should both converge to similar values if the agent can dispose of enough experience. Thus, after a sufficiently large number of time steps, whatever the agent’s mode, it will always make decisions following an exploitation logic.

6 Experimental Results on a Grid World

A survival reinforcement learning problem is non-episodic, forcing the agent to use its knowledge and exploit non-optimal rewards to keep its budget strictly positive (i.e. to survive). To evaluate the proposed method in a survival context, a 2D grid problem is defined. The agent is positioned into a corner of a grid map, corresponding to a matrix with size \( X \times Y \), where \( X = 25 \) is the number of columns and \( Y = 5 \) the number of rows in the grid. The reward distribution \( R \) is defined in function of the cost of movement, the influence of reward spots into a set \( \mathcal{R} \), and a variance factor \( \sigma^2 \). A big reward spot \( r_{\text{max}} \) is positioned at the opposite diagonal corner in relation to the agent’s initial position. Two small positive reward spots \( r_{\text{mid}} = +2 \) are placed in the diagonal between the agent and the big reward, around \( 1/3 \) and \( 2/3 \) of the total distance. A spreading factor \( \eta \in [0, 1] \) makes adjacent cells inherit part of those rewards, exponentially decreasing with the distance to the respective spots. That reward function, represented as a relief on the grid, is shown in the Figure 3.

The agent can choose between 4 actions: north, south, east, or west, with deterministic effect of moving the agent to the corresponding direction. But moving is costly, and the cost becomes bigger in the region near to the big reward. In fact, a linear variation is applied in function of the distance to the opposite corner in relation to the agent’s initial position, from \( r_{\text{mid}}(s) = -0.5 \) when coordinate \( x_s = 1 \), to \( r_{\text{mid}}(s) = -1.0 \), when \( x_s = 25 \). If the agent tries to go outside of the map, it remains in the same state, receiving the corresponding reward. Let \( \mathcal{N} \) be a Gaussian distribution with mean \( \mu \) and variance \( \sigma^2 \), and \( d \) be the distance (the number of necessary steps) between the cell represented by state \( s \), at coordinates \( (x_s, y_s) \) and the spot position at coordinates \( (x_r, y_r) \). The
Survival RL with Safety-Threshold Q-Learning

Fig. 2: Example of a typical execution with horizon $h = 2000$, and initial budget $b_0 = 200$. STQ using the same value ($w_q = w_k = 100$) for both safety and exploration thresholds explores the environment less than the one using separate thresholds ($w_q = 100$, $w_k = 200$).

Fig. 3: At the top, the reward map for the survival experience with rectangular map of size $25 \times 5$, $\gamma = 0.95$, one reward spot of +10 and two reward spots of +2, combined with costs varying from $-0.5$ to $-1.0$, from one side to another, and a reward spreading $\eta = 0.2$. At the bottom, the true Q-Values for the designed survival experience, obtained through dynamic programming.

effective reward returned when the agent steps into a cell is given by:

$$R(s) = \mathcal{N}(\mu(s), \sigma^2) \quad \text{where} \quad \mu(s) = r_{\min}(s) + \sum_{r \in R} \eta d(s, r) r \quad \text{and} \quad d(s, r) = |x_s - x_r| + |y_s - y_r| \quad (7)$$

The consequence is that, for the cells of the grid far from the reward spots, any movement executed by the agent will return a negative reward, reducing its budget. A good strategy should be using the small reward spots to recharge the budget, and then be able to explore the environment until reach the big reward, without being ruined before. This “reward desert” makes exploration difficult, and to make things worse, the moving costs are smaller near to the initial position, which can potentially inhibits exploration in the opposite side of the grid, where the big reward is placed. Whereas a conventional agent would take the risk of dying by exploring the environment without taking the budget into account, or, inversely, would accept to survive using a sub-optimal strategy, lying on the small positive rewards, STQ explores the environment while managing its budget, exploiting positively rewarded actions, even when sub-optimal, in order to ensure a sufficiently frequent budget recharging, then continuing exploration to discover an optimal policy. The true Q-values of the problem are presented in the Figure 3 (bottom).

Different scenarios with initial budgets $\{100, 150, 200, 250, 300, 350, 400\}$ have been tested into a $25 \times 5$ grid-world, with discount factor $\gamma = 0.95$. Each simulation ran until a maximal time-horizon $h = 5000$ steps, and was repeated 500 times to allow some statistical analysis. Three different variations of the classical Q-Learning method have been tested: a greedy one, another that explores
based on the optimism in the face of the uncertainty, with Q-tables initialized with value $q_0 = 200$, and $\varepsilon$-greedy with exploration rate fixed to 0.1. Those classical Q-Learning instances have been compared to Safety-Threshold Q-Learning, with Q-table values initialized to 0 and K-table values initialized to $k_0 = 200$. Different combinations of safety and exploration thresholds have been tested: \{100, 200\}, \{200, 400\}, {50, 800}. The parameter $\alpha$ (learning rate) was set to 0.5 for all algorithms. The experimental results have been produced into an Intel core i7 CPU. All the algorithms have been implemented in Python, from scratch.

The obtained results indicate that STQ highly improves the performance of classic Q-learning in terms of exploration (Figure 5). The agent is persistently exploratory when the budget is greater than the exploration threshold, allowing a very good approximation between the learned Q-function and the true Q-function for the entire state-action space. It can be surprising that, in terms of survival rates, the proposed heuristic is not significantly better than classic greedy or $\varepsilon$-greedy (Figure 4). It is because those methods are able to survive using the small rewards, which allows positive rewarded trajectories, even if not-optimal. But STQ presented a better average reward return in the long-term (Figures 6 and 7) thanks to the systematic exploration that allows to discover the big reward earlier. Figures 4 – 7 present the average survival time, average exploration rate, average final budgets, and the average budget evolution, for Greedy ($\varepsilon = 0$), Classic Q-learning ($\varepsilon = 0.1$), Optimistic-Greedy ($\varepsilon = 0, q_0 = 200$), and the combinations of STQ with $\varepsilon = 0.1$ or greedy, thresholds \{100, 200\}, \{200, 400\} and \{50, 800\}, with $k_0 = 200$, $\alpha = 0.5$, and $\gamma = 0.95$.

7 Discussion, Conclusion, and Future Work

Survival RL is still a class of understudied problems, with diverse open questions. In this paper, we propose a first heuristic approach to deal with Survival RL. Safety-Threshold Q-Learning introduces three new meta-parameters: $w_q$, $w_k$, and $k_0$, and allows the agent to change between two behaviors: in survival mode it uses a neutral Q-table, which has been initialized with 0, and follows a greedy policy, trying to exploit the most positive decisions given its current knowledge; in normal mode, the agent follows the policy suggested by a second, optimistic, K-table (“K” for knowledge), which
had been initialized with highly positive values, defined by the parameter $k_0$. The double threshold works like a thermostat: when the budget goes under $w_q$, the agent enters in survival mode and becomes greedy, trying to make the budget increase again using its best current policy, even if not
optimal. Then, when the budget becomes sufficiently high, greater than \( w_k \), it comes back to the normal mode, which is very explorative at the beginning, converging to the optimal policy with high probability when the K-table and Q-table approximate the true Q-values.

Experiments in a grid world show that STQ executes a kind of systematic exploration on the first rounds of the simulation, eventually finding a small reward, to which it returns when necessary to recharge the budget. The random exploration actions used by the classic Q-Learning makes more difficult to go to the other side of the board. At the same time, since Q-Learning is not aware about the risk of ruin, it depletes the budget more often.

Finally, even if most of the classic (tabular) RL methods have proven interesting theoretical guarantees, the need of storing a Q-table limits their practical application to complex real-world problems. The combinatorial nature of an exhaustive enumeration of states would make impossible to maintain a Q-table in memory. In addition, the discrete representation of states and actions can be inappropriate to domains that could be naturally represented by continuous dimensions. In recent years, the rise of Deep Neural Networks (DNNs) provided new possibilities for function approximation and representation learning [54]. Deep RL combines DNNs with classic RL methods, allowing to improve scalability and solve complexity issues faced by tabular methods, and enabling decision making in high-dimensional state and action spaces [55]. The advantage is that, when available, gradients provide a strong learning signal that can be estimated through sampling, offering great generalization capacities. Powerful open-source frameworks are available today allowing to implement and test new Deep RL algorithms easily, like PyTorch, TensorFlow, Keras, and AIDGE.

The next steps of this research involve the extension of the proposed heuristic to other methods, including Deep RL algorithms. The algorithm Deep Q-Networks (DQN) [56, 57], for example, is a deep version of Q-Learning and relies on approximating the state-action values \( Q^* \) using DNNs. The problem is that STQ implemented optimism by initializing a K-table optimistically. This kind of initialization is not possible in DNN, since the parameters of the network impact multiple states at once, and the relation between neurons and states changes during learning. A second improvement can be replacing the fixed thresholds by some adaptive function, making the agent changing between survival and normal mode smarter and smoothly.

Fig. 7: Average budget evolution over time, until horizon \( h = 5000 \). Simulation repeated 100 times in a grid map with dimensions 25 \( \times \) 5 with \( b_0 = 300 \).
References

Tabu Tenure Policy with Deep Reinforcement Learning

Anna Konovalenko, Lars Magnus Hvattum, and Swati Aggarwal
Molde University College, Norway
anna.konovalenko@himolde.no, hvattum@himolde.no, swati.aggarwal@himolde.no

1 Introduction

Tabu Search (TS) is a widely recognized metaheuristic for tackling complex combinatorial optimization problems. The metaheuristic falls under the category of local search algorithms, which explore the solution space by iteratively making small changes to a current solution. A fundamental aspect of the TS algorithm is its utilization of memory to guide the search process. The memory element can take the form of a tabu list, storing forbidden attributes of moves or solutions. The algorithm refers to this list to avoid revisiting previously explored solutions, enabling an efficient exploration of the solution space. A crucial parameter, referred to as the tabu tenure, represents the number of iterations an attribute remains tabu. The parameter impacts the algorithm’s performance and influences the direction of the local search.

Many variants and extensions of the basic tabu search can be found in [1]. Most of these extensions deal with the organization of the neighborhood of the search and how to choose the next solution in the neighborhood. We focus on methods to dynamically adjust the tabu tenure along the search. The objective of this study is to identify attributes that influence the determination of the tabu tenure policy and to establish guidelines for dynamically setting the tabu tenure parameter.

2 Background

The TS method heavily relies on a tabu tenure parameter, and establishing the value for this parameter requires careful consideration. The value can vary based on the types or combinations of attributes, as well as at different stages or intervals of the search process. Different values for the tabu tenure influence the search in different ways: small values for the tabu tenure allow the search to exploit solutions near a local optimum, while large values for the tabu tenure allow the search to explore a larger part of the search space. Varying the tabu tenure maintains a balance between intensification and diversification of the search, allowing for a thorough examination of one region before moving to a different part of the solution space.

There are several ways in which a dynamic tabu tenure can be implemented. For example, Glover and Laguna [1] proposed that the value can be randomly selected within a defined range or systematically chosen from a sequence of values within a range, where the sequence may be repeated until the end of search. Another option involves setting the tenure as a function of the current quality of a solution [2], or as a function of the last move, measuring its impact [3].

In general, there is no globally optimal tabu tenure for addressing all types of problems and instances, but certain guidelines can be established. We aim to formulate them using reinforcement learning (RL), one of the branches of machine learning. RL has demonstrated its potential in engaging with optimization algorithms. For example, in a study by [4] RL was employed to adaptively set parameters for a Genetic Algorithm aimed at solving a Vehicle Routing Problem. We are optimistic about applying RL to our specific settings to determine a dynamic tabu tenure policy and identify the tabu-active attributes influencing its determination.

3 Tabu search

We apply the tabu search algorithm with a dynamic move evaluation to address binary integer programming (BIP) problem. The tabu search for the BIP is outlined in the works of Arntzen et.al. [5], Hvattum et.al [6], and Bentsen et. al [7], where readers can find a detailed description of the algorithm. Our implementation is based on the tabu search version previously developed by Bentsen et. al [7].
The computational study is performed using benchmark instances from an optimization problem, which is a special case of the BIP: the optimum satisfiability problem [8]. Instances for this study are sourced from Bentsen et al. [7].

4 Preliminary Results

In their work, Bentsen et al. [7] presented a tabu tenure policy for a family of BIPs. The tabu tenure is drawn at random from a uniform distribution \( \{T_{MIN}, T_{MIN}+1, \ldots, T_{MAX}\} \), where \( T_{MIN} = 7 \) and \( T_{MAX} = 22 \) in the dynamic tabu search version. The results of this policy for some instances of the optimum satisfiability problem are detailed in the first row of Table 1. The column headers for the first two columns denote the tabu tenure range, while the others specify the family class of the instance along with the instance number in parentheses. The corresponding values represent the mean of the best-found solution across \( n_{episodes} \) iterations. These means are calculated over 400 runs with 5 different random initial solutions, totaling 2000 runs.

The tabu tenure policy in the study by Bentsen et al. [7] was tuned to solve a wide variety of BIPs, of which the optimum satisfiability problem is only one example. To explore the impact of tabu tenure variations, we conducted experiments with lower values for selected instances. Table 1 presents the effects of varying tabu tenure ranges. While the values in the second and third rows closely align due to minimal changes in the tabu tenure range, the values observed in the first and fourth row provide evidence that variations in tabu tenure ranges have an impact on the results. Both excessively small and excessively large tabu tenures can lead to a decrease in solution quality, emphasizing the delicate balance required for effective algorithm performance.

<table>
<thead>
<tr>
<th>( T_{MIN} )</th>
<th>( T_{MAX} )</th>
<th>Class 25 (0)</th>
<th>Class 25 (1)</th>
<th>Class 27 (0)</th>
<th>Class 27 (1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>22</td>
<td>22468</td>
<td>20747</td>
<td>20331</td>
<td>20407</td>
</tr>
<tr>
<td>1</td>
<td>14</td>
<td>22506</td>
<td>20784</td>
<td>20366</td>
<td>20447</td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>22502</td>
<td>20783</td>
<td>20361</td>
<td>20444</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>22451</td>
<td>20738</td>
<td>20317</td>
<td>20395</td>
</tr>
</tbody>
</table>

Løkketangen and Olson [3] demonstrate the advantages of incorporating individual instance search-attributes in deciding tabu tenure policies. Their work suggests that integrating more sophisticated heuristics to decide the tabu tenure policy can provide even better performance.

Building on this insight, various attributes of the search process can influence the determination of the tabu tenure policy. For example, in TS, the type of local search move—whether it is classified as improving or non-improving—can be used to influence the tabu tenure policy. Table 2 presents results where the tabu tenure ranges depend on the type of move (improving moves vs. non-improving moves). The results indicate that for improvement moves, the tabu tenure should be larger than for non-improvement moves.

The findings serve as motivation for the development of a advanced policy for dynamic tabu tenure using RL.

<table>
<thead>
<tr>
<th>Impr.</th>
<th>Non-Impr.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_{MIN} )</td>
<td>( T_{MAX} )</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>15</td>
</tr>
</tbody>
</table>
5 RL-Based Policy Formulation

For using RL model, the problem is formulated as a Markov Decision Process (MDP):

**Decision Epoch and Episode:** In our initial approach, we plan to decide the tabu tenure for each iteration of the tabu search. The algorithm starts with an initial random solution. To avoid influencing our adaptive policy by the initial random solution, for the first $n_{init}$ iterations we use the policy of Bentsen et al. [7]. Consequently, the episode will start at the point in the search where some variables are already in tabu. Given that the Tabu Search algorithm is a local search that can run millions of iterations to improve the solutions, it can comprise a lengthy episode for the RL algorithm. Therefore, each episode will have a limited number of $n_{episode}$ iterations.

**State:** The state can consist of any tabu-active attributes of the search. In the previous section we discussed potential factors from current literature that may influence the policy of determining tabu tenure. These factors can include the current quality of a solution, measuring the impact of the last move, or considering the history of the search. The objective is to identify the attributes that will impact the policy and quantify the extent of their influence.

**Action:** The action involves selecting a tabu tenure value for the current iteration, either as a discrete value from $[T^{MIN}, T^{MIN} + 1, ..., T^{MAX}]$ or determining the direction of change (increase/decrease) for the tabu tenure value.

**Reward:** The reward is given at the end of the episode and reflect the quality of solution have been found during the search.

6 Conclusion

Preliminary results indicate the potential for a more advanced tabu tenure policy. Various factors can influence the establishment of a better policy, and RL can serve as a valuable tool to acquire this information. While it is early in the experiments to make any claims, we are optimistic about gaining insights and shaping our dynamic tabu tenure policy in the future.

References

NeuroLGP-SM: A Surrogate-assisted Neuroevolution Approach using Linear Genetic Programming

Fergal Stapleton¹[0000−0002−5347−1573], Brendan Cody-Kenny² and Edgar Galván¹[0000−0001−8474−5234]

¹ Naturally Inspired Comp. Res. Group, CS Department, Hamilton Institute, Maynooth University, IE fergal.stapleton.2020@mumail.ie, edgar.galvan@mu.ie
² Sema, Baltimore, MD, USA brendan@semasoftware.com

Abstract. Evolutionary algorithms are increasingly recognised as a viable computational approach for the automated optimisation of deep neural networks (DNNs) within artificial intelligence. This method extends to the training of DNNs, an approach known as neuroevolution. However, neuroevolution is an inherently resource-intensive process, with certain studies reporting the consumption of thousands of GPU days for refining and training a single DNN network. To address the computational challenges associated with neuroevolution while still attaining good DNN accuracy, surrogate models emerge as a pragmatic solution. Despite their potential, the integration of surrogate models into neuroevolution is still in its early stages, hindered by factors such as the effective use of high-dimensional data and the representation employed in neuroevolution. In this context, we address these challenges by employing a suitable representation based on Linear Genetic Programming, denoted as NeuroLGP, and leveraging Kriging Partial Least Squares. The amalgamation of these two techniques culminates in our proposed methodology known as the NeuroLGP-Surrogate Model (NeuroLGP-SM). For comparison purposes, we also code and use a baseline approach incorporating a repair mechanism, a common practice in neuroevolution. Notably, the baseline approach surpasses the renowned VGG-16 model in accuracy. Given the computational intensity inherent in DNN operations, a singular run is typically the norm. To evaluate the efficacy of our proposed approach, we conducted 96 independent runs spanning a duration of 4 weeks. Significantly, our methodologies consistently outperform the baseline, with the SM model demonstrating superior accuracy or comparable results to the NeuroLGP approach. Noteworthy is the additional advantage that the SM approach exhibits a 25% reduction in computational requirements, further emphasising its efficiency for neuroevolution.

Keywords: Neuroevolution, Linear Genetic Programming, Surrogate-assisted Evolutionary Algorithms

1 Introduction

In the preceding decades, Evolutionary Computation (EC) [10] has attracted considerable attention in the burgeoning field of Neural Architecture Search (NAS) [12]. The combination of these two fields, commonly referred to as neuroevolution, is centred around automatically finding architectures of artificial neural networks (ANNs) by evolving network topologies, hyperparameters and/or weights. Typically, the architectures are evolved based on accuracy, but often consider other design characteristics, such as network latency [9]. Neuroevolution of deep neural networks (DNNs) [21], has seen growing interest in recent years as discussed in a recent survey [12], reviewing over 170 works, and has applications to state-of-the-art and emerging technologies such as in the field of autonomous vehicles [14, 28].

When considering DNNs, in general, whether using random search [2], neuroevolution [12] or other approaches, there is a high computational overhead required in order to effectively search
for well-performing architectures. To tackle this, research has turned to the use of surrogate-assisted evolutionary algorithms (SAEAs) [18]. SAEAs can be used to estimate the fitness of expensive DNNs without requiring them to be fully trained. To build an effective modelling strategy for fitness estimation, we need to impute the accuracy of untrained or partially trained networks based on knowledge from previously fully-trained neural networks. However, comparisons between models often require ingenious encoding strategies [30] to compare genotypes as generally, it is not possible to create distance metrics for differing network topologies [29]. A more natural approach is to consider the behaviour of the DNN architectures rather than relying on genotypic information [17, 29], however, to date, SAEAs based around phenotypic distance metrics have focused primarily on ANNs of only one to two layers of fully-connected layers [29]. Another limitation of this previous work is that Kriging, which is an interpolation technique widely used in SAEAs and is well suited for optimization problems [3], suffers computational overhead when using high-dimensional data which puts further restrictions on the size of phenotypic distance vectors that can be used.

The primary objective of this paper is to provide insight into the effective integration of surrogate models into neuroevolution, addressing a significant challenge posed by the commonly encountered high-dimensional data. The key contributions of this work are outlined as follows:

- Firstly, we implement and validate a baseline model employing a repair mechanism, a strategy frequently utilised in neuroevolutionary methods. This model surpasses the well-established VGG-16 model, setting a high-performance benchmark for comparison with our proposed approaches.
- Secondly, we introduce an innovative representation based on Linear Genetic Programming [5], termed NeuroLGP, facilitating the automatic discovery of well-performing DNNs that outperform the baseline model. The NeuroLGP approach is employed to compute the fitness of the entire population and serves as an excellent method to compare against a surrogate model (SM).
- Thirdly, to address the challenge of high-dimensional data and enable the use of a SM to reduce computational demands in neuroevolution, we employ Kriging Partial Least Squares [4]. The fusion of this technique with NeuroLGP results in our second proposed approach, referred to as NeuroLGP-SM. This approach consistently identifies DNNs that perform similarly to those discovered by NeuroLGP, simultaneously reducing fitness evaluations and training time required for these DNNs. In our previous work [27], we identified that it was not possible to use the original Kriging approach to this end.
- Fourthly, we demonstrate the effectiveness of our approach through a robust comparison involving 8 independent runs for each of the 2 aforementioned NeuroLGP approaches plus the baseline model, using 4 challenging image classification datasets. This extensive evaluation, deviates from the norm of a single run in the context of DNNs.
- Our fifth key contribution lies in the innovative management of the SM. This approach remains invariant to varying network topologies and robust to data augmentation techniques. Consequently, we can train our networks with a significantly reduced number of instances while maintaining the ability to generalise effectively to unseen data.

Section 2 details related work, Section 3 discusses background. Section 4 describes the NeuroLGP method along with the surrogate model in detail, Section 5 details the experimental setup, Section 6 offers analysis of our results and Section 7 offers concluding remarks.
2 Related Work

2.1 Neuroevolution using Surrogate Models

An old, but still highly relevant survey on surrogate models by Jin [18], covers some of the fundamental aspects of surrogate-assisted EAs, such as surrogate model management strategies and acquisition functions. A more recent survey by Khaldi and Draa [20], covers more state-of-the-art concepts, such as considering the computational complexity of a surrogate model when performing neuroevolution. Next, we will cover some of the most relevant works.

Performance prediction can be categorized into two branches: (i) approaches that infer the performance of unseen networks based on specific characteristics of previously evaluated networks and which typically have been fully trained, and (ii) approaches that instead used partially trained information to predict the future performance of a network.

A major work in recent years of the former approach is the End-to-End Performance Predictor (E2EPP) [30] proposed by Sun et al. This approach uses an offline surrogate model based on random forests to search for optimal Convolutional Neural Networks (CNN) architectures. The CNN is cleverly encoded such that it maps to a numerical decision variable, which can then be processed by a random forest surrogate. This approach alleviates restrictions found in other approaches such as assuming a smooth learning curve and not requiring large amounts of training. The authors approach was shown to speed up fitness evaluations while also achieving the best performance compared to other peer performance predictors.

Approaches falling into the second category include the Freeze-Thaw Bayesian Optimization (FBO) technique proposed by Swersky et al. [32]. It uses Bayesian optimization to decide if a partially trained network should be trained to completion. The fundamental idea is that a human expert is quickly able to assess if a network is likely to result in poor performance and as such can decide to halt training. Based on this notion the authors designed an approach that at its core is a form of performance prediction, where Bayesian Optimization is used to determine whether a particular partially trained network will yield preferable results compared to other networks. Unlike the E2EPP this approach does require a smooth learning curve which can be hampered if different learning rates are considered. Of interest, is that the FBO approach relies on the phenotypic behaviour of the network in order to decide which networks to evaluate.

Gaier et al. [11] devised a kernel-based surrogate model to use with NEAT [25] referred to as Surrogate-assisted Neat (SA-NEAT). To overcome the limitation of variable length genotypes, they exploited a distance metric inherent in the NEAT algorithm. This distance metric which is known as the ‘compatibility distance’ was originally designed to help promote diversity amongst the network topologies through speciation. Within this work they also use it as a distance metric to use for Gaussian Processes. They demonstrated that they were able to achieve similar performance to NEAT with much fewer evaluations.

Greenwood and McDonnell [16] proposed a grammar-based approach which generates a tensor representation of variable length DNN topologies. An advantage of using formal grammar as a representation is that they can be designed to ensure validity of models by describing the space of allowable topologies. Their approach modifies the DeepNeat algorithm first proposed by Miikkulainen [22]. The modelling strategy of this work is designed around a two-phase approach. Firstly, during the initialization phase, the DeepNeat algorithm is used to evolve the population of neural networks and these models are used to formulate the surrogate. Secondly, the active learning phase is implemented where new networks are evaluated using the surrogate model, along with a subset of networks that are fully trained and evaluated to further inform and improve the surrogate model. They noted a five times improvement in compute time.
3 Background

3.1 Surrogate assisted models: Kriging and Kriging Partial Least Squares

The aim of a surrogate model, also referred to as a meta-model, is to sufficiently approximate an estimate of the fitness values of a potential solution, while reducing the run time of the evolutionary process, compared to the run time of using the real fitness function alone. This requires that the surrogate model is well-posed and requires a robust model management strategy, otherwise, the evolutionary algorithm may converge to a false optimum [18].

Kriging allows for interpolation by assuming spatial correlation exists between known data points. The process itself is informed by the distance and variation between these data points. More explicitly, a kernel function \( K(\cdot) \) denotes the spatial correlation between two samples \( x_i \) and \( x_j \) as shown in Equation 1,

\[
K(x, x') = \prod_{i=1}^{m} \exp \left( -\theta_i (x_i - x_i')^2 \right)
\]

where the \( \theta \) parameter determines the rate at which the correlation decays to zero and \( m \) is the dimension. The \( \theta \) parameter is determined by the Maximum Likelihood Estimator (MLE). One drawback, however, is that for large \( m \), the cost increases significantly since the MLE algorithm requires calculating the inverse of the correlation matrix many times.

Kriging Partial Least Squares (KPLS) seeks to address this by effectively reducing the number of parameters calculated [4]. It does so using partial least squares which project the high-dimensional data into a lower dimension using principal components. Equation 2 details the KPLS kernel,

\[
K(x, x') = \prod_{k=1}^{h} \prod_{i=1}^{m} \exp \left( -\theta_k (w_{ik}(x_i - x_{i}')^2 \right)
\]

where \( w_{ik} \) are rotated principal directions which maximize the covariance and are a measure of how important each principal component is. The number of principal components \( h << m \) which allows for the substantial improvement in computational cost associated with the KPLS approach. For full details of the method, please refer to [4].

3.2 Phenotypic Distance

The rationale behind employing phenotypic distance draws inspiration not only from Stork et al.’s work [29] but also from our research in Genetic Programming (GP) [13, 15], focusing on semantics. Specifically, our contributions in the realm of semantic distance metrics include advancements in underexplored domains, such as multi-objective optimisation [13, 15, 26]. In traditional GP, semantics refer to the behaviour of a program given a finite set of inputs.

In the context of neuroevolution, we can define a solution sample \( x \) as having the semantic or phenotypic behaviour of the \( i^{th} \) program such that \( x_i = s(p_i) \), where the semantics \( s(p) \) of a program \( p \) is the vector of values from the output. From Equation 1 we can then define our phenotypic distance \( d \) as

\[
d(x_i, x_j) = d(s(p_i), s(p_j))
\]

As such the distance metric is dependent on the outputs of each network. In this work, we use the convention established by Stork et al. [29], where the \( x_i \) is a flattened vector containing the output of the nodes at the final layer for all data instances. As such, our phenotypic distance
vector length is given as the number of images of the validation dataset times the number of classes. It is important to note that this approach can be extended to any deep learning model architecture that can have its output represented in vectorised form, such as transformers, however, further research would be required to determine the limitations and scalability of applying this approach to other deep learning approaches.

4 Methodology

4.1 NeuroLGP

With the original LGP paradigm [5], representation is inspired by the von Neumann architecture of modern CPUs which use registers for storing or modifying small amounts of data. The content of each register in this architecture can be changed using instruction operations. An instruction, in this context, has three main components: an operand, registers for which the operand operates on (one register for 2-register instruction and two registers for 3-register instruction) and the destination register which stores the computed results. In the context of LGP, genetic operations work by either modifying these registers or the instructions that operate on them. While the original approach of LGP uses registers to store small amounts of data or instructions, it is possible to abstract this form so that the registers are themselves pointers to much larger amounts of data stored in memory. Our proposed approach does as such, using these pointers to instead control the flow of the initial and intermediary data from each outputted layer of our evolvable DNN. The left of Figure 1, demonstrates psuedocode how the expected representation would look, where \( r[i] \) denotes the \( i^{th} \) register. Each line of code is executed imperatively. This example contains both effective and non-effective lines of code. The non-effective lines of code are commented out and, as such, are not compiled. The register \( r[0] \) is a specially designated register for the final output of the program. To the best of our knowledge, LGP has not yet been used for neuroevolution.

```python
def neuroLGP (...)
{
   r[0] := Conv(r[1])
   // r[4] := BatchNorm(r[3])
   r[5] := MaxPool(r[0])
   r[11] := BatchNorm(r[5])
   r[0] := Dense(r[11])
   ...}
```

Fig. 1: left: NeuroLGP psuedocode for python. right: Functional API example in TensorFlow.

To understand how this representation can be useful for the case of neuroevolution, some example code in TensorFlow is given to demonstrate the imperative nature of defining models as seen in the right of Figure 1. This code demonstrates an example of a model definition using the functional API from TensorFlow. On Lines 3, 5 and 6, to the left of the assignment, a variable \( x \) will hold the outputs of each statement and for Lines 5 – 6, to the right of the assignment, \( x \) is passed to each layer. As such, the variable \( x \) holds transient data which updates as each line is executed. The aim is to replace \( x \) using abstract or virtual registers (i.e \( r[0] \), \( r[1] \), \( r[2] \) ...etc), where the \( x \) variables to the left of the assignment are represented using a destination register and values to the right represent registers to be operated on. Line 4 is not executed.
4.2 NeuroLGP with Surrogate Model (NeuroLGP-SM)

To determine which individuals should be fully evaluated, we use the expected improvement (EI) criteria [6, 19]. EI is an acquisition function where the next candidate solution to be evaluated is based on the expected improvement over the current best solution so far. As such, EI allows us to evaluate solutions from regions of the search space likely to see the best improvement. To estimate the fitness using the surrogate model, first, we evaluate the fitness based on a limited number of epochs. From here, we split our population based on the EI criterion, whose parameters are informed by the surrogate model. Based on the EI criterion we fully evaluate a subset of the population, leaving the remaining proportion of the population to be evaluated using the surrogate model. Using the fully evaluated fitness we can now update the surrogate training data and re-train our surrogate model. In this sense, the use of the EI criterion is two-fold. Firstly, it allows us to evaluate individuals we expect to offer the greatest improvement, ensuring we expend our resources on the most promising networks. Secondly, it allows us to select iteratively, informing our surrogate model with better and better solutions. Figure 2 demonstrates our framework. On the left, we can see the workflows of the evolutionary process and the surrogate model. On the right, we can see in more detail how the expected EI criterion is used within our surrogate model management strategy.

Fig. 2: Left: Diagram showing the interplay between a typical evolutionary algorithm and a surrogate model approach. Right: The surrogate model management strategy is shown on a more granular level.

4.3 Genetic Operations and Repair Mechanism

The mutation operator we use in this work mutates either a single input or output register or the operand. The mutation operator works on both effective code and non-effective code. Additionally, we make use of a novel effective crossover operator. This operator selects two crossover points from the effective sections of the parent code and then transfers segments to create offspring. While the ends of each segment contain effective code the code within may be non-effective. A further point mutation repair is applied at either segment end to ensure input
and output registers match after crossover has been applied. Ultimately, this results in always producing a valid network.

A repair mechanism is incorporated when performing the genotype-to-phenotype mapping to ensure models are compilable. There are several conditions for incorporating a repair mechanism, but it is important to note that in each case the repair is only performed on the effective code, by either inserting or deleting a specific layer.

- When there is no effective code in the genotype, we perform a single effective mutation inserting a single convolutional layer from our feature list.
- It is possible for the program to compile without a convolutional layer at the start (i.e., Max pooling, dropout, etc). We add an effective insertion of a convolutional layer before any layer that does not match this criteria.
- When the input data dimensions are too low or if the number of data-reducing operations exceeds the dimensions of the data, the neural network will fail to compile, we remove layers iteratively until the model is valid.
- If the input data dimensions are too high or if the number of data-reducing operations are too few, our fully-connected layer can be composed of an arbitrarily large number of nodes. Furthermore, adding further fully-connected layers can cause the models to become prohibitively expensive to evaluate, as such, we put additional conditions in to check and add a fully-connected layer before the final output layer. The size of the fully connected layer is dependent on the number of parameters from the CNN portion of the architecture.

5 Experimental Setup

The Breast Cancer Histopathological Image Classification (BreakHis) [24] dataset is a binary classification dataset consisting of microscopic images containing 2,480 benign and 5,429 malignant tumours, split across four types of magnification (40x, 100x, 200x and 400x). Each image consists of 700X460 pixels and 3-channel RGB with 8-bit depth in each channel. These images are re-scaled to a lower resolution of 64x64 pixels with 3-channels for each experiment. The overall data split for training, validation, test and test-2 is approximately (63.5%, 12.5%, 12.5%, 12.5%) or at the training level for each network, when only one test set is considered (70%, 15%, 15%). The first test set is used to evaluate the accuracy of each network on unseen data and is used to generate a fitness function for the NeuroLGP approach. The second data set is used to evaluate the NeuroLGP approach after evolution and as such is unseen to both the neural network and evolutionary processes. Since the dataset is imbalanced, synthetic minority oversampling technique (SMOTE) [8] is used to up-sample the minority class. Though other works sometimes use patient-level information to determine the accuracy, in our work we use image level accuracy (ILA), which is the standard classification accuracy at the image level [1].

We performed a substantial number of independent runs, 96 in all, totalling approximately 80 GPU days (3 approaches x 4 datasets x 8 runs). A generation size of 15 and a population size of 50 were selected for all 3 approaches. Experiments were conducted using Kay supercomputer provided by the Irish Centre for High-End Computing (ICHEC). Runs were run in parallel, with each individual run assigned to a single Nvidia Tesla V100 GPU with 16GB Ram. A set of three experiments were designed to test three approaches:

1. A baseline approach which is a random search approach that initializes the network architecture layers randomly. These networks are not evolved but are repaired in line with the repair mechanism as discussed in Section 4.3. All networks are fully trained to the full number of epochs (30 epochs),
2. The expensive approach where we evolve the structure of our networks using the NeuroLGP approach for the max number of epochs (Pop. size = 50, Gen. size =15).
3. The surrogate approach where we employ the NeuroLGP-SM. The surrogate model is informed using partially trained networks (10 epochs) for 60% of the population size.

6 Results

6.1 Preliminary Analysis of the Baseline Model
The baseline in our approach makes use of random search over the feature extraction portion of our network architecture with a set of repair operations to ensure the architecture can be compiled. Typical network sizes from this approach ranged from $\sim$20K to $\sim$12.5M parameters with the baseline. For example of VGG-16 has $\sim$40M parameters [23]. Figure 3 shows the distribution of accuracy values for the baseline, for magnification x200, for all networks across 8 runs (750 individuals per run, 6000 individuals overall). It is important to point out that across all 8 runs, every model found by the baseline was valid and compilable. Slight peaks at 0.33 and 0.66 represent poor performing networks that classify all the data as a single class. We can see that despite a tailed distribution to the left, the vast majority of individuals are centred around the median peak of 0.83. Additionally, the performance of VGG-16 is also shown in the figure with an accuracy of 0.87, which was verified in our own experiments and documented in the work by Cascianelli et al. [7]. The main takeaway is the baseline approach can find competitive architectures to compare against the surrogate and expensive approaches.

![Figure 3: Accuracy of 6000 individuals (read text).](image)

6.2 Comparison of Models
A recent survey by Benhammou on the BreakHis dataset lists a comprehensive comparison of various machine learning approaches in terms of accuracy [1]. The results demonstrated here match the state-of-the-art approaches in terms of performance and are particularly notable as the approach we use does not make use of transfer learning techniques.

![Validation accuracy (0.0 to 1.0) vs. Normalized density](image)
A comparison is conducted between the three experiments as outlined in Section 5 for the BreakHis dataset. The violin plots in Figures 4a - 4d plot the accuracies of the best individual in terms of fitness, for each run, for the x40, x100, x200 and x400 magnifications, respectively. For reference, if we were to look at baseline as an example (left hand side in each plot of Figure 4), then the individuals shown in these plots represent individuals found at the far right of the density plot in Figure 3. Initially, we ran the three models for 4 runs only, which took approximately 40 GPU days in total. It should be noted that in neuroevolution the norm is usually to do a single run. The initial results showed a tendency for the surrogate and expensive models to outperform the baseline model for the x40, x100 and x400 models however for x200 the surrogate and expensive underperformed. It was decided to extend this, to 8 runs totalling 80 GPU days in all. The updated results are represented here in the violin plots, where again we can see that in the case of x40 and x100 magnification, the surrogate (green) and expensive (blue) models tend to outperform the baseline (red) model. Results are more mixed when we look at x200, but is markedly improved over the initial set of 4 runs. The x400 magnification appears somewhat mixed, however, with more runs x400 may have more separation for surrogate and expensive models over the baseline given there are few better-performing individuals above the 92% mark. Comparing surrogate and expensive models in particular, an important observation can be made, in that for all magnifications they have similar accuracies. Given the distribution of the baseline is not fully separable from the other methods, compounded by the low run number, it was decided analysis of the best individuals across all runs would be more informative rather than relying purely on statistical testing.

The accuracies for the best-performing individuals across all runs for the baseline, surrogate and expensive models are listed respectively as such; for the x40 magnification the accuracies are 0.889, 0.913 and 0.930; for the x100 magnification are 0.869, 0.903 and 0.916; for the x200 magnification are 0.946, 0.970 and 0.960 and finally for the x400 magnification are 0.914, 0.925 and 0.925. In no instance did the baseline produce the best-performing individual. For the x40 and x100 magnifications, the expensive model produced the best-performing individual. For the x200 magnification the surrogate model produced the best-performing individual and the x400 magnification has a tie for the best-performing individual. In summary, while on average our surrogate and expensive models are as good as or better than the baseline, when we consider the best-performing individuals across all runs the surrogate and expensive models are better.

6.3 Analysis of the Surrogate Model

We use three metrics to determine how well our surrogate model performs in terms of predicting the fitness of our partially trained models. Firstly, we use the Mean Squared Error (MSE) between the predicted fitness and actual fitness. Values closer to 0 indicate our surrogate model is accurate in predicting fitness. Secondly, Kendall’s Tau is used to measure the correlation between the predicted fitness and the actual fitness [31]. A coefficient value of -1 indicates a perfect negative correlation, while a coefficient of +1 indicates a perfect positive correlation. Values close to 0 would indicate no discernible correlation. When considering all runs, a mean value close to 0 would indicate our surrogate model is performing poorly while a mean value closer to 1 would indicate an ideal-performing surrogate model for Kendall’s Tau. Thirdly, the $R^2$ score [3], is used to measure how close the predicted value is to its true value and ranges from -$\infty$ to 1, and explains how much variability is in the prediction model. $R^2$ scores close to 1 would represent a model which perfectly fits the data.

Table 1 summarizes the effectiveness of the surrogate model for each of the datasets, as seen in the ‘Quality of fit NeuroLGP-SM’ header of this table. The low MSE values (second column), show a relatively low error between the predicted and actual fitness, however, the MSE value for x40 magnification was comparatively high at 0.0037, compared to x100, x200 and x400 at
Fig. 4: Figures 4a through 4d plot violin plots, showing the distribution of accuracies of the architectures with the best fitness for each of the four data sets across 8 runs.

magnifications 0.0017, 0.0014 and 0.0009 respectively. Furthermore, Kendall’s Tau values range from 0.5647 to 0.6791 across the four datasets indicating a strong positive correlation between the actual and predicted fitness. We can see that for x40 we have $R^2$ of 0.5026 indicating a moderate level of fit being captured by the surrogate model while x100, x200 and x400 we have $R^2$ values of 0.6665, 0.7079 and 0.7786 indicating a strong level of fit. A deeper dive into why x40 had a comparatively lower performance revealed that for a particular run, the surrogate model estimated poorly on the lower fitness models but in general was better for higher fitness models and was better overall for the other seven runs.

The ‘GPU hours per run’ header of Table 1 shows a comparison between the expensive and surrogate model in average runtime per GPU hour, in the fifth and sixth columns respectively. Of note is that the surrogate model typically is an order of magnitude higher for standard deviation, meaning there is greater variance associated with runtime for the surrogate model approach. The last column shows the percentage in terms of time saved using the surrogate model over the expensive model. In general, we can see that given the parameters selected, we roughly save 25% in terms of GPU hours. Overall, if we consider the total time it took to run 8 runs for all 4 datasets, the expensive models took ~28 GPU days and the surrogate models ~21 GPU days, saving approximately ~7 GPU days in all.
Table 1: Average quality of fit and number of GPU hours per run.

<table>
<thead>
<tr>
<th>Mag.</th>
<th>MSE</th>
<th>Kendall's Tau</th>
<th>( R^2 )</th>
<th>Expensive (hrs) Mean</th>
<th>Expensive (hrs) Std</th>
<th>Surrogate (hrs) Mean</th>
<th>Surrogate (hrs) Std</th>
<th>Reduction Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>x40</td>
<td>0.0037</td>
<td>0.6019</td>
<td>0.6026</td>
<td>21.3 ( \pm 0.2 )</td>
<td>15.9 ( \pm 0.7 )</td>
<td>25.3%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x100</td>
<td>0.0017</td>
<td>0.6791</td>
<td>0.6665</td>
<td>22.7 ( \pm 0.1 )</td>
<td>16.6 ( \pm 0.7 )</td>
<td>26.7%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x200</td>
<td>0.0014</td>
<td>0.6225</td>
<td>0.7079</td>
<td>22.4 ( \pm 0.1 )</td>
<td>16.8 ( \pm 0.8 )</td>
<td>24.9%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x400</td>
<td>0.0009</td>
<td>0.5647</td>
<td>0.7786</td>
<td>20.0 ( \pm 0.1 )</td>
<td>15.3 ( \pm 0.6 )</td>
<td>23.8%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7 Conclusion and Future Work

In our work, we adapt the use of phenotypic distance vectors for surrogate-assisted neuroevolution of Deep Neural Networks (DNNs) using a variation of Kriging, entitled Kriging Partial Least Squares (KPLS). While phenotypic distance vectors have previously been used for traditional Artificial Neural Networks (ANNs) of only a few layers, their use in surrogate modelling for DNNs marks a significant leap forward in tackling high-dimensionality in neuroevolution. Our surrogate model management strategy makes use of a novel neuroevolutionary approach inspired by Linear Genetic Programming, entitled NeuroLGP which allows us to evolve compact, robust and variable-length architectures. This new approach was shown to outperform a competitive baseline for both the expensive (NeuroLGP) and surrogate-based variant (NeuroLGP-SM) using four magnification subsets of the BreakHis dataset when considering best-individuals across all runs. While we conducted an extensive set of experiments totalling 80 GPU days, in future, we would like to extend the number of runs from 8 to 16 to conduct a full statistical analysis. Furthermore, the runtime associated with NeuroLGP-SM was on average 25% faster compared to the expensive variant.

Acknowledgements

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant number 18/CRT/6049. For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

References


A Multi-Objective Metaheuristic Applied to the Airborne Wind Energy Farm Layout Problem

Rui Carvalho Da Costa∗1, Luís A.c. Roque2, Luís Tiago Paiva1, Manuel C.r.m. Fernandes1, Dalila B.m.m. Fontes3, and Fernando A.c.c. Fontes1

1SYSTEC–ISR ARISE, Universidade do Porto – Portugal
2SYSTEC–ISR ARISE, DMA/TID–ISEP, Politécnico do Porto – Portugal
3LIADD–INESC-TEC, Universidade do Porto – Portugal

Abstract

The power production enhancement and layout design of Airborne Wind Energy (AWE) farms for the dual objective of maximising power output and minimising the number of producing units (kites) are addressed. We employ a multi-objective optimization strategy combining two genetic algorithm metaheuristics: the Non-Dominated Sorting Genetic Algorithm-II (NSGA-II) and the Biased Random Key Genetic Algorithm (BRKGA). The approach blends elements from both algorithms, incorporating key features such as random keys, Pareto set determination for the selection of an elite set, a biased crossover strategy and immigration. (...

Please find the full extended abstract attached or in:

https://uporto-my.sharepoint.com/:b:/g/personal/up201806449_up_pt/EUn0xt1vKDBAilqZzp6pk0ABpM6357LWwZKhucCn55JNng?e=kU4fPD

∗Speaker
How is the objective function of the Feature Selection problem formulated?

First Author\textsuperscript{1}[0000–1111–2222–3333], Second Author\textsuperscript{2,3}[1111–2222–3333–4444], and Third Author\textsuperscript{3}[2222–3333–4444–5555]

\textsuperscript{1} Named University, Address NA 012344, Country

\textsuperscript{2} Named University, Address NA 012344, Country lnsc@springer.com

http://www.springer.com/gp/computer-science/lncs

\textsuperscript{3} Named University, Address NA 012344, Country {abc,lncs}@email.com

Abstract. This paper presents a comprehensive analysis of objective functions used in feature selection, a critical aspect of machine learning. We conducted a systematic literature review, categorizing objective functions into single-objective and multi-objective, with further classification into pure and weighted multi-objective functions. Our study spans research from 2019 to 2023, analyzing a total of 161 articles. We found that weighted multi-objective functions are most prevalent, highlighting their efficacy in balancing model performance and complexity. This work offers a detailed classification of these functions, contributing to a deeper understanding of their role and effectiveness in feature selection challenges. Our findings illuminate trends and preferences in objective function usage, providing valuable insights for researchers and practitioners in the field of machine learning.

Keywords: Future selection · Objective Functions · Literature Review · Multi-Objective Optimization.

1 Introduction

The rise in artificial intelligence, especially in machine learning, is attributed to enhanced computational power and the abundance of data. However, this data richness can lead to redundancy and irrelevance, potentially causing learning errors [32]. Feature selection is pivotal in managing this, aiming to identify a subset of features from large datasets that represent the original data effectively and eliminate unnecessary information, thereby enhancing classification algorithm performance [11]. This problem is complex since the search space is defined as \(2^n\) where \(n\) corresponds to the number of features that make up the data set [2].

Given the extensive number of investigations on the topic, many methods for solving the problem have emerged. Following the extensive related literature [2], the solution methods for feature selection problem can be classified as shown below:
Filter methods: These methods select features based on statistical analysis of the data set, independent of the classification algorithm. They are known for their speed and efficiency. Examples include the Correlation Coefficient and Chi-Squared Test.

Wrapper methods: These iteratively refine feature sets during the model’s training phase, optimizing for the model’s needs. Known for enhancing performance, common approaches include forward selection and metaheuristics.

Embedded methods: Integrated within learning algorithms, these methods aim for fast and accurate feature selection. Examples are lasso regression and decision trees.

Wrapper Methods are computationally more expensive than Filter Methods; however, the former delivers better results. Metaheuristics stand out within the Wrapper Methods. In this sense, Becerra-Rozas et al. [6] reviews the literature related to the binarization of continuous metaheuristics to solve combinatorial problems, finding that the feature selection problem is highly studied and inspiring to continue exploring the field of metaheuristic binarization. This study conducts a systematic literature review presenting a comprehensive taxonomy of objective functions, categorized into Single-Objective and Multi-Objective functions. Based on this, the contributions of this research is a detailed classification of objective functions, which offers a comprehensive understanding of the status of various aspects. The remaining of this document is structured as follows. In Section 2, the applied methodology is presented, also detailing research questions. The research question proposed in Section 2 are answered in Section 3. Finally, in Section 4, the conclusions of the research and some future work lines are presented.

2 Methodology

The methodology is crucial in ensuring a robust and comprehensive research analysis. This research was conducted following the Systematic Literature Review (SLR) framework, as outlined by Kitchenham [17].

In an SLR, one of the main steps involves defining the research questions. These questions act as a guide, directing the exploration and analysis in the field. In this work, the research question is defined as: RQ: How is the objective function of the Feature Selection problem formulated?

In our literature review, we searched six databases from 2019 to May 2023, focusing on ‘Feature Selection’ in titles. The databases and their initial document counts were: Scopus (8812), Web of Sciences (4713), IEEE Xplore (2204), ScienceDirect by Elsevier (1388), SpringerLink (3006), and Wiley (220).

Due to the varying capabilities of the databases used, a manual refinement of search results was necessary. This refinement involved two main phases. Initially, we applied inclusion criteria using database tools, selecting manuscripts that had a title, DOI, and abstract, were not duplicates, were published in journals, and contained ‘Feature selection problem’ in the abstract. Subsequently, we manually filtered the 190 identified papers, reaffirming these criteria and ensuring that they
Discussion

3.1 How is the objective function of the Feature Selection problem formulated?

Optimization problems are composed of an objective function subject to constraints. The objective functions can be classified into two main categories: single-objective focused on optimizing only one objective and multi-objective functions focused on optimizing several objective functions at the same time. There are two ways of representing multi-objective optimization problems, which are (1) Pure multi-objective function and (2) Weighted multi-objective function. A summary of the classification of the objective functions found in the papers collected is shown below.


2. Multi-Objective
   (a) Pure multi-objective function: Accuracy, Correlation and complexity of features, Cost of features, Error rate, Miscellaneous, Number of feature selected (NFS).
   (b) Weighted multi-objective function: Accuracy and NFS, Accuracy, mutual information and NFS, Dependence of rough set theory and NFS, Error rate and NFS, F-score and NFS.

The analysis of the collected data indicates that weighted multi-objective functions dominate the landscape of feature selection research, with 102 instances of use. Mono-objective functions are also notably prevalent, appearing in 35 instances, while pure multi-objective functions are cited in 16 cases. There are 9 instances where the objective function classification is not clearly specified. It is worth noting that in a particular case, such as the study by the authors of [27], multiple types of objective functions are employed, which accounts for the total exceeding the sum of individual categories.

Single-objective functions are focused on optimizing only one objective function subject to constraints. Mathematically, single-objective optimization problems are modeled as follows [9]:

$$\min \text{ or } \max f(X) \quad (1)$$
Subject to

\[ g_i(X) < 0 \quad i = 1, 2, \ldots, N_{\text{ineq}} \]
\[ h_i(X) = 0 \quad i = 1, 2, \ldots, N_{\text{eq}} \]

where \( f(X) \) represents the objective function, \( X \) corresponds to the solution vector composed of the decision variables, \( g_i(X) \) and \( h_i(X) \) are the inequality and equality constraints respectively. Within this category, eight different functions were detected:

(a) **Accuracy**: Measures how well a classification algorithm correctly predicts the classes of the data sets. As example of use, this objective function can be found in [15]. Accuracy is calculated as the ratio of correctly predicted cases to the total number of cases in the dataset. Mathematically is defined as follows:

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}
\]

where TP (true positive) is the number of positive instances that are correctly classified, FP (false positive) is the number of negative instances that are wrongly classified as positive, TN (true negative) is the number of negative instances that are correctly classified, and FN (false negative) is the number of positive instances that are wrongly classified as negative.

(b) **Error rate**: Sometimes referred to as the misclassification rate, it is an essential complement to accuracy. Measures the overall accuracy of a model in terms of the proportion of misclassified instances in a data set. An example of this objective function can been found in [28] and mathematically is defined as follows:

\[
\text{Error rate} = \frac{FP + FN}{FP + FN + TP + TN} \text{ or } 1 - \text{Accuracy}
\]

(c) **Fuzzy c-means (FCM)**: Clustering algorithm which returns a cost function used to calculate the performance of the metaheuristic [8]. This objective has been pursued in [4].

(d) **Redundancy and relevance**: Redundancy has been used to quantify the similarity level between selected features. Relevance represents the relevance between features and categorical variables reflecting the recognition ability of the selected features. We found two papers that pursued relevance and redundancy. In [24], the authors use these metrics to calculate the objective function and relate them by subtraction, as follows:

\[
F(X) = \text{Relevance} - \text{Redundancy}
\]

On the other hand, in [30], the authors relate these metrics by means of a division, as shown below:
The objective function of the Feature Selection problem is formulated as follows:

\[ F(X) = \frac{\text{Redundancy}}{\text{Relevance}} \]  

(e) **Accuracy and correlation:** In [12], the authors present an objective function that relates the correlation between the selected features without the presence of class labels and the Accuracy. This objective function has been pursued in [12] and mathematically is defined as follows:

\[ F(X) = \frac{A + (1 - M)}{2} \]  

where \( A \) is the Accuracy and \( M \) is the computed correlation.

(f) **Shannon entropy:** Measures the amount of information in a distribution. If a distribution has a high entropy value, it contains higher information. The authors of [13] used this information as an objective function, defining it mathematically as follows:

\[ F(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i) \]  

where \( n \) corresponds to the number of features and \( p(x_i) \) is the probability of occurrence of a feature.

(g) **Hamming loss:** Evaluates the performance of multi-label classification problems. In multi-label classification, each instance can be associated with multiple class labels, and the goal is to predict all the correct labels for each instance. The hamming loss quantifies how well the model performs in terms of correctly predicting all the labels for each instance. This objective function has been pursued in [10] and mathematically is defined as follows:

\[ \text{Hamming Loss} = \frac{1}{p} \sum_{i=1}^{p} \frac{1}{q} |h(x_i) \triangle Y_i| \]  

where \( p \) and \( q \) indicate the number of samples and labels. \( h(x_i) \) represents the result of classification for \( i \)-th sample and \( Y_i \) shows the actual labels of corresponding sample. \( h(x_i) \) and \( Y_i \) are binary vector in which 1’s indicate the class labels to which the instance belong, \( \triangle \) represents the hamming distance between \( h(x_i) \) and \( Y_i \).

(h) **Jaccard index:** Measures the similarity and overlap between two sets. It is often used in data analysis, information retrieval, and text mining. This objective function has been pursued in [20] and mathematically is defined as follows:

\[ F(X) = \frac{TP}{TP + FP + FN} \]
where TP (true positive) is the number of positive instances correctly classified. FP (false positive) is the number of negative instances wrongly classified as positive and FN (false negative) is the number of positive instances wrongly classified as negative.

(i) Miscellaneous: Finally, we detected two objective functions, which the authors explain in greater detail in the respective papers; see [7] and [31].

In the literature collected 'Accuracy' and 'Error rate' have emerged as the two most utilized criteria within the span of the last five years. The data reflects a consistent preference for 'Accuracy' as a objective function, with a total of 16 instances of its application across the period from 2019 to 2023. On the other hand, 'Error rate,' which is inversely related to 'Accuracy,' was the second most employed metric, tallying 10 instances in total. Other single objective functions such as 'Fuzzy c-means' and 'Redundancy' have been notably less prevalent, each recorded only once in the five-year period, indicating a more focused interest in measures directly related to performance accuracy.

Pure multi-objective functions Pure multi-objective functions are focused on independent optimization of each objective function. Thus, Pareto dominance is used to determine the best solution. Mathematically, multi-objective optimization problems are modeled as follows [21]:

\[
\begin{align*}
\min \text{ or } \max f_1(X), f_2(X), ..., f_m(X) \\
\text{Subject to} \\
g_i(X) < 0 \quad i = 1, 2, ..., N_{\text{ineq}} \\
h_i(X) = 0 \quad i = 1, 2, ..., N_{\text{eq}}
\end{align*}
\] (11)

where \( f_1(X), f_2(X), ..., f_m(X) \) represents the \( m \) objective functions to be optimized, \( X \) corresponds to the solution vector composed of the decision variables, \( g_i(X) \) and \( h_i(X) \) are the inequality and equality constraints respectively. Within the category of Pure multi-objective functions, six different functions were detected:

(a) Error rate: Defined in Section 3.1 and mathematically in equation 4. An example of this objective function used as a parte of a multi-objective function can be found in [22].

(b) Number of features selected (NFS): One of the essential aspects when solving the feature selection problem is to increase the performance of the classifiers to the smallest number of features possible. Given this, the number of selected features is an important objective an example can be found in [14].
How is the objective function of the Feature Selection problem formulated? 7

(c) Cost of features: In [33], the authors incorporate costs associated with features to the feature selection problem, minimizing the costs associated with the features and the error rate of the classification algorithms.

(d) Accuracy: Defined in Section 3.1 and mathematically in equation 3. This objective function has been pursued in [18].

(e) Correlation and complexity of features: In [5], the authors propose four different metrics to build the objective function, join to the error rate and the correlation and complexity of the features.

(f) Miscellaneous: In [19], the authors use six different metrics to build the objective function, named as follows:

- $F_1(S) =$ Number of features selected.
- $F_2(S) =$ Accuracy.
- $F_3(S) =$ Relevance.
- $F_4(S) =$ Redundancy.
- $F_5(S) =$ Interclass Distance.
- $F_6(S) =$ Intraclass Distance.

Thus, the objective function is defined as follows:

$$\min F(X) = F_1(S), -F_2(S), F_3(S), -F_4(S), F_5(S), F_6(S)$$

In the context of pure multi-objective functions used for feature selection from 2019 to 2023, ‘Error rate’ and ‘Number of Feature selected’ were equally the most referenced objectives, each accumulating a total of 12 citations over the five-year span. Other multi-objective functions, including ‘Accuracy’ and those assessing the ‘Complexity of features’ and ‘Correlation’, were acknowledged to a lesser extent, each being utilized only once within the period. It is also worth mentioning that specific objectives, such as ‘Cost of Feature’ and ”Miscellaneous”, were mentioned once, indicating niche but present interests in the field.

Weighted Multi-objective functions In general, metaheuristics are designed to solve single-objective optimization problems, and adapting them to multi-objective optimization problems is very costly both computationally and in development time. In [16], the authors present a way to translate a multi-objective optimization problem into a single-objective optimization problem. This procedure is a weighted sum of all the objective functions, and mathematically, it is defined as follows:

$$\min \text{or } \max f(X) = w_1f_1 + w_2f_2 + ... + w_m f_m$$
Subject to

\[ w_i \geq 0 \quad \forall i = 1, 2, ..., m \]
\[ w_1 + w_2 + ... + w_m = 1 \quad (15) \]

where \( w_1, w_2, ..., w_m \) are non-negative weights for \( m \) objective functions. Within the category of Weighted multi-objective functions, five different were detected:

(a) **Error rate and number of features selected (Error Rate & NFS):**
Within the feature selection problem, it is essential to improve the performance of the classifier and reduce the number of features. Given this, a weighted multi-objective function that relates these two terms was proposed as follows:

\[
F(X) = \alpha \cdot \text{Error Rate} + \beta \cdot \frac{S}{F} \quad (16)
\]

where \( S \) and \( F \) correspond to the number of features selected and the total number of features of the data set, and \( \alpha \) and \( \beta \) assign the importance of the error rate and the number of features selected. \( \alpha \) and \( \beta \in [0, 1] \), and there is no consensus on the values of these parameters. An example of this objective function can be found in [29].

(b-c) **Accuracy and number of features selected (Accuracy & NFS):**
Objective function similar to Error Rate & NFS. The difference is that the accuracy is the metric of the classification technique. In the literature, two objective functions that associate accuracy with the number of features selected were detected.

The first version found is defined as follows:

\[
F(X) = \alpha \cdot \text{Accuracy} + \alpha \cdot \left( 1 - \frac{S}{F} \right) \quad (17)
\]

where \( S \) and \( F \) correspond to the number of features selected and the total number of features of the data set, and \( \alpha \) assigns the importance of the number of features selected. \( \alpha \in [0, 1] \) and there is no consensus on the values of this parameter. An example where this objective function has been used is [26].

The second version found is defined as follows:

\[
F(X) = \alpha \cdot \text{Accuracy} + \beta \cdot \left( \frac{F - S}{F} \right) \quad (18)
\]

where \( S \) and \( F \) correspond to the number of features selected and the total number of features of the data set, and \( \alpha \) and \( \beta \) assign the importance of the accuracy and the number of features selected. \( \alpha \) and \( \beta \in [0, 1] \), and there is no consensus on the values of these parameters. An example this objective function can be found [23].
(d) **F-score and number of features selected (F-score & NFS):** This objective function has only been studied in [3] and mathematically, is defined as follows:

$$\min F(X) = w_1 z_1 + w_2 z_2 + w_3 z_3$$

(19)

where $w_1 + w_2 + w_3 = 1$ and the authors determine that $w_1 = 0.5$, $w_2 = 0.25$ and $w_3 = 0.25$. $z_1$, $z_2$ and $z_3$ are defined as follows:

$$\min z_1 = 1 - \text{F-score}$$

(20)

$$\min z_2 = \frac{|S|}{T}$$

(21)

$$\min z_3 = \frac{\max\{t|_{x_t \in S}\}}{T}$$

(22)

where $z_1$ aims to maximize the F-score, $z_2$ seeks to minimize the number of features selected per unit of time, and $z_3$ pursue minimize the last feature to be selected.

(e) **Accuracy, mutual information, and number of features selected:**

This objective function has only been studied in [25] and mathematically, is defined as follows:

$$F(X) = \alpha \cdot \text{Accuracy} + \beta \cdot \left(\frac{|F - S|}{F}\right) + \gamma \cdot \text{Mean}(I(X_k; Y))$$

(23)

where $|S|$ is the number of selected features, $\alpha \cdot \text{Accuracy}$, $\beta \cdot \left(\frac{|F - S|}{F}\right)$ and $\gamma \cdot \text{Mean}(I(X_k; Y))$ are considered for increasing classification accuracy, reducing the number of selected features, and increasing the mean of mutual information, respectively. $\alpha$, $\beta$, and $\delta \in [0, 1]$ and the sum equals 1.

(f) **Dependence of rough set theory and number of features selected:**

This objective function has only been studied in [1] and mathematically, is defined as follows:

$$F(X) = \alpha \cdot \text{dep}(X) + \beta \cdot \frac{1}{S}$$

(24)

where $X$ is the feature subset found. Fitness is calculated based on the dependency measure of rough set theory $\text{dep}(X)$, and $S$ is the length of the feature subset size($x_i$). $\alpha \in [0, 1]$ controls the relative weight of dependency value and feature subset length, and $\beta$ is $(1 - \alpha)$.

In the surveyed literature, objective functions frequently incorporate the number of features selected (NFS) as a key parameter. Predominantly, the combination of Error rate & NFS has been extensively examined, featuring in 85 articles. Another significant metric is the juxtaposition of Accuracy with NFS (Accuracy & NFS), which assesses model accuracy relative to the compactness
of the feature set. This metric appeared nine times in its second iteration and five times in its initial form, underscoring its relevance in optimizing the balance between model performance and simplicity.

4 Conclusion

This work studied the complex and broad field of research related to feature selection, scrutinizing the objective functions employed to address this challenge. Guided by our central research question, we uncovered a wide array of objective functions utilized in feature selection inquiries. These functions have been systematically categorized into single-objective and multi-objective classes, with further subdivisions into pure and weighted forms. Our findings reveal that weighted multi-objective functions enjoy a preeminent status in the literature, indicative of their comprehensive utility in addressing the complexities inherent in feature selection tasks.

References

How is the objective function of the Feature Selection problem formulated?

11


Evidence on the Regularisation Properties of Maximum-Entropy Reinforcement Learning

Rémy Hosseinkhan Boucher1,2, Onofrio Semeraro1,2, and Lionel Mathelin1,2

1 Université Paris-Saclay, Orsay, France
2 CNRS, Laboratoire Interdisciplinaire des Sciences du Numérique, Orsay, France
{remy.hosseinkhan, onofrio.semeraro, lionel.mathelin}@upsaclay.fr

Abstract. The generalisation and robustness properties of policies learnt through Maximum-Entropy Reinforcement Learning are investigated on chaotic dynamical systems with Gaussian noise on the observable. First, the robustness under noise contamination of the agent’s observation of entropy regularised policies is observed. Second, notions of statistical learning theory, such as complexity measures on the learnt model, are borrowed to explain and predict the phenomenon. Results show the existence of a relationship between entropy-regularised policy optimisation and robustness to noise, which can be described by the chosen complexity measures.

Keywords: Maximum-Entropy Reinforcement Learning · Robustness · Complexity Measures · Flat Minima · Fisher Information · Regularisation

1 Introduction

Maximum-Entropy Reinforcement Learning [44] aims to solve the problem of learning a policy which optimises a chosen utility criterion while promoting the entropy of the policy. The standard way to account for the constraint is to add a Lagrangian term to the objective function. This entropy-augmented objective is commonly referred to as the soft objective.

There are multiple advantages in solving the soft objective over the standard objective. For instance, favouring stochastic policies over deterministic ones allows learning multi-modal distributions [17]. In addition, agent stochasticity is a suitable way to deal with uncertainty induced by Partially Observable Markov Decision Processes (PO-MDP). Indeed, there are PO-MDP such that the best stochastic adapted policy can be arbitrarily better than the best deterministic adapted policy [42].

Furthermore, several important works highlight both theoretical and experimental robustness of those policies under noisy dynamics and rewards [14].

In this context, the term “stochastic adapted policy” is a conditional distribution on the control space $U$ given the observation space $Y$ since this type of policy is “adapted” from Markovian policies in fully observable MDPs.
Related to the latter notion of robustness, the maximum-entropy principle exhibits non-trivial generalisation capabilities, which are desired in real-world applications [18].

However, the reasons for such robustness properties are not yet well understood. Thus, further investigations are needed to grasp the potential of the approach and to design endowed algorithms. A clear connection between Maximum-Entropy RL and their robustness properties is important and intriguing.

Meanwhile, recent work in the deep learning community discusses how some complexity measures on the neural network model are related to generalisation, and explain typically observed phenomena [33]. In fact, these complexity measures are derived from the learnt model, bound the PAC-Bayes generalisation error, and are meant to identify which of the local minima generalise well.

As a matter of fact, a relatively recent trend in statistical learning suggests generalisation is not only favored by the regularisation techniques (e.g., dropout) but mainly because of the flatness of the local minima [22,12,27]. The reasons for such regularity properties remain an open problem. This work aims to address these points in the context of Reinforcement Learning, and addresses the following questions:

What is the bias introduced by entropy regularisation? Are the aforementioned complexity measures also related to the robustness of the learnt solutions in the context of Reinforcement Learning?

In that respect, by defining a notion of robustness against noisy contamination of the observable, a study on the impact of the entropy regularisation on the robustness of the learnt policies is first conducted. After explaining the rationale behind the choice of the complexity measures, a numerical study is performed to validate the hypothesis that some measures of complexity are good robustness predictors. Finally, a link between the entropy regularisation and the flatness of the local minima is treated through the information geometry notion of Fisher Information.

The paper is organised as follows. Section 2 introduces the background and related work, Section 3 presents the problem setting. Section 4 is the core contribution of this paper. This section introduces the rationale behind the studied complexity measures from a learning theory perspective, as well as their expected relation to robustness. Lastly, Section 5 presents the experiments related to the policy robustness as well as their complexity, while Section 6 examines the results obtained. Finally, Section 7 concludes the paper.

2 Related work

Maximum Entropy Policy Optimisation In [18], the generalisation capabilities of entropy-based policies are observed where multimodal policies lead to optimal solutions. It is suggested that maximum entropy solutions aim to learn all the possible ways to solve a task. Hence, transfer learning to more challenging objectives is made easier, as demonstrated in their experiment. This study investigates the impact of adopting policies with greater randomness on
their robustness. The impact of the entropy regularisation on the loss landscape has been recently studied in [3]. They provide experimental evidence about the smoothing effect of entropy on the optimisation landscape. The present study aims specifically to answer the question in Section 3.2.4 of their paper: *Why do high entropy policies learn better final solutions?* This paper extends their results from a complexity measure point of view. Recently, [32, 11] studied the equivalence between robustness and entropy regularisation on regularised MDP.

*Flat minima and Regularity* The notion of local minima flatness was first introduced in the context of supervised learning by [22] through the Gibbs formalism [19]. Progressively, different authors stated the concept with geometric tools such as first order (gradient) or second order (Hessian) regularity measures [47, 27, 37, 46, 12]. In a similar fashion, [7] uses the concept of local entropy to smooth the objective function.

In the scope of Reinforcement Learning, [3] observed that flat minima characterise maximum entropy solutions, and entropy regularisation has a smoothing effect on the loss landscape, reducing the number of local optima. A central objective of this present study is to investigate this latter property further and relate it to the field of research on robust optimisation. Lastly, among the few recent studies on the learning and optimisation aspects of RL, [15] shows how a well-chosen regularisation can be very effective for deep RL. Indeed, they explain that constraining the Lipschitz constant of only one neural network layer is enough to compete with state-of-the-art performances on a standard benchmark.

*Robust Reinforcement Learning* A branch of research related to this work is the study of robustness with respect to the uncertainty of the dynamics, namely *Robust Reinforcement Learning* (Robust RL), which dates back to the 1970s [38]. Correspondingly, in the field of control theory, echoes the notion of robust control and especially $H_{\infty}$ control [48], which also appeared in the mid-1970s after observing Linear Quadratic Regulator (LQR) solutions are very sensitive to perturbations while not giving consistent enough guarantees [13].

More specifically, the Robust RL paradigm aims to control the dynamics in the worst-case scenario, *i.e.*, to optimise the minimal performance for a given objective function over a set of possible dynamics through a min-max problem formulation. This set is often called *ambiguity set* in the literature. It is defined as a region in the space of dynamics close enough w.r.t. to some divergence measure, such as the relative entropy [35]. Closer to this work, the recent paper from [14] shows theoretically how Maximum-Entropy RL policies are inherently robust to a certain class of dynamics of fully-observed MDP. The finding of their article might still hold in the partially observable setting as any PO-MDP can be cast as fully-observed MDP with a larger state-space of probability measures [21], providing the ambiguity set is adapted to a more complicated space.
3 Problem Setup and Background

3.1 Partially Observable Markov Decision Process with Gaussian noise

First, the control problem when noisy observations are available to the agent is formulated. The study focuses on Partially Observable Markov Decision Processes (PO-MDP) with Gaussian noise of the form [10]:

\[ X_{h+1} = F(X_h, U_h) \]
\[ Y_h = G(X_h) + \epsilon, \epsilon \sim N(0, \sigma^2 I_d) \]  

(1)

with \( X_h \in \mathcal{X}, U_h \in \mathcal{U} \) and \( Y_h \in \mathcal{Y} \) for any \( h \in \mathbb{N} \), where \( \mathcal{X}, \mathcal{U} \) and \( \mathcal{Y} \) are respectively the corresponding state, action and observation spaces. The initial state starts from a reference state \( x^* \) on which centred Gaussian noise with diagonal covariance \( \sigma^2 I_d \) is additively applied, \( X_0 \sim N(x^*, \sigma^2 I_d) \). Associated with the dynamics, an instantaneous cost function \( c : \mathcal{X} \times \mathcal{U} \rightarrow \mathbb{R}^+ \) is also given to define the control model.

In this context, a policy \( \pi \) is a transition kernel on \( \mathcal{A} \) given \( \mathcal{Y} \), i.e., a distribution on actions conditioned on observations. This kind of policies are commonly used in the literature but can be very poor in the partially observable setting where information is missing. Together, a control model, a policy \( \pi \) and an initial distribution \( P_{X_0} \) on \( \mathcal{X} \) define a stochastic process with distribution \( P_{\pi,\epsilon} \) where the superscript \( \epsilon \) highlights the dependency on the observation noise \( \epsilon \). Similarly, one denotes by \( P_{\pi} \) the distribution of the process when the noise is zero almost-surely, i.e., \( P_{\pi} = P_{\pi,0} \). More details about the PO-MDP control problem can be found in [21, 6].

Here, the maximum-entropy control problem is to find a policy \( \pi^* \) which minimises the following performance criterion

\[ J_{m}^{\pi,\epsilon} = E_{\pi,\epsilon} \left[ \sum_{h=0}^{H} \gamma^h c(X_h, U_h) \right] + \alpha_m E_{\pi,\epsilon} \left[ \sum_{h=0}^{H} \gamma^h H(\pi(\cdot | X_h)) \right], \]  

(2)

where \( H \in \mathbb{N} \) is a given time horizon, \( E_{\pi,\epsilon} \) denotes the expectation under the probability measure \( P_{\pi,\epsilon} \), \( H \) denotes the differential entropy [9] and \( \alpha_m \) is a time-dependent weighting parameter that evolves over training time \( m \leq m_D = |\mathcal{D}| \) with \( |\mathcal{D}| \) being the total number of times the agent interacts with the system such that all observations used by the learning algorithm form the dataset \( \mathcal{D} \) at the end of the training procedure (when \( m_D \) environment interactions are done). In the \( \alpha_m = 0 \) case, \( J_{m}^{\pi,\epsilon} \) is denoted \( J^{\pi,\epsilon} \). The quantity \( J^{\pi,\epsilon} \) is called the value function or, more generally, loss.

Moreover, the performance gap for dynamics with noisy and noiseless observables will be considered in the sequel. In this context, the (rate of) excess risk under noise is defined as the difference between the loss under noisy dynamics and the loss under noiseless dynamics:
**Definition 1 (Excess Risk Under Noise).** The excess risk under noise of a policy $\pi$ for a PO-MDP with dynamics (1) is defined as:

$$ R_\pi = E_{\pi,\epsilon} \left[ \sum_{h=0}^{H} \gamma^h c(X_h, U_h) \right] - E_\pi \left[ \sum_{h=0}^{H} \gamma^h c(X_h, U_h) \right] = J_{\pi,\epsilon} - J_\pi $$

Similarly, the rate of excess risk under noise is defined as:

$$ \dot{R}_\pi = \frac{J_{\pi,\epsilon} - J_\pi}{J_\pi} = \frac{R_\pi}{J_\pi} $$

Note that in the above definition, expectations are taken with respect to the probability measure $P_{\pi,\epsilon}$ and $P_{\pi}$ respectively. The rate of excess risk under noise represents the performance degradation after noise introduction in value function units. In the rest of the paper, arguments to derive complexity measures will be developed, allowing to predict the excess risk under noise and provide numerical evidence showing maximum-entropy policies are more robust regarding this metric. Hence, maximum-entropy policies implicitly learn a robust control policy in the sense of Definition 1.

In the next section, some concepts of statistical learning theory are introduced. Then, complexity measures will be defined to quantify the regularisation power of the maximum-entropy objective of (2).

### 4 Complexity Measures and Robustness

#### 4.1 Complexity Measures

The principal objective of statistical learning is to provide bounds on the generalisation error, so-called generalisation bounds. In the following, it is assumed that an algorithm $A$ returns a hypothesis $\pi \in F$ from a dataset $D$. Note that the dataset $D$ is random and the algorithm $A$ is a randomised algorithm.

As the hypothesis set $F$ typically used in machine learning is infinite, a practical way to quantify the generalisation ability of such a set must be found. This quantification is done by introducing complexity measures, enabling the derivation of generalisation bounds.

**Definition 2 (Complexity measure).** A complexity measure is a mapping $M : F \rightarrow \mathbb{R}_+$ that maps a hypothesis to a positive real number.

According to [33] from which this formalism is inspired, an appropriate complexity measure satisfies several properties. In the case of parametric models $\pi_\theta \in F(\Theta)$ with $\theta \in \Theta \subset \mathbb{R}^b$, it should increase with the dimension $b$ of the parameter space $\Theta$ as well as being able to identify when the dataset $D$ contains totally random, spurious or adversarial data. As a result, finding good complexity measures $M$ allows the quantification of the generalisation ability of a hypothesis set $F$ or a model $\pi$ and an algorithm $A$. 

81
4.2 Complexity measures for PO-MDP with Gaussian Noise

This paper studies heuristics about generalisation bounds on the optimal excess risk under noise from Definition 1 when the optimal policy $\pi_\theta^*$ is learnt with an algorithm $\mathcal{A}$ on the non-noisy objective $J^{\pi}$, where $\alpha_m = 0$ for any $m$.

**Definition 3 ((Rate of) Excess Risk Under Noise Bound).** Given an optimal policy $\pi^*$ learnt with an algorithm $\mathcal{A}$ on the non-noisy objective $J^{\pi}$, the optimal excess risk under noise bound is a real-valued mapping $\varphi$ such that

$$R^{\pi^*} \leq \varphi(\mathcal{M}(\pi^*, D), m_D, \eta, \delta)$$

and $\varphi$ is increasing with the complexity measure $\mathcal{M}$ and the sample complexity $m_D$. The definition is similar for the rate of excess risk under noise bound where $\dot{R}^{\pi^*}$ is used instead of $R^{\pi^*}$.

Hence, considering a learning algorithm $\mathcal{A}$ with a parameterised family $\mathcal{F}(\theta) = (\pi_\theta)_{\theta \in \Theta}, \Theta \subset \mathbb{R}^b$, such that $\theta = (\theta_{\mu}, \theta_{\sigma})$ with $\pi_\theta(\cdot | x) \sim \mathcal{N}(\mu_\theta(x), \text{diag}(\theta_{\sigma}))$,

where $\mu_\theta$ is a shallow multi-layer feed-forward neural network (with depth-size $l = 2$, width $w = 64$ neurons, weights matrix $(\theta^i_{\mu})_{1 \leq i \leq l}$) and $\text{diag}(\theta_{\sigma})$ is a diagonal matrix of dimension $q = \dim(\mathcal{U})$ parameterising the variance.

4 Note this choice of state-independent policy variance is inspired by [3] and simplifies the problem.

**Norm based complexity measures** First, the so-called norm-based complexity measures are functions of the norm of some subset of the parameters of the model. For instance, a common norm-based measure calculates the product of the operator norms of the neural network linear layers. The measures are commonly used in the statistical learning theory literature to derive bounds on the generalisation gap, especially in the context of neural networks [34, 16, 30].

In fact, the product of the norm of the linear layers of a standard class of multi-layer neural networks (including Convolutional Neural Networks) serves as an upper bound on the often intractable Lipschitz constant of the network [30]. Thus, controlling the magnitude of the weights of the linear layers increases the regularity of the model.

Consequently, the following complexity measures are defined:

- $\mathcal{M}(\pi_\theta, D) = \|\theta_{\mu}\|_p$
- $\mathcal{M}(\pi_\theta, D) = \prod_{i=1}^l \|\theta^i_{\mu}\|_p$ where $\theta^i_{\mu}$ is the $i$th layer of the network $\mu_{\theta_{\mu}}$.

In this context $\| \cdot \|_p$ with $p = 1, 2, \infty$ denotes the $p$-operator norm while $p = F$ denotes the Frobenius norm, which is discarded for the first case of the full parameters vector $\theta_{\mu}$ (since Frobenius norm is defined for matrix).
Flatness based complexity measures

On the other hand, another measure of complexity is given by the flatness of the optimisation local minimum (see Section 2 for a brief overview). As [29, 33] have pointed out, the generalisation ability of a parametric solution is controlled by two key components in the context of supervised learning: the norm of the parameter vector and its flatness w.r.t. to the objective function.

One might wonder if a similar robustness property still holds in the setting of Reinforcement Learning. In this manner, complexity measures quantifying the flatness of the solution are needed. Concretely, the interest lies in the flatness of the local minima of the objective function $J^\pi$. As stated earlier, there are several ways to quantify the flatness of a solution with metrics derived from the gradient or curvature of the loss function at the local optimum, such as the Hessian’s largest eigenvalue - otherwise spectral norm [27] or the trace of Hessian [12].

Moreover, as discussed in Section 2, [3] observed that maximum entropy solutions are characterised by flat minima while entropy regularisation has a smoothing effect on the loss landscape. Hence, a central objective of this present study is to investigate this latter property further and relate it to the robustness aspect of the resulting policies.

However, instead of dealing directly with the Hessian of the objective $J^\pi$ this work proposes a measure based on the conditional Fisher Information $I$ of the policy due to its link with a notion of model regularity in the parameter space.

**Definition 4 (Conditional Fisher Information Matrix).** Let $x \in \mathcal{X}$ and $\pi_\theta$ a policy identified by its conditional density for a parameter $\theta \in \Theta \subset \mathbb{R}^b$ and suppose $\rho$ is a distribution over $\mathcal{X}$. The conditional Fisher Information Matrix of the vector $\theta$ is defined under some regularity conditions as

$$I(\theta) = - \mathbb{E}^{X \sim \rho, U \sim \pi_\theta(|X)} \left[ \nabla_\theta^2 \log \pi_\theta(U | X) \right], \quad (6)$$

where $\nabla_\theta^2$ denotes the Hessian matrix evaluated at $\theta$.

Note that the distribution over states $\rho$ is arbitrary and can be chosen as the discounted state visitation measure $\rho^\pi$ induced by the policy $\pi$ [1] or the stationary distribution of the induced Markov process if the policy is Markovian and the MDP ergodic\(^5\) as it is done in [25].

As a matter of fact, it has already been mentioned in the early works of policy optimisation [25] that this quantity $I$ might be related to the Hessian of the objective function. Indeed, the Hessian matrix of the standard objective function reads (see [41] for a proof):

$$\nabla_\theta^2 J^\pi_\theta = \mathbb{E}^{\pi_\theta} \left[ \sum_{h,i,j=0}^H c(X_h, U_h) \left( \nabla_\theta \log \pi_\theta(U_i | X_i) \nabla_\theta \log \pi_\theta(U_j | X_j)^T + \nabla_\theta^2 \log \pi_\theta(U_i | X_i) \right) \right]. \quad (7)$$

\(^5\) With these choices, the following holds: $\mathbb{E}^{\rho^\pi(da)\pi(da|x)} = \mathbb{E}^\pi$ up to taking the expectation w.r.t. the state-action space (no subscript under $X$ and $U$) or the trajectory space (with subscripts such as $X_h$ and $U_h$ as trajectory coordinate) [1].
As suggested by the author mentioned above (S. Kakade), (7) might be related to the state-conditional log-likelihoods \( \nabla_\theta^2 \log \pi_\theta \) on the rightmost part of the expectation of (7) belongs to the objective-function Hessian \( \nabla_\theta^2 J_\pi \) while the Fisher Information \( I(\theta) \) is an average of the Hessian of the policy log-likelihood.

In any case, the conditional FIM measures the regularity of a critical component of the objective to be minimised. Thus, the trace of the conditional FIM of the mean actor network parameter \( \theta_\mu \) is suggested as a complexity measure

\[
- \mathcal{M}(\pi_\theta, D) = \text{Tr}(\mathcal{I}(\theta_\mu)) = \text{Tr}(\mathbb{E}_{X \sim \rho^\pi, U \sim \pi_\theta(\cdot | X)} \left[ \nabla_\theta^2 \log \pi_\theta(U | X) \right]).
\]

Moreover, in the context of classification, a link between the degree of stochasticity of optimisation gradients (leading to flatter minima [31, 45]) and the FIM trace during training has recently been revealed in [23]. Magnitudes of the FIM eigenvalues may be related to loss flatness and norm-based capacity measures to generalisation ability [26] in deep learning.

5 Experiments

5.1 Robustness under noise of Maximum Entropy Policies

The first hypothesis is that maximum entropy policies are more robust to noise than those trained without entropy regularisation (which play the role of control experiments). Consequently, the robustness of the controlled policy \( \pi_\theta^* \) is compared with the robustness of the maximum entropy policy \( \pi_\theta^\alpha \) for different temperature evolutions \( \alpha = (\alpha_m)_{0 \leq m \leq m_D} \). In this view, and since inter-algorithm comparisons are characterised by high uncertainty [20, 8, 2], only one algorithm \( A \) (Proximal Policy Optimisation (PPO) [40]) is retained while results on multiple entropy constraint levels \( \alpha = (\alpha_m)_{0 \leq m \leq m_D} \) are examined.

In this regard, ten independent PPO models are trained for each of the five arbitrarily chosen entropy temperatures \( \alpha^i = (\alpha^i_m)_{0 \leq m \leq m_D} \) where \( i \in \{1, \ldots, 5\} \), on dynamics without observation noise, i.e., where \( \sigma^2 = 0 \). The entropy coefficients linearly decay during training, and all vanish \( (\alpha_m = 0) \) when \( m \) reaches one-fourth of the training time \( m_{1/4} = \left\lfloor \frac{m_D}{4} \right\rfloor \) in order to replicate a sort of exploration-exploitation procedure, ensuring that all objectives \( J^i_m \) are the same whenever \( m \geq m_{1/4} \), i.e., \( J^i_m = J^\pi \). This choice is different but inspired by [3] as they optimise using only the policy gradient and manipulate the standard deviation of Gaussian policies directly, whereas, in the present approach, it is done implicitly with an adaptive entropy coefficient. An algorithm that learns a model with a given entropy coefficient \( \alpha = (\alpha_m)_{0 \leq m \leq m_D} \) is denoted as \( A_\alpha \).

The chosen chaotic systems are the Lorenz [43] (with \( m_D = 10^6 \)) and Kuramoto-Sivashinsky (KS) [5] (with \( m_D = 2 \cdot 10^6 \)) controlled differential equations. The defaults training hyper-parameters from Stable-Baselines3 [36] are used.
5.2 Robustness against Complexity Measures

So far, three separate analyses on the $5 \times 10$ models obtained have been performed on the Lorenz and Kuramoto-Sivashinsky (KS) controlled differential equations. First, as mentioned before, the robustness of the models for each of the chosen entropy temperatures $\alpha^i$ is tested against the same dynamics but now with a noisy observable, i.e., $\sigma_Y > 0$. Second, norm-based complexity measures introduced in Section 4.2 are evaluated and compared to the generalisation performances of the distinct algorithms $A_{\alpha^i}$. Third, numerical computation of the conditional distribution of the trace of the Fisher Information Matrix given by (6) is performed to test the hypothesis that this regularity measure is an indicator of robust solutions. The state distribution $\rho_{\pi_{\theta}}$ is naturally chosen as the state visitation distribution induced by the policy $\pi_{\theta}$. The following section discusses the results of those experiments.

6 Results

This section provides numerical evidence of maximum entropy’s effect on the robustness, as defined by the Excess Risk Under Noise defined by (3). Then, after quantifying robustness, the relation between the complexity measures defined in Section 4.2 and robustness is studied.

6.1 Entropy Regularisation induces noise robustness

In the first place, a distributional representation of the rate of excess risk under noise defined in (3) is computed for each of the $5 \times 10$ models obtained with the PPO algorithm $A_{\alpha^i}$, $i \in \{1, \ldots, 5\}$ and different levels of observation noise $\sigma_Y > 0$.

First and foremost, the results shown in Figure 1 indicate that the noise introduction to the system observable $Y$ of KS and Lorenz leads to a global decrease in performance, as expected. The robustness to noise contamination of the two systems is improved by initialising the policy optimisation procedure up to a certain intermediate threshold of the entropy coefficient $\alpha^i > 0$. Once this value is reached, two respective behaviours are observed depending on the system. In the case of the Lorenz dynamics, the robustness continues to improve after this entropy threshold, whereas the opposite trend is observed for KS (particularly with the maximal entropy coefficient chosen).

\footnote{By replacing the expectation operator $\mathbb{E}$ with the conditional expectation $\mathbb{E}[\cdot \mid X_0]$ in the definition of $R^\pi$ in (3), the quantity becomes a random variable for which the distribution can be estimated by sampling the initial state distribution $X_0 \sim \mathcal{N}(x^*_e, \sigma_e^2 I_d)$. In fact, taking the conditional expectation gives the difference of the standard value functions under $P^\pi$ and $P^{\pi,\epsilon}$.}
Fig. 1: Distributional representation of the rate of excess risk under noise $\tilde{R}^x$ conditioned on the $\alpha^i$ used during optimisation for different initial state distribution $X_0 \sim \mathcal{N}(x^*_e, \sigma^2_e I_d)$. Each of the rows corresponds to one of the dynamical systems of interest. Each of the columns corresponds to one of the initial state distributions of interest. There are two non-zero fixed points (equilibria) $x^*_e$ for Lorenz and three for KS. From top to bottom: KS; Lorenz.

For each box plot, three intensities $\sigma_Y$ for the observation noise $\epsilon$ are evaluated. As expected, when the uncertainty regarding the observable $Y$ increases through the variance $\sigma_Y$ of the observation signal noise $\epsilon$, the policy performance decreases globally ($\tilde{R}^x$ increases). Moreover, the rate of excess risk under noise tends to decrease when $\alpha^i$ increases in the Lorenz case, whereas it decreases up to a certain entropy threshold for KS before increasing again.

Hence, the sole introduction of entropy-regularisation in the objective function impacts the robustness. This behaviour difference between Lorenz and KS might be explained by the variability of the optimisation landscapes that can be observed with respect to the chosen underlying dynamics as underlined in [3].

6.2 Maximum entropy as a norm-based regularisation on the policy

Norm-based complexity measures introduced in Section 4.2 are now evaluated. For a complexity measure $\mathcal{M}$ to be considered significant, it should be correlated with the robustness of the model.

Accordingly, the different norm-based measures presented in Section 4.2 are estimated. Figure 2 shows the layer-wise product norm of the policy actor net-
Evidence on the regularization properties of Maximum-Entropy RL

Fig. 2: Measures of complexity $\mathcal{M}(\pi_{\theta}, D) = \Pi_{i=1}^{l} \|\theta_{i}\|_{p}$ with $p = 1, 2, \infty, F$ conditioned on the $\alpha^i$ used during optimisation. Each row corresponds to one of the dynamical systems of interest while column represents a different norm order $p$. From top to bottom: Lorenz and KS.

For the Lorenz case, the barycenters of the measures tend to decrease when $\alpha^i$ increases. Regarding KS, passing a threshold, the complexity increases again with the entropy. In addition, the measures are much more concentrated when $\alpha^i > 0$. For $p = 2, F$, the separation of the measures w.r.t. the different $\alpha^i$ is more pronounced.

work parameters ($\mathcal{M}(\pi_{\theta}, D) = \Pi_{i=1}^{l} \|\theta_{i}\|_{p}$) w.r.t. to their associated entropy coefficient $\alpha^i$ for all the 50 independently trained models.

Again, policies obtained with initial $\alpha^i > 0$ exhibit a trend toward decreasing complexity measure values as $\alpha$ increases up to a certain threshold of the entropy coefficient. Similarly to Section 6.1, the complexity measure continues to decrease after surpassing this threshold for the Lorenz system. On the other hand, in the KS case, $\mathcal{M}(\pi_{\theta}, D)$ increases again once its entropy threshold is reached, notably for the larger entropy coefficient.

Moreover, the measures tend to be much more concentrated when $\alpha^i > 0$, especially in the case of KS (except for the higher $\alpha^i$).

This may indicate that the entropy regularisation acts on the uncertainty of the policy parameters. Likewise, similar observations can be made for the total norm of the parameters but are not introduced here for the sake of brevity.

Consequently, this experiment highlights an existing correlation between maximum entropy regularisation and norm-based complexity measures. As this complexity measure is linked to the Lipschitz continuity of the policy, one might wonder if the regularity of the policy is more directly impacted. This is the purpose of the next subsection.
6.3 Maximum entropy reduces the average Fisher-Information

Another regularity measure is considered: the average trace of the Fisher information \( M(\pi_\theta, D) = \text{Tr}(\mathcal{I}(\theta)) = \text{Tr}(- \mathbb{E}_{X \sim \rho, U \sim \pi_\theta(X)} \left[ \nabla^2_{\theta} \log \pi_\theta(U | X) \right]) \). As discussed in 4.2, this quantity reflects the regularity of the policy and might be related to the flatness of the local minima of the objective function.

Figure 3 shows the distribution under \( \pi_\theta \) of the trace of the state conditional Fisher Information of the numerical optimal solution \( \theta_{\mu, \alpha}^* \) for the policy w.r.t. the \( \alpha^i \) used during optimisation. In other words, a kernel density estimator of the distribution of \( \text{Tr}(\mathcal{I}(\pi_{\theta^*}_{\mu, \alpha}^*(\cdot | X))) \text{ when } X \sim \rho^{\theta^*} \) is represented. The results of this experiment suggest first, this distribution is skewed negatively and has a fat right tail. This means some regions of the support of \( \rho^{\theta^*} \) provide FIM trace with extreme positive values, meaning the regularity of the policy may be poor in these regions of the state space.

A comparison of the distribution w.r.t. the different \( \alpha^i \) sheds further light on the relation between robustness and regularity. In fact, there appears to be a correspondence between the robustness, as indicated by the rate of excess risk under noise \( \mathcal{R} \) shown in Figure 1 and the concentration of the trace distribution toward larger values (i.e. more irregular policies) when the model is less robust.

Meanwhile, under the considerations of 4.2 and since it is known that entropy regularisation favours flat minima in RL [3], these experimental results support the hypothesis of an existing relationship between robustness, objective function flatness around the solution \( \theta^* \) and conditional Fisher information of \( \theta^* \).

For a complementary point of view, a supplementary experiment regarding the sensitivity of the policy updates during training w.r.t. to different level of entropy is also presented in Appendix A.

7 Discussion

In this paper, the question of the robustness of maximum entropy policies under noise is studied. After introducing the notion of complexity measures from the statistical learning theory literature, numerical evidence supports the hypothesis that maximum entropy regularisation induces robustness under noise. Moreover, norm-based complexity measures are shown to be correlated with the robustness of the model. Then, the average trace of the Fisher Information is shown to be a relevant indicator of the regularity of the policy. This suggests the existence of a link between robustness, regularity and entropy regularisation. Finally, this work contributes to bringing statistical learning concepts such as flatness into the field of Reinforcement Learning. New algorithms or metrics, such as in the work of [28], may be built upon notions of regularity, e.g., Lipschitz continuity, flatness or Fisher Information of the parameter in order to achieve robustness.
Evidence on the regularization properties of Maximum-Entropy RL

Fig. 3: Distribution of the trace of the (conditional) Fisher information of the numerical optimal solution $\theta^*_{\mu, \alpha}$ for the policy w.r.t. the $\alpha^i$ used during optimisation. From left to right: Lorenz and KS environments. Colours: control experiment $\alpha^i = 0$ (black); intermediate entropy level $\alpha^i$ (blue); largest $\alpha^i$ (red). A skewed distribution towards (relatively) larger values is observed for all controlled dynamical systems. Moreover, those right tails exhibit high kurtosis, especially for the control experiment (black) and the model with the larger entropy coefficient (red) for the KS system. Finally, solutions with intermediate entropy levels (blue) are much more concentrated - have lower variance than the others. About Lorenz, the barycenter of the more robust model (red) is shifted towards lower values than the others.

Acknowledgements

The authors acknowledge the support of the French Agence Nationale de la Recherche (ANR), under grant ANR-REASON (ANR-21-CE46-0008). This work was performed using HPC resources from GENCI-IDRIS (Grant 2023-[AD011014278]).

A Weights sensitivity during training

This section is intended to provide complementary insights on the optimisation landscape induced by the entropy coefficient $\alpha$ during training from the conservative or trust region policy iteration point of view [24, 39].

Let $\{\theta^\alpha_m\}_{m=1}^k$ be the sequence of weights of the policy during the training of the model for some initial entropy coefficient $\alpha$. The conditional Kullback-Leibler divergence between the policy identified by the parameters $\theta^\alpha_m$ and the subsequent policy defined by the parameters $\theta^\alpha_{m+1}$ is given by

$$D_{KL}(\theta^\alpha_m, \theta^\alpha_{m+1}) = \mathbb{E}_{X \sim \rho} \left[ \int dX \log \left( \frac{\pi_{\theta^\alpha_m}(dX | X)}{\pi_{\theta^\alpha_{m+1}}(dX | X)} \right) \pi_{\theta^\alpha_{m+1}}(du | X) \right].$$

The above quantity is a measure of the divergence from the policy at time $m$ to the policy at time $m + 1$. Thus it may provide information on the local stiffness of the optimisation landscape during training.

Figure 4 shows the evolution of the Kullback-Leibler divergence between two subsequent policies during training for the Lorenz and KS controlled differential equations. Regarding the Lorenz system, the maximal divergence is reached for
the optimisation performed with the two lowest $\alpha^i$ while increasing entropy seems to slightly reduce the divergence. On the other hand, the highest divergence values observed for the KS system are reached for $\alpha^i = 0$ and the maximal entropy coefficient. This observation is coherent with the results of the previous sections and suggests that the entropy coefficient $\alpha$ impacts the optimisation landscape during training.

Interesting questions regarding the optimisation landscape and its link with the Fisher Information (through the point of view of Information Geometry [4]) are raised by the results of this section but are left for future work.

![Fig. 4: Evolution of $\bar{D}_{KL}(\theta^i_{\alpha^i}, \theta^i_{\alpha^i+1})$ during training for the Lorenz and KS controlled differential equations. For Lorenz, the maximal divergence is reached for the optimisation performed with $\alpha^i = 0$ and the second lowest $\alpha^i$. Regarding KS, the highest divergence values are observed for $\alpha^i = 0$ and the maximal entropy coefficient.](image)

References

Evidence on the regularization properties of Maximum-Entropy RL


Benchmarking the operational efficiency of major container ports using machine learning and bootstrap DEA

Jose Humberto Ablanedo-Rosas
Marketing, Management and Supply Chain Department,
University of Texas at El Paso, El Paso TX 79968, USA
jablanedorosas2@utep.edu

Abstract. This study aims to incorporate a machine learning approach when benchmarking the operational efficiency of 40 container ports located in five different developing regions including North Africa, West Africa, South America, North and Central America, and the Caribbean. A bootstrap data envelopment analysis (DEA) reveals that seven out of ten pairs of geographic regions differ in their operational efficiency scores, which can be explained by the diversity of the regions, port infrastructure dissimilarities, and the heterogeneity of their container port operations. Hence, this study applies a machine learning approach for clustering the container ports by similar infrastructure rather than by geographic region. The ports’ infrastructure was defined in terms of quay length, land area, draft length, and number of berths. Preliminary results from this study confirm that infrastructure heterogeneity can be addressed by utilizing an unsupervised machine learning algorithm. The results generated similar efficiency distributions between groups. These results are well aligned with the current challenges faced by decision makers and suggest benchmarking best practices with container ports that are similar in terms of port infrastructure.

Keywords: Machine learning, cluster analysis, bootstrap DEA, container port efficiency

1. Introduction

Container ports play a major role in global supply chains and are important economic drivers. Hence, it is critical to increase the operational efficiency of container ports located in developing countries given the significant impact in the developing countries’ economies. There are research studies aimed at estimating the operational efficiency in developing countries (Al-Eraqi et al., 2009; Suárez-Alemán et al., 2014; Julien et al., 2018; Hlali et al., 2023). Recently, Hlali et al. (2023) assessed the bias corrected operational efficiency of container ports in the North African, West African, Caribbean, South American, North American, and Central American developing regions. This research utilized a panel dataset of the ports, which corresponds to the period 2013–2018. An interesting finding is that the dynamism of the ports can be explained by their dynamism of operations rather than their infrastructure dynamism. Furthermore, most of the ports were operating in the increasing returns to scale zone. Hence, it is expected that a better benchmarking of the operational efficiency can be obtained when classifying the ports by their port infrastructure characteristics. Therefore, this research in progress proposes the use of an unsupervised machine algorithm for clustering the container ports by their port infrastructure before estimating their bias corrected efficiency scores. To the best of my knowledge, there is not any research project combining both rigorous approaches. Preliminary results confirm that this robust hybrid approach provides better results and supports a more effective benchmarking process, which will be of significant relevance for port decision makers.
2. Methodology and Empirical Analysis

This research utilizes a k-means approach for clustering 40 container ports based on their port infrastructure characteristics (Cabral et al., 2014). The ports’ infrastructure was defined in terms of quay length, land area, draft length, and number of berths. The sum of the squared distance between each point and the centroid in a cluster suggests using three clusters. Cluster 1 has four ports from North Africa only; clusters 2 and 3 have 16 and 20 ports from all five geographic regions, respectively.

A bootstrap DEA approach is applied to each cluster next (Hlali et al., 2023). It is expected that the bias corrected efficiency density between clusters is similar. Figure 1 shows the kernel distribution of the bias corrected Constant Returns to Scale (CRS) efficiency scores per cluster.

![Figure 1: Aggregate bootstrap CRS efficiency scores per cluster](image)

Figure 1 suggests the efficiency densities are similar. Hence, the Simar and Zelenyuk (2006) bootstrapped test is used to estimate the statistical significance of the bootstrap CRS score densities. The test p-values are given in Table 1. The null hypothesis is that two given clusters have similar score densities, whereas the alternative hypothesis states that they differ.

<table>
<thead>
<tr>
<th>Table 1. Simar &amp; Zelenyuk test for the CRS scores</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster 2</td>
</tr>
<tr>
<td>Cluster 1</td>
</tr>
<tr>
<td>Cluster 2</td>
</tr>
</tbody>
</table>

Table 1 confirms that the bootstrap efficiency densities are similar, which is expected by clustering the container ports based on their infrastructure. This result has immediate implications for benchmarking purposes. The decision maker should benchmark against ports with similar infrastructure despite their geographic location. This proposed approach enhances the analysis presented by Hlali et al. (2023) where seven out of ten pairs of regions differed in their efficiency densities, which was explained by the heterogeneity of the geographic regions and the dissimilarities of the ports’ infrastructure. This research in progress suggests that the port infrastructure diversity can be controlled by a machine learning algorithm that clusters the ports, allowing a fairer bootstrap DEA efficiency estimation.
3. Conclusions

The combined machine learning and bootstrap DEA approach provides a better opportunity for more significant benchmarking processes. Improvement opportunities are easier to identify and apply when comparing ports with similar infrastructure.

The bootstrap CRS score encompasses technical and scale efficiency scores. Hence, the next step is to analyze the bootstrap technical and scale efficiency individually. It is expected to have similar results where ports from the same cluster attain similar efficiency densities. Once these results are confirmed, the next step is to develop a supervised machine learning algorithm for estimating the operational efficiency based on the port infrastructure. This algorithm will assist decision makers in estimating the efficiency increase when implementing infrastructure improvements. Furthermore, it will allow us to identify the improvement contribution of each enhancement in the main infrastructure factors such as quay length, land area, draft length, and number of berths.

4. References


A Multi-Objective Approach for Assessing Environmental Responsibility in the Textile and Clothing Industry Supply Chain

EP. Mezatio\textsuperscript{1,2,**}, MM. Aghelinejad\textsuperscript{1,***}, L. Amodeo\textsuperscript{1,***}, and I. FERREIRA\textsuperscript{2,+}  

\textsuperscript{1} University of Technology of Troyes, 12 Rue Marie Curie, 10300 Troyes  
\textsuperscript{2} Institut français du textile et de l'habillement, 270 Rue du Faubourg Croncels, 10000 Troyes  
\texttt{ferreira@ifth.org}

1 Abstract  
Today, analyzing decision-makers’ environmental responsibility index is crucial as it enables us to assess their performance in terms of considering environmental issues in their decisions. This assessment can include criteria such as reducing carbon emissions, improving energy efficiency, managing waste, and utilizing sustainable materials. By evaluating decision-makers’ impact on the environmental performance of the supply chain, it becomes possible to identify areas for improvement and develop targeted strategies for promoting more responsible and sustainable management. This study presents a bi-objective mathematical model aiming to minimize both cost and carbon emissions. We used the weighted sum approach method to solve the model and assess the environmental responsibility index of decision-makers in the direct supply chain of the textile and clothing industry. It demonstrates the influence of different factors, such as cost and carbon emissions, on the supply chain configuration.

2 Introduction  
Since the 1950s, extensive research has been conducted on production decisions in manufacturing companies, yielding significant outcomes. However, most of these studies primarily focus on meeting customer needs while minimizing costs or maximizing revenues, without considering the environmental and energy impacts of production activities. The latest report from the Intergovernmental Panel on Climate Change (IPCC\textsuperscript{[1]}), published on March 22, 2023, warns that global warming is on track to exceed 1.5°C by the early 2030s compared to pre-industrial levels. Key sectors with high GHG emissions must be particularly targeted for significant emissions reductions. In this context, analyzing the environmental responsibility index of decision-makers in the supply chain and optimizing supply chain configuration becomes crucial. It is imperative to reevaluate production and supply chain management decisions by integrating environmental and energy considerations comprehensively.

In the field of sustainability and carbon emissions reduction in the supply chain, several studies have been conducted. Cui et al.\textsuperscript{[2]} emphasize the significance of sustainable supplier selection in managing sustainable supply chains. They propose a model validated through a case study and offer practical recommendations for selecting sustainable suppliers in diverse supply chain structures. Mezatio et al.\textsuperscript{[3]} integrate a carbon taxation approach into the supply chain and propose a single-objective mathematical model that minimizes overall costs while considering planning deadlines and horizons. However, this method lacks a concise assessment of the supply chain’s environmental responsibility and becomes less effective with very low carbon taxes. Naghi Beiranvand et al.\textsuperscript{[4]} present a conceptual model for assessing the performance of sustainable product and service supply chains (PSSCs) in the oil and gas industry. They conduct an extensive literature review to identify relevant indicators, and through exploratory and confirmatory factor analyses, they establish ten criteria for the model, including environmental, customer, financial, IT, social, risk, logistics, operational, organizational, and innovation and growth performance. In this paper, we propose a multi-objective model that incorporates the cost minimization objective from Mezatio et al.\textsuperscript{[3]} as the first objective, and a second objective of minimizing carbon emissions. This model employs the Weighted Sum method to find the optimal balance between the two objectives and enables the assessment of a supply chain’s environmental responsibility. It also facilitates the analysis of various supply chain configurations at each decision level.
3 Problem statement and mathematical modelling

The supply chain examined in this article, as shown in Figure (1) and based on Mezatio et al. ([3]), simultaneously integrates economic, environmental, and energy constraints. In the depicted figure, there are two primary flows in the supply chain model. Firstly, there is an information flow, represented by the numbers 1, 2, and 5 in the figure. They signify the flow of information regarding the production requirements or demands within the supply chain. Secondly, there are material flows represented by numbers 3, 4, 6, and 7 in the figure. These material flows represent the movement of physical goods or materials within the supply chain.

3.1 Mathematical modelling

The mathematical model consists of two objectives: The first objective, $f_1$, aims to minimize total costs in the supply chain, while the second objective, $f_2$, aims to minimize carbon emissions throughout the supply chain.

The model has eight indices: F (Factories), U (Subcontractors), S (Suppliers), R (Raw materials), P (Products), W (Warehouses), N (Retailers), and V (Transport modes). These indices represent different entities or components within the supply chain. Additionally, the model includes 52 parameters, and 26 decision variables.

The $f_1$ cost function is defined by:

\[
\text{minimize } f_1 = P_c + M_c + H_c + T_c
\]

Where, $P_c$ represents the procurement cost, $M_c$ represents the manufacturing cost, $H_c$ represents the holding cost, and $T_c$ represents the transportation cost.

The $f_2$ carbon emission function is defined by:

\[
\text{minimize } f_2 = P_{em} + M_{em} + T_{em}
\]

Where $P_{em}$ represent the procurement emission of the raw materials, $M_{em}$ represent the manufacturing emission, and $T_{em}$ represent the transportation emission.

3.2 Resolution and Experimentation

Resolution Method

The model presented is solved using the Weighted Sum method. The advantage of this method is that it allows us to better assess the degree of importance given to each of the objective. The aim of this method is to minimize the function:

\[
\text{minimize } f = \alpha \cdot f_1 + (1 - \alpha) \cdot f_2
\]

In the experiment conducted, the parameter $\alpha$ is used to determine the relative importance assigned to each objective function in the evaluation process. To explore the trade-off between cost and environmental impact, 10 values of $\alpha$ were considered, ranging from 0 to 1 with a step size of 0.1. As the magnitudes of the results for the cost function ($f_1$) and the environmental function ($f_2$) differ significantly, with $f_1$ typically in the million range and $f_2$ in the thousand range, the normalization technique used is Min-Max normalization, which scales the values of a feature between 0 and 1.
Experimentation

In the study, the test proposed by Mezatio et al. ([3]) was used. The instance is defined as follows: 8 suppliers (S=8), 3 factories (F=3), 4 warehouses (W=4), 8 customers (N=8), 9 raw materials (R=9), 8 different products (P=8), 9 subcontractors (U=9), and 3 transport modes (V=3). The customer demands for each product range between 100 and 500 units. The mathematical models were solved using the Cplex 20.1.0 solver on a Core i5 PC with a frequency of 2.3 GHz.

Table 1 in the paper presents the results regarding the increase in costs and overall emissions based on the variation of the weight parameter $\alpha$. The table provides insights into the trade-off between cost and emissions, showcasing the impact of different weightings given to these two objectives. In the comparison analysis, the paper considers the case where costs and emissions are minimized, serving as a baseline. This baseline scenario corresponds to $\alpha = 0$ for carbon emissions, indicating a focus solely on reducing emissions, and $\alpha = 1$ for overall costs, emphasizing cost minimization.

When $\alpha = 0$, decision-makers prioritize carbon emissions reduction, aiming to minimize the environmental impact of the supply chain without considering costs. Conversely, when $\alpha = 1$, the emphasis is solely on minimizing costs, with little consideration given to carbon emissions.

Table 2 showcases the different supply chain configurations obtained for each $\alpha$ value. In these configurations, the values of 1 represent the selected actors (such as suppliers, factories, subcontractors, and warehouses), while 0 indicates that the respective actor is not selected or not present in the configuration.

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suppliers</td>
<td>1 1 1 1 1 0</td>
<td>1 1 1 0 1 1 0</td>
<td>1 1 1 1 0 1 1 0</td>
<td>1 1 1 1 0 1 1 0</td>
<td>1 1 1 1 0 1 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subcontractor</td>
<td>0 0 0 0 1 0 0 0 0</td>
<td>0 0 0 0 1 0 0 0 0</td>
<td>0 0 0 0 1 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Factories</td>
<td>1 0 1</td>
<td>1 0 1</td>
<td>1 0 1</td>
<td>1 0 1</td>
<td>1 0 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Warehouse</td>
<td>1 1 1</td>
<td>1 1 1</td>
<td>1 1 1</td>
<td>1 1 1</td>
<td>1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4 Conclusion

The paper introduces an approach for evaluating the responsibility index of a supply chain, taking into account both cost and carbon emissions. Table 1 presents the results, demonstrating the trade-off between cost and emissions when the weight parameter $\alpha$ is varied. For instance, when $\alpha = 0.5$, indicating equal importance given to cost and emissions, the results indicate that costs can increase by nearly 10% compared to the minimum cost, while emissions only increase by 4% compared to the minimum emission. Table 2 showcases the different configurations obtained for various supplier, factory, subcontractor, and warehouse choices, considering different values of $\alpha$.

The number of configurations varies depending on the decision level, with four configurations for supplier choice, two configurations for factory choice, two configurations for subcontractor choice, and a single configuration for warehouse choice across different $\alpha$ values. Future studies will focus on assessing the cost associated with different supply chain configurations and exploring variations in the supply chain structure. We will experiment with new approaches using the pareto front and myticriteria analysis methods.

References

Learning Insertion Patterns to Enhance Operational Efficiency in Large-Scale Dial-a-Ride Systems

Chijia LIU, Alain QUILLOT, Hélène TOUSSAINT, and Dominique FEILLET

1 Université Clermont Auvergne, CNRS, UMR 6158 LIMOS, 63178 Aubière France
{chijia.liu, alain.quilliot, helene.toussaint}@uca.fr
2 Mines Saint-Étienne, Univ Clermont Auvergne, INP Clermont Auvergne, CNRS, UMR 6158 LIMOS, F - 42023 Saint-Etienne France
feillet@emse.fr

Abstract. We study a large-scale dial-a-ride system considering around 300,000 dynamic requests. An efficient routing algorithm is crucial to guaranteeing the viability of the system. Large-scale requests are assumed to be dominated by daily commuting needs and thus should exhibit similar mobility patterns from one day to another. Consequently, daily vehicle trajectories should also be recurring if similar requests can be served in the same manner. We introduce a greedy insertion algorithm integrating a Guided Insertion Mechanism that learns insertion patterns from reference resolution to enhance the operational efficiency of the underlined systems while maintaining high-quality solutions.

Keywords: Large-Scale Dial-A-Ride · On-Demand Transportation System · Ride-Sharing.

1 Introduction

In recent years, we have witnessed the emergence of novel transportation systems that offer efficient and sustainable alternatives to traditional modes of transportation. In this paper, we consider a prospective Dial-A-Ride (DAR) system where a mid-capacity vehicle fleet offers services to a very large volume of passengers (around 300,000 per day), catering to the transportation needs of a vast user base. In such a system, ride-sharing is allowed, so that passengers can share common journeys in the same vehicle. The system operates in a dynamic context, where user requests are submitted and processed immediately. To ensure the viability of the system, we need to implement a routing algorithm that is both efficient and capable of delivering high-quality solutions. We address a dynamic large-scale dial-a-ride problem. Due to the very large-scale aspect, we rely on the classic greedy insertion heuristic for its simplicity and effectiveness. Furthermore, we consider that large-scale requests should be dominated by daily commute needs which exhibit recurring characteristics. Therefore, daily requests...
should be similar and repetitive, and we should be able to recreate similar travel patterns in vehicle trajectories. Based on this idea, as our main contribution to this work, we propose a two-phase algorithmic framework that involves a Guided Insertion Mechanism (GIM). GIM learns the insertion patterns of requests from representative reference solutions to enhance the operational viability and efficiency of the routing process in such systems, while maintaining a high solution quality.

In Section 2, we introduce some related works in the literature. Section 3 defines the problem. In Section 4, the algorithmic framework is presented. Then, in Section 5, we detail the GIM. Experimental results are presented and discussed in Section 6 and we conclude in Section 7.

2 Related Works

Within the context of DAR transportation systems, we naturally turn to the Dial-A-Ride Problem (DARP) for formulation. As a variation of the Vehicle Routing Problem (VRP), DARP is NP-hard. Very few studies seek to solve the DARP using exact optimization methods, unless for solving small-sized static problems [3, 11]. More studies tend to propose approximate methods capable of solving larger instances [4, 2, 6].

On the other hand, on-demand DAR transportation systems offer personalized, efficient services via digital platforms and shared mobility. They cater to urban transportation needs, with studies focused on, for example, managing congestion [9], addressing electric vehicle recharging challenges [1], and exploring ride-sharing benefits [5]. Rapid user request processing is critical in large-scale systems, making it crucial to expedite the routing and scheduling processes. For instance, [12] examines a taxi system handling over 330,000 requests, introducing a dual-side taxi searching algorithm to rapidly retrieve plausible (nearest) vehicles to service a target request. Similarly, [10] proposes a filtering system that not only provides candidate vehicles but corresponding candidate insertion positions within the routes as well.

In large-scale scenarios, daily travel requests should follow repetitive patterns, particularly those coming from regular commuting needs. Capturing these mobility patterns using machine learning or deep learning approaches is widely explored [14, 15]. Meanwhile, vehicle trajectories supporting these requests should also exhibit regular travel patterns. Studies such as [8, 7] analyze spatial and temporal travel patterns in the underlined traffic networks, utilizing, for instance, graph-based approaches, trajectory clustering, etc. Our paper assumes the availability of a representative reference request instance in our DAR system, bypassing the need to extract historical travel patterns. We introduce a Guided Insertion Mechanism (GIM) that accelerates the resolution of large-scale dynamic DARs by learning the patterns of requests from vehicle travel patterns. These patterns, derived from the solution to the related reference problem, guide the insertion of requests on-the-fly. To the best of our knowledge, we are the first to consider the correlation between vehicle travel patterns and the resolution of
DARP. Furthermore, we present a formal representation of these vehicle travel patterns, offering an efficient framework for resolving DARP.

3 Problem Statement

In this section, we define our Large-Scale Dial-A-Ride Problem (LSDARP).

We consider a transit network $G = (N, A)$, where $N$ contains all the intersections, and $A$ contains all the arcs in the network. The graph contains only one depot, $n_0 \in N$. We use $t(u, v)$ to denote the shortest travel time from any node $u$ to any node $v$ in the network.

A request $r \in R$ is submitted at time $t_{sub}^r$, while specifying a pickup service $O^r$, including an origin $o^r \in N$ and a pickup time window $[e_{O^r}, l_{O^r}]$, and a drop-off service, including a destination $d^r \in N$ and a maximum ride time $T^r$. We note that service times are not considered, each request involves one passenger, and all requests are supposed to be feasible and non-preemptive, which means that each request must be fulfilled exactly once by exactly one vehicle.

Passengers are serviced by a fleet $V$ of vehicles of capacity $Q$. A route $\theta^v \in \Theta$ followed by a vehicle $v$ is a list of key points $K$ that aggregates services happening at the same location at the same time. Typically, a key point $K$ contains: $n_K \in N$, the location of the service; $q^v_K$, the load of $v$ before departing from $n_K$; $R^o_K$, the list of requests scheduled to get onboard at $K$; $R^d_K$, the list of requests scheduled to get off at $K$; $[e^o_K, l^o_K]$, the arrival time window at $K$; and $[e^d_K, l^d_K]$, the departure time window from $K$. For any request $r$ assigned to $v$, we use $K(O^r) \in \theta^v$ (resp. $K(D^r)$) to denote the key point where $O^r$ (resp. $D^r$) is inserted.

Vehicle routes must start and end at the depot $n_0$, and have a load that never exceeds $Q$. And for every request $r$ assigned to the vehicle, $O^r$ precedes $D^r$, and the schedule must not violate the pickup time window and the maximum ride time constraints. When providing services, $v$ follows the earliest arrival time $e^o_K$ and earliest departure time $e^d_K$.

We consider a lexicographic objective function. We assume that the number of vehicles is unlimited, so minimizing the fleet size is the primary objective. The total drive time of vehicles is considered the second criterion.

4 Algorithm Framework

We define a set of decision epochs $E$. Each decision epoch lasts $I_e$ (for example, $I_e = 10$ min) time units. For each decision epoch $E \in E$,

1. the system proceeds with the insertion of requests submitted during the previous epoch;
2. once all requests are inserted, the system updates the vehicle schedules and informs passengers of the inserted requests about the updated information about their pickup (e.g. the vehicle’s passage time);
3. after the update, vehicles start following the new routes until the next update.

Regarding the very large problem size and the need to make decisions on-the-fly, we propose a two-phase resolution framework to be applied during each decision epoch $E$ based on the greedy insertion heuristic. Specifically, given $R_E$ containing all the requests to be inserted and the current route collection $\Theta = \{\theta^v, v \in V\}$:

- **Phase 1**: A Guided Insertion Mechanism (GIM) is invoked to proceed with rapid insertions of all requests in $R_E$. For any $r \in R_E$, if a feasible insertion is found, we keep it;

- **Phase 2**: For all the requests that failed to be inserted via GIM, we apply the best-fit insertion heuristic based on a Filtering System, meaning that $r$ should be inserted while minimizing the additional detour. For any request $r$, if the best-fit insertion failed, we activate a new vehicle $v$ to service it and add $\theta^v$ to the current route set $\Theta$.

About the framework, we shall make some clarifications on the notion of insertion of a request $r$, the GIM, and the Filtering System.

Regarding the notion of insertion, let us introduce the procedure to insert $r$ into the route $\theta^v$ regarding a given insertion parameters, $(\theta^v, K^o, K^d)$. We first insert the pickup service $O^r$. If $n_{O^r} = n_{K^o}$, we aggregate $O^r$ to $K^o$ and $K(O^r) = K^o$; otherwise, a new key point $K(O^r)$ supporting $O^r$ is inserted between $K^o$ and its successive key point. Then the load $q_{K(O^r)}$, the inbound request list $R^+_K(O^r)$, and the pickup and drop-off time windows on $K(O^r)$ are updated regarding the constraints about the vehicle and requests mentioned in Section 3. The same rules applied for the insertion of the drop-off service $D^r$ at $K^d$ while considering the maximum ride time $T^r$. Once $O^r$ and $D^r$ are inserted, we increase the load of key points between $K(O^r)$ and $K(D^r)$ by 1 while ensuring that the new loads never exceed the vehicle capacity $Q$. Finally, we need to update the time windows of the key points along $\theta^v$ and check that these time windows are always feasible. For that, we implement a classic constraint propagation procedure [13] considering the above time constraints. This procedure has a complexity of $O(|\theta^v|^2)$, where $|\theta^v|$ is the number of key points in $\theta^v$.

Regarding the mechanism GIM, it is introduced to be applied upstream of the best-fit insertion. GIM aims to rapidly and wisely select well-fitted insertion parameters under the guidance of a set of insertion patterns learned from some representative reference instances and the corresponding routing solutions. More intuition and details about GIM will be presented in Section 5.

Regarding the Filtering System, we implement the methodologies presented in [10] to effectively reduce the search space of the best-fit insertion. Generally speaking, when inserting a request $r$, the Filtering System first utilizes a spatial-temporal indexed vehicle filtering matrix to rapidly retrieve a set of potential vehicles capable of simultaneously visiting the spatial-temporal neighborhoods of $O^r$ and $D^r$ while respecting the time window constraints of routes. Then, for each
Learning Insertion Patterns for Efficient Large-Scale DAR Systems

A candidate vehicle $v$, a spatial-temporal indexed insertion position filtering matrix is used to provide all the plausible insertion positions within $\Theta^v$ for $O^v$ and $D^v$. Then, the time-consuming insertion feasibility tests are only conducted on this much smaller search space. To further accelerate the process, we can integrate a stopping mechanism. Selected candidate vehicles are sorted and explored based on the descending order of their insertion score. The score of $v$ estimates its possibility of resulting in a feasible insertion. The stopping mechanism keeps track of the computational effort spent during the exploration process and stops it once the effort reaches a pre-defined threshold, and we should insert $r$ based on the best-fitted insertion parameters found so far.

Two algorithms can be derived from our two-phase framework: BIM-PFS (Partial Filtering System) and BIM-FS (Filtering System). The former lacks the stopping mechanism and aims for better routing results, while the latter includes it to improve the efficiency.

5 The Guided Insertion Mechanism

In this section, we describe the Guided Insertion Mechanism (GIM).

Let us use $\text{LSDARP}(\mathcal{R})$ to denote the problem with an input instance $\mathcal{R}$. Considering two similar instances, $\mathcal{R}_1$ and $\mathcal{R}_2$, where most requests in $\mathcal{R}_1$ have similar counterparts in $\mathcal{R}_2$, we believe that the optimal routing solution $\Theta_1$ for $\text{LSDARP}(\mathcal{R}_1)$ should resemble $\Theta_2$, the optimal solution for $\text{LSDARP}(\mathcal{R}_2)$. Because if $r_1 \in \mathcal{R}_1$ and $r_2 \in \mathcal{R}_2$ are similar, they should be able to be inserted in a similar manner. GIM is conceived based on this idea. Due to the large-scale nature and prevalence of recurring daily commute mobility, we assume that daily requests processed in DAR systems exhibit similarities. Consequently, the travel patterns of vehicles are also anticipated to be similar from one day to the next.

Supposing that we have a representative reference set $\bar{\mathcal{R}}$ capturing the basic distribution (origin and destination and pickup times) of requests, the static (i.e., off-line) optimal solution $\bar{\Theta}$ to the problem $\text{LSDARP}(\bar{\mathcal{R}})$ should be able to guide any dynamic (i.e., on-line) resolution of any real problem $\text{LSDARP}(\mathcal{R})$, where $\mathcal{R}$ is the set of real dynamic requests to be processed.

Extracting mobility patterns of requests and optimally solving the associated static problem are two distinct challenges. Instead, this paper does not address neither of them and assumes that a representative enough reference set $\bar{\mathcal{R}}$ is in our possession, along with its off-line high-quality solution $\bar{\Theta}$. Our focus is how to learn from the references $(\bar{\mathcal{R}}, \bar{\Theta})$ the insertion patterns to guide the insertion process when solving a similar real dynamic problem $\text{LSDARP}(\mathcal{R})$.

5.1 Preprocessing: Obtain Vehicle Travel Patterns

For each reference route $\bar{\theta} \in \bar{\Theta}$, we compute a specific travel pattern $\gamma(\bar{\theta}) \in \Gamma$. A travel pattern $\gamma(\theta)$ is a simplified route defined as a list of pattern points. A pattern point $\bar{P}$ represents a cluster of key points in $\theta$. The notion of key point cluster is defined as follows:
Definition 1 (key point cluster). Given a route $\bar{\theta} = \{\bar{K}_0, \ldots, \bar{K}_i, \ldots, \bar{K}_{M-1}\}$, $\{\bar{K}_i, \bar{K}_{i+1}, \ldots, \bar{K}_{i+m}\}$ is a cluster if and only if:

- $t(\bar{K}_j, \bar{K}_{j+1}) \leq \delta^K$, for $i \leq j \leq i + m - 1$,
- $t(\bar{K}_{i-1}, \bar{K}_i) > \delta^K$,
- $t(\bar{K}_{i+m}, \bar{K}_{i+m+1}) > \delta^K$,

where $\delta^K$ defines the maximum travel time from a key point to its successor allowed in the same cluster.

A temporal neighborhood $[t_{\min}^P, t_{\max}^P]$ for any $\bar{P}$ can be derived from the corresponding cluster, defined by the maximum spanning time of the cluster. For example, in Fig. 1, a pattern point $\bar{P}_2 \in \gamma(\bar{\theta})$ represents the key point cluster $\{\bar{K}_2, \bar{K}_3, \bar{K}_4\}$. And its temporal neighborhood is defined by $[v_{\bar{K}_2}^o, b_{\bar{K}_4}^i]$.

Fig. 1. Illustration of the relationships between the reference route, the travel pattern, and the real route

In GIM, one travel pattern is related to at most one real route. And for $\gamma \in \Gamma$ related to $\theta \in \Theta$, each pattern point $\bar{P}$ is associated with a subset of consecutive real key points in $\theta$ whose passage time windows exhibit an overlap with $[t_{\min}^P, t_{\max}^P]$. These real key points are called the children of $P$. We define two pointers, $lcp$ and $rcp$, where $lcp$ points at its left-most child, and $rcp$ points at its right-most child. For example, in Fig. 1, the children of $P_2$ are $K_1$, $K_2$ and $K_3$.

5.2 Guided Insertion Process

Given a real request $r$, GIM operates according to the following steps.

Step 1: Retrieve similar reference requests We first identify from $\bar{R}$ all the reference requests $\bar{r}$ that are similar to $r$. The similarity between requests is defined as follows:
Definition 2 (similarity between requests). Two requests $r_1$ and $r_2$ are similar if and only if $t(o_1^1, d_2^1) \leq \delta^s$, $t(d_1^1, d_2^1) \leq \delta^s$, and $|e_{O_1} - e_{O_2}| \leq \delta^t$, where $\delta^s$ is the spatial threshold measuring the maximum distance (travel time) between two locations, and $\delta^t$ is the temporal threshold defining the maximum difference in the earliest pickup time between two similar requests.

If $r_1$ and $r_2$ are similar, we use $|e_{O_1} - e_{O_2}|$ to quantify their similarity. Let $\mathcal{R}'$ denote the set of retrieved similar reference requests. The set $\mathcal{R}'$ is sorted based on the descending order of their similarity values with $r$.

Step 2: Construct guide object set. We construct a set of guide objects $GO^r$. A guide object is a triplet $(\gamma, P^o, P^d)$ used to guide the insertion of $r$, where $P^o$ and $P^d$ are pattern points in the travel pattern $\gamma$. As illustrated in Figure 1, a reference request $\bar{r}$ inserted in $\theta$ at $K(O')$ and $K(D')$ corresponds to the guide object $(\gamma(\theta), P^o, P^d)$. Consequently, $GO'$ contains all the distinct guide objects of reference requests in $\mathcal{R}'$ in alignment with the order specified in $\mathcal{R}'$.

Step 3: Insert $r$ according to the guide object. Given $(\gamma, P^o, P^d) \in GO^r$, we use $P^o$ and $P^d$ to guide the insertion of $O'$ and $D'$, respectively. If $\gamma$ has not been related to any real route in $\Theta$, then a vehicle $v$ is activated to serve $r$, and we relate its route $\theta^v$ to $\gamma$ by correctly setting the pointers $lc_P$ and $rc_P$ of all $P \in \gamma$. If $\gamma$ is already related to a route $\theta \in \Theta$, then all the key points between the two key points pointed by $lc_{P^o}$ and $rc_{P^o}$ are considered as candidate insertion positions for $O'$. We do the same to identify candidate insertion positions for $D'$. A best-fit scheme is then implemented while trying the insertion feasibility of $r$ at the selected candidate key points. This means that if at least one feasible insertion is found, we keep the candidates that minimize the insertion cost.

We note that when inserting $r$, GIM encourages replicating the insertion pattern of its most similar reference request. A reference request is more similar to $r$ when its guide object is positioned on top of $GO'$. Therefore, we implement a first-fit scheme to explore $GO'$. If $r$ can be inserted under the guidance of $(\gamma, P^o, P^d)$, we proceed with the insertion, stop the exploration of $GO'$, update the pointers $lc_P$ and $rc_P$ of pattern points $P$ along $\gamma$, and continue to insert the next request. Otherwise, we explore the next guide object in $GO'$.

6 Numerical Experiments

We programmed the algorithms in C++ language and solved the problem on a 512 GB RAM machine with an AMD EPYC 7452 32-Core Processor.

We take the transit network of the city of Clermont-Ferrand, France, and its peri-areas. The selected area contains 13,839 nodes and 31,357 arcs. Among all nodes, 1,469 are valid pickup and drop-off locations.

The tested instances in this paper are self-generated. The number of requests in each instance is set at 300,000. For any request, the length of the pickup
window is 15 minutes, and the maximum ride time $T_r$ is twice the shortest travel time from $o$ to $d$.

The service period lasts $T = 24$ hours, from 00:00 to 24:00. In order to simulate the system’s intended use cases, i.e., catering to various travel requests that are dominated by daily commute demands, we define the following basic request distribution, which intentionally creates two peak periods: 06:00 ~ 10:00, and 15:00 ~ 19:00. Each of the peak periods contains 35% of the requests generated. During the morning peak, around 50% are typical that travel from a residential location to a working location. And the evening one includes 50% of requests moving from a working location to a residential location. For requests departing at other time periods, their origin and destination are randomly distributed over the network.

A reference request set $\overline{R}$ is randomly generated based on the basic distribution, and the corresponding reference solution $\overline{Θ}$ containing 1,621 vehicles is obtained with a best-fit insertion heuristic. Real request instances can be decomposed into two parts: the “random” part and the “similar” part. Requests in the “random” part are randomly generated using the above-defined basic distribution. The “similar” part simulates the stable and recurring pattern of daily requests. When generating a request $r$ in this part, we randomly select a reference $\overline{r}$ from $\overline{R}$. Then $o'$ (resp. $d'$) is randomly selected among the nodes that are reachable within 3 arcs from $o$ (resp. $d$). The earliest pickup time $e_{O\overline{r}}$ is a random value selected between $e_{O\overline{r}} - 7.5$ minutes and $e_{O\overline{r}} + 7.5$ minutes, with 7.5 being half the length of the pickup time window. In addition, the submission time of $t_{sub}$ of any $r$ is randomly set between 0 and $e_{O\overline{r}}$. Requests are submitted on average one hour before $e_{O\overline{r}}$. The system being prospective, the similarity pattern in the daily requests is unclear. We consider three scenarios: high, moderate and low, where the “similar” part respectively accounts for 90%, 50%, and 30% of the requests generated.

The length of each decision epoch $e$ lasts $I_e = 10$ minutes. The GIM parameters $\delta_K$ (key point cluster threshold), $\delta^c$ and $\delta^t$ (request similarity thresholds) are fixed at 2 minutes, 2 minutes, and 15 minutes, respectively. For the sake of comparison, in addition to the two approaches proposed in this paper, GIM-PFS and GIM-FS, we also consider an approach GIM-BF, where a Best-Fit insertion heuristic over the whole search space is invoked after GIM. Additionally, their corresponding approaches without GIM, BF, PFS and FS are considered as baselines. We note that in FS, parameters of the stopping mechanism are selected so that around the top 10% best vehicles from the candidate pool are kept. This number is also bounded between 40 and 140.

### 6.1 General Analyses of the Effectiveness of Algorithms Under the Basic Configuration

Let us begin by analyzing the effectiveness of the approaches with GIM proposed in this paper in comparison to the corresponding baseline approaches.

Table 1 summarizes the general results when solving the LSDARP using different approaches. For any scenario, in terms of the global processing time, all
the approaches integrated with **GIM** effectively outperform **BF** as well as their corresponding baseline approaches. The most efficient method **GIM-FS** exhibits a drastic reduction of more than 91% compared to the CPU time used in **BF**. In the meantime, it is observed that methods employing **GIM** successfully maintain the fleet size at a nearly consistent level compared to their respective baseline approaches. Additionally, thanks to the additional guide of the reference solution which helps better inserting similar requests, the most efficient method **GIM-FS** merely increases the fleet size by about 10%, surpassing its baseline approach **FS** whose search space is much smaller compared to other methods. We observe a similar effect regarding the performance to the total drive time of vehicles: **GIM-BF** and **GIM-PFS** keep the values at the same level as the baselines, and **GIM-FS** outperforms **FS**. Moreover, in terms of travel comfort, the average in-vehicle times of passengers when using **GIM** are almost similar, even slightly better, to the baseline results, demonstrating less detour in the routes that are better organized under the guide of **GIM**. Finally, across different scenarios, regarding the fleet size, total drive time and average in-vehicle time, we see a slight improvement in highly similar scenarios (high and moderate) compared to low, but the difference is not big. In terms of the CPU time, however, we notice that the reduction becomes more pronounced when the instance contains more similar requests. Because this leads to a higher success rate of insertions using **GIM**. From now on, we only focus on the scenario high.

**Table 1.** Results with different approaches under different similarity scenarios (The relative variations are computed compared to the baseline approach **BF**)

<table>
<thead>
<tr>
<th>scenario</th>
<th>approach</th>
<th>CPU time (min)</th>
<th>fleet size</th>
<th>total drive time (h)</th>
<th>average in-vehicle time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>high</td>
<td>BF</td>
<td>176.7</td>
<td>2,183</td>
<td>25,717.5</td>
<td>16.9</td>
</tr>
<tr>
<td></td>
<td>PFS</td>
<td>120.4 (-31.9%)</td>
<td>2,183 (+0.0%)</td>
<td>+0.0%</td>
<td>16.9</td>
</tr>
<tr>
<td></td>
<td>FS</td>
<td>17.9 (-89.9%)</td>
<td>2,551 (+16.9%)</td>
<td>+22.7%</td>
<td>16.7</td>
</tr>
<tr>
<td></td>
<td>GIM-BF</td>
<td>115.8 (-34.5%)</td>
<td>2,224 (+1.9%)</td>
<td>+0.6%</td>
<td>16.6</td>
</tr>
<tr>
<td></td>
<td>GIM-PFS</td>
<td>81.6 (-53.8%)</td>
<td>2,221 (+1.7%)</td>
<td>+0.6%</td>
<td>16.6</td>
</tr>
<tr>
<td></td>
<td>GIM-FS</td>
<td>14.3 (-91.9%)</td>
<td>2,390 (+9.5%)</td>
<td>+17.8%</td>
<td>16.6</td>
</tr>
<tr>
<td>moderate</td>
<td>BF</td>
<td>168.4</td>
<td>2,191</td>
<td></td>
<td>17.0</td>
</tr>
<tr>
<td></td>
<td>PFS</td>
<td>118.0 (-29.9%)</td>
<td>2,176 (-0.7%)</td>
<td>+0.1%</td>
<td>17.0</td>
</tr>
<tr>
<td></td>
<td>FS</td>
<td>17.0 (-89.9%)</td>
<td>2,548 (+16.3%)</td>
<td>+23.0%</td>
<td>16.8</td>
</tr>
<tr>
<td></td>
<td>GIM-BF</td>
<td>115.8 (-31.2%)</td>
<td>2,224 (+1.5%)</td>
<td>+0.7%</td>
<td>16.7</td>
</tr>
<tr>
<td></td>
<td>GIM-PFS</td>
<td>81.6 (-51.5%)</td>
<td>2,220 (+1.3%)</td>
<td>+0.6%</td>
<td>16.7</td>
</tr>
<tr>
<td></td>
<td>GIM-FS</td>
<td>14.3 (-91.5%)</td>
<td>2,413 (+10.1%)</td>
<td>+19.3%</td>
<td>16.6</td>
</tr>
<tr>
<td>low</td>
<td>BF</td>
<td>163.2</td>
<td>2,159</td>
<td></td>
<td>17.1</td>
</tr>
<tr>
<td></td>
<td>PFS</td>
<td>114.8 (-29.7%)</td>
<td>2,151 (-0.4%)</td>
<td>-0.1%</td>
<td>17.0</td>
</tr>
<tr>
<td></td>
<td>FS</td>
<td>17.4 (-89.3%)</td>
<td>2,518 (+16.6%)</td>
<td>+23.1%</td>
<td>16.8</td>
</tr>
<tr>
<td></td>
<td>GIM-BF</td>
<td>117.8 (-27.8%)</td>
<td>2,207 (+2.2%)</td>
<td>+0.7%</td>
<td>16.7</td>
</tr>
<tr>
<td></td>
<td>GIM-PFS</td>
<td>83.9 (-48.6%)</td>
<td>2,205 (+2.1%)</td>
<td>+0.6%</td>
<td>16.7</td>
</tr>
<tr>
<td></td>
<td>GIM-FS</td>
<td>14.6 (-91.1%)</td>
<td>2,413 (+11.8%)</td>
<td>+19.9%</td>
<td>16.7</td>
</tr>
</tbody>
</table>
Fig. 2 further proves the strength of GIM in reducing the processing time. It is shown that apart from the first few decision epochs, all methods with GIM are more efficient compared to their baseline method. The two surges in the execution time in the figure correspond to the two peak periods, where more requests are submitted and processed. We see that during these epochs, the advantage of using GIM becomes more pronounced. This is highly important in real-life dynamic scenarios, where in each decision epoch, the initial routing decisions should be made as quickly as possible, in order to leave temporal margin for time-consuming communications between the system, vehicles and customers, and the management of stochastic situations like no-shows or unexpected traffic conditions.

6.2 Analyses of the Sensibility

In this section, we analyse the sensibility of approaches with GIM to its parameters $\delta^s$ and $\delta^t$, while focusing on the results of global processing time, the final fleet size, and the proportion of requests inserted via GIM among all the requests (see Column $\text{succGI}$).

According to Table 2, when the request similarity criteria become more relaxed (i.e., bigger values of $\delta^s$ and $\delta^t$), the reduction in the CPU time becomes more evident. Because for both $\delta^s$ and $\delta^t$, when their value becomes bigger, it is easier for a request $r$ to be matched with other reference requests, which promotes the insertions via GIM, leading to a higher successful guided insertion rate. On the other hand, in terms of the fleet size, more vehicles are utilized when the values of $\delta^s$ and $\delta^t$ get bigger. Because in these situations, we may force a request $r$ to be inserted via GIM under the guide of some $\bar{r}$, whereas $r$ and $\bar{r}$ are not very similar. This will not only deprive the opportunity of $r$ to be inserted into other better-fitted vehicles via the normal best-fit insertion
process, but also perturb the original travel pattern of the vehicle in question, both of which worsen the quality of the solution.

Table 2. General results with different values of GIM parameters (The relative variations are computed compared to the results of each respective baseline approach without GIM)

<table>
<thead>
<tr>
<th>δ^s</th>
<th>δ^t</th>
<th>approach</th>
<th>CPU time</th>
<th>fleet size</th>
<th>succGI</th>
</tr>
</thead>
<tbody>
<tr>
<td>2min</td>
<td>15min</td>
<td>GIM-BF</td>
<td>115.8 (-34.5%)</td>
<td>2,224 (+1.9%)</td>
<td>37.3%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GIM-PFS</td>
<td>81.6 (-32.2%)</td>
<td>2,221 (+1.7%)</td>
<td>37.4%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GIM-FS</td>
<td>14.3 (-20.1%)</td>
<td>2,390 (-6.3%)</td>
<td>37.8%</td>
</tr>
<tr>
<td>2min</td>
<td>3min</td>
<td>GIM-BF</td>
<td>148.5 (-16.0%)</td>
<td>2,155 (-1.3%)</td>
<td>20.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GIM-PFS</td>
<td>105.9 (-12.0%)</td>
<td>2,160 (-1.1%)</td>
<td>19.8%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GIM-FS</td>
<td>14.0 (-21.8%)</td>
<td>2,428 (-4.8%)</td>
<td>20.0%</td>
</tr>
<tr>
<td>2min</td>
<td>7.5min</td>
<td>GIM-BF</td>
<td>124.9 (-29.3%)</td>
<td>2,193 (+0.3%)</td>
<td>31.7%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GIM-PFS</td>
<td>88.9 (-26.3%)</td>
<td>2,197 (+0.6%)</td>
<td>31.5%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GIM-FS</td>
<td>14.8 (-17.3%)</td>
<td>2,394 (-6.2%)</td>
<td>31.9%</td>
</tr>
<tr>
<td>0.5min</td>
<td>15min</td>
<td>GIM-BF</td>
<td>167.3 (-51.3%)</td>
<td>2,117 (-3.0%)</td>
<td>11.8%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GIM-PFS</td>
<td>118.6 (-15.5%)</td>
<td>2,115 (-3.1%)</td>
<td>11.9%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GIM-FS</td>
<td>14.7 (-17.9%)</td>
<td>2,429 (-4.8%)</td>
<td>11.8%</td>
</tr>
<tr>
<td>5min</td>
<td>15min</td>
<td>GIM-BF</td>
<td>84.7 (-52.1%)</td>
<td>2,399 (+9.9%)</td>
<td>46.2%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GIM-PFS</td>
<td>54.2 (-55.0%)</td>
<td>2,404 (+10.1%)</td>
<td>46.2%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GIM-FS</td>
<td>11.8 (-34.1%)</td>
<td>2,616 (+2.5%)</td>
<td>47.5%</td>
</tr>
</tbody>
</table>

7 Conclusion

We introduce a two-phase algorithmic framework involving a GIM upstream of a best-fit Filtering Process. The experiment results show that by learning and imitating a reference static solution, our approach hugely reduces the execution time while exhibiting almost the same performance as the baseline approaches by encouraging the fleet to follow well-organized vehicle travel patterns. We firmly believe that with a more representative reference problem and a more optimal reference solution, GIM should emerge as a highly effective algorithm that is especially suitable for addressing dynamic online problems. GIM is currently in its early stages as an emerging technology. Our upcoming focus aims to enhance its performance by maximizing the proportion of successful guided insertions within each epoch, while minimizing the perturbation of the pre-defined travel patterns.

8 Acknowledgement

This work was supported by the International Research Center "Innovation Transportation and Production Systems" of the I-SITE CAP 20-25.
References

Flexible Routing and Optimization for the Last Mile: An Approach Based on Smart Lockers and Heuristic Techniques

Mohamed-Ali EJJANFI¹ and Jamal BENHRA²

¹² LARILE, ENSEM, University Hassan II of Casablanca, Morocco
¹ejjanfimed@gmail.com
²jamalbenhra@gmail.com

Abstract. This research paper presents a comprehensive investigation into the development and application of flexible routing strategies to address the Vehicle Routing Problem (VRP) in last-mile delivery scenarios. The study, uniquely, considers the constraint of parcel locker unavailability, an aspect often overlooked in previous research but integral to the optimization of modern urban logistics systems.

The research leverages advanced heuristic methods including Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Simulated Annealing (SA) to tackle these complex combinatorial problems. By harnessing the strengths of each method and engineering them to adaptively respond to the dynamic nature of last-mile delivery, the study develops innovative strategies that efficiently solve the VRP under this novel constraint.

The findings demonstrate promising routes towards improving logistical efficiency and reducing delivery times, therefore enhancing the overall customer experience in the rapidly evolving e-commerce landscape. Further, the research lays the groundwork for future studies addressing dynamic and stochastic elements in last-mile delivery routing problems, opening new avenues for exploration in this critical field of logistics and supply chain management.

Keywords: Routing optimization, VRP, Genetic Algorithms, Ant Colony Optimization, Particle Swarm Optimization, Simulated Annealing, smart lockers, last-mile delivery logistics, flexible routing

1 Introduction

The explosion The rise of e-commerce has dramatically transformed the logistics and supply chain sector, particularly emphasizing the importance of 'last mile delivery'. This phase, where goods are transferred from distribution centers to the final consumer, poses intricate logistical challenges. Central to these challenges is the Vehicle Routing Problem (VRP), a pivotal component in optimizing logistics operations under various constraints.
The VRP represents an optimization problem central to designing the most efficient delivery routes, especially when facing constraints such as the absence of parcel lockers. This study addresses these complexities using heuristic methods, notably Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Simulated Annealing (SA). These methods are chosen for their proven efficacy in solving complex optimization problems. For instance, GA is versatile in optimizing routes with extensive problem spaces, as shown by Chunhua et al. [10]. ACO, following the insights from Negulescu et al. [12], excels in managing multiple constraint scenarios like vehicle route allocation, thus ensuring optimal resource utilization. PSO, as highlighted by Akhtar et al. [13], is noted for its effectiveness in constraint-laden contexts, enhancing the adaptability of the heuristic approach.

This research compares the performance of these algorithms based on key performance indicators (KPIs), such as total tour distance (in kilometers) and execution time (in seconds). The objective is to offer valuable insights for practitioners and researchers in logistics and supply chain management, aiding them in selecting and employing suitable heuristic methods for specific operational challenges.

Our analysis not only deepens the understanding of these heuristic algorithms in the context of last mile delivery but also contributes to the ongoing search for innovative solutions to increasingly complex logistics issues in the digital era. This work aims to enhance operational efficiency and customer satisfaction in the rapidly evolving e-commerce sector.

2 State of art

Flexible routing optimization in the Traveling Salesman Problem (TSP) and Vehicle Routing Problem (VRP) is at the heart of addressing complex challenges encountered in logistics, distribution, and smart cities. By exploring a multitude of methodologies such as integrating production and transportation activities, enhancing genetic algorithm crossover methods for TSP, developing variations of Particle Swarm Optimization (PSO) for specific routing problems like Vehicle Routing Problem with Time Windows (VRPTW) and Time-Dependent Vehicle Routing Problem (TDVRP), and considering the use of drones for parcel delivery, innovative and effective solutions are being devised. These solutions are capable of catering to the present and future needs of businesses and smart cities concerning logistics, distribution, and transportation. Let's delve into the specifics of some noteworthy studies in this domain.

The first study emphasizes the integration of production task planning with vehicle route optimization using a hybrid algorithm based on the Ant Colony System and local search [1]. This approach has demonstrated its efficiency and effectiveness in solving the combined problem.
The second study uses a genetic algorithm to optimize the daily routes of distribution employees in a Portuguese company [2]. The application of this algorithm has significantly reduced the total distance traveled, reinforcing the relationships between suppliers and customers, reducing fuel-related costs, and promoting environmental sustainability [2].

The third study presents a new crossover method for solving the TSP using genetic algorithms [3]. This innovative method of combining existing solutions has shown better results than previous methods.

The fourth study proposes a new variant of the PSO algorithm for solving the VRPTW [4]. Three different adaptive strategies are used in the Multi-Adaptive Particle Swarm Optimization (MAPSO) algorithm, which was tested on classic benchmark instance sets and compared to other PSO versions and the most efficient algorithms in the literature [4].

The fifth study investigates the application of PSO for solving TDVRP cases [5]. After reviewing 37 benchmark articles, the extraction and analysis reveal that the research on TDVRP addresses the travel time duration between two locations and that the route optimization parameter is determined from the cost of the journey, including the total distance traveled, the total travel time, the number of routes, and the number of vehicles used [5].

The sixth study explores the use of drones in parcel delivery in tandem with trucks [6]. An Ant Colony Optimization (ACO) algorithm is developed to solve the Vehicle Routing Problem with Drones (VRPD) by assigning customers to pairs of drone-truck, determining the number of drone-truck units to be dispatched, and obtaining optimal service routes while minimizing the fixed and moving costs of both vehicles [6]. Two new methods are proposed to investigate the efficiency of the drone-truck combination, allowing drones to perform additional delivery services to a single feasible customer or multiple feasible customers while the truck waits at a customer location [6]. Experimental results show that the proposed ACO algorithm can effectively solve the VRPD for different instance sizes and different distributions of customer locations and succeeds in providing timely solutions for small test instances within 1% of optimal solutions [6].

The optimization of vehicle routing with maximum capacity constraints but no time constraints, in the context of e-commerce logistics and distribution, is a critical area of research. Various optimization algorithms, such as Intra-Route Local Search, Inter-Route Local Search, and Tabu Search, are proposed and compared to solve this NP-hard problem [7]. The impact of the instance size on the performance of these algorithms is evaluated, and a comprehensive comparison of performances is presented, aiming to optimize last-mile delivery by reducing the global carbon footprint and achieving profitability through sustainable distribution modes [7].

A new routing model for the synchronized operation of trucks and drones has been introduced. This model allows multiple drones to fly from a truck, serve one or multiple customers, and return to the same truck for battery exchange and parcel retrieval. The model addresses two levels of delivery and aims to minimize the total arrival time of trucks and drones at the depot. Two effective heuristic algorithms, the Truck-Drone
Route Construction (DTRC), and the Large Neighborhood Search (LNS), are developed to tackle large-sized problems [8].

A novel Traveling Salesman Problem (TSP) with time windows for last-mile delivery in online purchases is presented, emphasizing customer pickup at shared delivery facilities [9]. A general variable neighborhood search heuristic is developed to address this problem, and computational results confirm that the proposed heuristic is competitive compared to well-known algorithms [9].

In summary, recent studies in the field of flexible routing optimization for the TSP and VRP showcase a variety of approaches for solving complex logistics, distribution, and smart cities challenges. By integrating production and transportation activities, optimizing daily routes of distribution employees, improving crossover methods in genetic algorithms for TSP solutions, developing Particle Swarm Optimization algorithm variants for VRPTW and TDVRP problems, and exploring the use of drones for parcel delivery, researchers continue to develop innovative and effective solutions to address current and future business and smart city challenges in logistics, distribution, and transportation.

This work takes the current state of the art to new heights by implementing a comprehensive analysis and critical evaluation of multiple dimensions in VRP. Not only do I provide a meticulous metric analysis that measures the performance and efficiency of various variables in VRP, but also execute an authentic mapping of TSP VRP, thereby creating a more accurate and detailed model of real-world delivery scenarios. Additionally, my model is enhanced with an innovative element of considering constraints due to parcel locker unavailability, paving the way for a more realistic and pragmatic approach in addressing and solving real-life logistic challenges.

3 Methods

In our exploration of routing optimization, we focused on the Vehicle Routing Problem (VRP) and its extended scenarios, employing a range of algorithms to tackle distinct challenges and additional constraints. These scenarios included the standard VRP and the VRP with the added complexity of 'smart lockers', an emerging factor in last-mile delivery logistics.

For the standard VRP, which includes challenges like vehicle capacity, pickup points, and varying demands, we utilized Genetic Algorithms (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Simulated Annealing (SA). These algorithms were selected for their versatility and capability to handle the complexities inherent in the VRP. Each of these methods has shown proficiency in managing large search spaces and optimizing routes while considering various operational constraints.

The VRP was further extended to include the innovative concept of 'smart lockers'. This addition introduced a unique constraint: the availability and strategic placement of these lockers within the delivery network. For this scenario, we continued with the application of GA, ACO, PSO, and SA, leveraging their adaptability to accommodate this additional dimension of complexity.
The objective behind employing these diverse algorithms across different VRP scenarios was to identify the most effective strategies for each specific challenge. This approach was aimed at developing flexible and robust solutions, adaptable to the complex and evolving requirements of logistics and distribution. By focusing on these variations of the VRP, we sought to enhance our understanding of route optimization in the face of ever-increasing logistical demands.

3.1 A Subsection Sample Data Preparation:

In our investigation, we focused on comparing the performance of four distinct algorithms - Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Simulated Annealing (SA) - in solving Vehicle Routing Problem (VRP) scenarios within the city of Marrakech. Marrakech, a historic and tourist city in Morocco, poses unique challenges due to its traffic conditions and urban density, making the resolution of VRP particularly relevant and interesting in this context.

The goal of this comparison was to ascertain the effectiveness and efficiency of each algorithm in terms of the quality of the solutions achieved, the computation time required, and the robustness of the algorithms under different problem parameter variations. We utilized a dataset comprised of 40 points in Marrakech, loaded from a CSV file, to serve as a test case. This dataset was chosen to mimic the complexities and realities of logistics and routing within a densely populated urban setting.

This analysis allowed us to highlight the strengths and weaknesses of each method, contributing to the identification of the most suitable approaches to solve VRP in similar urban environments. As such, we were able to provide valuable insights into the comparative performance of each algorithm, thereby aiding in the decision-making process for routing optimization in similar contexts.

![Fig. 1. Spatial Distribution of Parcel Lockers: A Geographic Representation](image-url)
3.2 Optimization Algorithms:

**Genetic Algorithms (GA):** is a search heuristic inspired by the process of natural selection. It is an iterative algorithm that optimizes solutions by simulating biological evolution. The Genetic Algorithm works with a population of possible solutions, each represented as chromosomes. These solutions undergo operations analogous to genetic mutation and crossover (or reproduction) to generate new potential solutions. In our analysis, we used the Genetic Algorithm to optimize last-mile delivery routes. This technique was particularly advantageous in our study due to its ability to traverse large search spaces and provide near-optimal solutions, despite the complexity of the problem.

The pseudocode can be seen below:

```
Initialize a population of individuals
While not termination condition do
    Calculate fitness for each individual in the population
    Select individuals for crossover
    Perform crossover and create offspring
    Perform mutation on offspring
    Select individuals for the next generation
    Replace the current population with the new generation
End while
```

**Selection:** Individuals are selected from the population for reproduction. This is often done probabilistically, where the probability of selection is based on the fitness of an individual.

**Crossover:** Two individuals (parents) are combined to form one or more offspring. This is often done by swapping parts of the parents’ representations. A simple representation could be a binary string, in which case the crossover could be a single point crossover, where the string is cut at a point, and the ends are swapped.

**Mutation:** Small random changes are made to an individual. In the case of a binary string representation, this could be flipping a bit from 0 to 1 or vice versa.

**Ant Colony Optimization (ACO):** is a probabilistic technique designed to solve optimization problems by simulating the behavior of ants searching for food. Ants in the algorithm move through a graph representing the problem space, laying down and following pheromone trails, which guide the rest of the colony to favorable solutions. In our analysis, we employed ACO to optimize delivery routes, capitalizing on its capability to discover the most efficient paths by using a swarm intelligence-based search method. It proved particularly effective in handling dynamic changes in the problem environment, mirroring the real-world unpredictability of last-mile deliveries.

The pseudocode can be seen below:

```
Initialize pheromone trails
While not termination condition do
    Each ant constructs a solution
    Update best solution (if current solution is better)
    Evaporate pheromone trails
    Deposit pheromone on the trail according to the quality of the best solution
End while
```
Pheromone Update:

\[ P_{ij}(t) = \frac{[\tau_{ij}]^\alpha[\eta_{ij}]^\beta}{\sum[\tau_{ik}]^\alpha[\eta_{ik}]^\beta} \quad (1) \]

\[ \tau_{ij} = (1-\rho)\tau_{ij} + \sum \Delta \tau_{ij}^k \quad (2) \]

- \( P_{ij} \): The probability of moving from node i to node j.
- \( \tau_{ij} \): The intensity of the pheromone trail from node i to node j.
- \( \eta_{ij} \): The heuristic information from node i to node j.
- \( \rho \): The pheromone evaporation rate
- \( \alpha \): Parameter to control the influence of the pheromone trail.
- \( \beta \): Parameter to control the influence of the heuristic information.
- \( \Delta \tau_{ij}^k \): Pheromone deposited by \( k^{th} \) ant.

**Particle Swarm Optimization (PSO):** is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. The technique is inspired by the social behavior of bird flocking or fish schooling. In PSO, each particle (representing a potential solution) adjusts its flying according to its own and its companions' flying experience. In our study, we applied the PSO to optimize the delivery routes. PSO's power lies in its simplicity and ability to converge rapidly towards an optimal or near-optimal solution. The algorithm demonstrated excellent performance in our problem scenario, finding good solutions within reasonable time frames.

The pseudocode can be seen below:

```plaintext
Initialize a swarm of particles with random positions and velocities

While not termination condition do
  For each particle in the swarm
    Calculate fitness of the particle
    If the fitness is better than the best fitness of the particle, update the best fitness
    If the fitness is better than the best fitness of the swarm, update the best fitness of the swarm
  End for
  For each particle in the swarm
    Update velocity using the best fitness of the particle and the best fitness of the swarm
    Update position using the new velocity
  End for
End while
```

Velocity Update:

\[ \dot{v}_i(t+1) = Wv_i(t) + c_1r_1\left(x_{\text{best}}^i(t) - x_i(t)\right) + c_2r_2\left(g_{\text{best}}(t) - x_i(t)\right) \quad (3) \]

Position Update:
\[ x_i(t + 1) = x_i(t) + v_i(t + 1) \quad (4) \]

\[ X_i^{\text{best}}(t) : \text{Personal best of the particle number } i \text{ at iteration number } t. \]

\[ v_i(t) : \text{Velocity of the particle number } i \text{ at iteration number } t. \]

\[ x_i(t) : \text{Position of the particle number } i \text{ at iteration number } t. \]

\[ G_{\text{best}} : \text{Global best} \]

\[ r_1, r_2 : \text{Random numbers between 0 and 1 open} \]

\[ c_1 : \text{Cognitive coefficient} \]

\[ c_2 : \text{Social coefficient} \]

\[ W : \text{Inertia weight} \]

**Simulated Annealing (SA):** is a probabilistic optimization technique inspired by the annealing process in metallurgy. The algorithm works by initially accepting both better and worse solutions to escape local optima, then gradually reducing the acceptance of worse solutions as the 'temperature' cools. In our analysis, we employed SA to tackle the delivery routing problem. Given the algorithm's ability to avoid getting trapped in local optima and explore the search space more comprehensively, SA was highly beneficial in our problem context, which involved large-scale and complex routing optimization.

The pseudocode can be seen below:

```
Initialize an initial solution
Initialize initial temperature
While not termination condition do
    Generate a new solution by a small modification of the current solution
    If the new solution is better than the current solution, accept it as the current solution
    Else, accept the new solution as the current solution with a certain probability, which is a function of the difference in solution quality and temperature
    Decrease the temperature according to a cooling schedule
End while
```

**Acceptance Probability:**

\[ P_k = e^{\frac{e - e'}{T_k}} \quad (5) \]

**Temperature Schedule:**

\[ T_k = T_0 \alpha^k \quad (6) \]

\[ e : \text{Distance of the current solution.} \]

\[ e' : \text{Distance of the new solution.} \]

\[ T_k : \text{The current temperature.} \]

\[ P_k : \text{The acceptance probability.} \]

\[ T_0 : \text{The initial temperature.} \]

\[ \alpha : \text{The cooling rate.} \]
3.3 Deployment Of Methods

In this research, we addressed three distinct logistical problems, each involving the implementation of different optimization algorithms to find the most efficient solutions. The algorithms chosen for this study included Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Simulated Annealing (SA).

Vehicle Routing Problem (VRP): is a complex optimization challenge that involves designing optimal routes for a fleet of vehicles operating from a central depot. The primary goal is to service a number of dispersed customers and return to the depot, while minimizing the total travel distance or time. This problem integrates various constraints, such as vehicle capacities and customer demands. Each of the four algorithms (GA, ACO, PSO and SA) was adapted to handle these additional constraints, demonstrating their flexibility and effectiveness for solving complex logistical problems.

\[
V = |C| \quad (7)
\]
\[
N = |D| \quad (8)
\]
\[
\sum D_i \leq \sum C_i \quad (9)
\]
\[
0 \leq d \leq N \quad (10)
\]

\( V = \text{The number of vehicles must be equal to the number of capacities:} \)
\( C = \text{set of capacities} \)
\( D = \text{set of demands} \)
\( d = \text{Depot index} \)
\( N = \text{number of points} \)

VRP with "Smart Lockers" constraints: The third problem is a variant of the VRP, incorporating the additional constraint of the availability of "smart lockers". These lockers represent autonomous pickup points that can serve as temporary storage for parcels. The algorithms had to be further adapted to handle scenarios where some lockers might be unavailable. In such cases, the algorithms were designed to reroute the deliveries to the next nearest available locker. This problem scenario is reflective of real-world situations and showcases the algorithms' ability to handle dynamic problem environments.

\[
D_{NP} + = D_{CV} \quad (11)
\]
\[
D_{CV} = 0 \quad (12)
\]

\( D = \text{set of demands} \)
\( NP(int): \text{The index to add the demand to} \)
\( CV(int): \text{The index from which the demand is moved} \)
4 Results and Discussion

The machine used was an i7-10750H CPU @ 2.60GHz with 16GB RAM. The route on the map was visualized by HTML. The application was developed using Tkinter. The heuristic was implemented in Python and run thirty times for each instance. The average value is reported for analysis.

In this study, our primary objective was to assess the effectiveness of different optimization algorithms in solving the Vehicle Routing Problem (VRP). To accomplish this, we used two key performance metrics - the total tour distance and the execution time of the algorithm.

Total Tour Distance: the VRP, this metric is the sum of the distances of all sub-tours taken by each vehicle. We obtain these distances from a distance matrix, which we generate using the Bing Maps API to provide accurate route distances between points. The total tour distance serves as an indicator of the efficiency of a given solution, with lower distances indicating more optimal solutions.

\[
(TTD) = \sum \sum d(P_i, P_{i+1}) \quad (14)
\]

\(d(P_i, P_{i+1})\) = Distance from point \(i\) to point \(i+1\) in the tour.

Execution Time: This is the time taken by the algorithm to generate a solution. It serves as a measure of the algorithm's efficiency, with faster execution times being more desirable. This is especially important in real-world applications where timely solutions are essential.

\[
(ET) = T_{end} - T_{start} \quad (15)
\]

\(T_{end}\) = End time of algorithm execution.
\(T_{start}\) = Start time of algorithm execution.

4.1 Vehicle Routing Problem (VRP)

For the Vehicle Routing Problem (VRP), four heuristic optimization algorithms were employed: Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Simulated Annealing (SA). In this scenario, each vehicle had its unique capacity, which remained consistent across all algorithms. The same parameters of demand and depot were utilized for each algorithm, ensuring a fair comparison.

The Genetic Algorithm was configured as follows: number of generations (100,000), population size (100), mutation rate (0.1), and crossover rate (0.9). The GA took significant time (301.10s) to solve the VRP and reached convergence at the 70,000th iteration. The algorithm found a solution with a total distance of 87.66 km (as seen in Figure 6).

The Ant Colony Optimization algorithm was set to 2000 iterations, 40 ants, alpha as 0.1, beta as 1, and rho as 0.5. The ACO finished in 226.68 seconds and converged after 500 iterations, securing a shorter route of 76.43 km (Figure 7).
Particle Swarm Optimization was configured with 2000 iterations, 40 particles, c1 (personal influence) as 0.1, c2 (social influence) as 0.9, and w (inertia weight) as 0.1. Despite taking only 14.55 seconds to complete, the PSO’s score was higher at 110.63 km, and it converged at 1350 iterations (Figure 8).

Simulated Annealing had parameters of 100000 iterations, initial temperature of 1000, final temperature of -100, and a cooling rate of 0.1. SA was the fastest algorithm, finishing in a mere 2.42 seconds and converging at the 55500th iteration, but it found a longer route of 96.95 km (Figure 9).

Upon analysis, it can be seen that each algorithm presents trade-offs between execution time and solution quality. The shortest route was discovered by ACO, although it took a comparatively longer time to solve the problem. The fastest algorithm was SA, but it did not find the shortest path. This shows the importance of choosing the right algorithm and tuning its parameters according to the specific needs of the problem.
In a further study of the Vehicle Routing Problem (VRP), we considered "Smart Lockers" constraints where certain lockers might be unavailable. In such cases, the algorithms were adapted to redirect demands to the nearest available locker. As before, each vehicle had its own capacity, and the same capacities, demands, and depot were used across all the algorithms. In this scenario, we removed parcel locker number 3 from the set of available options.

The Genetic Algorithm (GA) was set with a number of generations at 100000, a population size of 100, a mutation rate of 0.1, and a crossover rate of 0.9. It took a substantial 252.95 seconds to converge at iteration 70000, with the final route totaling a distance of 84.71 km, as shown in Figure 10.

The Ant Colony Optimization (ACO) algorithm was run for 2000 iterations with 40 ants, with an alpha of 0.1, a beta of 1, and rho at 0.5. ACO converged at iteration 1200, taking 229.85 seconds to compute. The resulting route was the shortest among the algorithms at 75.61 km, as shown in Figure 11.

Particle Swarm Optimization (PSO) had parameters set to 2000 iterations, 40 particles, with c1 (personal influence) as 0.1, c2 (social influence) as 0.9, and w (inertia weight) as 0.1. PSO was relatively quick, taking only 15.23 seconds to converge at iteration 250, with the total distance of the route being 103.61 km, as shown in Figure 12.

Simulated Annealing (SA) was the fastest algorithm, running 100000 iterations with an initial temperature of 1000, a final temperature of -100, and a cooling rate of 0.1 in a short 2.25 seconds. SA converged at the 72000th iteration, with the final route totaling a distance of 95.98 km, as illustrated in Figure 13.

Upon analysis, the ACO algorithm again provided the shortest route, proving its effectiveness in handling the VRP with Smart Lockers constraints. However, it's important to note that the fastest solution, offered by SA, didn’t provide the shortest route, underscoring the trade-off between computational efficiency and solution quality.
4.3 Critical Analysis

In our study, we compared the efficacy of various heuristic algorithms including Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization
(PSO), and Simulated Annealing (SA) in tackling two complex combinatorial optimization problems: the Vehicle Routing Problem (VRP). Our findings highlighted the superiority of the ACO heuristic over the others in these problem contexts in terms of quality.

Further validating these findings, we benchmarked our ACO implementation using three standard instances from the Vehicle Routing Problem library - small (n39k5), medium (n69k9), and large (n120k5). The results were compared against the known optimum values for these instances under identical demand capacity constraints. The following table presents our findings:

**Table 1. Comparative analysis of ACO performance against optimum solutions for VRP instances**

<table>
<thead>
<tr>
<th>Instance</th>
<th>Optimum Value</th>
<th>ACO Result</th>
<th>% Deviation from Optimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>n39k5</td>
<td>822</td>
<td>842.93</td>
<td>2.54%</td>
</tr>
<tr>
<td>n69k9</td>
<td>1159</td>
<td>1203.45</td>
<td>3.83%</td>
</tr>
<tr>
<td>n120k5</td>
<td>13332</td>
<td>14001.05</td>
<td>5.01%</td>
</tr>
</tbody>
</table>

5 Conclusion

Throughout our study of heuristic optimization algorithms, we’ve concentrated on two key scenarios: the standard Vehicle Routing Problem (VRP) and the VRP with "Smart Lockers" constraints. We employed four distinct optimization techniques: Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Simulated Annealing (SA).

In the standard VRP, the ACO emerged as a standout performer, delivering the shortest routes while effectively managing the additional complexity of vehicle capacities. On the other hand, Simulated Annealing (SA) was notable for its rapid computation, underscoring the balance between solution quality and computational speed.

The VRP with "Smart Lockers" constraints introduced a new layer of complexity. In this scenario, ACO proved its versatility and efficiency, adeptly balancing capacity constraints with the availability of smart lockers. While SA offered the quickest solutions, it did not always yield the most optimal routes.

In conclusion, ACO has demonstrated a consistent ability to find the shortest routes across different VRP scenarios. However, the fastest algorithm does not necessarily equate to the most optimal solution. The selection of an algorithm should therefore be tailored to the specific needs of the problem, considering factors like computational resources, time constraints, and precision requirements. Vehicle capacity and the unique constraints of each scenario also play a critical role in determining the most appropriate algorithm. This highlights the necessity for a nuanced approach to heuristic optimization, where the choice of algorithm is carefully customized to ensure the best possible outcomes.

**Table 2. Performance of optimization algorithms VRP and VRP with constraints**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>VRP (Time / Score)</th>
<th>VRP with Constraint (Time / Score)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA</td>
<td>301.10s / 87.66 km</td>
<td>252.95s / 84.71 km</td>
</tr>
</tbody>
</table>
Acknowledgments. The MILEX project, a cornerstone of this research, was significantly enhanced by the steadfast support from Hassan II University of Casablanca, Ecole Centrale Casablanca, and Mohammed VI Polytechnic University. Their collaborative and essential contributions were instrumental in the success of this endeavor. We are deeply grateful for their invaluable involvement throughout our research journey. We also extend our heartfelt thanks to the referees for their insightful and constructive feedback, which greatly enhanced the study's quality and rigor.

References

Contribution to Demand Forecasting Based on Artificial Intelligence Algorithms Applied to the Occupation of Smart Lockers Prediction

Jamal BENHRA1, Mohamed-Ali EJJANFI2 and Hamza EL KASSIMI EL ALAOUI3

12 LARILE, ENSEM, University Hassan II of Casablanca, Morocco
3Computer science engineer, ESITH Casablanca, Morocco
1jamalbenhra@gmail.com
2ejjanfimed@gmail.com
3elkassimihamza04@gmail.com

Abstract. The aim of this paper is to provide a comprehensive examination and comparison of five key forecasting techniques: Triple exponential smoothing (ETS), ARIMA (autoregressive moving average), SARIMA (Seasonal autoregressive moving average), ANN (Artificial Neural Networks), and LSTM (Long Short-Term Memory). Prior to model training, we perform an extensive hyperparameter optimization to determine the best parameters for each forecasting technique. These techniques have been carefully selected due to their significant relevance and potential in the field of demand prediction, especially in the context of smart parcel locker systems. Each forecasting method is scrutinized in-depth, with a focus on their mathematical underpinnings, their practical applications, and their strengths and weaknesses in predicting demand patterns. Our analysis emphasizes the importance of precise forecasting in ensuring the flexibility of parcel locker systems, by facilitating optimal storage management and adaptation to fluctuations in parcel sizes and quantities. The insights generated from this study offer a roadmap for businesses and organizations to select the most suitable forecasting technique for their specific needs. With a nuanced understanding of these methods, fostered through rigorous hyperparameter tuning, they can enhance the utilization of their smart locker systems, thereby meeting the ever-changing demands of their customers more effectively and efficiently.

Keywords: Demand forecasting, parcel size, parcel quantity, ANN, LSTM, Holt-Winters Exponential Smoothing, ARIMA, SARIMA, GMDH SHELL, Smart lockers.

1 Introduction

Smart lockers are advanced storage systems that employ cutting-edge technologies such as the Internet of Things (IoT) to enable users to deposit and collect items with ease and flexibility [1]. These systems are highly adaptable, making them suitable for handling various storage demands, especially in terms of package sizes and quantities. Accurate demand forecasting for these specific attributes is crucial to fully optimize their utilization [2].
This review seeks to examine the versatility of smart lockers in addressing the storage needs for diverse package quantities by exploring pertinent literature and investigating demand forecasting approaches that integrate these factors. We conducted an exhaustive search of electronic databases, including Web of Science, google scholar and IEEE Xplore, focusing on peer-reviewed articles published from 2019 to 2023. Our research uncovered several articles that specifically tackle demand forecasting [7].

By distilling and analyzing information from these sources, we aim to offer valuable insights into the efficacy and challenges of adaptive smart lockers and demand forecasting methods that consider package dimensions and volume.

The findings of this review have the potential to assist businesses and organizations in better comprehending the advantages and challenges of implementing smart lockers while optimizing their storage management to meet evolving customer requirements. Moreover, this study could contribute to a broader understanding of urban logistics, supply chain management, and the optimization of delivery services.

The main contributions of this research can be summarized as follows:

State of the Art: This research provides a comprehensive overview of the current state of the art in forecasting techniques, focusing on their applicability and performance in predicting the number of parcels. By reviewing the latest advancements in the field, the study offers valuable insights for both practitioners and researchers.

Methods: This research applies a range of forecasting techniques such as Exponential Smoothing, SARIMA, ARIMA, LSTM, and ANN to tackle the challenge of predicting the quantity and volume of parcels. Initially, we conducted a benchmark analysis using GMDH Shell to set a comparative standard for our models. These methods were scrutinized under a two-pronged approach - hyperparameter tuning to seek the best parameters and a manual configuration of parameters for comparison. The contrasting results from these methods provide a comprehensive understanding of their respective strengths and limitations. This offers a valuable perspective for readers to make informed decisions when selecting suitable forecasting techniques for similar problems. The contribution of this work lies not only in its exploration of various forecast methodologies but also in its methodical comparison, which helps to highlight the diverse capabilities of these models.

Data Preparation: The research utilizes a rich dataset containing parcel volume information over different periods (4, 6, and 10 years), which allows for a thorough evaluation of each method’s performance. The use of varying dataset lengths also enables an investigation of the impact of dataset size on model performance.

Results and Discussion: The research presents a comprehensive analysis of the results obtained from each forecasting technique, discussing the implications of these findings in the context of parcel prediction. By examining the performance of each method in terms of Mean Absolute Error (MAE) and Root Mean Square Error (RMSE), we provide a detailed examination of the model’s forecasting accuracy and error margins. These metrics allow for the quantification of the differences between the forecasted and actual values, thus helping to evaluate the effectiveness of each forecasting model in accurately predicting future parcel demand.

Conclusion: The research culminates in a conclusive summary, highlighting the key findings and offering guidance for future research and practical applications in the field.
of demand forecasting. By synthesizing the findings from the state of the art, methods, dataset, and results, the conclusion offers a clear direction for further exploration and improvement in this critical area of study.

2 State of art

Demand forecasting is a critical aspect of the smart locker industry, as it helps optimize the distribution and management of locker spaces to meet the fluctuating needs of consumers. Several techniques are commonly employed in demand forecasting for smart lockers, each with its unique set of advantages and limitations. One technique is time series analysis, which focuses on historical data to identify patterns and trends. For example, Autoregressive Integrated Moving Average (ARIMA) models have been used to forecast parcel lockers demand [5]. This method is effective for capturing seasonality and trends in demand, but it assumes that future demand will mirror past patterns, which may not always be the case. Another technique is machine learning-based forecasting, where algorithms such as artificial neural networks (ANN) and support vector machines (SVM) are employed [7]. Found that an SVM-based model outperformed traditional time series models for predicting demand. Machine learning techniques can learn complex patterns and relationships in data, making them well-suited for forecasting in dynamic and rapidly changing markets. In addition, hybrid models have gained traction as they combine the strengths of multiple methods. For instance, a study developed a hybrid model that combined an LSTM (Long Short-Term Memory) neural network with a seasonal decomposition of time series (STL) to forecast Metro Ridership Using Seasonal and Trend Decomposition Using Loess and LSTM [4]. The results showed that the hybrid model performed better than traditional time series and machine learning models alone, demonstrating the potential of combining multiple techniques. Lastly, causal models, such as regression analysis, are used to identify the factors affecting smart locker demand. The study employed multiple linear regression to determine the relationship between smart locker demand and factors such as service quality and customer satisfaction in the adoption of smart parcel lockers, dividing service quality into five dimensions: service price, service reliability, convenience, fault handling capability, and service diversity [3]. This approach can help businesses and policymakers understand the drivers of demand and tailor their smart locker deployment strategies accordingly. In summary, various techniques are employed in demand forecasting for smart lockers, including time series analysis, machine learning, hybrid models, and causal models. Each method has its strengths and limitations, and the choice of technique depends on the specific needs of the business or organization. By selecting the most appropriate forecasting technique, smart locker providers can optimize their operations to meet consumer needs and maximize profits.

3 Methods

In this study, we employed various forecasting techniques to predict the demand for different parcel sizes using a dataset of IoT-based smart parcel lockers. We explored the performance of five models, namely Exponential Smoothing (ETS), Long Short-Term Memory (LSTM), Artificial Neural Network (ANN), Autoregressive Integrated Moving Average (ARIMA), Seasonal Autoregressive Integrated Moving Average (SARIMA). These methods were chosen due to their effectiveness in handling time-
series data and ability to capture complex patterns, trends, and seasonality. By comparing the mean absolute error (MAE) and the root mean square error (RMSE) for each model, we aimed to identify the most suitable forecasting technique for predicting parcel locker demand. This comprehensive approach provided valuable insights into the comparative strengths and weaknesses of each method, enabling a more informed decision-making process for optimizing last-mile delivery services using smart parcel lockers.

3.1 Data Preparation:
In our analysis, we utilized a custom-generated dataset representing the demand for smart parcel lockers over a period of four years. This dataset contained daily records of the total number of parcels and the number of parcels. To ensure better model stability and reduce the influence of daily volatility, we transformed our daily data into weekly sums. We generated this data using Python’s random module and datetime functionality, simulating a real-world scenario of fluctuating parcel demand. The dataset was created by starting from a given date, iterating through each day until the present, and generating random parcel numbers on a daily basis. For model training and testing purposes, we split our data into two parts, using 70% of the data for training the model and the remaining 30% for testing its predictive performance. This common practice in machine learning projects allows us to gauge how well our model can generalize its predictions to unseen data, thereby providing us with a realistic estimate of its performance in real-world scenarios. We used three distinct datasets to examine the accuracy and reliability of our predictions over various time frames. The first dataset encompasses data from the past four years (data_small.csv), providing a shorter time span to analyze recent trends and fluctuations. The second dataset extends this period to six years (data_medium.csv), offering a more comprehensive view of the market’s performance and allowing us to identify potential cyclical patterns. Finally, the third dataset covers an entire decade (data_large.csv), granting us the opportunity to observe long-term trends and investigate how the market has evolved over time. By examining these three datasets, we can gain valuable insights into the forecasting performance across different time horizons and develop a more robust understanding of the market dynamics at play.

3.2 Forecasting Techniques:
Exponential Smoothing: In our research, we have employed Triple Exponential Smoothing, also known as Holt-Winters Exponential Smoothing, to forecast the num-

![Fig. 1. small dataset for 4 years](image1)
![Fig. 2. Medium dataset for 6 year](image2)
![Fig. 3. Large dataset for 10 years](image3)
ber of parcels. Triple Exponential Smoothing is particularly advantageous for our dataset, as it accounts for both seasonality and trend components. This method is an extension of the simple exponential smoothing technique, which considers only the level (i.e., the weighted average of past observations). The Triple Exponential Smoothing technique incorporates three components: level, trend, and seasonality, making it a more comprehensive and accurate forecasting tool for our dataset, spanning the last four years [13]. The Holt-Winters method involves three smoothing parameters: alpha for the level, beta for the trend, and gamma for the seasonality. By optimizing these parameters, the model can effectively capture the underlying patterns in the data and generate more accurate forecasts [15]. In our implementation, we have used the Exponential Smoothing function from the statsmodels library, which automatically optimizes these parameters by default, ensuring the best possible forecasts for our data. Given the presence of both trend and seasonality in our dataset, the choice of Triple Exponential Smoothing is well-justified and expected to provide reliable predictions for the number of parcels from the text.

**ARIMA/ SARIMA:** ARIMA (Autoregressive Integrated Moving Average) and SARIMA (Seasonal Autoregressive Integrated Moving Average) are popular time series forecasting methods. Both models aim to capture the underlying patterns in time series data, such as trends and seasonality, to make accurate predictions. The ARIMA model consists of three main components: autoregression (AR), differencing (I), and moving average (MA). The AR component measures the relationship between a data point and its preceding data points, the I component is used to make the time series stationary by differencing it, and the MA component estimates the error between the actual and predicted values [6]. SARIMA is an extension of the ARIMA model that incorporates seasonality, making it more suitable for time series data with regular seasonal fluctuations. In addition to the AR, I, and MA components, SARIMA introduces seasonal components for autoregression (SAR), seasonal differencing (SI), and seasonal moving average (SMA). These seasonal components allow the model to capture and predict seasonal patterns more accurately. When comparing the performance of ARIMA and SARIMA, SARIMA generally performs better on time series data with clear seasonal patterns [18]. If seasonality is not present in the data, ARIMA may be more suitable. It is essential to understand the nature of the data and the presence of any seasonal fluctuations when choosing between ARIMA and SARIMA models for accurate forecasting.

**Artificial neural networks (ANN):** The Artificial Neural Network model is a computational learning system inspired by the structure and functional aspects of biological neural networks. It is widely used for various machine learning tasks, including time-series forecasting [7]. In our analysis, we employed the ANN model to predict parcel size demand, taking advantage of its ability to learn complex patterns, trends, and seasonality in the data.

**Long short term memory (LSTM):** The Long Short-Term Memory (LSTM) model is a type of recurrent neural network (RNN) that is particularly effective in handling time-series data with long-term dependencies. In our analysis, we employed the LSTM model to predict parcel size demand, focusing on its ability to capture complex patterns, trends, and seasonality in the data [4].
3.3 Hyperparameter tuning

In this study, we will employ a dual approach of parameter tuning to optimize the predictive performance of our models. First, we will utilize hyperparameter tuning methods, which systematically search the parameter space using techniques such as Random Search using Keras Tuner for LSTM and ANN models [14]. This approach allows us to uncover optimal configurations that may not have been intuitive or immediately evident. Simultaneously, we will also engage in manual tuning of the parameters for SARIMA and ARIMA. In the course of our study, we developed a dedicated application aimed at facilitating the hyperparameter tuning process for ANN and LSTM models. The application harnesses the power of Keras Tuner, a Python library specifically designed for efficient hyperparameter tuning of neural networks in Keras. This approach allowed us to perform an exhaustive search of the hyperparameter space for these models, optimizing their structure and parameters to achieve superior forecasting performance. In the case of the HW Exponential Smoothing model, we employed the itertools library, a standard Python library that provides efficient looping and iteration functions [15]. It was particularly useful in our context for generating a comprehensive grid of hyperparameter combinations, which we then used to tune the smoothing model. This allowed us to identify the best set of hyperparameters that minimized the forecast error, improving the model’s predictive accuracy.

Exponential smoothing: In our exploration of the Holt-Winters forecasting model, an important facet of our analysis was the optimization of the model's parameters: alpha, beta, and gamma. These parameters respectively represent the level, trend, and seasonal components of the model, and their optimal values can significantly influence the accuracy of the forecasts. To find the best set of parameters, we employed the Python library, itertools, which allows for efficient iteration over all possible combinations of parameter values within a specified range. However, as the size of the time series expanded, we observed a decrease in the model's performance. Specifically, the Mean Absolute Error (MAE) escalated from 201.44 for smaller datasets to 280.08 for larger ones. This indicates that the Holt-Winters model, while effective for smaller time series, may lose its predictive accuracy as the size of the data expands. It's a valuable insight that prompts further research into optimizing forecasting models for larger time series datasets.

Artificial neural network: We also used an artificial neural network (ANN) to model our time series. We proceeded to a preliminary optimization step to determine the best possible values for the parameters of our model. Following the hyperparameter tuning, we identified that the optimal configuration of our ANN was the following:

- 3 Dense layers with 120,72,104 neurons
- A Rectified Linear Unit (ReLU) activation function for the hidden layers
Adam optimizer

**LSTM:** We also used a long-term recurrent neural network (LSTM) model to analyze our time series. This model is particularly suitable for processing data sequences over time, thanks to its ability to memorize long-term dependencies. Just like for the ANN, we performed a hyperparameter step to optimize the performance of our LSTM model. Following this optimization, we determined that the best configuration for our LSTM model consists of:

- 2 LSTM Layers with 8 neurons for each
- Single Dense layer with 8 neurons
- A Rectified Linear Unit (ReLU) activation function for the hidden layers
- Optimizer: Adam

![](image1.png)

Fig. 7. ANN Hyperparameter tuning

![](image2.png)

Fig. 8. LSTM Hyperparameter tuning

### 4 Results and Discussion

In our review, we chose to use the mean absolute error (MAE) and root mean square error (RMSE) as key performance metrics for comparing the different forecasting techniques. The MAE measures the average absolute difference between the predicted and actual values, providing a clear and intuitive representation of the accuracy of a forecasting model [16]. Meanwhile, RMSE is a quadratic scoring rule that measures the average magnitude of the error, effectively penalizing large errors more severely than MAE [17]. By using these metrics, we can better understand the magnitude of the errors produced by each technique and gauge their overall performance in predicting smart locker demand. The choice of MAE and RMSE as performance metrics is particularly relevant in the context of smart lockers, where accurately forecasting the demand for different package sizes and quantities is crucial for efficient storage management and meeting customer needs. By comparing the MAE and RMSE values across the various forecasting techniques, we can identify which models perform best at minimizing prediction errors and, consequently, provide the most reliable demand forecasts.

\[
MAE = \sum_{i=1}^{n} (\hat{y}_i - y_i) \tag{1}
\]

\[
RMSE = \sqrt{\frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{n}} \tag{2}
\]

$n$ = Total number of data point

$y$ = Actual output value

$\hat{y}$ = Predicted output value
4.1 Critical Analysis
As a prerequisite to the comparative analysis of the five forecasting techniques, a benchmark analysis was carried out using the GMDH Shell 3.8.9. This software, which is a widely recognized forecasting tool, was applied to predict demand in three different datasets. GMDH Shell has been used extensively in previous literature due to its advanced capabilities in dealing with complex datasets and delivering highly accurate predictions [8]. Our aim with this benchmarking is to establish a point of comparison for assessing the accuracy and reliability of the other forecasting techniques. Furthermore, it provides us with an opportunity to evaluate the performance of the other methods against a proven standard in the field of demand forecasting. The comparison will ultimately aid in identifying the most suitable forecasting technique for our particular datasets and use case. This exercise of benchmarking adds a crucial dimension to our comparative study, enhancing its validity and comprehensiveness.

Summary: We noticed some notable trends during this process. Particularly, the algorithm seemed to struggle with larger datasets. As we increased the size of the dataset from small to medium and large, the model's performance deteriorated, reflected in the increase in forecast error metrics. This suggested a challenge in the model's ability to capture and extrapolate complex patterns present in larger datasets effectively. Simultaneously, we observed that the computation time for the forecasts also increased in tandem with the data size. The small dataset took approximately 37 seconds to process, while the medium dataset took slightly longer, around 45 seconds. Predictably, the large dataset took the longest, with an execution time of 1 minute and 6 seconds.

4.2 Exponential Smoothing (ETS)
We observed that the Mean Absolute Error (MAE) of the Triple Exponential Smoothing model decreases as we increase the length of the dataset. When using data from the last 4 years, the MAE was 201.44, while expanding the dataset to include 10 years of data led to an increased MAE of 280.08. This improvement in forecasting accuracy can be attributed to the increased amount of historical data, which provides a more comprehensive picture of the underlying patterns and trends in the data.
4.3 ARIMA/ SARIMA

In this step, it is worth noting that, unlike exponential smoothing, these models do not require hyperparameter tuning. Thus, we were able to directly train our ARIMA and SARIMA models without having to perform a prior optimization step.

After training the two models, we evaluated their performance using the mean absolute error (MAE) and root mean square error (RMSE) as an evaluation metric. This approach allowed us to quantify the accuracy of the ARIMA and SARIMA models and identify potential areas of improvement to further refine our forecasts and optimize predictive performance.

- ARIMA gives MAE 151.07/ 154.91/ 183 successively for Small, Medium and Large dataset.
- SARIMA gives MAE 173.16/ 141.11/ 251.35 successively for Small, Medium and Large dataset.
4.4 Artificial neural network

We also used an artificial neural network (ANN) to model our time series. Following the hyperparameter we identified that the optimal configuration of our ANN was the following: 3 dense layer, 120,72,104 neurons, the Adam optimizer and the ReLU activation function. Once these parameters were determined, we trained our ANN model on our time series and evaluated its performance using the mean absolute error (MAE) as an evaluation metric. It gives MAE of:

- 11.54 for small dataset.
- 21.55 for medium dataset.
- 15.80 for large dataset.

![Fig. 24. ANN forecasting Plot (Actual vs Predicted) for small dataset](image1)

![Fig. 25. ANN forecasting Plot (Actual vs Predicted) for medium dataset](image2)

![Fig. 26. ANN forecasting Plot (Actual vs Predicted) for large dataset](image3)

4.5 Long Short Term Memory

We finished our analysis with long-term recurrent neural network (LSTM) model to analyze our time series. This model is particularly suitable for processing data sequences over time, thanks to its ability to memorize long-term dependencies. Just like for the ANN, we performed a hyperparameter step to optimize the performance of our LSTM model. Following this optimization, we determined that the best configuration for our LSTM model consists of 2 layers of LSTM cells, 8 neurons in each layer, a single dense layer contains 8 neurons, the Adam optimizer and the ReLU activation function. Once these parameters were selected, we trained our LSTM model on our time series and evaluated its performance using the mean absolute error (MAE) as an evaluation metric. LSTM gives MAE of:

- 8.74 for small dataset
- 4.50 for medium dataset
- 5.92 for large dataset.
5 Conclusion

In conclusion, the objective of our study was to systematically compare the performance of five distinct forecasting techniques - Triple Exponential Smoothing, ARIMA, SARIMA, LSTM, and GMDH Shell - in predicting the demand for smart parcel lockers. Our analysis was predicated on the understanding that accurate and reliable demand forecasting is pivotal to the optimal management and deployment of these lockers. Each technique, characterized by its unique mathematical and statistical properties, was rigorously evaluated over multiple datasets of varying sizes. The study highlighted the strengths and limitations of each approach, demonstrating that the performance was dependent on the underlying characteristics of the data, including its size, trend, and seasonality. It provided valuable insights into the interplay between the model complexity, dataset size, and forecast accuracy. Ultimately, our study serves as a guide to selecting the most suitable forecasting technique that balances accuracy, computational efficiency, and ease of interpretation for operationalizing flexible parcel locker systems in the context of last-mile delivery services.

**TABLE 1.** Comparative Analysis of Mean Absolute Error (MAE) Across Various Forecasting Techniques

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Small</th>
<th>Medium</th>
<th>Large</th>
</tr>
</thead>
<tbody>
<tr>
<td>GMDH SHELL</td>
<td>7.21</td>
<td>10.52</td>
<td>19.71</td>
</tr>
<tr>
<td>ETS</td>
<td>201.44</td>
<td>240.89</td>
<td>280.08</td>
</tr>
<tr>
<td>ARIMA</td>
<td>156.01</td>
<td>154.91</td>
<td>183</td>
</tr>
<tr>
<td>SARIMA</td>
<td>141.11</td>
<td>173.16</td>
<td>251.35</td>
</tr>
<tr>
<td>ANN</td>
<td>11.54</td>
<td>21.55</td>
<td>15.80</td>
</tr>
<tr>
<td>LSTM</td>
<td>8.74</td>
<td>4.50</td>
<td>5.92</td>
</tr>
</tbody>
</table>

**TABLE 2.** Comparative Analysis of Root Mean Square Error (RMSE) Across Various Forecasting Techniques

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Small</th>
<th>Medium</th>
<th>Large</th>
</tr>
</thead>
<tbody>
<tr>
<td>GMDH SHELL</td>
<td>8.40</td>
<td>14.20</td>
<td>25.49</td>
</tr>
<tr>
<td>ETS</td>
<td>221.40</td>
<td>249.55</td>
<td>297.10</td>
</tr>
<tr>
<td>ARIMA</td>
<td>171.39</td>
<td>169.31</td>
<td>258.43</td>
</tr>
</tbody>
</table>
Acknowledgments. The success of the MILEX project owes much to the steadfast support provided by Hassan II University of Casablanca, Ecole Centrale Casablanca, and Mohammed VI Polytechnic University. Their collaborative and constructive engagement was crucial in bringing this project to fruition, and we are deeply grateful for their significant contributions during the research process. We also extend our heartfelt thanks to the referees, whose thoughtful feedback and constructive critiques have greatly enhanced the study's quality and rigor.

References


<table>
<thead>
<tr>
<th>Model</th>
<th>SARIMA 162.90</th>
<th>187.55</th>
<th>265.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANN</td>
<td>12.80</td>
<td>23.02</td>
<td>29.97</td>
</tr>
<tr>
<td>LSTM</td>
<td>10.07</td>
<td>5.92</td>
<td>7.45</td>
</tr>
</tbody>
</table>
Parallel Strategies for Iterative Kriging-Based Surrogate Model Optimization: Bayesian Inference in Engineering Simulations

Yunus Emre Sunay\textsuperscript{1} and Nilay Sezer-Uzol\textsuperscript{2}.

\textsuperscript{1} Dept. of Aerospace Engineering, Middle East Technical University (METU) Ankara, Turkey
\texttt{emre.sunay@metu.edu}

\textsuperscript{2} Dept. of Aerospace Engineering, Middle East Technical University (METU) Ankara, Turkey
\texttt{nuzol@metu.edu.tr}

Abstract. The advancement of computer hardware technology significantly contributes to resolving complex engineering challenges that require solution of complex, highly nonlinear equations. With high-fidelity numerical simulators, obtaining a solution can be time-consuming, often requiring several hours. This study aims to integrate high-fidelity analytical tools into the design cycle and thereby to reduce the overall duration of the design process through enhanced computational power. An optimization tool based on Bayesian inference process is developed to find the optimum design. In this study, deterministic numerical simulations are employed to construct a Kriging surrogate model, which represents a stochastic relationship between inputs and outputs, independent of the underlying physics of the simulations. Computational fluid dynamics (CFD), an expensive high-fidelity tool, is replaced by the Kriging surrogate model in Bayesian optimization. Latin Hypercube Sampling, which has randomness and homogeneity, is used to initialize the samples in the design space; however, it is not possible to increase the number of samples for iterative processes. Furthermore, this study investigates a hybrid methodology for parallel incremental sampling, which uses randomness and even distribution of LHS. This method is applicable for any initial quantity of samples to target number of samples. The optimization tool, which contains hybrid parallel algorithms for sampling, is tested with well-known Rosenbrock function and applied for shape optimization of delta wing using meshless CFD.

Keywords: Kriging Surrogate Model, Bayesian Optimization, Meshless Computational Fluid Dynamics, Vortex Particle Method, Parallel Design Strategies.

1 Introduction

The development of computer hardware technology is helping to solve very complex engineering problems using solutions of highly nonlinear equations. Fidelity of this analysis depends on the computer abilities and increases with the developments in computer power; on the other hand, it can still take hours to reach a solution for a single case with the supercomputer system. Combination of the size of the design space and
the number of variables results in numerous numbers of alternatives in the design space. In addition to that, optimization methods based on differentiation such as gradient descent cannot be parallelized since the successive step depends on the results coming from the previous run. Increasing the number of CPUs can speed up the runtime of the analysis; however, it is still limited by the hardware. All these situations make it impossible for high-fidelity analysis tools to be involved in the design cycle.

The motivation of this study is to involve high-fidelity analysis tools in the design cycle and decrease the total time required for the design process with the help of quantity of CPU and GPU and optimization algorithms. This issue can be handled with the help of surrogate model optimization methods. Most remarkable feature of these methods is their ability of parallelization that concludes with decreasing the required design time. Surrogate model optimization consists of three main components which are sampling, building the surrogate model and optimization of the surrogate model. This optimization method can be used as both one-time run and iterative process to obtain the optimum points. The challenging part in the iterative process is how to update samples both keeping initial points and increment samples by parallelization strategy. This study concerns not only adding optimum points to the design space but also increasing the homogeneity of the design space.

This study focuses on the design and the analysis of numerical experiments by using stochastic surrogate models, namely the Kriging model. The main idea in the surrogate model, also known as meta model, is to build a mathematical model between the input and the output regardless of the physics behind the simulations. Theoretical part of the Kriging model is dependent on the study of the French mathematician Georges Matheron [1] and its name is coming from South African geostatistician Danie G. Krige [2].

Sampling, known as design of experiment (DoE), is related to choosing data in the design space to help surrogate model to describe the space accurately. Full factorial is the best DoE to model the entire space, but it is not possible to put it into practice with multiple inputs and large design spaces. On the other hand, random sampling results in uneven distributions along the space and it can cause missing region in the optimization procedure. Latin hypercube sampling (LHS) [3] is capable of ensuring randomness and even distribution in the design space. The importance of DoE stems from the determination of the initial point to represent the design space.

After sampling, surrogate model is coming into process to fit the behavior of I/O relationship and the data itself. There are various types of surrogate model which can be based on classical deterministic approach such as linear approximation, Response Surface Methodology (RSM), Radial Basis Function (RBF) or on recent stochastic approach such as the Kriging model, Bayesian framework, neural network model. All the aforementioned models are actively used in accordance with the complexity and dimensionality of optimization problem itself. In the modelling, the main assumption is that the observation points are noise-free points. There are some approaches about how to include the noise in the calculations [4]. On the other hand, the popularity of surrogate model optimization stems from the success of the design and the analysis of numerical experiments which is accepted as having noise-free results.
The Kriging model and the stochastic approach in Bayesian frame extends back a long time. It has been used in many different applications in engineering design cycle; ([5], [6], [7]) for choosing experiments in materials science; medicine molecular design ([8], [9], [10]) and in reinforcement learning ([11], [12]). It became popular thanks to two important studies, namely the study of the Jones et al. [6] and study of Snoek et al. [13]. The study of Jones et al. [6] represents the definition of the acquisition function for selection of the successive point for optimization process which is known as Efficient Global Optimization (EGO), which will be mentioned in the methodology part in detail. EGO proved itself as finding global optima for expensive black-box derivative-free functions [14]. The study of Snoek et al. [13] which is tuning hyperparameters in machine learning, especially very complicated version deep neural networks, supported this idea and made it very popular for the machine learning society and research.

2 Methodology

Design optimization using surrogate model consists of choosing samples in the design space, building surrogate model instead of using high-fidelity analysis tools and finding the optimum point in the design space using the stochastic approach. Among the sampling methods, Latin hypercube sampling (LHS) is used for the determination of initial points in the design space. The Kriging surrogate model, also known as Gaussian process regression (GPR), is built to represent the trend of the function which is being tried to be optimized. Optimization cycle using surrogate model is based on the assumption that the model and the real function are having the same trend. Literature proves that this assumption concludes with satisfactory results ([15], [16], [13]). Bayesian inference is used to find the optimum point in the design space.

The design cycle starts with the definition of the problem and the limits of the design space. Following that, with the help of design of experiment algorithms, which is LHS in this study, initial points are determined. Initial points are analyzed with the high-fidelity analysis tool which is determined as FLOWUnsteady for this design study. Initial points and corresponding results of the solver are used to construct a Kriging surrogate model. Leave-one-out cross-validation (LOOCV) methods are used to analyze the accuracy of the surrogate model and the CFD analysis tool. Well-known performance parameters such as root mean square error (RMSE) and R² score are calculated for LOOCV which is the most reliable validation methods to investigate the accuracy of the model. The next step is to optimize the surrogate model to find minimum/maximum points. Efficient optimization methods for black box functions are gradient-free probabilistic optimization methods and all surrogate models can be accepted as black box functions. Bayesian inference, a statistical inference process, which is based on GPR and acquisition function, is used to improve/optimize the surrogate model. “Expected Improvement” is selected as an acquisition function. The combination of “Expected Improvement” and Bayesian optimization which is the study of Jones in 1998 [6] is called as “Efficient Global Optimization” (EGO). The design process should be finalized according to predefined criteria which can be determined as validation of the optimum point, accuracy of the surrogate model and enhancement in the results. The
optimum point is then validated with the high-fidelity analysis tool, FLOWUnsteady. If all or some of criteria are not satisfied, the cycle goes back to the sampling step with the addition of new samples to construct again the surrogate model which is called as hybrid incremental sampling method in this study.

LHS is a sampling method using randomness regarding the even distribution in the design space, and randomness does not guarantee to use same initial points while updating or increasing the number of the samples with LHS method. Incremental sampling is not an easy task in this process and needs a new approach to achieve it. In this study, a hybrid approach which is a combination of the EGO algorithm and strategy coming from even distribution and randomness of the LHS is developed for incremental sampling. All steps in the design cycle are verified first with a test function, Rosenbrock function, which is used to test the optimization algorithms and strategies. Then, to show the effectiveness of the hybrid optimization algorithm, a real problem regarding the design of the delta wings is investigated. The surrogate model optimization algorithms are coded in Python environment. FLOWUnsteady is a meshless unsteady solver which is coded in Julia program language.

2.1 Sampling Algorithms/Strategies

The sampling algorithm is the critical part of the design cycle since it directly affects the required computer power in the design cycle and the accuracy of the surrogate model. In this study, sampling algorithms are used not only for determination of the initial points but also for increasing the number of samples in each iteration. Initial points are determined by the Latin hypercube sampling (LHS). Literature shows the effectiveness of LHS in the Kriging surrogate model. In addition to that, optimization tools based on LHS, and surrogate model are suitable for parallelization. On the other hand, no incremental sampling method, which is both effective and can also be parallelized in improving accuracy of the surrogate model, became prominent in the literature. The randomness of LHS leads to no common points with the use of LHS for both equal and different number of initial samples, which makes it impossible for LHS to be used in incremental sampling process. This study proposes a hybrid incremental sampling method for increasing samples which uses both randomness and even distribution of LHS and the result of the Bayesian inference process.

Latin Hypercube Sampling (LHS): LHS, which is proposed by McKay et al. [17], is a very popular option for the design and optimization of computer experiments. The idea is to fill the space homogenously regarding the equal distribution of the variables. In the sampling methodology, one of the challenging points is to determine the number of points in the first step. In the literature and applications conclude with a rule of thumbs which corresponds that number of initial points is equal to 10 times the number of input variables [18]. The number of samples depends on the problem itself and should be updated in the process according to criteria to meet the accuracy of the model, on the other hand, the rule of thumbs is a good start.

There are different libraries, which are especially related to surrogate models and optimizations, that contain the LHS method. In this study, Surrogate Model Toolbox [19], which is a Python library, is used for modeling and optimization with LHS.
Hybrid Incremental Sampling Methods: The number and distribution of samples are critical in the surrogate model consistency and accuracy. The potential of LHS stems from both the randomness and the even distribution over the variables in the design space. In the optimization process, it is required to add new samples at each iteration of the design cycle to improve the consistency and accuracy of the model which results in finding better optimum point. Adding new samples to an initial existing design space to increase the number of samples or creating a new random design space with a new increased number of samples will not be the same. The randomness and even distribution will still be important with increased number of samples. Using LHS will result in a new design space distribution and there may be no common points between the initial and new distribution. This will require expensive numerical CFD simulations to be performed for all the points in the new distribution at each iteration and with an increased number of samples.

The hybrid sampling method, the incremental sampling, which is proposed in this study, will allow the simulation runs to be done only for the new points added to the design space. Initial design space is obtained by LHS, and then the number of samples is increased by using the optimum point from the Bayesian inference and incremental sampling method which proposed in this study. Thus, it will combine the effectiveness of LHS with the accuracy of Bayesian inference.

It is required to find a better strategy which uses the LHS and adds new samples by keeping the initial points. Incremental samples are results of a hybrid algorithm which consists of two different parts and can be parallelized. The first part comes from the result of the optimization tool as an optimum point which has only one point. In this study, EGO algorithm, which is the combination of Bayesian rules and EI algorithm, is used to find the optimum point. Sample coming from EGO consists of one element and corresponds to the first part. The second part of the hybrid methodology proposed a new strategy which uses remarkable features of LHS such as even distribution, randomness and parallelization and still keeps the initial points. It is not possible to use LHS for the delta samples, because the initial sampling and new sampling with more sample points will have no common points using LHS.

For the second part of the incremental sampling methods, the Euclidean norm of each new points obtained using LHS to every initial point are calculated and summed. After calculating Euclidean norm of each new point to every point in the initial samples, Euclidean norms are summed up for each new point which corresponds to the overall distance of a new point to the initial points. The largest value coming from summation is the outmost or most different point in the updated samples which are added the design space.

Formulation for the adding new points are represented as follows:

\[
\text{distance score} = \sum_{i=1}^{n} \| x_i - x_u \| \quad \forall x \in A
\]  

(1)

where, subscript \( i \) corresponds to initial point, subscript \( u \) corresponds to updated point, \( n \) is total number of existing samples, and \( A \) is subset of the samples.

Fig. 1(a) shows that there are no common points between the initial samples and the new incremental samples. On the other hand, Fig. 1(b) shows the same initial points
and the new incremental samples that keeps the initial points and additional points obtained by the new strategy that keeps the homogeneity of the design space.

![Distribution of samples using LHS with 20 and 31 points](image1.png)

![Distribution of incremental samples with 20 initial and 31](image2.png)

Fig. 1. Demonstration of difference between LHS and the hybrid incremental sampling method.

### 2.2 Kriging Surrogate Model

The Kriging surrogate model is a prediction model between the input and output, it consists of a known function which can be called deterministic part and the realization of stochastic process part. Its popularity is the result of the success in the design based on the deterministic discrete simulations. CFD fits the definition for deterministic discrete simulations. It has the capability of modeling spatial space with a limited set of samples. It is differentiated from other predicting models by making a correlation between samples in spatial space and predict values of the new points by using this correlation coming from spatial space. Basic assumption behind the Kriging is that the sample points are correlated, and the correlation depends on the distance between the sample points. The distance and direction between the sample points determine the spatial correlation which is used to predict the response surface. In addition to that, Kriging also generates an estimation region for all interpolated regions. Detailed theoretical background is explained in Kleijnen’s studies ([18], [20], [21]).

The Kriging surrogate model consists of two main parts which are deterministic part and stochastic part. The deterministic part tries to catch up the general trend of the functions and stochastic part comes with confidence level of the points. The deterministic part models the general trend of the function and is kept as simple as it can be. Most common function which is used in deterministic part is constant. Other options for it are linear and low order polynomials. The Kriging model is the summation of deterministic part and realization of stochastic process and can be formulized as follows:

$$y = \sum_{i=1}^{k} \beta_i f_i(x) + Z(x)$$

(2)
Stochastic part assumes that points have random distribution in nature and the relations between points are based on the distance and direction between sample points. The stochastic part is a probabilistic model with no mean and only concerning the variance since the trend is coming from the deterministic part. As it is mentioned, the relation of points is based on the distance and direction of the points which concludes with the correlation/covariance matrix containing the relations of each point with the others. There is no restriction to represent the correlation matrix except being a positive semi-definite matrix. In this study, Matern 3/2 correlation function is used. The stochastic nature can be formulized with the correlation/covariance between the spatial properties with zero mean as follows:

\[
cov(Z(x^{(i)},x^{(j)})) = \sigma^2 R(x^{(i)},x^{(j)})
\]

### 2.3 Leave-One-Out Cross-Validation (LOOCV)

K-fold cross validation is the most common method used in the validation of machine learning models. The best way to measure the performance of the model is to investigate the performance parameters such as root mean square error (RMSE) and \(R^2\) score for the non-training data. For this purpose, entire data should be splitted into training and test data. It is obvious that train data should not be used for measuring performance and test data should not be used for training. Although there are some suggestions for splitting in the literature, results highly depend on the discretization and distribution of the test and training data. K-fold cross validation can avoid this dependency by splitting data into k subset and making k times cross-validation for all subsets. Each data set is both tested and trained in different models. As a result, this methodology avoids the dependency of the results on chance. K-fold cross-validation concludes with very good results about the accuracy of the model. However, on the other hand, it is not efficient for total times of validation since the same procedure is repeated k times. Leave-one-out cross-validation (LOOCV) is a K-fold cross validation method in which k is exactly equal to the number of data points. In LOOCV, all points except one point are used to train the model and separated one point is used to test the performance of the model. LOOCV is the edge of cross validation which means that cross validation applied as many times as the amount of data. Regarding process time it is totally inefficient for larger data sets such as thousands of samples. In contrast to that, it is very reliable to use in small data sets such as surrogate model optimization for less than 150 samples. For large data sets k is generally taken as 10 and range is changed from 20 to 50 according to computer power and complexity of the model. \(R^2\) and RMSE are calculated to investigate the consistency of the model as performance parameters.

### 2.4 Bayesian Optimization

Bayesian optimization, first defined by [22], is a derivative free optimization technique which has an ability to find global minima/maxima points for very expensive black box functions. It became very popular with the achievement of the tuning hyperparameter in neural networks [13]. Common points for application of the Bayesian optimization are optimizing functions acting as black boxes which means that no close form, such as derivatives, is expensive to evaluate.
The strategy underlying the Bayesian is to act as a random function and make a prior relation on it. It is assumed that prior relation captures the behavior of the function. After evaluating the functions over the sample points, prior is updated to posterior distributions, which concludes with an acquisition function to find the extreme points in the space. Bayesian optimizations can be grouped into sampling, Gaussian regression, Bayes rules and acquisition function. Sampling methodology is quite the same as the sampling methods which are discussed in the first part of this section.

Gaussian process regression (GPR), which can be considered the same Krigeing model, is a stochastics approach behind the Bayesian optimization. GPR assumes that any finite numbers of samplers have no relation to each other. In the Bayesian frame accept that prior distribution with no evidence is acting like a random distribution in nature. GPR uses random prior distribution to make a probabilistic model with a specific mean vector and covariance matrix. The mean vector can be defined by evaluating the mean function at every point. Constructing covariance matrix is calculated by covariance functions or kernels for each pair of the points. Kernel functions, which have large positive correlations between close points, are generally used. Only requirement for the correlation matrix is positive semi definite. The idea of the correlation depending on the distance in the spatial space, is same with the Krigeing model. Parameters which are in the kernel and mean function should be defined with some approaches. Three main approaches to determine them are maximum likelihood estimate (MLE), maximum a posteriori (MAP) and fully Bayesian approach. MLE is used in this study.

Gaussian regression concludes with posterior distribution for any points. Next step is how to determine the next sample using surrogate model and posterior distributions. Acquisition function is defined to find the following points in the space. The most common and successful one is “Expected Improvement” (EI) which is also selected in this study. Its success comes with a special name for this algorithm, EGO which is basically Bayesian optimization using EI. EI is instead of looking for the best improvement point, searching for expected improvement in the next point. It can be achieved by probability density function of normal distribution over the expected value. Theoretical background is explained detailed in [4]. Other well-known acquisition functions are entropy search and knowledge gradient.

2.5 Unsteady Flow Solver (FLOWUnsteady)

FLOWUnsteady, the doctoral study of [23], is an open-source unsteady flow solver for unsteady aerodynamics and aeroacoustics which is coded in Julia. The theory behind the solver is the reformulated vortex particle methods (rVPM) which is a mesh-free CFD framework solving the large eddy simulation (LES) filtered incompressible Navier-Stokes equations in their vorticities form. The solver uses the Lagragian (meshless) scheme which can not only prevent the disadvantages coming from generation of mesh, but also retain the structure of the vorticity over the distance while minimizing the numerical dissipation. Generally, classical VPM formulation ignored the filtered terms. In addition to that, neighbor effect of the close particles is also vanished in the solution of the classical VPM solution. Classical VPM also assumes that the shape of particle is constant along the streamlines. This assumption violates the conservation of mass and momentum in the transition and turbulent regimes which is accepted as the
main reason of the numerical instability in the classical VPM. The classical VPM is numeric instability which causes breaking structure of the vorticity close to the turbulent regime.

FLOWUnsteady avoids this disadvantage by reformulating VPM as LES which is numerically stable which also does not increase the computational time. Classical VPM assumes that shape of particle is constant along the streamlines, in contrast to that, reformulating VPM formulates the change of the particle shape by conserving angular momentum of the particles. Deformation on the particle intensifies the vorticity in the direction that the element is stretched. The other approach for modeling deformation of fluid particles is conservation of mass by using existence of vortex lines and vortex tubes. In this approach, vorticity evolves as the material lines, being identically stretched, and reoriented by the velocity field.

The advantages of the mesh-free solver are listed as not suffering from dissipation due to mesh itself, integration with coarser discretization without losing physical accuracy and derivatives calculated analytically. Most impressive feature is 100 times faster than conventional mesh CFD solver. FLOWUnsteady has capability of solving tilting wing and rotor, rotors with variable RPM and variable pitch. It can also be used for different levels of fidelity.

The validation test case of unsteady flow solver for delta wing is investigated in the Avarez’s doctoral study [23] for FLOWUnsetady. The experiment, used in the validation of the low-speed flow over the 45-deg swept back wing, was conducted by Weber et al. [24].

3 Results

The aim of this study is to develop an optimization tool which is based on stochastic approach and can be parallelized and updated if needed. In this section, one test function which is Rosenbrock test function and one design problem which is design of delta wing are investigated to present the performance of the optimization tool.

3.1 Surrogate Model Optimization for Rosenbrock Function

The surrogate model optimization is validated by applying optimization process to the Rosenbrock test function which is used to test optimization algorithms and has a definite optimum point. It has a valley or banana shape which causes difficulty in finding the optimum point. The first step in optimization process is the defining the problem which is “optimizing $X_1 \in [-4, 4]$ and $X_2 \in [-4, 4]$ to find minimum point of the Rosenbrock function”. Using LHS method to initialize 20 samples for Rosenbrock function. Fig. 2 shows that the real values of Rosenbrock functions for initial samples and dashed lines corresponds to optimum point after first iteration which only uses initial samples to find the optimum point. Despite the minimum value of the Rosenbrock function for initial samples is equal to nearly 20, optimization tool finds the optimum point value as 0.9914.

Fig. 1 shows how the samples are distributed for incremental sampling methods and LHS. As expected from the algorithms, the homogeneity in the space is increasing over the iterations. Instead of increasing the accuracy of the model in the local region, it
improves all over the space, so that it minimizes the chance of missing a global optimum point.

![Fig. 2. Initial samples’ results of the Rosenbrock functions.](image)

**Table 1** demonstrates that $R^2$ is nearly 1 in all models. $R^2$ is sometimes not enough to explain the accuracy of the real and the model results. It is advised to check other error parameters such as RMSE. In addition to $R^2$, it is better to investigate RMSE as a more accurate parameter to investigate the accuracy of the model. From **Table 1**, decreasing RMSE values, closing $R^2$ value to 1 and investigating new optimum point with increasing iteration show that increment sampling algorithm improves the optimization process and helps to find better optimum point.

**Table 1.** Results of surrogate model and numerical simulation and $R^2$, RMSE ($F_m$ is result of surrogate model, $F_r$ is result of Rosenbrock function).

<table>
<thead>
<tr>
<th># of samples</th>
<th>$X_{opt} = [X1,X2]$</th>
<th>$F_m(X_{opt})$</th>
<th>$F_r(X_{opt})$</th>
<th>$R^2$</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>[1.9956, 3.9840]</td>
<td>-0.5087</td>
<td>0.9914</td>
<td>0.999993</td>
<td>5.781</td>
</tr>
<tr>
<td>26</td>
<td>[0.7403, 0.5527]</td>
<td>0.1172</td>
<td>0.0695</td>
<td>0.999999</td>
<td>3.692</td>
</tr>
<tr>
<td>31</td>
<td>[0.7403, 0.5527]</td>
<td>0.1632</td>
<td>0.0695</td>
<td>1.000000</td>
<td>3.142</td>
</tr>
<tr>
<td>36</td>
<td>[0.7403, 0.5527]</td>
<td>0.1772</td>
<td>0.0695</td>
<td>1.000000</td>
<td>1.417</td>
</tr>
<tr>
<td>42</td>
<td>[0.9449, 0.9059]</td>
<td>0.0070</td>
<td>0.0020</td>
<td>1.000000</td>
<td>0.628</td>
</tr>
<tr>
<td>53</td>
<td>[0.9853, 0.9670]</td>
<td>-0.0400</td>
<td>0.0016</td>
<td>1.000000</td>
<td>0.439</td>
</tr>
<tr>
<td>58</td>
<td>[0.9853, 0.9670]</td>
<td>-0.0387</td>
<td>0.0016</td>
<td>1.000000</td>
<td>0.357</td>
</tr>
<tr>
<td>63</td>
<td>[0.9853, 0.9670]</td>
<td>-0.0345</td>
<td>0.0016</td>
<td>1.000000</td>
<td>0.302</td>
</tr>
<tr>
<td>68</td>
<td>[0.9853, 0.9670]</td>
<td>-0.0028</td>
<td>0.0016</td>
<td>1.000000</td>
<td>0.287</td>
</tr>
</tbody>
</table>

### 3.2 Surrogate Model Optimization for Delta Wing

Surrogate model optimization tool is applied for practical relevance which is determined as shape optimization of delta wing using CFD. Root and tip twist angles are selected as optimization variable parameters to maximize ($C_l/C_D$). Twist angles directly affect the vortex shading behind the wings which is not easy to capture with empirical or semi-empirical methods. The baseline configuration, which is tested by Weber et al. [24], is selected as the low-speed flow over the 45-deg swept back wing. It is recommended to use CFD tools to capture this phenomenon. Optimization process
with initial sampling points is represented in Fig. 3 which includes CFD results of samples, the optimum point after 1st iteration and the baseline configuration. CFD results of samples form a scattered distribution of various values in the design spaces. Fig. 3 shows that the first iteration has already improved the objective function \((C_L/C_D)\) according to the baseline configuration.

\(R^2\), which also represents the consistency of both the surrogate model and CFD, are tabulated in Table 2. Closeness of \(R^2\) to 1 and small differences between the surrogate model and CFD results which are tabulated in Table 2 support the trustworthiness of the results of the surrogate model. The behavior of the numerical simulation can be varied in accordance with the variables and the design space. In this case, many local optimum points are very close to the global optimum point, which can also be realized from Table 2. The difference between CFD tool and the surrogate model can be more than the difference between local and global optimum points. For such a situation, although the surrogate model comes with different optimum points, results of the real function in local and global optimum points are expected to get closer to each other. It is concluded that the optimization process leads to efficient results. Small decrement in the 3rd iteration is an example of this phenomenon. As \(R^2\) highly depends on the location of the sampling points, adding new points to the design space causes fluctuation in \(R^2\) values. Decrement in \(R^2\) is caused by the change of the confidence level resulting in adding new points.

![CFD results](image)

Fig. 3. CFD results \((C_L/C_D)\) of initial samples and optimum design.

The configuration which is performed in the wind tunnel test has zero tip and root twist angle. The CFD results for wind tunnel test configuration for \(C_L\), \(C_D\) and \((C_L/C_D)\) are measured 0.238, 0.005 and 48.57 respectively. The objective function, which is \((C_L/C_D)\), is being tried to be maximized in the optimization problem. The optimum point found by the surrogate model built in the 1st iteration with initial sample points is [0.4628, -2.3741]. The \((C_L/C_D)\) coming from the optimization cycle is 51.77. After the optimization loop \((C_L/C_D)\) value increased by 6.6%.
Table 2. Results of surrogate model and numerical simulation and $R^2$, RMSE ($F_m$ is result of surrogate model, $F_{CFD}$ is result of CFD, $\theta$ is twist angle)

<table>
<thead>
<tr>
<th># of samples</th>
<th>$X_{opt} = [\theta_{root}, \theta_{tip}]$</th>
<th>$F_m(X_{opt})$</th>
<th>$F_{CFD}(X_{opt})$</th>
<th>$R^2$</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>[0.4628°,-2.3741°]</td>
<td>51.67</td>
<td>51.78</td>
<td>0.997</td>
<td>0.550</td>
</tr>
<tr>
<td>26</td>
<td>[0.4628°,-2.3741°]</td>
<td>51.78</td>
<td>51.78</td>
<td>0.998</td>
<td>0.591</td>
</tr>
<tr>
<td>31</td>
<td>[0.5029°,-2.4188°]</td>
<td>51.78</td>
<td>51.75</td>
<td>0.997</td>
<td>1.520</td>
</tr>
<tr>
<td>37</td>
<td>[0.4186°,-2.2099°]</td>
<td>51.80</td>
<td>51.81</td>
<td>0.975</td>
<td>1.014</td>
</tr>
<tr>
<td>42</td>
<td>[0.4186°,-2.2099°]</td>
<td>51.81</td>
<td>51.81</td>
<td>0.987</td>
<td>0.827</td>
</tr>
<tr>
<td>47</td>
<td>[0.4186°,-2.2099°]</td>
<td>51.81</td>
<td>51.81</td>
<td>0.992</td>
<td>0.688</td>
</tr>
<tr>
<td>52</td>
<td>[0.4186°,-2.2099°]</td>
<td>51.81</td>
<td>51.81</td>
<td>0.995</td>
<td>0.617</td>
</tr>
<tr>
<td>57</td>
<td>[0.4186°,-2.2099°]</td>
<td>51.81</td>
<td>51.81</td>
<td>0.996</td>
<td>0.546</td>
</tr>
<tr>
<td>62</td>
<td>[0.4186°,-2.2099°]</td>
<td>51.81</td>
<td>51.81</td>
<td>0.997</td>
<td>0.479</td>
</tr>
<tr>
<td>67</td>
<td>[0.4186°,-2.2099°]</td>
<td>51.81</td>
<td>51.81</td>
<td>0.998</td>
<td>0.369</td>
</tr>
</tbody>
</table>

4 Conclusion

In this study, a surrogate model-based optimization algorithm is developed and tested for the test function of Rosenbrock, and delta wing design applications. Finding minimum point of Rosenbrock test function as a validation test case and delta wing optimization test cases as an aerodynamic design application, are investigated to test the optimization algorithm and the parallel design strategy. In the delta wing design case, the twist angle of the selected delta wing configuration is optimized.

The conclusions drawn from this study shows that The Kriging surrogate model is used as an effective stochastic method to model a high-fidelity numerical simulation, which are obtained by an unsteady flow solver FLOWUnsteady and the Bayesian inference successfully finds the optimum point for the selected test function and the delta wing design optimization cases.

Increasing the number of samples in the design algorithm, which is used to determine samples in each iteration, is suitable for parallelization. The parallel design strategy demonstrates that, with iterations, design time decreases and the consistency of the surrogate model with the real mathematical functions (i.e., exact solutions) increases. Although this strategy may increase the total CPU time to find an optimum point, the parallelization of the runs results in a decrease in design time.

The results of test function show that the design algorithm works properly. This can be observed from the $R^2$ score and the difference between the numerical simulation and the surrogate model results.

The future work regarding this study may be recommended as enlarging by including more related design parameters such as aspect ratio, taper ratio, sweep angle, twist angle, and incidence angle and can be conducted for different flow regimes. In addition, interdisciplinary design test cases can be investigated. In addition to that, the hybrid parallel optimization tool developed in this study can also be used together with higher fidelity numerical simulation tools such as Navier-Stokes equations based CFD flow solvers or Lattice Boltzmann flow solvers.
References

2. Krige, D.: A statistical approach to some mine valuation and allied problems on the Witwatersrand, University of the Witwatersrand, Faculty of Engineering, Witwatersrand, (1951)
14  Y.E. Sunay and N. Sezer-Uzol

Optimizing Object Detection Training Over Diverse Datasets

Pramit Dutta¹, Edgar Galvan² and John McDonald³

¹ Dept. of CS, Hamilton Institute, Maynooth University, Ireland
pramit.dutta.2021@mumail.ie

² Dept. of CS, Hamilton Institute, Maynooth University, Naturally Inspired Computation Res. Group, Ireland
edgar.galvan@mu.ie

³ Dept. of CS, Hamilton Institute, Maynooth University, Ireland
john.mcdonald@mu.ie

1 Introduction

Object detection in images is one of the classical problems in computer vision and has been widely studied [1]. Numerous models and methods have been proposed in the literature to cater to object detection problems in different application areas [2]. In most recent works the object detection is usually solved by pre-training either the model or the backbone on a large dataset, e.g., MS-COCO[3] or ImageNet [4] and then fine-tuning the model using a application specific curated dataset. Many pretrained models are widely available in the literature [5–7]. However, there are certain caveats to this approach. Firstly, this method works best when the final application and data characteristics are similar to the pre-training dataset. Secondly, the problem significantly changes when there are limited datasets available to pre-train the model for a specific application. This work addresses these two problems in the context of human search and rescue using infra-red (IR) images from unmanned aerial vehicles (UAV). There are a few reasons behind selecting this use case. Firstly, unlike object detection in ground-based platforms, object detection using drones pose some additional challenges [8]. Some of these include, varying altitude, thus significantly changing the bounding box scales for the same object class. Similarly, varying camera angles provide different visual perception of the same object class. Secondly, for IR-based human detection, only a few open datasets are available to effectively train the model. Lastly, as in most computer vision problems, the final application, can vary significantly from the training setup. This can be in terms of camera, environmental and topographic parameters, etc. All these datasets provide a diverse set of training samples. However when training a single model to learn over diverse datasets, various imbalances like class distribution, scale in-variance, etc., can skew the model learning and its performance towards certain parameters. This works aims to study and develop a strategy to train an object detection model by posing the learning over datasets as an optimization problem. Using an empirical approach, a model is trained to detect humans from IR based images by mixing of available datasets. The main contributions of the work are, firstly, to understand if object detection training over different datasets can be posed as a multi-objective optimization problem and secondly, to evaluate mixing strategies to improve zero-shot cross-dataset performance where the model has not seen certain datasets during the training.

2 Experimental Setup & Methodology

Model Architecture: Several deep learning models for object detection are available in the literature. These models are typically classified into anchor-based [3, 6] and anchor-free models [7]. The choice of the object detector architecture, for this work, is guided by a lightweight model with reasonable training and inference time. An anchor-based Faster-RCNN [6] model is chosen with a mobile-net V2 [9] backbone as a common architecture for all experiments.
Table 1: Thermal Pedestrian Detection Datasets

<table>
<thead>
<tr>
<th>Dataset Name and Reference</th>
<th>Object Classes in dataset</th>
<th>Number of Images</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAU[10]</td>
<td>Pedestrians</td>
<td>1941</td>
<td>Train: 1000 / Test/Val: - Fixed camera, low altitude (~9m)</td>
</tr>
<tr>
<td>NII[12]</td>
<td>Pedestrians</td>
<td>2633</td>
<td>Train: 485 / Test/Val: - UAV based, mid altitude (20-50m)</td>
</tr>
<tr>
<td>OSU[13]</td>
<td>Pedestrians</td>
<td>227</td>
<td>Train: 57 / Test/Val: - Fixed Camera, low altitude (~20m)</td>
</tr>
</tbody>
</table>

Fig. 1: Bounding-Box to Image Area Distribution of Datasets

Fig. 2: Model performance (mIoU score) per test dataset

Datasets: Selection of datasets for UAV based IR detection of humans is a crucial part of the work. There are only a limited number of thermal datasets available for training an object detector model for human/ pedestrian object detection from UAV perspective. Table 1 gives an overview of the publicly available datasets used as a part of the work reported here in. While some of the datasets have been crafted from actual IR camera footage obtained from drones flying at various altitude ranges [11, 12], other datasets are created by capturing IR footage from stationary cameras mounted at a certain altitude overlooking a street or field [10, 13]. The datasets also provide a mix of city and rural/open environments. Nevertheless, when considered in tandem they all provide a good variation of flight altitudes, perspectives and environmental conditions that is essential for this work. To evaluate the mixing strategies of the datasets and optimize the training of the model, we analyse the distribution of the bounding boxes (b-box) to image area ratio for all datasets, as shown in Fig 1.

Methodology: The experiments evaluate two specific objectives of this work. Firstly, they compare the effect of random mixing or simple bagging (SB) of the datasets to a more structured approach using multi-objective optimization (MOO). In SB strategy we combine two datasets using a B/L method[14], where B is the total mini-batch size and L is the number of datasets used in the mixing. Thus, for every mini batch we select equal number of images from each of the mixing datasets, find the total loss for the B images and then update the model using the final loss. In MOO, we use the Pareto optimal approach (POA) as proposed in [15] for multi-task learning. Unlike in SB approach, for each mini-batch individual loss and gradients are calculated for each set of B/L images of the mixing datasets. A multiple gradient descent algorithm[15] is then used to calculate a weighted set of gradients across the datasets which are used to update the model weights. The second objective is to compare the model performance from training setup to actual field data. To represent this, in each setup, the model is trained with both SB and POA approaches for two datasets, and its performance is tested not only on mixed datasets used in training but also on datasets that the model has not seen during training.

Training Details: The models are trained from scratch without any pre-initialized weights. A standard object-detection loss function[6] is used for Faster-RCNN. The model training is stopped when there is no significant improvement in the test set of the training datasets. The training uses
Table 2: Preliminary Results of Model Training Using SB and POA Approach
†: results on test set of the datasets used in training; ∗: results on test set of unseen datasets

<table>
<thead>
<tr>
<th>Mix #</th>
<th>Datasets Mixed</th>
<th>Strategy</th>
<th>mIoU Results on Test Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>AAU</td>
</tr>
<tr>
<td>Mix 1</td>
<td>AAU + HIT</td>
<td>SB</td>
<td>0.16†</td>
</tr>
<tr>
<td></td>
<td></td>
<td>POA</td>
<td>0.17†</td>
</tr>
<tr>
<td>Mix 2</td>
<td>NII + AAU</td>
<td>SB</td>
<td>0.29†</td>
</tr>
<tr>
<td></td>
<td></td>
<td>POA</td>
<td>0.28†</td>
</tr>
<tr>
<td>Mix 3</td>
<td>HIT + NII</td>
<td>SB</td>
<td>0.05∗</td>
</tr>
<tr>
<td></td>
<td></td>
<td>POA</td>
<td>0.03∗</td>
</tr>
</tbody>
</table>

SGD optimizer with an initial learning rate of 5e-3, momentum of 0.9 and weight decay of 5e-4. The learning rate is halved every 50 epochs. Hyper-parameter tuning is not performed during the training. All models are trained using a batch-size of 6 images per dataset in both the SB and POA approaches using two Nvidia 1080Ti GPU. The mean Intersection over Union (m-IoU) [16] score is used to evaluate the model performance.

3 Preliminary Results

Table 2 shows the initial results from each of the experiments performed as a part of this work. Column 2 shows the various mixing of the datasets followed by the training strategy in column 3. The final m-IoU scores on the test sets of the datasets for the final trained model is given from column 4-7. The first set of experiments with Mix-1 shows marginal improvement of results using POA approach over the SB approach. For Mix-2, results for POA closely flows the SB approach for the test sets of the training datasets. However, it is interesting to note that the performance on the unseen datasets, using POA, has increased multi fold times. This can be attributed to the wide range of bounding-box to image area ratio covered by the two datasets, refer Fig. 1. The POA approach helps to drive the model performance where none of the datasets dominates the model training which helps to boost the performance on unseen datasets as well. As expected, for Mix-3, where the training only covers a small subset of the bounding-box to image area ratio, although performance in the self-training datasets is closely matched, there is no significant improvement in the performance for the unseen dataset. Our results indicate that when training models over multiple diverse datasets for object detection, multi-objective optimisation strategies can provide a principled approach to improve model performance not only over the datasets on which they is trained but also on zero-shot cross dataset tasks.

4 Conclusions

Training object detection models for real-world applications can be challenging, where coverage of the problem domain requires training across a diverse datasets. This also becomes an impediment towards deployment of model for actual applications where the camera setup and environmental parameters might be different from the datasets used for model training. Recently, Pareto optimal approaches have been evaluated for optimization of model learning for both single and multi-task training. This work shows that an object detection model can be optimally trained by mixing of datasets using a Pareto optimal approach. This improves the model performance on both known and unknown datasets. Further investigation and ablation studies are necessary to generalise this approach for a broader set of models, datasets, and algorithms.

5 Acknowledgments

This work has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number SFI 18/CRT/6049 and in part by a research grant from Science Foundation Ireland (SFI) under Grant Number 16/RI/3399.
References

Nature-Inspired Techniques for Combinatorial Reverse Auctions in Electricity Consumption

Sifat E. Jahan\textsuperscript{1}, Malek Mouhoub\textsuperscript{1}\textsuperscript{[0000–0001–7381–1064]}, and Mehdi Sadeghilalimi\textsuperscript{1}

Department of Computer Science, University of Regina, Regina SK, Canada
\{sjt588, mouhoubm, msv368\}@uregina.ca

\textbf{Abstract.} With the growing electricity demand, the likelihood of experiencing power outages is also rising. Utility companies have started buying electricity through online auctions to address this issue. To meet the increasing electricity demand, we propose a solution to procure energy from various sources, trading off multiple objectives while solving a complex winner(s) determination problem for resource procurement optimization. Winner determination is an NP-hard problem, and applying exact methods is impractical. Instead, we rely on nature-inspired techniques as they are appropriate for trading the quality of the returned solution for the required running time. In particular, Genetic Algorithms (GAs), Whale Optimization Algorithms (WOAs), Ant Colony Optimization (ACOs), Particle Swarm Optimizations (PSOs) and Firefly Algorithms (FAs) are explored and evaluated in terms of effectiveness in producing high-quality solutions for several instances of the Combinatorial Reverse Auction (CRA) problem.

\textbf{Keywords:} Combinatorial reverse auctions · Electricity auctions, Retail markets · Winner determination · nature-inspired techniques

\section{1 Introduction}

The growing population, infrastructure, and economic growth have heightened the electricity demand, prompting numerous countries to consider infrastructure reforms to meet this growing demand\cite{1, 2}. Online electric auctions are gaining popularity due to their potential to enhance efficiency and competitively boost suppliers’ markets\cite{1, 3, 4}. Purchasing electricity online from different sources is a new strategy that utility companies are considering to resolve power outage issues during peak hours. Previous strategies were based on a recommendation system controlling the customer’s consumption rate \cite{5}. These strategies have several concerns about the consumer’s privacy as recommendations were based on their lifestyles. Online electricity auctions can utilize various energies like variable energy (wind or solar), active controllable load (batteries, heat storage), and controllable renewable energy (biomass, hydroelectricity, geothermal heat)\cite{6}. Very few studies have been done on online electricity auctions employing CRAs, and the majority merely discuss the advantages of doing so. Few re-
searchers have explored the potential of addressing the electricity demand problem using exact and approximate methods. Moreover, online electricity trading is a complex and challenging task, with constraints making it difficult to find a solution for determining the winner [3]. In this paper, we report on a study that explores practical strategies for efficiently solving the winner determination problem for electricity consumption. More precisely, the paper addresses two objectives. The first one is to evaluate a CRA model for electricity consumption that will provide quality solutions efficiently, considering different constraints for both the buyer and suppliers, such as electricity price, quantity, and time slot. The second objective is to study and analyse variants of GAs and WOAs, and other nature-inspired techniques, including ACO, PSO and FA to determine their suitability for CRAs.

2 Background

A Combinatorial Reverse Auction (CRA) is a complex problem that determines the auction winner and can be solved using different types of approximate or exact algorithms. However, there is minimal research on CRAs for electricity markets, and designing auctions with bidding techniques can be challenging. Multi-attribute auctions are also complex and time-consuming, with features beyond price, such as service quality or duration, proposed in [7]. Whether sealed or open, online auctions have distinct properties and relationships with buyers and sellers[8, 9]. Sealed bidding involves investments from new and existing sellers, while open bidding favors existing sellers. These factors influence the design of e-auctions. Most of the literature on electricity auctions highlights their benefits. However, it lacks a dedicated mechanism for design and experimental analysis, often limiting auction features to reduce the complexity of the CRA problem.

2.1 Electricity Auction

Despite their efficiency in resource allocation and profit maximization, combinatorial electricity auctions have yet to be thoroughly researched due to their complexity. Designing CRAs for the electricity sector is quite challenging [3]. Teleich et al. proposed theoretical foundations for multi-attribute e-auctions in B2B transactions, including a negation mechanism for purchasing multiple product units[10]. Few studies have demonstrated the significant impact of CRAs on the electricity market in both the government and private sectors [11], and few studies have efficiently conducted electricity combinatorial auctions, which consider only price [12]. In [13], the authors propose a double combinatorial auction protocol called probability bidding mechanism (PBM) for electricity auction mechanisms, aiming for price minimization and quantity maximization, but claim it lacks actual implementation. Moreover, a recent study on electricity auctions aims to reduce buyer costs by using single units and attributes, with a user-friendly interface and IBM CPLEX optimizer for winner determination[11]. Electricity auctions have significant economic impacts, particularly in Brazil, a
leading country in renewable energy production [4]. These auctions can maximize buyer profits and have been studied in European countries for time management, bidding strategies, and electricity production prices. Proper implementation can lead to significant annual cost savings in electricity market auctions [14]. Online auctions face a high likelihood of misbehavior and fraudulent activities, necessitating careful consideration of fraud mechanisms to ensure fairness. A study analyzing fraud types and detection mechanisms has been reported in [15], introducing a novel method for monitoring in-action fraud (IAF).

2.2 Evolutionary Winner Determination Techniques

Metaheuristics are efficient alternative solving methods that trade running time for the quality of the returned solutions when solving a given optimization problem [16]. Metaheuristics include nature-inspired techniques. The Winner Determination Problem, involving multiple attributes, is a complex task [17] with limited studies. In [18], the authors discuss different contexts for multi-attribute auctions. Using the Nash Equilibrium principle and assuming all bidders and sellers follow the same principle can ensure the stability of CRAs in large market auctions as discussed in [19]. Evolutionary techniques efficiently solve combinatorial reverse auctions. The study in [20] introduces a Genetic Algorithm-based technique for optimizing resource allocation in combinatorial auctions, thereby maximizing sellers’ sold quantities and efficiently solving the winner determination problem. Moreover, GAs have been utilized in energy sectors to optimize warehouse service time and storage policies and to solve multi-attribute combinatorial dispatching problems in multiprocessor environments [21, 22]. In [17, 23], the GAMICRA technique (based on GA) is proposed as a highly achievable method for solving the winner determination problem. Another study proposes an innovative GA-based solution for CRA, considering features, limitations, and associated constraints in electricity market auctions [24].

3 Proposed Approach

3.1 Problem Description

The multi-attribute reverse auction is a winner determination problem. This type of problem can lead to infeasible solutions in real life using exact methods. Even though an optimal solution can be achieved, it will be time-inefficient. Approximate methods are preferred for solving this problem due to their practicality, efficiency, and potential for optimal or nearly optimal solutions [25]. In this research, we will explore evolutionary methods to solve the electricity auction winner determination problem. We will consider constraints like time slots and electricity prices, aiming to meet buyer and supplier demands for maximum quantity with low price. Moreover, buyers can purchase electricity from various suppliers, while suppliers can produce it from various sources. Winners may be multiple suppliers, but only one for each electricity demand item. To model the
Table 1. CSP Model and CRA Features

<table>
<thead>
<tr>
<th>CSP Features</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Variables</strong></td>
<td>Buyer (a buyer who wants to buy energy; normally, it should be a public utility company)</td>
</tr>
<tr>
<td></td>
<td>Sell items</td>
</tr>
<tr>
<td></td>
<td>Cost of the item</td>
</tr>
<tr>
<td></td>
<td>Type of energy</td>
</tr>
<tr>
<td></td>
<td>Time-slots</td>
</tr>
<tr>
<td><strong>Domains</strong></td>
<td>Domain(Buyer) = public utility company, individual buyer</td>
</tr>
<tr>
<td></td>
<td>Domain(Seller) = small utility companies, homeowners</td>
</tr>
<tr>
<td></td>
<td>Domain(Item) = [1 - N] (Here, N is a positive number)</td>
</tr>
<tr>
<td></td>
<td>Domain(Cost) = [minPrice – maxPrice] (Here, minPrice and maxPrice will be set according to the auction conditions; the maxPrice of an item is given by the buyer, and minPrice of an item is given by the seller)</td>
</tr>
<tr>
<td></td>
<td>Domain(Type of energy) = solar, wind, battery storage, battery of electric vehicles, heat storage, hydroelectricity, biomass, and geothermal heat</td>
</tr>
<tr>
<td></td>
<td>Domain(Time-slots) = 8 am - 5 pm (Period of 15 minutes)</td>
</tr>
<tr>
<td><strong>Constraints</strong></td>
<td>Constraint 1: bidPrice $\geq$ minPrice and bidPrice $\leq$ maxPrice</td>
</tr>
<tr>
<td></td>
<td>The energy bid must be between the minimum and maximum prices.</td>
</tr>
<tr>
<td></td>
<td>Constraint 2: itemCost $\geq$ minPrice and itemCost $\leq$ maxPrice</td>
</tr>
<tr>
<td></td>
<td>The cost of each item cannot exceed the maximum price and cannot be less than the minimum price.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CRA Features</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Reverse</td>
<td>Public utility companies or buyers can purchase energy from various sellers for peak hour supply, typically meeting demand for one or two hours. A seller can be a small company or home owner, and there will be different sources of energy as already mentioned in the CSP modeling.</td>
</tr>
<tr>
<td>Combinatorial</td>
<td>The auction will feature multiple items representing 15-minute time slots, allowing small sellers, such as home owners, to easily accommodate those 15-minute slots.</td>
</tr>
<tr>
<td>Conflicting attributes</td>
<td>Seller and buyer must compete in energy quantity and price, with buyers attempting to reduce prices and increase energy quantity. Buyers must specify demand period, power quantity, and price, while sellers must provide minimum price and production details like energy type and quality.</td>
</tr>
<tr>
<td>Auction</td>
<td>It must be sealed-bid, with no seller information shared, to promote competition and prevent collusion.</td>
</tr>
</tbody>
</table>

Table 2. CRA Process Step wise

<table>
<thead>
<tr>
<th>Steps</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electricity Demand</td>
<td>The buyer will specify the quantity desired (stating the maximum and minimum quantity to avoid any blackout period), the demand period (different time slots for the desired quantity), as well as different constraints such as the maximum price and the maximum electricity quantity.</td>
</tr>
<tr>
<td>Sellers Registration</td>
<td>The actual bidding will take place in this step, and there will be two rounds: in the first round, only variable energy source generators will be involved, and in the second round, the rest of the sources will be considered.</td>
</tr>
<tr>
<td>Winner Determination</td>
<td>The proposed algorithm will be applied to determine the winning winners.</td>
</tr>
<tr>
<td>Electricity Supply</td>
<td>The winners will provide the required electricity to the buyer,meeting all the necessary requirements.</td>
</tr>
</tbody>
</table>

auction problem, we need to consider constraints with reverse auction features, requiring the use of two concepts: Constraint Satisfaction Problem (CSP) [26] and Combinatorial Reverse Auction (CRA) features. The considered features are shown in Table 1. In the CRA system, there will be five steps until the winner is determined as shown in Table 2. The proposed CRA system will consider buyer requirements and sellers’ constraints at the beginning of the algorithm, requiring submission of quantity and price bids as input.

3.2 Nature-inspired techniques for the CRA

3.2.1 Genetic Algorithms

A Genetic Algorithm (GA) is an evolutionary algorithm based on natural selection, generating high-quality optimization solutions [27, 28]. A GA requires the selection, crossover, and mutation operators to create a GA-based solution. The selection operator utilizes an objective fitness function to select the best offspring for the next generation. The crossover merges parental genetic information to produce offspring, based on the current population. The mutation
Table 3. Variants of Selection, Crossover and Mutation operators

<table>
<thead>
<tr>
<th>Selection Techniques</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gambling-Wheel Disc Selection</td>
<td>Similar to spinning a roulette wheel, where each chromosome has a slice proportional to its fitness. The wheel randomly stops and chooses the parent from a slice, with a higher fitness value indicating a higher likelihood of selection.</td>
</tr>
<tr>
<td>Rank Selection</td>
<td>The selection probability of a chromosome depends on the fitness rank of the population.</td>
</tr>
<tr>
<td>Tournament Selection</td>
<td>The chromosomes are chosen from the set of chromosomes based on their highest fitness values.</td>
</tr>
<tr>
<td>Stochastic Universal Sampling (SUS)</td>
<td>Selection is based on the Gambling Wheel Disc Selection, with multiple selection points and a single wheel turn based on principles of proportion.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Crossover Techniques</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-point Crossover</td>
<td>A random point is selected and the bits to the right of that point are swapped between the two parent chromosomes.</td>
</tr>
<tr>
<td>Two-point Crossover</td>
<td>Two random points are selected and the bits from that chunk are swapped between the two parent chromosomes.</td>
</tr>
<tr>
<td>Uniform Crossover</td>
<td>Every chromosome segment is given equal weight, and if any segment is deemed relevant, it is changed using the coin flipping method.</td>
</tr>
<tr>
<td>Order Crossover</td>
<td>A permutation crossover involves swapping consecutive alleles from the first parent with remaining values placed in the child in the order of their appearance in the second parent.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mutation Techniques</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swap Mutation</td>
<td>Two points are randomly selected and the bits of their parents are exchanged.</td>
</tr>
<tr>
<td>Bit-Flip Mutation</td>
<td>One or several bits are chosen and flipped accordingly for the mutated offspring.</td>
</tr>
<tr>
<td>Scramble Mutation</td>
<td>Random points from the chromosome are chosen as per required bits length and scramble their content.</td>
</tr>
<tr>
<td>Inversion Mutation</td>
<td>The chromosome is randomly selected based on the required bits length and its contents are reversed for mutated offspring.</td>
</tr>
</tbody>
</table>

Aims to enhance genetic diversity in a population. Variants of these operators are listed in Table 3. The bid's feasibility will be checked for constraints, and initial chromosomes will be generated randomly. Later on, fitness values will be calculated using objective functions like quantity maximization and price minimization. The GA operators listed above will be used to improve chromosomes. To improve the quality other techniques will be considered such as crowding distance, which will be calculated using relative fitness function to achieve diversity without similar solutions. Also, the elitism technique will be considered for efficiency using an external population. Among the two solutions, the first one based on the current GA solution and the second one obtained from elitism and based on the fitness values from both populations, the best two offspring will be chosen for the next generation, and the process will be repeated for certain iterations if a winning solution is not found before. The algorithm will proceed to the second round with other sources if it cannot find a solution for all items using variable energies in the first round. The following formula is considered to define the chromosomes, and a binary string will be used for representation: 

\[ S < 2^{\text{reqBits}}, \text{ where } S \text{ is the number of sellers.} \]

The fitness function is a bi-objective function with two objectives: The quantity of the required item to maximize, and the related price to minimize. To find the fitness value, the weighted sum method [29] has been used. These two objectives have been considered, and they have been formulated as given in Equation 1, 2 and 3. For the diversity mechanism, the relative fitness formula as stated in Equation 4 has been used, where the fitness of the previous chromosome (\( \zeta - 1 \)) and next chromosome (\( \zeta + 1 \)) is considered, and the maximum and minimum fitness values of the population (p) are considered.

\[
\text{Fitness}(x) = \Sigma_{i=1}^{p} \text{utility}(i) \ast \text{weight}(i) \quad (1)
\]

\[
\text{utility}(i) = \begin{cases} 
(bidPrice_i - minPrice_i) & \text{if } (maxPrice_i - minPrice_i) > 0 \\
(bidPrice_i - minPrice_i) & \text{otherwise} 
\end{cases} \quad (2)
\]
weight(i) = \begin{cases} 
\frac{(\text{bidQuant}_i - \text{minQuant}_i)}{(\text{maxQuant}_i - \text{minQuant}_i)}, & \text{if } (\text{maxQuant}_i - \text{minQuant}_i) > 0 \\
((\text{bidQuant}_i - \text{minQuant}_i), & \text{otherwise}
\end{cases} 

relFitness(x) = \frac{|\text{fitness}_{\zeta+1} - \text{fitness}_{\zeta-1}|}{\text{maxFitness}(p) - \text{minFitness}(p)} \in \{0, 1\}

For each item, the corresponding seller’s utility and weight have been considered. The fitness value of chromosome x is the sum of all the items utility and weight based on the seller. The selection is based on the objective fitness function of the CRA system, which aims to maximize the quantity for the buyer at a reduced price. Gambling-Wheel Disc Selection, Modified Two-point Crossover, and Swap Mutation are considered as based techniques for comparison between the different techniques in [17] and a sample problem from [24].

### 3.2.2 Whale Optimization Algorithm (WOA)

WOA [30] has proven effective in addressing combinatorial optimization problems by incorporating shrinking encircling and spiral motions for exploitation and exploration, inspired by humpback whales. WOA can be seen as a combination of both the moth flame and the grey wolf techniques [31]. To adapt WOA to the discrete context of the CRA, we redefine the distance by the Hamming Distance (HD), representing the count of differing entries between two whales (chromosomes). In the exploitation phase’s shrinking encircling attack, whales approach their prey by rotating around it. For exploitation, we use the equations established for the spiral attack in [30] to guide the whales, utilizing HD to calculate their dissimilarity. We introduce a random parameter for increased diversity to balance shrinking and spiral attacks. During the exploration phase, we adopt a strategy similar to the exploitation phase in redefining the operators, with the only difference being the random selection of a whale instead of the optimal whale. In this context, we consider the following mutation variants.

- **Random Resetting Mutation (RRM):** Randomly select a set of entries from a whale followed by the random alteration of their values.
- **Swap Mutation (SwM):** Random select and exchange pairs of variable values.
- **Scramble Mutation (ScM):** Randomly select contiguous entries and scramble their values.
- **Inverse Mutation (IM):** Randomly select contiguous entries and invert their values.

### 3.2.3 Ant Colony Optimization (ACO)

ACO [32] has proven effective in tackling combinatorial optimization problems [33]. ACO leverages Swarm Behaviors and Self-Organizing principles to facilitate interaction and cooperation among population members. Specifically, ACO
Nature-Inspired Techniques for CRAs

draws inspiration from the natural and social behavior of worker ants in their quest for optimal paths to food sources. As ants move towards the food source, they release a chemical substance known as pheromone, which naturally dissipates over time. The ant discovering the shortest path back to the nest returns earlier, releasing pheromones along the way to reinforce the trail. Subsequently, other ants instinctively follow the path with the highest pheromone concentration, further fortifying the trail. Ultimately, the path with the most pheromones is deemed the best and represents the optimal route from the nest to the food source. To apply ACO for the CRA, a three-stage methodology is adopted. The initial stage involves the random construction of the ant population’s candidate solutions. Parameters $\alpha$ and $\beta$ dictate the relative importance of pheromone information (candidate solution variable values) and heuristic information. In the second stage, a local search is executed to determine which pheromones should be updated. The third stage involves the enhancement of previously identified pheromones. This process includes decreasing the influence of poor candidates through pheromone evaporation. These stages iteratively continue until the algorithm meets its termination condition.

3.2.4 Particle Swarm Optimization (PSO)
PSO [34] simulates the social behaviour of flying birds. Based on its own and its partners’ flying experiences, every particle modifies its flight path accordingly. In other words, it is a computational method that optimizes the candidate solution by iteratively improving the solution based on a quality measurement. To solve the CRA using PSO, a group of candidate solutions (swarms) are randomly generated with random velocities and positions at the beginning. Later, based on the objective function defined according to price minimization and quantity maximization, it is evaluated. Iteratively, the candidate solutions are improved to reach the optimal solutions. The standard parameters for inertia ($w$) and coefficients ($c_1, c_2$) considered are $w = 0.72984$ and $c_1 = c_2 = 2.05$.

3.2.5 Firefly Algorithm (FA)
FA is a bio-inspired optimization algorithm [35], inspired by the flashing behaviour of fireflies, where using bio-luminescence, fireflies communicate and attract mates. To solve the CRA using the FA, a group of candidate solutions (fireflies) is randomly generated with random positions initially. Later, based on the objective function, it is evaluated. The positions of each firefly are updated by considering its attractiveness towards other fireflies and their light intensity. The standard parameters for light intensity ($\gamma$), attractiveness ($\beta$), randomness ($\alpha$ and $\delta$) and considered are $\alpha, \beta = 1$, $\gamma = 0.975$ and $\delta = 0.01$.

4 Experimentation
The aim of the experiments reported in this section is to evaluate the performance of variants of the GA algorithm. The best variant is then compared to
the WOA algorithm and PSO algorithm. The experiments are implemented in Python and executed on Intel(R) Xeon(R) CPU with 2.20GHz processor speed and RAM of 12.7 GB. Each dataset is run 20 times as part of the CRA system implementation to determine the average, best, and standard deviation of both the computation time and solution quality. To compare the different metaheuristics variants, we use five datasets where the number of items and sellers are respectively: 8 and 5 for Dataset 1, 15 and 20 for Dataset 2, 75 and 50 for Dataset 3, 100 and 200 for Dataset 4, and 250 and 500 for Dataset 5. The execution times related to Dataset 5 is shown in Figure 1.

GA performs better when paired with a few attributes on larger datasets. The top charts in Figure 2 reveal that Rank selection and Tournament selec-
Fig. 3. Mutation Results (Computational Time)

Table 4. Computational Time Comparison Results

<table>
<thead>
<tr>
<th>Dataset</th>
<th>GA</th>
<th>WOA</th>
<th>ACO</th>
<th>PSO</th>
<th>Firefly</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg</td>
<td>Best</td>
<td>STD</td>
<td>Avg</td>
<td>Best</td>
<td>STD</td>
</tr>
<tr>
<td>1</td>
<td>24</td>
<td>21.43</td>
<td>2.33</td>
<td>24</td>
<td>18.87</td>
</tr>
<tr>
<td>2</td>
<td>35</td>
<td>32.80</td>
<td>2.99</td>
<td>25</td>
<td>16.29</td>
</tr>
<tr>
<td>3</td>
<td>152</td>
<td>133.87</td>
<td>31.83</td>
<td>61</td>
<td>56.12</td>
</tr>
<tr>
<td>4</td>
<td>240</td>
<td>179.44</td>
<td>83.08</td>
<td>104</td>
<td>69.24</td>
</tr>
<tr>
<td>5</td>
<td>601</td>
<td>479.21</td>
<td>188.95</td>
<td>386</td>
<td>301.33</td>
</tr>
</tbody>
</table>

Table 5. Quality Comparison based on Fitness Called

<table>
<thead>
<tr>
<th>Dataset</th>
<th>GA</th>
<th>WOA</th>
<th>ACO</th>
<th>PSO</th>
<th>Firefly</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>Avg</td>
<td>Best</td>
<td>STD</td>
<td>Avg</td>
<td>Best</td>
</tr>
<tr>
<td>1</td>
<td>250</td>
<td>200</td>
<td>145.59</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>2</td>
<td>200</td>
<td>200</td>
<td>0</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>3</td>
<td>200</td>
<td>200</td>
<td>0</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>4</td>
<td>200</td>
<td>200</td>
<td>0</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>5</td>
<td>200</td>
<td>200</td>
<td>0</td>
<td>200</td>
<td>200</td>
</tr>
</tbody>
</table>

In terms of all datasets, Tournament Selection offers better timing and high-quality solutions outperform Gambling Wheel Disc selection, while One-point crossover and Uniform crossover outperform Modified Two-point crossover due to longer chromosome feasibility checks. The bottom charts in Figure 2 show that Rank selection and Random selection outperform Gambling Wheel Disc selection, while One-point crossover, Uniform crossover, and Order crossover outperform other techniques. The study found that rank selection, random selection, and tournament selection were effective selection techniques for dataset 3. For dataset 4 and 5, Tournament selection outperforms all selection techniques in time and quality. Moreover, Uniform crossover is the most effective crossover method for datasets 3, 4, and 5. For mutation criteria, Swap mutation technique outperforms any other technique for the CRA model in all datasets. Overall, in terms of all datasets, Tournament Selection offers better timing and high-quality solutions.
The research study plans to explore larger-scale electricity markets and different consumption to assess the quality of solutions in terms of time and complexity. Experimental analysis, comparing GA and WOA algorithms for electricity market restructuring. The proposed methodology benchmark was tested through appropriate CRA model, offering economic benefits and potential infrastructure.

CRA techniques can significantly enhance the electricity market by enhancing technology and addressing rising energy demand. This study proposes an electricity market auction design, addressing constraints and implementing an appropriate CRA model, offering economic benefits and potential infrastructure restructuring. The proposed methodology benchmark was tested through experimental analysis, comparing GA and WOA algorithms for electricity market consumption to assess the quality of solutions in terms of time and complexity. The research study plans to explore larger-scale electricity markets and different nature-inspired techniques to determine the best trading off mechanism, while also exploring exact techniques like ant colony or brute-force.

## Table 6. Quality Comparison based on Solution Fitness

<table>
<thead>
<tr>
<th>DS</th>
<th>Rnd</th>
<th>GA</th>
<th>WOA</th>
<th>ACS</th>
<th>PSO</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1st</td>
<td>0.95</td>
<td>1.00</td>
<td>0.75</td>
<td>0.00</td>
<td>88.00</td>
</tr>
<tr>
<td>2</td>
<td>1st</td>
<td>0.95</td>
<td>1.00</td>
<td>0.75</td>
<td>0.00</td>
<td>146.30</td>
</tr>
<tr>
<td>3</td>
<td>1st</td>
<td>0.95</td>
<td>1.00</td>
<td>0.75</td>
<td>0.00</td>
<td>32.18</td>
</tr>
<tr>
<td>4</td>
<td>1st</td>
<td>0.95</td>
<td>1.00</td>
<td>0.75</td>
<td>0.00</td>
<td>82.10</td>
</tr>
<tr>
<td>5</td>
<td>1st</td>
<td>0.95</td>
<td>1.00</td>
<td>0.75</td>
<td>0.00</td>
<td>31.08</td>
</tr>
<tr>
<td>6</td>
<td>1st</td>
<td>0.95</td>
<td>1.00</td>
<td>0.75</td>
<td>0.00</td>
<td>1.15</td>
</tr>
<tr>
<td>7</td>
<td>1st</td>
<td>0.95</td>
<td>1.00</td>
<td>0.75</td>
<td>0.00</td>
<td>75.95</td>
</tr>
<tr>
<td>8</td>
<td>1st</td>
<td>0.95</td>
<td>1.00</td>
<td>0.75</td>
<td>0.00</td>
<td>63.78</td>
</tr>
<tr>
<td>9</td>
<td>1st</td>
<td>0.95</td>
<td>1.00</td>
<td>0.75</td>
<td>0.00</td>
<td>45.86</td>
</tr>
<tr>
<td>10</td>
<td>1st</td>
<td>0.95</td>
<td>1.00</td>
<td>0.75</td>
<td>0.00</td>
<td>33.96</td>
</tr>
<tr>
<td>11</td>
<td>1st</td>
<td>0.95</td>
<td>1.00</td>
<td>0.75</td>
<td>0.00</td>
<td>21.96</td>
</tr>
<tr>
<td>12</td>
<td>1st</td>
<td>0.95</td>
<td>1.00</td>
<td>0.75</td>
<td>0.00</td>
<td>9.15</td>
</tr>
<tr>
<td>13</td>
<td>1st</td>
<td>0.95</td>
<td>1.00</td>
<td>0.75</td>
<td>0.00</td>
<td>79.90</td>
</tr>
<tr>
<td>14</td>
<td>1st</td>
<td>0.95</td>
<td>1.00</td>
<td>0.75</td>
<td>0.00</td>
<td>18.04</td>
</tr>
<tr>
<td>15</td>
<td>1st</td>
<td>0.95</td>
<td>1.00</td>
<td>0.75</td>
<td>0.00</td>
<td>19.13</td>
</tr>
<tr>
<td>16</td>
<td>1st</td>
<td>0.95</td>
<td>1.00</td>
<td>0.75</td>
<td>0.00</td>
<td>10.05</td>
</tr>
<tr>
<td>17</td>
<td>1st</td>
<td>0.95</td>
<td>1.00</td>
<td>0.75</td>
<td>0.00</td>
<td>9.53</td>
</tr>
<tr>
<td>18</td>
<td>1st</td>
<td>0.95</td>
<td>1.00</td>
<td>0.75</td>
<td>0.00</td>
<td>33.43</td>
</tr>
<tr>
<td>19</td>
<td>1st</td>
<td>0.95</td>
<td>1.00</td>
<td>0.75</td>
<td>0.00</td>
<td>34.47</td>
</tr>
<tr>
<td>20</td>
<td>1st</td>
<td>0.95</td>
<td>1.00</td>
<td>0.75</td>
<td>0.00</td>
<td>33.47</td>
</tr>
</tbody>
</table>

solutions, increasing dataset size, while **Uniform crossover** performs best in crossover and **Swap mutation** in mutation for this problem. WOA, ACO, PSO and FA are compared with the best GA operators from previous results, including tournament selection, uniform crossover, and swap mutation. Table 4 reveals that WOA demonstrated superior execution time compared to GA, PSO and FA. Table 5 shows FA was worst for smaller dataset to get a complete solution and WOA, ACO and GA performed better. For larger dataset all did well based on solution quality. Table 6 shows that for the first round WOA got better quality solutions in most datasets except dataset 1, while in the second round GA performed better in all datasets. ACO performance was not better than WOA. PSO performance is better in larger datasets than the smaller ones and FA’s performance is not good enough compared to other approaches.

## 5 Conclusion

CRA techniques can significantly enhance the electricity market by enhancing technology and addressing rising energy demand. This study proposes an electricity market auction design, addressing constraints and implementing an appropriate CRA model, offering economic benefits and potential infrastructure restructuring. The proposed methodology benchmark was tested through experimental analysis, comparing GA and WOA algorithms for electricity market consumption to assess the quality of solutions in terms of time and complexity. The research study plans to explore larger-scale electricity markets and different nature-inspired techniques to determine the best trading off mechanism, while also exploring exact techniques like ant colony or brute-force.
References


A Bayesian Optimization Approach to Algorithm Parameter Tuning in Constrained Multiobjective Optimization

Jordan N. Cork\textsuperscript{1,2}[0069–0066–6849–4088] and Bogdan Filipi\textsuperscript{c}\textsuperscript{1,2}[0000–0003–4428–4255]

\textsuperscript{1} Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia
\textsuperscript{2} Jožef Stefan International Postgraduate School, Jamova cesta 39, 1000 Ljubljana, Slovenia
\{jordan.cork, bogdan.filipic\}@ijs.si

Abstract. Algorithm parameter tuning is an often neglected step in the optimization process. This study shows that constrained multiobjective optimization can benefit significantly from tuning, in both the specialized (for an individual problem) and generalized (over a number of problems) parameter setting approaches. Numerical experiments conducted with three multiobjective optimization algorithms on 139 test problems from 13 benchmark suites quantify the algorithm performance improvement on individual problems. Additionally, regarding the generalized approach, alternative default parameter settings are identified. The study also identifies Bayesian optimization as an adequate method for tuning multiobjective evolutionary algorithms with constraint handling. Overall, it is concluded that, given sufficient computational resources to apply to a problem, parameter tuning, using an approach such as Bayesian optimization, should be conducted. If computational resources do not allow such tuning, then the proposed default parameters are applicable.

Keywords: Constrained multiobjective optimization · Multiobjective evolutionary algorithms · Algorithm parameter tuning · Bayesian optimization · Benchmark problems.

1 Introduction

Algorithms are rarely parameter free. They generally involve several parameter settings of varying types and ranges. It is often up to the person implementing or applying the algorithm to decide upon these settings. To make matters more difficult, settings are often codependent, meaning they must be treated as a whole [11]. To account for the human error, algorithm parameter tuning can be automated. There are various such approaches, including, among others [17], Bayesian optimization (BO) [28].

Evolutionary algorithms are no exception to having multiple tunable settings [11]. This paper concerns the tuning of multiobjective evolutionary algorithms (MOEA) used to handle constrained multiobjective optimization problems (CMOP). Determining appropriate parameter settings for an MOEA to
suit the specific CMOP, prior to the algorithm run, is called parameter tuning. The process can be automated or done by hand, and is a critical aspect of the algorithm design step. Different from tuning is parameter control, where the parameters are changed during the algorithm run [11, 12]. This paper deals specifically with parameter tuning.

CMOP function evaluations vary in time and resource costs. For some problems, however, expensive evaluations are less of a concern and significant resources can be allocated towards finding the best approximations of the optimum. These may be called design problems and require specialized parameter settings [13]. In other cases, this opportunity cannot be afforded and a suitable set of generalized parameter settings are required to have the best chance of success [12, 17]. For single-objective evolutionary algorithms, in both these domains, automated parameter tuning has been well studied, including using BO [17, 27]. Multiobjective optimization, and in particular constrained multiobjective optimization, lacks such studies. We address this lack by conducting an empirical study of automated parameter tuning of MOEAs for solving CMOPs, considering both specialized and generalized parameter settings.

Due to its efficiency in handling costly evaluation processes, BO is used as the algorithm parameter tuner. To test the effects of parameter tuning using BO, MOEAs of three types are considered: dominance-, indicator- and decomposition-based, with one algorithm tested of each type. These are applied to a large set of test problems from several benchmark suites, both artificial and real-world. To address both specialized and generalized settings, two experiments are conducted. The first one tunes the three algorithms on each problem individually, while the second one tunes them on an aggregation of all problems.

The work primarily contributes to the field of parameter tuning for MOEAs through the results and findings from the two experiments. The first experiment outlines the applicability of parameter tuning to MOEAs and reveals the effect of parameter tuning on the considered algorithms. The second experiment provides an insight into the ability to generalize, in the sense of searching for robust parameter values, and results in improved default settings.

The paper is further organized as follows. Constrained multiobjective optimization is detailed in Section 2. Following this, in Section 3, is the methodology, where the conduct of each experiment is presented, along with details of BO. The experimental setup is then described in Section 4. This is followed by the presentation of results in Section 5 and a discussion on the findings and limitations of the work, in Section 6. Finally, in Section 7, the paper concludes with a summary and plans for future work.

### 2 Constrained Multiobjective Optimization

A CMOP can be formulated as:

\[
\begin{align*}
\text{minimize} & \quad f_m(x), \quad m = 1, \ldots, M \\
\text{subject to} & \quad g_k(x) \leq 0, \quad k = 1, \ldots, K,
\end{align*}
\] (1)
where \( x = (x_1, \ldots, x_D) \) is a \( D \) dimensional solution vector, \( f_m(x) \) are objective functions, \( g_k(x) \) are constraint functions, \( M \) is the number of objectives, and \( K \) is the number of constraints.

If a solution \( x \) satisfies all constraints, \( g_k(x) \leq 0 \), for \( k = 1, \ldots, K \), it is considered feasible. Given two feasible solutions, \( x \) and \( y \), \( x \) dominates \( y \) if \( f_m(x) \leq f_m(y) \) for all \( 1 \leq m \leq M \), and \( f_m(x) < f_m(y) \) for at least one \( 1 \leq m \leq M \). Within the set \( S \), if there exists no feasible solution \( x \in S \) that dominates the feasible solution \( x^* \), then \( x^* \) is a Pareto-optimal solution. The set of all feasible solutions constitutes the feasible region \( F \), while the set of all non-dominated feasible solutions forms the Pareto set \( S_o \). The image of the Pareto set in the objective space is called the Pareto front, \( P_0 = \{ f(x) \mid x \in S_o \} \).

The ideal point is the point in the objective space consisting of the best objective values \( z_{\text{ideal}} = (\min_{x \in S_o} f_1(x), \ldots, \min_{x \in S_o} f_M(x)) \). The point in the objective space consisting of the worst fitness values across all solutions in the Pareto front is the nadir point \( z_{\text{nadir}} = (\max_{x \in S_o} f_1(x), \ldots, \max_{x \in S_o} f_M(x)) \).

The hypervolume indicator [36] measures the region dominated by a set of solutions, bounded by a given reference point. Its reliance on a reference point is, however, its downside. This is addressed in [16] with \( I^{\text{MOP}} \), represented by:

\[
I^{\text{MOP}}(A^T) = \begin{cases} \frac{1}{T} \frac{1}{d(A^T, Z)} & \text{if } A^T \leq \{ z_{\text{nadir}} \} \\ d(A^T, Z) & \text{otherwise} \end{cases}
\] (2)

where \( A^T \) represents an algorithm run after performing \( T \) evaluations and \( Z \) is the region of interest set by \( z_{\text{nadir}} \). In (2), \( I^{\text{HV}}(A^T) \) is the original hypervolume function, calculated following the objective space normalization using \( z_{\text{ideal}} \) and \( z_{\text{nadir}} \). Further,

\[
d(A^T, Z) = \inf_{(x,z) \in A^T \times Z} \| f(x) - z \| \quad \text{(3)}
\]

is the Euclidean distance of the solution closest to \( Z \).

An additional difficulty is encountered when considering quality indicators for CMOPs. In [32], Vodopija et al. proposed a quality indicator, where infeasible solutions are assigned the value of their overall constraint violation, and feasible solutions always score better than infeasible solutions:

\[
I^{\text{CMOP}}(A^T) = \begin{cases} \min_{x \in A^T} v(x) + \tau^* & A^T \cap F = \emptyset \\ \min(I^{\text{MOP}}(A^T \cap F), \tau^*) & \text{otherwise} \end{cases}
\] (4)

where \( \tau^* \) is a threshold value indicating the feasible region has been reached and \( v(x) \) is the solution’s overall constraint violation. Normalization of the distance and constraint values is conducted by taking the median values of these from a set of 100 random sample solutions. For more details on the implementation of \( I^{\text{CMOP}} \), see [32].

3 Methodology

Here, the use of the \( I^{\text{CMOP}} \) performance indicator is first justified, followed by the means for mediating the issues related to normalization. Then, BO is detailed,
and the parameter settings kept fixed in the experiments are mentioned. Finally, the processes underlying the two experiments are outlined.

As we were dealing with CMOPs, the use of the $f^{CMOP}_I$ quality indicator in assessing algorithm performance was appropriate. This was also, beneficially, usable for aggregation purposes, given its normalization of the objective, distance and constraint violation values.

For objective value normalization, $f^{CMOP}_I$ requires both the nadir and ideal points, or at least good approximations of them. These are available for all test problems, except for the RCM real-world problem suite [19], where only the nadir points were available. Obtaining the ideal points for these problems required finding good approximations of the Pareto fronts. This was achieved by running each of the MOEAs tested in the study 30 times and gathering non-dominated solutions from all the runs. From the final set of non-dominated solutions, the best value across each objective was taken to give the problem’s ideal point and the worst value for each objective for the nadir point. This nadir point approximation was then used in the experiments.

In BO, a probabilistic surrogate model is built, generally using a Gaussian process, to predict the quality and uncertainty around new solutions [28]. It was used in our study as an algorithm tuner, due to its ability to perform well on black-box optimization tasks. It was also well suited to handling the expensive quality functions found in algorithm parameter tuning [5]. One downside, however, to BO is that it requires a stochastic initialization to obtain a set of starting solutions. Additionally, a stochastic acquisition function was used. Due to the cost of parameter tuning, accounting for this stochasticity with multiple runs was impracticable and, therefore, a certain degree of variation is to be expected from further runs. A guideline for the use of certain covariance functions, i.e., kernels, such as the Matérn kernel, is that they are applied to stationary problems. The problem considered in this paper, concerning tuning of MOEAs, can be considered stationary, despite its stochastic nature. This is because it retains the same statistical properties at each execution. Additionally, the stochasticity is reduced through the conduct of multiple MOEA runs [26].

Tuning was conducted on quantitative parameters, which altered the instance of the MOEA, as opposed to qualitative parameters, which would have made the MOEA a different algorithm altogether. Two settings were kept consistent throughout all experimental runs. The first was the number of solution evaluations. This was kept at 10,000 for each algorithm run. This number was selected for consistency and to limit computation time. The second setting kept consistent was the number of algorithm runs. To derive a robust measure, 30 runs were conducted on each problem. From this, the mean was taken, providing a singular result.

The first experiment addressed the following questions. Does algorithm parameter tuning improve the performance of MOEAs, and, if so, by how much? Is it worth the effort to tune, given the cost? What can be learnt from the distributions of parameter settings? For this, each algorithm/problem combination was tested, which is known as competitive testing [12]. The process was
Algorithm Parameter Tuning in Constrained Multiobjective Optimization

The results (the $I_{CMOP}$ performance indicator values) using untuned algorithms, i.e., those with default parameter settings, were gathered. The algorithms were then tuned using BO, where performance improvement was the optimization objective. Finally, the difference in performance between the untuned and tuned algorithms and the parameter setting distributions were analyzed.

The second experiment tested the generalizability of parameter settings, by testing the MOEAs on the aggregation of all problems. The purpose was to determine the possibility of generalizing parameter settings for improved average performance and to find what these were for the tested algorithms. This involved first taking the mean of the $I_{CMOP}$ values of the untuned algorithms from the first experiment. This was used as a baseline. Then, each algorithm was tuned on all problems collectively, by testing on all the problems and taking the mean of the results, i.e., of the $I_{CMOP}$ values. The improvement of average performance was the optimization objective for BO and the resulting algorithm parameter values were proposed as alternative default settings.

4 Experimental Setup

The details of the experimental setup were as follows. Python was used as the sole programming language, with Scikit-optimize used for BO [23] and Pymoo [3] for all other optimization tasks. As mentioned, $I_{CMOP}$ was used as the performance indicator. The value for $\tau^*$ was set to one.

A total of 139 test problems from 13 benchmark suites, including artificial and real-world ones, were used in the experiments. Their characteristics are summarized in Table 1. The problems referred to as Classic consist of the following: BNH [2], TNK [31], SRN [29], OSY [25], and WB [7]. The implementations came from various sources. Pymoo was one particular source, while some others were taken from [33] and original Python source code. Those without prior implementations in Python were implemented by hand.

Due to restrictions in computational resources, certain problems were omitted. The number of problem objectives was limited to two and three. There are 95 two-objective and 44 three-objective problems. Five-objective problems, present in some suites, were omitted. Other omissions, such as those problems from the DOC and ZXHCF suites, were done due to unstable performance of the algorithms on them. Moreover, the number of problem decision variables, when able to be scaled, were kept at their default values, specified in the original publications or implementations. As well, the DAS-CMOP difficulty setting was set to one, i.e., the first difficulty triplet, for each problem in the suite. Additionally, none of the available problems contained equality constraints. While the RCM suite originally contained equality constrained problems, these problems are missing Python implementations and, therefore, were not included in the study.

Three MOEAs were tested. These consisted of one of each algorithm type: NSGA-II [8] as a dominance-based algorithm, SMS-EMOA [1] as an indicator-
Table 1. The test problem suites used in the algorithm parameter tuning experiments.

<table>
<thead>
<tr>
<th>Problem suite</th>
<th>No. of variables</th>
<th>No. of objectives</th>
<th>No. of constraints</th>
<th>Real-world</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-DTLZ [18]</td>
<td>10</td>
<td>3</td>
<td>1–3</td>
<td>no</td>
</tr>
<tr>
<td>CF [34]</td>
<td>10</td>
<td>2, 3</td>
<td>1, 2</td>
<td>no</td>
</tr>
<tr>
<td>Classic</td>
<td>5/5</td>
<td>2–6</td>
<td>2–6</td>
<td>mixed</td>
</tr>
<tr>
<td>CRE [30]</td>
<td>7/8</td>
<td>3–7</td>
<td>2, 3</td>
<td>1–11</td>
</tr>
<tr>
<td>CTP [9]</td>
<td>8/8</td>
<td>5</td>
<td>1, 2</td>
<td>no</td>
</tr>
<tr>
<td>DAS-CMOP [15]</td>
<td>9/9</td>
<td>30</td>
<td>2, 3</td>
<td>7, 11</td>
</tr>
<tr>
<td>DCDTLZ [21]</td>
<td>6/6</td>
<td>10</td>
<td>3</td>
<td>1–4</td>
</tr>
<tr>
<td>DOC [22]</td>
<td>7/9</td>
<td>6–16</td>
<td>2, 3</td>
<td>6–10</td>
</tr>
<tr>
<td>LIR-CMOP [14]</td>
<td>14/14</td>
<td>30</td>
<td>2, 3</td>
<td>2, 3</td>
</tr>
<tr>
<td>MW [24]</td>
<td>14/14</td>
<td>10</td>
<td>2, 3</td>
<td>1–4</td>
</tr>
<tr>
<td>NCTP [20]</td>
<td>18/18</td>
<td>10</td>
<td>2</td>
<td>1, 2</td>
</tr>
<tr>
<td>RCM [19]</td>
<td>21/50</td>
<td>2–10</td>
<td>2, 3</td>
<td>1–11</td>
</tr>
<tr>
<td>ZXHCF [35]</td>
<td>15/16</td>
<td>13</td>
<td>3</td>
<td>1–4</td>
</tr>
</tbody>
</table>

based algorithm and C-TAEA [21] as a decomposition-based algorithm. These algorithms were selected due to their availability in Pymoo, from where the implementations were obtained, and their in-built ability to handle constraints. For NSGA-II and SMS-EMOA, constraint handling takes place during survivor selection, where the constrained dominance principle is applied [8]. C-TAEA applies a unique two-archive approach to constraint handling. The default parameter values used in the experiments were those set in Pymoo.

The implementation of the BO algorithm was taken from Scikit-Optimize [23]. To avoid the case of tuning the algorithm tuner, the BO algorithm was applied with its default settings. Primarily this included using the Gaussian process (GP) method and the GP hedge acquisition function, which probabilistically selects from the following acquisition functions: lower confidence bound, negative expected improvement and negative probability of improvement. Additionally, a Matérn kernel was used and the algorithm was run for 100 solution evaluations. What must be set during the setup of the BO algorithm are the MOEA parameters to be tuned. These are different for each of the MOEA types. Note that crossover and mutation types were kept consistent for all algorithms, with simulated binary crossover (SBX) and polynomial mutation (PM), from [10], used, respectively.

The parameters for all algorithms are noted in Table 2. The dominance- and indicator-based algorithms have the same set of parameters. These are population size (pop_size) and parameters for SBX and PM. The SBX parameters are: the probability of crossover occurring (prob_cross), the probability of a variable participating in crossover (prob_var) and a magnitude parameter for crossover (eta_cross). The PM parameters are: the probability for a variable to be mutated (prob_mut) and the magnitude of mutation (eta_mut). The decomposition-based algorithm parameters also include parameters for SBX and PM. The differences lie in the parameters related to reference vectors. Two types (ref_type) are used:
Algorithm Parameter Tuning in Constrained Multiobjective Optimization 7

Table 2. The algorithm parameters tuned in this work, divided according to algorithm type.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Range</th>
<th>Dominance</th>
<th>Indicator</th>
<th>Decomposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>pop_size</td>
<td>integer</td>
<td>10–250</td>
<td>x</td>
<td>x</td>
<td>–</td>
</tr>
<tr>
<td>ref_type</td>
<td>integer</td>
<td>1, 2</td>
<td>–</td>
<td>–</td>
<td>x</td>
</tr>
<tr>
<td>ref_points</td>
<td>integer</td>
<td>10–250</td>
<td>–</td>
<td>–</td>
<td>x</td>
</tr>
<tr>
<td>partitions</td>
<td>integer</td>
<td>4–64</td>
<td>–</td>
<td>–</td>
<td>x</td>
</tr>
<tr>
<td>prob_cross</td>
<td>float</td>
<td>0.01–1.0</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>prob_var</td>
<td>float</td>
<td>0.01–1.0</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>eta_cross</td>
<td>integer</td>
<td>1–50</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>prob_mut</td>
<td>float</td>
<td>0.01–1.0</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>eta_mut</td>
<td>integer</td>
<td>1–50</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

Table 3. Default parameter settings for the algorithms used. The term ‘das-dennis’ refers to the Das-Dennis method. It requires the partition parameter, as opposed to the ref_points parameter.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>NSGA-II</th>
<th>SMS-EMOA</th>
<th>C-TAEA</th>
</tr>
</thead>
<tbody>
<tr>
<td>pop_size</td>
<td>100</td>
<td>100</td>
<td>–</td>
</tr>
<tr>
<td>ref_type</td>
<td>–</td>
<td>–</td>
<td>das-dennis</td>
</tr>
<tr>
<td>ref_points</td>
<td>–</td>
<td>–</td>
<td>NA</td>
</tr>
<tr>
<td>partitions</td>
<td>–</td>
<td>–</td>
<td>12</td>
</tr>
<tr>
<td>prob_cross</td>
<td>0.9</td>
<td>0.9</td>
<td>1.0</td>
</tr>
<tr>
<td>prob_var</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>eta_cross</td>
<td>15</td>
<td>15</td>
<td>30</td>
</tr>
<tr>
<td>prob_mut</td>
<td>0.9</td>
<td>0.9</td>
<td>0.9</td>
</tr>
<tr>
<td>eta_mut</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

In the Riesz s-energy method [4], where the number of reference points (ref_points) must be specified, and the Das-Dennis method [6], where the number of partitions must be specified. The ref_type setting is binary, i.e., discrete, and, therefore, had to be dealt with accordingly by the BO algorithm. This was handled by using two integers to represent the respective settings. The default parameter values used are documented in Table 3.

5 Results

The approximated Pareto fronts for the RCM problems were attained, using the approach described in Section 3. In 90% of cases, at least one point on the approximated Pareto front was able to dominate the approximate nadir point, proposed originally in [19]. The nadir points taken from the newly attained Pareto fronts were then used as reference points in the \(I_{CMOP}\) calculations.

To assist in the readability of the \(I_{CMOP}\) results, we provide the following explanation. Values are minimized and generally lie between –1 and 2. Values lower than this range are due to approximated Pareto fronts being bettered. Indicator values greater than this range are due to exceptionally high constraint
Table 4. Average $I_{CMOP}$ results from the first experiment, showing lower (i.e., better) results with tuning. Also presented are the number of problems for which the results were improved through tuning.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Untuned algorithm ($I_{CMOP}$)</th>
<th>Tuned algorithm ($I_{CMOP}$)</th>
<th>Difference</th>
<th>Problems improved</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSGA-II</td>
<td>$-0.2025$</td>
<td>$-0.3662$</td>
<td>$-0.1637$</td>
<td>135/139</td>
</tr>
<tr>
<td>SMS-EMOA</td>
<td>$-0.1611$</td>
<td>$-0.3369$</td>
<td>$-0.1758$</td>
<td>137/139</td>
</tr>
<tr>
<td>C-TAEA</td>
<td>$-0.0912$</td>
<td>$-0.3095$</td>
<td>$-0.2183$</td>
<td>138/139</td>
</tr>
</tbody>
</table>

Table 5. The number of problems for which feasible solutions were found in the first experiment.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Untuned algorithm (feasible solutions)</th>
<th>Tuned algorithm (feasible solutions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSGA-II</td>
<td>133/139</td>
<td>135/139</td>
</tr>
<tr>
<td>SMS-EMOA</td>
<td>134/139</td>
<td>135/139</td>
</tr>
<tr>
<td>C-TAEA</td>
<td>124/139</td>
<td>132/139</td>
</tr>
</tbody>
</table>

violation values. With $\tau^*$ set to 1, this constitutes the feasibility barrier, with values $\leq 1$ being feasible and values $> 1$ being infeasible.

In the first experiment, the tuned algorithms outperformed the untuned ones on the vast majority of problems and the mean improvement was significant. These results are shown in Figure 1 and Table 4. Feasibility was also improved. This is shown in Table 5. C-TAEA proved most tunable, however, this is likely due to the poorer performance using untuned parameters.

When the parameter value distributions for each algorithm across all problems were calculated, they showed there is no one setting applicable to all problems. The shape of each parameter distribution was relatively flat across the entire available range, with the exception of upticks in the end values. As well, from observations taken during tuning, various setups were able to perform similarly, indicative of the combinatorial nature of parameter settings. For example, some setups encouraging crossover performed similarly to setups doing the opposite.

As seen in Table 6, the outcomes of the second experiment were also positive. Tuning was able to improve the algorithms, in the attempt to generalize on the whole set of problems, with SMS-EMOA being most tunable in this sense. Therefore, alternative default parameter settings can be recommended, as shown in Table 7.

6 Discussion

The first experiment demonstrated the significant benefit of tuning MOEAs on individual problems. Conducting this, one can expect an improvement in the quality of the obtained Pareto front. The experiments showed untuned MOEAs were unable to reach the feasible region on certain problems. While this was only
Fig. 1. The results from the first experiment. The x-axis shows the difference between the $J_{CMOP}$ values of untuned and tuned algorithms, each averaged over 30 algorithm runs. The y-axis displays the benchmark problems, divided into suites. Suites are distinguished by color.
Table 6. The second experiment results, showing improvement with tuning, as $I_{CMOP}$ values are being minimized.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Untuned algorithm ($I_{CMOP}$)</th>
<th>Tuned algorithm ($I_{CMOP}$)</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSGA-II</td>
<td>−0.2025</td>
<td>−0.2561</td>
<td>−0.0536</td>
</tr>
<tr>
<td>SMS-EMOA</td>
<td>−0.1611</td>
<td>−0.2507</td>
<td>−0.0896</td>
</tr>
<tr>
<td>C-TAEA</td>
<td>−0.0912</td>
<td>−0.1225</td>
<td>−0.0313</td>
</tr>
</tbody>
</table>

Table 7. The proposed alternative default parameter settings for each MOEA, as derived from the second experiment. The term 'energy' refers to the Riesz s-energy method. It requires the ref_points parameter, as opposed to the partitions parameter, which is not applicable in this case.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>NSGA-II</th>
<th>SMS-EMOA</th>
<th>C-TAEA</th>
</tr>
</thead>
<tbody>
<tr>
<td>pop_size</td>
<td>31</td>
<td>21</td>
<td>−</td>
</tr>
<tr>
<td>ref_type</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>ref_points</td>
<td>−</td>
<td>−</td>
<td>10</td>
</tr>
<tr>
<td>partitions</td>
<td>−</td>
<td>−</td>
<td>NA</td>
</tr>
<tr>
<td>prob_cross</td>
<td>0.38</td>
<td>0.53</td>
<td>0.01</td>
</tr>
<tr>
<td>prob_yar</td>
<td>0.49</td>
<td>0.03</td>
<td>0.10</td>
</tr>
<tr>
<td>eta_cross</td>
<td>1</td>
<td>39</td>
<td>44</td>
</tr>
<tr>
<td>prob_mut</td>
<td>0.42</td>
<td>0.42</td>
<td>0.04</td>
</tr>
<tr>
<td>eta_mut</td>
<td>10</td>
<td>7</td>
<td>5</td>
</tr>
</tbody>
</table>

Despite the benefits, however, tuning is computationally expensive. Therefore, when addressing a single problem, the takeaway is, if computational resources allow, such as for design problems, tuning should be conducted.

The distributions of the tuned parameter values from the first experiment provide certain insights. The distributions, being broadly consistent across all parameters, showed no one particular algorithm setup, or rough approximation of one, suited all problems. In fact, the settings showed a rather equal distribution across the available range. This, and the observation that different setups performed similarly well, supports the notion, identified in [11], that parameters are codependent. The lesson from this is parameters are best tuned in a combinatorial manner, as opposed to individually.

In the second experiment, improvement was made with regards to the default parameter settings. Despite this, the average $I_{CMOP}$ values using the generalized parameter settings were still well below those of the specialized ones. This supports the notion, that should there be an allowance in computational resources, one should tune on the specific problem at hand. Should these resources not be afforded and should one desire to use the algorithms used in this experiment, then the proposed parameter settings should be used.

Bayesian optimization proved to be a high-performing and reliable tuner for MOEAs. This performance was well demonstrated in both experiments, where
significant performance improvements were observed. These improvements were achieved in at most 100 solution evaluations, far less than would be required of a meta-evolutionary algorithm for instance [17]. Bayesian optimization is therefore well suited to parameter tuning for MOEAs.

The following limitations were encountered during the study. Firstly, for simplicity’s sake, a subset of algorithm parameters were selected for tuning. The BO algorithm may have performed differently given a different subset. Several limitations relate to restrictions in computational resources. The most significant was the inability to conduct multiple runs of the BO algorithm. Another was the restriction of the number of solution evaluations to 10,000. Depending on the nature of the problem, a larger number of solution evaluations could be applied. Computational resource limitations also restricted which problems and algorithms were used. It is possible the outcomes would differ given different algorithms or problems with more objectives. These limitations are to be addressed in future work.

7 Conclusion

This study set out to determine the benefits of parameter tuning to MOEAs and made an attempt at generalization. It conducted these tasks through two experiments: one handling problems individually and the other aggregating results across all the considered problems. The benefit of tuning on singular problems was demonstrated in the first experiment and, in the second experiment, alternative default parameter settings were proposed for the algorithms used. In the vast majority of cases, the tuner of choice, Bayesian optimization, was able to improve upon the default settings, often significantly. Despite this, tuning is a computationally expensive task, requiring many runs of the optimization algorithm being tuned and remains only possible given the allowance of computational resources. This leads to the conclusion that, should this allowance be available, parameter tuning should be conducted.

In future work, as mentioned in Section 6, the experiments can be scaled up to, for instance, include many-objective problems or to cover a larger number of solution evaluations. Also of interest is the application of machine learning to the data acquired from the experiments. Given a set of features describing a CMOP, can the tunability of an algorithm or the parameter settings themselves be predicted? In future work, answers to these questions shall also be addressed.

Acknowledgements The authors acknowledge the financial support from the Slovenian Research and Innovation Agency (research core funding No. P2-0209 “Artificial Intelligence and Intelligent Systems”, and project No. N2-0254 “Constrained Multiobjective Optimization Based on Problem Landscape Analysis”). The publication is also based upon work from COST Action CA22137 “Randomised Optimisation Algorithms Research Network” (ROAR-NET), supported by European Cooperation in Science and Technology (COST).
References


Biologically-Inspired Algorithms for Adaptive Non-Player Character Behavior in Video Games

Sina Alizadeh1 and Malek Mouhoub1[0000–0001–7381–1064]

Department of Computer Science, University of Regina, Regina SK, Canada
{sau564, mouhoubm}@uregina.ca

Abstract. Video game designers frequently employ different movement algorithms to give players the impression of the intelligent and unpredictable motions of non-player characters or enemy agents. This paper explores the application of metaheuristic biological algorithms in video games to enhance the unpredictability and realism of enemy characters, moving away from traditional pre-programmed NPC behaviors. Developing the bespoke game "Run and Catch" incorporates the utilization of Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) for directing the movement of non-player characters. We tackle the premature convergence challenge by introducing an adaptive Finder–Tracker agent approach. This mechanism maintains diversity in enemy movements to ensure responsiveness to real-time player character activities. The findings illustrate the adverse effects of early convergence on both immersion and the variety of non-player character movements within a 2D game environment. Moreover, the study underscores the success of ACO and PSO, equipped with the Finder–Tracker agent framework, in reshaping group movements of non-player characters and effectively dealing with the convergence challenge.

Keywords: Bio-inspired Algorithms · Video Games · Non-Player Characters (NPCs).

1 INTRODUCTION

Over a decade since early AI research in gaming, "game AI" needs redefinition. AI developers are dealing with new issues relating to implementation, workflow, and game design in the context of growing game development activities. The gap between academia and industry hindered the adoption of traditional methods. Recent research focuses on player experience modeling, content generation, massive-scale player data mining, and alternative NPC capability enhancement, reshaping the game AI field [1]. Jesper Juul’s study analyzes the relationship between rules and storytelling in video games, using The Legend of Zelda and Grand Theft Auto as examples. By incorporating insights from diverse disciplines, the research charts the evolution of video games and highlights the fundamental connection between games and computers [2]. Researchers aiming for intelligent NPC behavior in video games particularly explore Champandard's
classification of planning models categorized as reactive and automated planning [3]. The primary focus of reactive planning is “how to do something”. This method entails constructing a decision system based on explaining the character’s behavior in response to environmental inputs. Finite State Machines (FSMs) and Behavior Trees (BTs) are two extensively used decision-making tools for reactive planning [4]. FSMs, an enduring decision-making technique, have found application in renowned video games like Pac-Man and Half-Life. FSMs boast a straightforward and intuitive design, but their complexity significantly escalates when attempting to implement more sophisticated behaviors [5]. Unlike reactive planning, automated planning focuses on “what the character can do” [6], and the answer to an artificial intelligence problem is found at runtime, that is, when the transitions between actions are dynamically established [7]. Traditional techniques, such as those modeling behavior using finite state machines and rule-based systems, are proving unable to meet the growing needs of today’s video games. In 2004, Orkin argued that a regressive real-time planning system can successfully meet the issues that game makers encounter. Furthermore, the study introduces and explores symbolic representation methodologies that have been applied to aid practical planning in the modern gaming world [8]. Goal-Oriented Action Planning (GOAP) is the most well-known type of automated planning, gaining popularity following its initial implementation in the FEAR video game. The GOAP approach is based on dynamically controlling a character’s repertoire of possible actions to attain specific goals systematically [8, 9]. In-game algorithm evaluation analyzes decision-making mechanisms shaping character behavior, AI, content generation, and overall game dynamics [10].

An NPC behaves differently from a human player because it follows computer program instructions. When discussing the creation of bot behaviors, NPC usu-
ally means an agent that can play a game on its [11]. Developing group-based movement algorithms is crucial for a group of NPCs, including ensuring purposeful behavior, emphasizing realism in movement, diverse actions, and achieving agent goals. Various artificial intelligence techniques are used to create game bots. Among various factors, a key element of NPC behavior revolves around their in-game movements. When an NPC is part of a collective of NPCs, implementing a movement algorithm based on group dynamics becomes crucial to simulating cohesive group behavior. These algorithms, collectively known as flocking algorithms, contribute to creating the appearance of purposeful group actions [12]. Flocking denotes the collective behavior of numerous interacting agents with a shared goal. Scholars across diverse disciplines, including animal behavior, physics, biophysics, social sciences, and computer science, have been intrigued by phenomena like flocking, swarming, and schooling in groups displaying local interactions. Examples of these agents include birds, fish, penguins, ants, bees, and crowds [13]. The primary drawback of flocking algorithms is that they use a central force model, which causes strange consequences, such as guiding all members of a dispersed flock to converge toward the flock’s centroid simultaneously [14]. Unlike the traditional method of pre-programmed NPCs’ movement (the flocking method), swarm intelligence algorithms enable emergence, meaning dynamic behavior. Researchers and game developers strategically utilize emergence to make enemy characters more unpredictable, immersive, and realistic. This approach enhances the gaming experience by introducing a level of complexity that mimics the spontaneity found in nature [15]. However, there needs to be more research on applying swarm intelligence to video games, as evidenced by [16]. When considering the analogy of collective behavior in a super-organism, exploring the application of swarm intelligence to video games emerges as a promising field. Swarm intelligence draws inspiration from the behavior of various groups of insects or animals, with key properties including self-organization, resilience to failures, and adaptive responses to the environment [17]. Ant colonies and particle-swarm intelligence are two notable examples of biological inspiration.

This research aims to develop bio-inspired methods (namely, ACO [18] and PSO [19, 20]) capable of producing successful enemy agents for video games. The ability of these methods to explore the game environment and find player characters is the main reason for choosing these methods for this study. In particular, we show that ACO and PSO are appropriate techniques for creating a variety of NPC behaviors in dynamic gaming situations. We will also show how bio-inspired methods can boost the diversity of NPC’s behavior equipped with the “Finder – Tracker agents” algorithm to deal with the convergence challenge, which leads to the limitation of movement variety. To demonstrate the advantage of enforcing ACO and PSO methods using the suggested “Finder – Tracker agents” approach in the diversity of NPC behavior, we introduce the designed and implemented bespoke titled “Run and Catch”. This comprehensive approach highlights the potential of bio-inspired metaheuristics [21, 22] for updating the NPCs’ behavior. This research provides practical solutions for addressing pre-mature convergence issues in NPC behavior within gaming.
2 Bio-inspired methods for game NPCs’ behavior design

As the complexity of modern games grows, developers are moving away from traditional artificial intelligence (AI) technologies in search of AI that is more scalable, versatile, and enables different behaviors in non-player characters (NPCs) [23]. The application of bio-inspired methods proves to be an effective approach to achieving realistic decision-making and movement in NPC behaviors. Using these methodologies, developers can tap into the potential of collective intelligence, driving inspiration from natural principles. Bio-inspired mechanisms, similar to the different behaviors displayed by individuals in a population, provide a flexible and dynamic method for creating variability in the behaviors of NPCs within the gaming environment. Bio-inspired approaches, which are frequently inspired by biological processes, are linked to the concept of population-based metaheuristics [24, 22]. These strategies make use of natural selection, adaptation, and collective intelligence characteristics observed in biological systems. Bio-inspired algorithms use natural processes to solve complex optimization and problem-solving problems. They have successfully solved numerous real-world problems by replicating the principles driving biological, physical, and ecological systems. The iterative decision-making process used in bio-inspired algorithms mimics the intricate interactions found in real systems. Each iteration needs an agent’s determination of its directional travel, which is based on three critical factors: alignment, cohesiveness, and separation. Alignment refers to an agent’s inclination to move in harmony with the general direction of nearby agents. Cohesion, on the other hand, drives the agent to converge with the average location of its neighbors, promoting collective motion. In contrast, the imperative of separation requires the agent to keep a safe distance from its neighbors, reducing the possibility of collisions. Here are some popular bio-inspired algorithms: Genetic Algorithms (GAs) [25], ACO [18], PSO [19, 20], Firefly Algorithm (FA) [26–28], Differential Evolution (DE) [29], and Grey Wolf Optimizer (GWO) [30]. Bio-inspired algorithms’ abilities to locate global optima in continuous search spaces, particularly, have been a source of substantial interest and success in the field of optimization. Bio-inspired metaheuristics exhibit key characteristics that contribute to their success in continuous search spaces, where the objective function operates over a continuous game domain. In this study, we consider the game environment as a 2D search space and the global optima as the player character.

3 ”Run and catch”: A bespoke game of algorithms evaluation

The custom-made game developed for this research is called “Run and Catch”. To show the abilities of bio-inspired methods in the variety of behaviors of NPCs in following the player character, we defined two different scenarios and levels based on PSO, and ACO. In this study, we consider the game environment as a 2D search space, with all the different levels and configurations represented as
The primary focus is modeling the game environment’s dynamic nature, treating it as a continuous and ever-changing landscape that NPCs and the player character navigate. The player character, representing the global optima in this context, serves as a dynamic point within this 2D search space, constantly changing its position based on the player’s character tendency and movement direction. Unlike traditional optimization problems where the global optima is a fixed solution, the global optima is inherently dynamic and subject to real-time changes in the game environment. Since all NPCs are attempting to find the position of the player character in the game environment, finding the dynamic global optima (player position) adds a layer of complexity to the study, requiring the ability to explore the free spaces of the game environment and avoid the convergence of non-player members that can effectively adapt to real-time changes. NPCs have two responsibilities: maintaining various behaviors while tracking the player’s character and occupying vacant spaces in the game environment to limit the player’s movement. The enemy members are unaware of empty spaces, and their movement is updated based on PSO and ACO algorithms. The player character’s position in the game environment is assumed to be the global optima, which is the goal for the NPCs to find and catch. The game features two distinct groups of characters. The first group comprises the player character, representing the individual controlled by the game’s user. The second group consists of non-player characters (NPCs), autonomously controlled entities within the game environment. In ”Run and Catch,” the gameplay unfolds as a thrilling adventure of growing challenges, with each level presenting players with a new encounter with NPC behaviors. A red square represents the player character, while NPCs are depicted as blue squares in the 800 x 600-pixel game environment. The main objective of the two levels (PSO and ACO) remains the same: outwitting and escaping the relentless pursuit of NPC enemies. The capacity to avoid these enemies and survive for a limited time defines success. Players win by mastering the art of evasion. The game becomes a super-exciting race where players stay focused and quickly escape from the computer-controlled characters to win. It is all about being fast and smart to come out on top. It is worth noting that the player character cannot engage in combat with or eliminate these NPCs. Both players and NPCs enjoy unrestricted freedom in their movement—there are no constraints on the directions they can go. Furthermore, the game environment lacks any barrier, allowing seamless navigation for players and NPCs as they explore the game environment. The number of enemies (NPCs) remains constant across the two levels. Specifically, the population for both PSO and ACO equals 50 non-player characters.

4 Player character movement

We calculate the player’s movement in a 2D game environment based on the mouse’s x and y positions and compares them to the mouse’s current x and y locations. The difference between the recorded x/y values and the mouse’s current x/y values indicates the mouse’s speed. The Euclidian distance from the last
and previous positions of the player character in the game environment is shown in Eq.2. The formula for updating the player character position and speed $v$ is given by Eq.1. $x_{\text{current}}$ and $y_{\text{current}}$ are the current mouse coordinates. $x_{\text{recorded}}$ and $y_{\text{recorded}}$ are the recorded mouse coordinates. $\Delta t$ is the time difference between the current and recorded positions. Significantly, the player’s character changes direction and speed depending on whether the player chooses to run or face enemies.

$$v = \frac{(x_{\text{current}} - x_{\text{recorded}})^2 + (y_{\text{current}} - y_{\text{recorded}})^2}{\Delta t}$$ (1)

$$d = \sqrt{(x_{\text{current}} - x_{\text{recorded}})^2 + (y_{\text{current}} - y_{\text{recorded}})^2}$$ (2)

The conscious direction and movement speed of the player vary based on the choice to run or stay in a spot.

4.1 NPC movements using PSO and ACO

**Level 1:** We use PSO to update the movements of non-player characters (NPCs) in the first level. PSO is a population-based method that generates a random initial population and employs the global communication of the swarm particles to find the global optima. PSO allows the movement of particles to their local best and stores and updates the best particle positions and the best global particle at each iteration. Each opponent particle is aware of its historical best position in games. This awareness is used to determine the direction of movement. Figure 1 depicts the game duration of updating NPCs using PSO. Notably, the NPC members reveal a convergence challenge in the 11th second (figure 1a), and this challenge continues until the final seconds of the game (figure 1b). The early convergence issue gives the player more areas (black areas) of the game space to move in and decreases the game’s attractiveness. Figure 1c shows how the NPCs members (blue squares) undergo early convergence by losing the distance among them. The average distance between NPC members and the player character determines the extent to which NPC members can react to sudden movements of the player character, essentially measuring their ability to surround the player, shown in Figure 1d.

**Level 2:** We apply ACO [18, 31] in the second game level to control non-player characters. Inspired by ant foraging behavior, ACO was developed to simulate how ants utilize pheromones to communicate and establish optimal pathways [18]. Artificial ants investigate solutions, laying virtual pheromones to indicate path quality. The iterative interplay of exploration and exploitation in ACO enables the search to converge toward optimal or near-optimal solutions over time. Like an ant, each NPC in the game has the power to express its findings and learn from the community’s collective knowledge. Like ants leaving pheromone trails, these characters communicate information about their prior successful journeys and the most successful route, affecting their movement selections. All
NPCs recalculate and adjust their paths as the player moves to adapt to the changing game situation. Figures 2a and 2b show the NPC movement updated based on the 11th and 40th seconds of the game and the inability of ACO to avoid the NPCs’ early convergence. Notably, due to the early convergence of NPCs, the black areas represent the available spaces for the player to navigate within the game. Figure 2c illustrates the average Euclidean distance among NPCs, showcasing a decreasing trend from the initial moments to the final seconds of the game’s runtime. Figure 2d shows the average Euclidean distance between NPCs and the player character is between 200 and 250, showing the inability of ACO to spread the NPCs members in the game environment.

4.2 Finder − Tracker agents mechanism

To regulate the convergence rate among the non-player characters (NPCs), represented by blue dots, within the context of this study, a mechanism called Finder − Tracker agents is proposed. The major purpose of this method is to prevent early convergence by allowing for a more robust exploration and maintaining NPCs’ movements in the “Run and Catch” game space. Besides, another advantage of the Finder − Tracker agents mechanism is that it evenly distributes NPCs in the game space to avoid creating free areas for the player character to move to. The proposed Finder − Tracker agents approach includes a predetermined threshold − distance and is used to improve convergence control with a population of enemies (NPCs). In this study, the NPC population is fixed.
at 50 blue dots, with 5 individuals designated as finders and the remaining 45 as trackers. Finder members consist of the 5 NPCs with the closest distance to the player character. These finder members play a crucial role in storing the last position as the best particles in both PSO and ACO for levels 1 and 2 respectively. Meanwhile, trackers are responsible for exploration and have the chance to replace finder members if they are closer to the player character. To manage the convergence of the blue dots during code execution, a threshold measure is introduced. During the Run and Catch game, the Euclidean distance between all tracker members is continuously computed. If the calculated distance between any pair of NPCs (blue tracker members) falls below the predefined threshold, these members are regenerated randomly in the game environment. It is noteworthy that finder members are exempt from this reset mechanism; otherwise, the implemented bio-inspired algorithm loses the best positions. Importantly, finder members are defined by their proximity to the red dot, and their positions do not change until they are no longer the closest members to the red dot. In such circumstances, due to the sudden movement of the player’s character (red), finder members may be replaced by tracker members, who have now become the red dot’s new nearest individuals. Algorithm 1 shows the steps of the Finder – Tracker agents approach to avoid convergence, which is compatible with both ACO and PSO.
Algorithm 1 Finder – Tracker agents algorithm

1: Initialize:
2: Set population size of the bio-inspired algorithm (PSO and ACO)
3: Set threshold – distance
4: Initialize positions of NPCs randomly
5: Main Loop:
6: while game play duration do
7:   Calculate Euclidean distances between all tracker members
8:   for each tracker1 in trackers do
9:     for each tracker2 in trackers do
10:       if tracker1 ≠ tracker2 and euclidean distance(tracker1, tracker2) < threshold – distance then
11:         Reset tracker position (tracker1)
12:         Reset tracker position (tracker2)
13:       if tracker1 ≠ tracker2 and euclidean distance<tracker1, tracker2) < threshold – distance then
14:         Update NPCs positions using the given bio-inspired method
15:       Find the closest NPC members to the player character:
16:       Set finders as the closest num_finders members
17:       Store Finder Positions in the game space (x, y)
18:       Check for movement of the player character:
19:       if finders are not the closest to the Player character then
20:         Replace Trackers that are closest to the player character with Finder members
21:       Store Best Finder Positions
22:   end while

5 In-game algorithm comparisons

Figures 3 and 4 depict the improved versions of PSO and ACO integrated with the Finder – Tracker agents algorithm. The effectiveness of the proposed method in preventing early convergence is evident, as reflected in the eye-catching reduced percentage of black areas, and ensures the preservation of a diverse range of NPC movements. Figures 3c and 4c illustrate the average Euclidean distance among NPCs updated based on PSO and ACO, respectively. Following each declining slope in both figures, there is an observable upturn in the NPCs distance, signifying the need for new NPCs’ regeneration. In other words, according to algorithm 1 (lines 8 to 12), the proposed method regenerates new members when the distance between them falls below the predefined threshold – distance which is the reason for the upward trend in figure 3c. Figures 3d and 4d indicate the average distance of NPC movements based on the enhanced PSO and ACO, incorporating the Finder – Tracker agents method, respectively. The proposed approach successfully maintains the average distance between NPCs and the player character, as indicated by the figures’ steady range, which fluctuates between 450 and 250. The significant difference observed in these numerical values implies a dual characteristic within the game environment. On one hand, it suggests that all non-player characters effectively converge to surround the player character closely, fostering an environment where potential interactions or challenges are close. also, the presence of numerous NPCs in the farthest areas of the game environment highlights the expansive and varied nature of the surroundings, contributing to a dynamic and diverse gameplay experience. This duality, reflected in the considerable spread of distances, underscores the nuanced spatial distribution and strategic positioning of NPCs throughout the gaming landscape.
(a) NPCs converge within 11 seconds
(b) NPCs converge within 40 seconds
(c) NPCs converge within 11 seconds
(d) NPCs converge within 40 seconds

Fig. 3: NPCs movement based on PSO + Finder − Tracker agents method

(a) NPCs converge within 11 seconds
(b) NPCs converge within 40 seconds
(c) NPCs converge within 11 seconds
(d) NPCs converge within 40 seconds

Fig. 4: NPCs movement based on ACO and Finder − Tracker agents method
6 Conclusion

Artificial intelligence is progressively utilized in computer games to emulate agency and intelligence in artificial entities, specifically non-player characters or NPCs. One approach to creating unpredictable movement in population-based game scenarios uses bio-inspired algorithms like PSO and ACO. ACO and PSO are promising for modeling non-player characters in this study. Their ability to generate emergent, adaptive, and realistic movement patterns, along with decentralized decision-making and efficiency in pathfinding, makes them valuable methods for creating immersive and dynamic NPC movement. As influenced by bio-inspired approaches, early convergence in NPC motions presents an issue because it results in a lack of diversity in NPC behaviors and contributes to exhausting game scenarios. This study’s proposed Finder – Tracker agents mechanism introduces a threshold-based control for tracker members. It ensures that finder members consistently strive to maintain proximity to the dynamically moving player character. This dynamic approach aims to enhance natural movement and maintain the diversity of spreading among the enemies’ members (NPCs) while chasing and catching the player character. The statistical results demonstrate that implemented ACO and PSO algorithms that use the proposed Finder – Tracker agents method resolve the early convergence problem, maintain stable movement variety, and enhance spread variation throughout the game.

References

Adapted Q-learning for the Blocking Job Shop Scheduling Problem

Karima Rihane¹, Adel Dabah², and Abdelhakim AitZai¹

¹ University of Sciences and Technology Houari Boumediene, Algeria
krihane@usthb.dz
² Computer, Electrical and Mathematical Science and Engineering, King Abdullah University of Science and Technology, Saudi Arabia
adel.dabah.1@kaust.edu.sa

Abstract. Reinforcement Learning (RL) approaches are becoming increasingly popular in the field of combinatorial optimization problems due to their ability to learn and adapt in unknown environments for decision-making problems. In this paper, we propose an RL agent based on the Q-Learning method to address Blocking Job Shop Scheduling (BJSS). This latter can be used to represent many manufacturing and real-world situations by handling storage-capacity constraints. To evaluate the performance of RL, we first model the BJSS problem as an alternative graph and map it to a Markov decision process. Our adapted Q-learning is based on three reward functions and two action selection methods. The results have shown that the Q-Learning method is capable of finding feasible solutions to complex BJSS problems while also learning effective selection strategies for minimizing Makespan. This is the first time a Reinforcement Learning approach has been used to tackle the BJSS problem. It presents a good opportunity for further research and the development of more advanced methods.

Reinforcement learning, Q-learning, Job shop Scheduling, Blocking constraints.

1 Introduction

Job Shop Scheduling (JSS) is a well-known Combinatorial Optimization Problem (COP). It can be used to model many industrial problems, from manufacturing and planning to transportation and resource management. Classical JSS problems consist of allocating a set of resources (machines) to a set of activities (jobs) under some constraints to optimize one or multiple objective functions. The classical JSS problem assumes an unlimited intermediate buffer capacity between machines. This assumption is impossible in most real-world problems. The Blocking Job Shop Scheduling (BJSS) is an extension of the classical JSS problem obtained by changing the unlimited buffer capacity to a limited or no storage capacity between machines. Thus, a job may block a machine until its next machine becomes available for processing. Adding this constraint makes the problem even more difficult to solve and restricts the solution space. Despite the complexity of the BJSS, several approaches have been proposed to tackle this problem. The most effective approaches are based on an exhaustive search like Branch and Bound (B&B) algorithms [1] or feasibility recovery strategy such as [2]. These methods depend on an in-depth understanding of the problem and cannot easily be generalized for other constraints.

In contrast, Machine Learning (ML) dispenses the need for expert knowledge in decision-making processes. It provides a learning algorithm able to learn from data and hence reinforce its experience and make decisions on its own. Reinforcement Learning (RL) is an ML approach that involves designing an agent (learner) to make decisions in an unknown environment by interacting with it and selecting the decision that will yield the highest reward. In other words, the goal of the RL agent is to maximize the feedback values it receives from its environment. Indeed, RL is a powerful tool that can be used to solve challenging COPs like the BJSS problem.

In this study, we propose an RL framework for the BJSS problem based on a Q-learning method. First, we map the BJSS problem to a Markov Decision Process (MDP) based on the well-known alternative graph model. In this process, the goal is to learn a suitable selection of alternative arcs. In other words, we design an RL agent for learning good scheduling decisions. The learning is performed using an adapted Q-learning method which relies on three proposed...
reward functions. The designed rewards aim primarily to ensure the feasibility of the solution and to keep the makespan as minimized as possible. We employ two action selection strategies (Softmax and $\epsilon$-greedy) to balance between the exploration and the exploitation of the search space. The obtained results demonstrate the strength of the designed Q-learning method for learning effective scheduling actions that guarantee solutions’ feasibility while reducing the makespan. To our knowledge, our study is the first implementation of an RL approach for the BJSS problem.

The rest of the paper is organized as follows. Section 2 introduces the BJSS problem. Section 3 focuses on the BJSS and RL literature review. Section 4 is the core of our study, gives some basic concepts of RL, and reports the adapted Q-learning for the BJSS problem. Section 5 is devoted to experimentation and results. Finally, Section 6 concludes the paper.

2 Blocking Job Shop Scheduling Problem

2.1 Problem formulation

The BJSS is an extension of the classical JSS problem with no intermediate buffers. The classical problem can be presented as a set of $n$ Jobs ($J_1, J_2, ..., J_n$) and a set of $m$ Machines ($M_1, M_2, ..., M_m$). Each job $J_i, i \in [1, ..., n]$ consists of a set of operations ($O_{1i}, O_{2i}, ..., O_{ni}$) a predefined processing order ($O_{1i}$ must be processed before $O_{2i}$, etc.), $n_i$ is the number of operations in $J_i$. Each operation $O_{ij}, i \in [1, ..., n]$ and $j \in [1, ..., n_i]$ needs to be processed on a given machine with a given duration of time noted $P_{ij}$. Each machine can execute only one operation at one time. The solution to this problem is to find a schedule that respects the given constraints and minimizes the total completion time, which is the Makespan.

In the BJSS, the storage capacity of the machines is limited, so the job has to wait on the current machine until the availability of its next machine for processing. The BJSS problem is presented as an alternative graph [3], which is an adaptation of the disjunctive graph model in [4]. The alternative graph is defined as $G(N,F,C)$ where $N$ is the set of nodes (Operations) with two additional dummy nodes modeling the starting and finishing of a schedule. Two kinds of operation exist in $N$ blocking operations and ideal operations. The last operation of a job is named an ideal operation because the machine becomes automatically available after the end of its processing time. All other operations are called blocking operations due to a possible blocking situation. A job can wait on its current machine. Thus, blocks it from processing other jobs until its next machine becomes available. $F$ represents the set of fixed arcs obtained from the precedence constraints between every two consecutive operations of a job. Finally, $C$ is the set of alternative pairs modeling the processing order between every two concurrent operations (operations that need the same machine for processing). The alternative pair generation depends on the nature of the involved operations, whether they are blocking operations or not. Figure 1 shows the alternative pairs generation for two concurrent operations. The left-hand case shows the alternative pairs for two concurrent blocking operations, $O_1$ and $O_2$. Let’s consider $\sigma(O_1)$ the operation that follows operation $O_1$ in $J(O_1)$. Since $O_1$ and $O_2$ can not be executed at the same time, we associate them with an alternative pair modeling the processing order of these two operations. The first possibility is $O_1$ is executed before $O_2$. This translate into an alternative arc ($\sigma(O_1), O_2$) with length 0, indicating that $O_2$ can not start before $\sigma(O_1)$ starts. In this case, $(P \geq P(O_1))$ contains both the processing and blocking times. Similarly, for the second alternative arc ($\sigma(O_1), O_2$). The right-hand side of Figure 1 shows the alternative pair between a blocking and an ideal operation. The first alternative arc ($\sigma(O_1), O_2$) is similar to what we saw earlier since $O_1$ is a blocking operation. However, since $O_2$ is an ideal operation, the machine becomes immediately available after its processing time $P(O_2)$. This translates in an alternative arc ($(O_3), O_1$) with length $P(O_3)$. To generate the alternative graph, we generate the alternative pairs for every two concurrent operations.
A selection (SL) is a set of alternative arcs constructed by selecting at most one arc from each pair in C, and $G(SL) = (N, F \cup SL)$ its graph representation. SL is called a complete selection if one arc is selected from each alternative pair in C, i.e., $|C| = |SL|$. Moreover, SL is feasible if $G(SL)$ does not contain a positive length cycle. Finally, a BJSS solution (schedule) is a complete feasible selection and its evaluation (Cmax) is the longest path in $G(SL)$. Thus, the goal of the BJSS is to find a feasible selection that minimizes the makespan.

Table 1 represents BJSS instance with two jobs and three machines. The first job ($J_1$) has 5 min processing time on machine $M_1$, 3 min on $M_2$, and 8 min on machine $M_3$. The second job ($J_2$) has 8 min processing time on machine $M_2$, 2 min on $M_1$, and 7 min on machine $M_3$.

In Figure 2, subfigure 2(a) represents the alternative graph of the BJSS instance in Table 1. This graph has three alternative pairs, two between blocking operations and one between two deal operations. Both operations 2 and 4 need the same machine $M_2$, and since $M_2$ can not process both operations at the same time, we associate with them an alternative pair. Since operations 2 and 4 are blocking operations, the first alternative arc (red arcs) (3, 4) represents the choice whereby operation 2 must be finished before the beginning of operation 4. His mate, arc (2, 5) represents the choice whereby operation 4 must be finished before the beginning of operation 2. We use the same process to generate the alternative pair ((2, 5), (6, 1)) (green arcs) between operations 1 and 5. The alternative pair between operations 3 and 6 is ((3, 6), (6, 3)) (blue arcs) because both operations 3 and 6 are ideal. While Subfigure 2(b) represents a feasible schedule (solution) for the BJSS instance in Table 1, obtained by choosing one arc from each alternative pair. The Makespan (Cmax = 26) of this schedule is the longest path in the obtained graph.
3 Related work

As an optimization problem, the blocking job shop is addressed using exact and approximation methods. Among the exact methods, the Branch and Bound (B&B) algorithms are the most used. Based on the alternative graph, [3] proposed a B&B method with four different selection processes and solve optimally 10 × 10 benchmark instance for the first time. Using some graph theory, [5] developed a B&B method combined with the alternative graph. In order to accelerate the process of the B&B for the BJSS problem, different parallelization techniques are proposed [6], [1], [7], and [8]. These approaches have contributed to the improvement of the BJSS’s problem state of the art. Recently the authors in [9] proposed a B&B with a learning-based selection and exploration strategies.

The aforementioned methods succeed in solving the problem optimally but fail to find solutions for large-scale instances due to the problem’s complexity. To tackle larger size instances, approximation methods are suitable for large BJSS instances. Several heuristics and metaheuristics are proposed. Among them, we mention Genetic Algorithms (GA) proposed in [10] and [11], which are based on priority roles encoding strategy. Other greedy metaheuristics are presented in [12], [13] and [14] In particular, Tabu Search (TS) algorithm has shown good success in solving the BJSS problem. In [16], the authors presented a TS method that employs a neighborhood structure based on two permutation moves on the disjunctive arcs. However, this latter does not guarantee the feasibility of generated solutions at each iteration. To explore feasible solutions, the authors in [3], [17], [11], and [2] proposed TS algorithms with a neighborhood function that guarantees feasible solutions, either by using priority rules encoding or via a feasibility recovery strategy. In [18], the authors proposed parallel TS with a new recovery strategy. The authors showed the positive impact of parallel strategies and improved most of the state-of-the-art BJSS results.

Nowadays, ML approaches and RL in particular have become a new tool for solving COPs due to the ability to make decisions in unknown environments with additional constraints. Most of the existing RL approaches are proposed for classical job shop scheduling problems. In [19], the authors proposed an RL approach based on a Q-learning to select a proper dispatching rule for each machine in the classical JSS problem. In [20], the authors associated with each JSS machine agent to learn a suitable job sequence in a reactive production environment based on Q-learning. In [21], the authors explored a multi-agent RL to tackle three variants of JSS problem. In [22], the authors implemented a Q-learning algorithm to learn how to select the proper operation for each JSS machine in order to maximize the makespan of the problem. In [23], the authors integrated an RL agent into a neighborhood search method to enhance its performance in solving the dynamic JSS problem. The learning task is assured by an adapted Q-learning method. Recently, [24] proposed a Multi-Agent RL approach for the JSS problem based on Q-learning.

The numerical results show that RL can be an efficient scheduling tool, and we can see that Q-learning is widely used among the RL methods. This can be explained by its high adaptation to new constraints due to trial-and-error interactions. This motivates us further to apply and adapt Q-learning to the BJSS problem.

4 Reinforcement Learning for the BJSS problem

4.1 Reinforcement Learning Basic Concepts

Reinforcement Learning is how to map situations to actions in order to maximize a numerical reward signal. It aims to discover the actions that yield the bigger reward by trial and error. Straightforward, RL is a machine learning approach, where an agent interacts with an unknown environment in discrete time steps. In Figure 3, we describe the RL conversation between the agent and its environment. At time t, the agent observes its state st and decides to perform an action at. It receives a feedback signal reward rt, and moves to the next state st+1. We can describe the RL process as a simple trajectory Tr, which is a sequence of states, actions, and rewards: 

\[ Tr = (s_0, a_0, r_0, s_1, a_1, r_1, ..., s_f) \]

is the initial state and \( s_f \) is the final or terminal state of the agent. The trajectory \( Tr \) can also refer to the episode of the RL.
Adapted Q-learning for the Blocking Job Shop Scheduling Problem

Fig. 3: Reinforcement learning one time step.

In a formal way, RL is introduced as a Markov Decision Process (MDP), it is defined using this 5-tuple $(S, A, T, R, \gamma)$:

- $S$: Set of states, $s \in S$.
- $A$: Set of actions, $a \in A$.
- $T$: Transition function, $T(s_{t+1} = s' | s_t = s, a_t = a)$. It is the probability of moving from state $s$ to $s'$ when the agent executes an action $a$ at time $t$.
- $R$: Reward function $R(s, a)$, is the environment feedback signal received by the agent at time $t$, in state $s$, when it selects an action $a$.
- $\gamma \in [0, 1]$: Discount factor, $\gamma$ expresses the importance given to the expected reward to long-term time steps. i.e. When ($\gamma = 0$) the agent only considers immediate rewards, and when ($\gamma = 1$) the agent strives towards long term rewards.

The aim of the RL agent is to learn a good policy $\pi(s) = a$ that maps the states into actions in order to maximize the cumulative rewards. To learn the optimal policy $\pi^*$ that earns the maximum rewards, RL approaches can be categorized into two types; Model-free and Model-based approaches.

- Model-free approaches: In this type, a model learns the policy directly from the rewards without learning the transition function.
- Model-based approaches: In this type, a model estimates the transition function for all states.

**Q-learning** Q-learning is a well-known model-free RL approach. It was introduced by Watkins in 1992 [25]. Its main objective is to find an optimal policy that maximizes the total reward over all the process steps. Q-learning associates to each state-action pair a Q-value function $Q(s, a)$. This value is the expected cumulative reward from taking action $a$ in state $s$. In order to find the optimal Q-value for each state-action pair, Q-value is updated according to the learning rate $\alpha$, the discount factor $\gamma$, and the expected maximum value of the next state $s'$, see Equation (1).

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[ R(s, a) + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$$  \hspace{1cm} (1)

As shown in Algorithm 1, iteratively Q-learning initializes the state $s$, selects an action $a$, receives the reward $r(s, a)$, updates the $Q(s, a)$ value, then moves to the next state $s'$ until $s$ is the final state, marking the end of one Q-learning episode.
Algorithm 1 Q-Learning Algorithm

Initialize $Q(s,a)$ arbitrarily

repeat (for each episode:)
  Initialize $s$
  repeat (for each step of episode):
    Select an action $a \in A$
    Take action $a$ and observe $r,s'$
    Update $Q(s,a)$ using Equation (1)
  until $s$ is terminal;

The selection of actions in Q-learning is a critical decision, and two of the most popular strategies for action selection are Epsilon-Greedy and Softmax. It is important to choose the right strategy in order to maximize the impact of the selected action. In the following, we will present these two selection algorithms and their adaptation for BJSS problem.

**Epsilon Greedy action selection method**

This is a common exploration strategy used in RL. It is a probabilistic approach that balances exploration and exploitation using a random uniform number $\mu \in [0,1]$ and a small positive number $\epsilon \in [0,1]$. The basic idea of this method is that the agent chooses the action $a$ whose Q-value is largest with a probability $1 - \epsilon$ where $\epsilon$ is a small positive number. Otherwise, it selects an action randomly with a probability $\epsilon$, as described in the Equation 2.

$$a = \begin{cases} 
\text{random action} & \text{if } \mu \leq \epsilon \\
\text{best current action} & \text{otherwise}
\end{cases} \quad (2)$$

This method ensures the exploration-exploitation trade-off in the actions space. In exploitation, the agents make the best decision based on the current information. While in exploration, the agent works on gathering more information. Thus, it may select a random action that may lead to the best overall decision.

**Softmax action selection method**

Like the $\epsilon$-greedy, the softmax algorithm seeks to control the exploration and exploitation of the agent. It attributes a probability $Pr$ for each selecting action $a_l$ overall the possible $k$ actions in a given state $s$. This probability is derived from the Boltzmann distribution and controlled by a positive parameter $\tau$ called temperature. The probability is calculated in Equation 3 as follows:

$$Pr_l(a_t = a_l|s_t = s) = \frac{e^{Q(s,a_l)/\tau}}{\sum_k e^{Q(s,a_k)/\tau}} \quad (3)$$

A high temperature ($\tau \to \infty$) makes the probabilities too near (exploration), while a low temperature ($\tau \to 0$) will make a great difference between actions probability (exploitation).

Both of the methods are critical and have advantages and disadvantages. There are no careful comparative studies of the two simple action selection rules, and the challenge is to set the parameters $\epsilon$ and $\tau$.

4.2 Adapted Q-learning for the BJSS problem.

Applying a Q-learning method for the BJSS problem requires defining the space of states and actions, as well as the reward function. Then set the discount factor and the learning rate to train the RL agent. Based on the alternative graph, we can define the following:

**State space $S$**: A state $s \in S$ is a snapshot of the alternative graph at a given time. It includes a set of selected alternative arcs $LS$ and a list of unselected alternative pairs $LU$. $s_0$ denote the initial state where $LS(s_0) = \emptyset$ and $LU(s_0)$ contains all alternative arcs in $C$, i.e.,
LU(s₀) = \{arc, \forall arc \in C\}. Therefore, |LU(s₀)| = 2 \times |c|\). We denote by sf the final state where |LS(sf)| = |C| and LU(sf) = Φ.

**Action space A**: An action a ∈ A chooses an alternative arc from the unselected arcs list LU. For each state s, we consider A(s) the set of all possible actions, which is defined by the list of unselected arcs LU(s). When an action a (an alternative arc) is selected in a given state s, we remove this action a and Comp(a) (is the alternative arc in the same pair of a) from A(s′), the available action set of the next state s′.

**Reward function R**: Our approach proposes three ways to reward the state-action pair. At a given state s, an action a can receive a reward r(s,a) according to the following cases:

- First case: choosing action a doesn’t produce a positive length cycle in the generated graph.
  In this case, the action a receives a positive reward r1.
- Second case: choosing the action a produces a positive length cycle in the generated graph.
  In this case, the action a receives a negative reward −r1. In the selection LS, we replace the action a with its companion action comp(a) in the alternative pair, the former receives r1.
- Third case: choosing action a and/or Comp(a) produces a positive length cycle. In this case, every action in LS producing the positive length cycle receives a negative reward r2.

In Figure 4, we describe the RL conversation with the scheduling environment of the BJSS instance in Table 1. At the beginning, the initial state is initialized to s₀, LS(s₀) = Φ, A(s₀) = LU(s₀) = \{(2,5); (6,1); (3,4); (5,2); (3,6); (6,3)\}. First, the agent chooses an available action a₀ = (3,4), as feedback for the action decision, the environment sends an immediate reward r(s₀,a₀) = r1, and moves to the next state s₁, defined by LS(s₁) = \{(3,4)\}, where the a₀ is included. While A(s₁) = LS(s₁) = \{(2,5); (6,1); (3,6); (6,3)\} \{ LS(s₁) is the initial list of actions A(s₀) excluding a₀ = (3,4) and the action in the same pair (comp(a₀) = (5,2). The agent repeats this conversation until the final state sf where A(sf) = Φ and LS(sf) is a complete feasible selection.

![Fig. 4: RL Conversation with the BJSS environment.](image-url)
The Adapted Q-learning is described in Algorithm 2.

**Algorithm 2** Adapted Q-Learning Algorithm for BJSSP

1. Initialize $Q(s,a)$ to zero;
2. Set the parameters $\alpha, \gamma$, and $BV$;
3. Repeat (for each episode)
   1. Initialize $s = s_0$;
   2. Repeat (for each step of episode)
      1. Choose $a$ from $A(s)$ using a selection action method;
      2. Take action $a$, add it to $LS_s$;
      3. If there is no positive length cycle in $G(LS_s)$ then
         1. Observe $r(s,a) = r_1$, and move to $s'$;
         2. Update the Q-function;
      4. Else
         1. Remove $a$ from $LS_s$, Add the $\text{comp}(a)$ companion of $a$ to $LS_s$;
         2. Update the Q-function;
   4. Until
5. If there is a cycle in $G(LS_s)$ then
   1. Update the Q-function;
   2. $LS_s \leftarrow \emptyset$;
   3. $s \leftarrow s_0$;
6. Until $s$ is terminal $s = s_f$;
7. If $C_{max} < VB$ then
   1. $r_3 = VB - C_{max}$;
   2. Update the Q-function;
   3. $VB = C_{max}$;
8. End

The blocking constraint in the BJSS problem makes the search space too restricted. i.e., The number of infeasible solutions is very large. Hence, the research methods take a huge amount of time to explore unfeasible solutions rather than feasible ones. For this reason, our rewards’ objective is to penalize the actions that lead to infeasible solutions and to award the actions that lead to feasible ones. To this aim, the $r_1$ value is the makespan of the $G(LS_s)$. We also set the second reward $r_2 = -|LU_s|$ (the number of unselected alternative arcs at the state $s$). The second goal is to minimize the makespan of the problem, which is the goal of reward $r_3$. Its objective is to reward the actions of a solution according to the value of the obtained makespan. At the beginning of the Q-learning process, the method fixes a big value $BV$, then compares it with the obtained makespan at the end of each episode. The actions of the solution are rewarded by $r_3$, $r_3 = C_{max} - BV$. Obviously, $r_3$ is positive when the obtained makespan is lower than $BV$. In this case, the makespan becomes the new value of the $BV$, as described in Algorithm 2.

5 Experimentation

In this section, we test the effectiveness of our Q-learning approach. Our method is implemented using Python and tested on a personal computer with Intel Core (TM) i5-7200 CPU at 2.50 GHz and 8GB memory and Ubuntu 21.04. We begin by setting the learning rate $\alpha$ and discount factor $\gamma$, as they are essential for Q-learning convergence. We test our adapted Q-learning using $\epsilon$-greedy with 100 episodes, where $\epsilon = 0.5$ and $BV = 5000$. We vary the learning rate and discount factor between $[0.1, 0.3, 0.6, 0.9]$, and Figure 5 shows the reward at the end of each episode for La01 instance [27]. It is evident from Figure 5 that $\alpha = 0.1$ and $\gamma = 0.9$ lead to faster convergence to the maximum reward, so we maintain these values for the rest of our experimentation.

In Q-learning, the action selection strategies (Greedy and Softmax) aim to control the exploration/exploitation trade-off using their control parameters ($\epsilon$ and $\tau$). Both strategies propose to
Adapted Q-learning for the Blocking Job Shop Scheduling Problem

Fig. 5: Obtained Reward from using different learning rates and discount factors for La01 BJSS instance.

The first observation from the graphs in Figure 6, is that the Epsilon-greedy and Softmax exhibit the same behavior. Using both strategies the Q-learning agent shows an ability to learn and improve its obtained rewards and the makespan. Indeed, the first few episodes have a low cumulative reward. This indicates that the agent is falling into unfeasible solutions when it explores the state-action space. This phase reflects the fact that the agent performs some random actions (exploration) according to the big value of the $\epsilon$ and $\tau$. After that, we can see a fast increase in the agent’s reward with better makespan values. This reflects the good actions learned by the RL agent in less than twenty episodes, explained by the small values of the control parameters (exploitation phase). Decreasing the value of the control parameter stabilizes the learning and affects the convergence of the adapted Q-learning.

In the following, we present the experimental results of our Q-learning method using both $\epsilon$-Greedy and Softmax selection methods on fifteen BJSS instances from Lawrence benchmarks [27]. To the best of our knowledge, no RL approaches have been proposed for the BJSS problem. Therefore, we compare our Q-learning results to the optimal B&B results in [1] and the effective greedy algorithm SMPC [3].
Table 2: Makspan results of the Adapted Q-learning on Lawrence [27] BJSS instances.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>La01 931</td>
<td>1059</td>
<td>1137</td>
<td>793</td>
<td></td>
</tr>
<tr>
<td>La02 1039</td>
<td>1002</td>
<td>1055</td>
<td>793</td>
<td></td>
</tr>
<tr>
<td>La03 975</td>
<td>944</td>
<td>1088</td>
<td>715</td>
<td></td>
</tr>
<tr>
<td>La04 969</td>
<td>972</td>
<td>1378</td>
<td>743</td>
<td></td>
</tr>
<tr>
<td>La05 926</td>
<td>904</td>
<td>1116</td>
<td>664</td>
<td></td>
</tr>
<tr>
<td>La06 1407</td>
<td>1494</td>
<td>1493</td>
<td>1060</td>
<td></td>
</tr>
<tr>
<td>La07 1590</td>
<td>1542</td>
<td>1737</td>
<td>1016</td>
<td></td>
</tr>
<tr>
<td>La08 1456</td>
<td>1484</td>
<td>1569</td>
<td>1040</td>
<td></td>
</tr>
<tr>
<td>La09 1719</td>
<td>1689</td>
<td>1943</td>
<td>1141</td>
<td></td>
</tr>
<tr>
<td>La10 1593</td>
<td>1602</td>
<td>1833</td>
<td>1096</td>
<td></td>
</tr>
<tr>
<td>La16 1378</td>
<td>1441</td>
<td>1593</td>
<td>1060</td>
<td></td>
</tr>
<tr>
<td>La17 1659</td>
<td>1610</td>
<td>1719</td>
<td>929</td>
<td></td>
</tr>
<tr>
<td>La18 1679</td>
<td>1718</td>
<td>-</td>
<td>1025</td>
<td></td>
</tr>
<tr>
<td>La19 1715</td>
<td>1801</td>
<td>1786</td>
<td>1043</td>
<td></td>
</tr>
<tr>
<td>La20 1468</td>
<td>1502</td>
<td>1626</td>
<td>1060</td>
<td></td>
</tr>
</tbody>
</table>

Table 2 summarizes the obtained results. The first two columns give the name and the size of the instance. Columns $\epsilon$-Greedy and Softmax give the makespan (Cmax) results of our Q-learning method using respectively the $\epsilon$-Greedy and Softmax selection strategies. While column SMCP is devoted to the results of heuristic SMCP in [3]. Finally, Column B&B reports the optimal makespan results reported in [1]. The makespan results of the proposed Q-learning methods are obtained after one hundred episodes. The results in Table 2, demonstrate that both selection strategies yield nearly the same makespan. The adapted Q-learning succeeds in learning a selection policy that yields the feasible search space. The Q-learning policy avoids investigating unfeasible schedules and minimizes the Makespan as possible. Compared to the SMCP heuristic, the adapted Q-learning shows its performance by achieving better Makespan values for the fifteen instances. Against the B&B, our Q-learning results are significantly optimal Cmax values. This is expected since the design of the proposed rewards aims primarily to guarantee solution feasibility instead of the solutions makespan.

6 Conclusion

This paper investigates the use of a learning-based approach for solving the blocking job shop scheduling (BJSS) problem, which is a new challenge. This later assumes a zero buffer capacity between machines which deeply affects the ability of metaheuristics to find feasible solutions. We first modeled the BJSS as a Markov decision process. Based on this model, we proposed a Q-learning algorithm with two action selection strategies (greedy and softmax). We proposed three rewards aiming to explore feasible solutions and reduce the makespan. Furthermore, we analyzed the impact of different parameters on the learning process and convergence speed. Obtained results demonstrated the ability of the Q-learning algorithm to learn an effective policy that always produces feasible solutions, demonstrating its ability to address the changing constraints of real-world problems.

In future works, we plan to extend this study by exploring the use of deep reinforcement learning approaches such as Deep-Q-Network (DQN) to address the BJSS problem.
References


27. S. Lawrence, "Resource-constrained project scheduling: an experimental investigation of heuristic scheduling techniques (supplement)". Graduate School of Industrial Administration, (1984), 1571980073974705920.
An hybrid approach based on Graph Attention Network for the Team Orienteering Problem

Iván Peña-Arenas¹ Rym Nesrine Guibadj¹ and Cyril Fonlupt¹

Université du Littoral Côte d’Opale, EA 4491 – LISIC – Laboratoire d’Informatique Signal et Image de la Côte d’Opale, F-62228 Calais, France
{ivan.pena-arenas, rym.guibadj, cyril.fonlupt}@univ-littoral.fr

1 Introduction

The Team Orienteering Problem (TOP) is a combinatorial optimization problem that has been proven NP-Hard [1], and which has different practical applications ranging from logistics to telecommunications. End to end machine learning models have been proven effective to solve discrete optimization problems [2]. Advancements in deep learning techniques designed for managing data structures of varying sizes, such as attention mechanisms and sequence to sequence approaches, have significantly increased their usage. Nevertheless, as the scale of problem instances grows, deep learning models experience a decline in performance, leading to a challenge in generalization (see, [3], [4]).

Heuristics methods are efficient solution algorithms designed for particular problems, which rely on their programmed behaviour. Several contributions involve merging optimization algorithms with machine learning [5], wherein the machine learning model aids the optimization algorithm. Following the same idea, we propose leveraging both solution methods, although in this case, the heuristic supports the learning algorithm, guiding the learning process. While the concept has been previously suggested [6], as far as we are aware, this is the first implementation of this idea in the context of the TOP.

We present a learning framework that integrates an efficient splitting algorithm applied to the TOP [7] within a deep learning model. This hybrid approach works in two steps. Initially, a giant tour (a sequence of customers/locations) is generated at once using a deep neural network, and subsequently, it is evaluated using the split algorithm (see Fig. 1). The main objective is to narrow down the solution space in which the deep learning model works (i.e. one optimal set of sub-tours for each giant tour), expecting an overall better performance. In addition, two handcrafted solution methods are used to compare the performance and the quality of the results. Section 2, briefly describes the method, Section 3 shows results and comparisons, and conclusions are presented in Section 4.

Fig. 1. Solution schema.
2 Hybrid Graph Attention Model

During the first stage, a reinforcement-trained model is used to generate a 'giant tour'. Our deep learning procedure is based on a Graph Attention Network that operates as an encoder-decoder system [4]. The encoder is responsible for processing the input instance and generating embeddings for all input nodes, including clients and the depot. This transformation of the instance's representation aims to capture underlying structures beneficial for problem-solving. Subsequently, the decoder utilizes the encoder embeddings along with the history of previously visited clients to compute a singular vector summarizing the current solution. Following this, a multi-head attention module uses this vector to establish a probability distribution function for selecting the subsequent client. The decoder progressively constructs candidate solutions, sequentially selecting the next node until all clients have been visited.

In the following stage, we employ an optimal splitting algorithm to identify the collection of sub-tours that maximizes the total sum of their profits. This method ensures that if a series of sub-tours, constituting an optimal solution for the Team Orienteering Problem, exists as sub-sequences within the 'giant tour' π, applying the splitting process to π will yield the optimal TOP solution. Consequently, the neural network operates within a more confined solution space, determined by the collection of giant tours resulting from the splitting procedure.

During the learning process, the Graph Attention Model defines a stochastic policy \( p_\theta(\pi|s) \) for selecting a giant tour (sequence) \( \pi \) given a TOP instance \( s \) and the parameters \( \theta \). Later, by applying the split procedure this sequence \( \pi \) is decomposed in a set of optimal sub-tours considering the maximum length duration for a tour \( L \). Moreover, we define \( J(\theta | s) \) as the policy objective function, which is the total expected score of the sub-tours evaluated by the split procedure given the instance \( s \).

\[
J(\theta|s) = \mathbb{E}_{p_\theta(\pi|s)} \left[ (\text{split}(\pi|s)) \right]
\]

We use policy gradients methods to search for a local maximum in \( J(\theta|s) \) by ascending gradient policy, w.r.t parameters \( \theta \), defined as:

\[
\nabla_\theta J(\theta|s) = \mathbb{E}_{p_\theta(\pi|s)} \left[ \nabla_\theta \log p_\theta(\pi|s)(\text{split}(\pi|s) - b(s)) \right]
\]

Where \( b(s) \) is the baseline, which is used to reduce the gradient variance. The value of the gradient function \( \nabla_\theta J(\theta|s) \) is equal to the expected value of the multiplication between the functions score and advantage. The gradient \( \nabla_\theta \log p_\theta(\pi|s) \) is a measure of the movement of the function in the solution space (score function). While, the difference between the score \( \text{split}(\pi|s) \) of the tour \( \pi \), and the baseline \( b(s) \) is the advantage function. To optimize the expected score we use REINFORCE gradient estimator, and an exponential moving average with a decay \( \beta = 0.2 \) as a baseline.

3 Results

We consider three types of TOP instances with number of clients \( n \) equal to 20, 50 and 100. For each problem size we generate 800000 instances for training, and two sets of 10000 instances for validation and test. The depot location as well as \( n \) node locations are sampled uniformly at random in the unit square. We consider a constant distribution of prices \( p_i = 1 \), thus the goal becomes to visit as many nodes as possible within the length constraint. The number of teams \( m \) is set to 2, and the maximum length \( T \) is fixed to 1.

We initialize parameters Uniform\((-1/\sqrt{d}, 1/\sqrt{d})\), where \( d = 128 \) represents the embedding dimension. We train 100 epochs, using a fixed batch size \( B \) equal to 500 instances and the training and validation data were generated on the fly. We use \( K = 3 \) layers in the encoder and a constant learning rate \( \eta = 10^{-5} \). All the experiments were conducted on a single Nvidia RTX-3500 Ada with 12 GB of VRAM. The Hybrid graph attention model (HGAM) was implemented with Python, using the Pytorch libraries, while the approximation algorithms were coded in C++.

To evaluate the effectiveness of the HGAM, we compare it with a tailored heuristic known as Construction Heuristic (CH) and with the metaheuristic Iterative Destruction/Construction Heuristic (IDCH)[8]. To apply the acquired policies, we utilize both a sampling decoding method,
applied at each decoding step, and a sampling strategy that generates 100 candidate solutions for each test instance. We then select the best solution based on their scores. These solutions are sampled from the probability distributions provided by the models. This evaluation was conducted using test instances that had never been seen during the training and validation processes. We report in Table 1 the average solution score (Objective), the average GAP (in percentage) to the best average solution score and the running time (in seconds) to solve a test instance.

<table>
<thead>
<tr>
<th>Method</th>
<th>n=20</th>
<th>n=50</th>
<th>n=100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Objective</td>
<td>GAP(%)</td>
<td>Time(s)</td>
</tr>
<tr>
<td>IDCH</td>
<td>6.71</td>
<td>0.00</td>
<td>0.0057</td>
</tr>
<tr>
<td>CH</td>
<td>6.21</td>
<td>7.31</td>
<td>0.0002</td>
</tr>
<tr>
<td>HGAM</td>
<td>6.45</td>
<td>2.38</td>
<td>0.0097</td>
</tr>
</tbody>
</table>

IDCH obtains the highest average solution score for all types of instances. The relative gap between the solution methods widens as the instance size increases, reaching up to 14.85% and 15.66% for HGAM and CH, respectively, in instances with 100 clients. On the other hand, CH is the fastest solution method, averaging 0.0035 seconds to solve an instance with 100 clients.

On scenarios involving 20 customers, there is a 2.38% variation in performance between IDCH and HGAM. For instances with 50 and 100 customers, this difference increases to an average of 12.09% and 15.66%, although HGAM is almost 7.4 (≈ 0.0959/0.0129) and 12.2 (≈ 1.0158/0.0829) times faster, respectively. In contrast, while CH is two orders of magnitude faster than HGAM, HGAM achieves better scores for all types of instances. In this sense, HGAM presents a good trade-off between the quality of the solutions and their running times.

4 Conclusion

Our work is part of a research trend aiming to integrate machine learning and optimization approaches to effectively solve combinatorial optimization problems. For this purpose, we proposed a hybrid architecture in which a deep neural network based on attention mechanisms seeks to prioritize clients/nodes. An optimal split procedure is then applied to (1) incorporate the constraints, (2) identify the routes, and (3) produce an optimal solution considering the order generated by the GAM. Preliminary findings suggest that our method is competitive when compared to specialized heuristics. Furthermore, unlike heuristic approaches, our model could be used with minimal modifications to solve several other problem variants with benefits.

References

8. H. Bouly, D.-C. Dang, and A. Moukrim.: A memetic algorithm for the team orienteering problem. 4OR, 8(1), 49–70(2010)

211
A Parallel Genetic Algorithm for Qubit Mapping on Noisy Intermediate-Scale Quantum Machines

Jérôme Rouzé¹, Nouredine Melab² and Daniel Tuyttens¹

¹ Mathematics and Operations Research Department, University of Mons, Belgium
{jerome.rouze,daniel.tuyttens}@umons.ac.be
² Univ. Lille, CNRS, Centrale Lille, Inria, UMR 9189 - CRIStAL - Centre de Recherche en Informatique Signal et Automatique de Lille, F-59000 Lille, France
nouredine.melab@univ-lille.fr

Abstract. Quantum computers are getting increasingly large and available thanks to some major technological advancements, but they remain in the realm of NISQ (Noisy Intermediate Scale Quantum) devices. On such devices, due to the limited connectivity of the physical qubits, most quantum circuit-based programs cannot be executed without transpilation. This latter includes an important step, referred to as qubit mapping, which consists in converting the quantum circuit into another one which best matches the graph of physical qubits taking into account its limited connectivity constraint.

In this paper, we propose a Parallel Genetic Algorithm to Qubit Mapping (PGA-QM). The challenge is to minimize the depth of the transformed circuit and the execution time and error rate consequently. PGA-QM has been experimented using various medium-to-large scale circuit benchmarks. It is compared against the SABRE heuristic currently implemented in Qiskit, the IBM’s library for quantum computing. The reported results show that PGA-QM can provide better solutions and with better consistency than its counterpart while parallelism greatly reduces its execution time during the transpilation.

1 Introduction

In recent years, quantum computers have become increasingly available, including a growing number of qubits. Using the principles of quantum physics such as superposition and entanglement, these systems are promised to offer significant speedup over classical ones for a variety of applications. While Grover’s search algorithm [9] and Shor’s integer factorization [15] can be cited as two significant historical quantum-related contributions, other notable advancements include quantum combinatorial optimization [5,7], quantum machine learning [12], and so forth [10].

However, such promising results, often referred to as the quantum advantage, cannot be easily achieved without quantum error correction, which would require much more qubits than currently supplied. Actually, current quantum computers are noisy, and the results of the computations they return have a non-zero chance of being wrong. To make use of current quantum devices, one needs to minimize the error rate so that the returned results can be trusted. In addition, like in classical computing, a quantum program (circuit in this paper) needs to be compiled (or transpiled) to be executable on a device. This transpilation consists in several steps, including gate decomposition and mapping of the quantum (or logical) circuit on a graph of physical qubits composing a quantum machine.

This mapping step has a great impact on the error rate of the executed quantum circuit. In this paper, we investigate this circuit mapping problem referred to in the literature as qubit allocation problem [16], qubit initialization problem [6] or qubit mapping problem [11,4]. Depending on how the logical qubits of the circuit are mapped to the physical ones in the hardware, one may need to add a lot of gates to the circuit to match the hardware connectivity. That will increase the depth of the circuit, increasing the likelihood of decoherence, thus increasing the error rate of the circuit. Given that gates are noisy, adding more gates increases the error rate. For these two reasons, finding a correct mapping can greatly limit the error rate.

The main contributions of this paper are the following:

- We propose a parallel genetic algorithm the solve the qubit mapping problem (PGA-QM). The parallel contribution is done at the evaluation step of the GA as computing the cost of each individual is costly (and doable in parallel).
We compared the performance of this PGA-QM using various benchmarks, against the currently used SABRE [11] heuristic supplied in IBM’s Qiskit [13] library for quantum computing. The reported results highlight the ability of the PGA-QM to outperform SABRE and find better mappings, albeit at a higher computational cost. The added parallelism limits the computational expensiveness of the genetic algorithm.

The rest of this paper is organized as follows. Section 2 starts with a brief introduction to quantum computing followed by a proper formulation of the qubit mapping problem and some related works. In Section 3, the proposed parallel GA approach in presented. The experimental protocol and benchmarks are described in Section 4 and the results are discussed in Section 5. Finally, a conclusion is drawn in Section 6 with some future works.

2 Background, Problem Formulation and Related Works

In this section, we introduce the basic concepts of quantum computing necessary to understand the qubit mapping problem. Then, a proper problem formulation is given. Lastly, related works are presented.

2.1 Background

To introduce the problem at hand in this paper, let us give a brief introduction to the main concepts of quantum computing.

Quantum bits While traditional computing relies on bits as the unit of information, quantum computing relies on quantum bits or qubits. Based on the principles of quantum mechanics, a qubit can be a superposition of the two basis state denoted $|0\rangle$ and $|1\rangle$. That is, bits are deterministic being in either one of those states while qubits are in a state $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $\alpha, \beta \in \mathbb{C}$ can be normalized such that $|\alpha|^2 + |\beta|^2 = 1$. From a theoretical point of view, the basis state $|0\rangle$ and $|1\rangle$ can be seen as a basis of a vector space, while $\alpha$ and $\beta$ are the coefficients of the vector $|\Psi\rangle$ in said basis.

Quantum gates An operation on one or more qubit(s) is a quantum gate, similarly to logical gates in classical computing. Mathematically, quantum gates are unitary operations that transform the current quantum state of the system into another. Being unitary operations, they are also reversible. For example, the Hadamard gate, usually denoted as $H$ is a very commonly used single-qubit gate creating a superposition state. The Control-NOT, or CNOT, gate is a two-qubit gate that creates entanglement. It involves a control qubit and a target one applying the NOT operation to the target if the control is in the state $|1\rangle$, else leaving the target unchanged. It has been proven in [3] that the set of all one-qubit gates and the CNOT gate form a universal set of gates. This means that any quantum operation can be decomposed into a sequence of one-qubit and CNOT gates. Actually, only a few carefully selected one-qubit gates and CNOT ones are enough to form a universal set. For that reason, one can consider the CNOT gate as the only multi-qubit gate without loss of generality.

Measurement The only operation that is not a quantum gate is the measurement. Measurement means observing in which state one or more qubit(s) is (are). With the normalization of $\alpha$ and $\beta$ mentioned before, $|\alpha|^2$ and $|\beta|^2$ are the probabilities of measuring the states $|0\rangle$ and $|1\rangle$ respectively. Due to the properties of quantum mechanics, this measurement operation changes the quantum state in a destructive way\(^3\), meaning that this operation is not reversible and therefore not unitary. Measurement is the way to obtain some results, usually at the end of the computation.

Quantum circuits One of the most commonly used quantum programming paradigms is based on quantum circuits. A quantum circuit represents the sequence of gates applied to its qubit(s). Figure 1 illustrates a four-qubit circuit made of $H$ gates and CNOTs, followed by measurements. On such a representation of a quantum circuit, each line represents a qubit, and the list of gates

\(^3\) One says that the quantum state collapses to the measured state.
on that same line is the list of operations sequentially applied to that qubit.

**NISQ Computers** In this paper, Noisy Intermediate Scale Quantum (NISQ) computers are considered. On these systems, quantum gates and measurements are noisy, meaning there is a non-zero chance that a gate is wrongly applied or that a measurement does not return the correct state. They also present a limited number of qubits (< 10³) that are scarcely interconnected (see Figure 2). In theory, when designing a quantum circuit, the CNOT\((q_i, q_j)\) between two qubits \(q_i \neq q_j\) can always be applied. However, on NISQ hardware, \(q_i\) and \(q_j\) need to be connected, which is a strong restriction given the scarcity of the connectivity graph. Finally, on these hardware, a limited set of gates is feasible in practice.

**Circuit mapping** To get around these restrictions, a quantum circuit must be transformed into an equivalent circuit to match the hardware requirement. Gates are decomposed into the basis gates of the system, and extra SWAP operations are added to match the connectivity. A SWAP gate is actually made of three consecutive CNOT gates, and exchanges the quantum states of two qubits. These transformations, and especially the added SWAPs heavily depend on the one-to-one mapping between abstract circuit qubits and physical ones illustrated in Figure 3. Finding the mapping leading to the least noisy circuit is known as the Qubit Mapping Problem.

![Fig. 1. A 4-qubit quantum circuit made of a some Hadamard and CNOT gates, a barrier (dashed line) followed by measurements.](image1)

![Fig. 2. IBMQ Washington's connectivity graph](image2)

### 2.2 Problem formulation

Formally speaking, the qubit mapping problem can be formulated as follows. The data include a circuit of \(d\) virtual qubits \(Q = \{q_0, \ldots, q_{d-1}\}\) and a graph representing the connectivity of a physical machine made of \(m \geq d\) qubits \(P = \{p_0, \ldots, p_{m-1}\}\). The objective is to find the best mapping \(\pi : Q \rightarrow P\), which can be simply represented by an ordered list of \(d\) different integers in \(\{0, \ldots, m-1\}\). For example, \([1, 3, 0]\) designates the mapping \(\pi(q_0) = p_1, \pi(q_1) = p_3, \text{and } \pi(q_2) = p_0\).

A solution is then made of \(d\) ordered integers among \(m\), leading to \(\frac{m!}{(m-d)!}\) possible mappings in total. The problem can also be considered as a partial permutation problem.

To each mapping is associated a cost, which is ideally the error rate of the resulting transpiled circuit. However, the error rate is hard to compute. For that reason, it is common to use other metrics that influence the error rate instead. In this paper, we consider the depth of the circuit as the cost function. Essentially, our cost function takes a mapping, i.e. a list of \(d\) different integers, and returns the depth of the transplied circuit to fit the mapping. The depth of a quantum circuit is defined as the largest number of gates the quantum device has to execute sequentially to run the whole quantum program. Another way to view this is to represent the quantum circuit by regrouping the gates that can be applied simultaneously in one layer. The depth is then the number of layers.

The depth is chosen among other possible choices, namely, the total cost of circuit transformations as defined in [16] or the number of additional gates [11]. Let us justify the choice made here.
Recall that the goal is to generate a resulting circuit with the lowest possible error rate. This error rate depends not only on the individual gate error rates but also on the circuit’s execution time, due to decoherence. Although negligible at our scale, as it takes a microsecond at worst, longer execution times increase the likelihood of decoherence, leading to an inaccurate quantum state of the outcome. Minimizing the circuit depth reduces the execution time since each layer is executed in parallel, which consequently diminishes the risk of decoherence. A lower depth also translates to fewer SWAP gates as they increase the circuit depth by 3. Moreover, since SWAP gates have a high error rate\(^4\), their number has to be minimized. Hence, a reduced depth addresses both gate-related errors and decoherence likelihood.

The other previously mentioned choices primarily focus on minimizing added CNOT gates to lower the circuit error rate given their higher individual error rates compared to 1-qubit gates (7.524\(e^{-3}\) for CNOT, 2.332\(e^{-4}\) for 1-qubit gates)\(^5\). However, these approaches do not directly reduce circuit depth, and consequently, the risk of decoherence.

### 2.3 Related works

The qubit mapping problem has been tackled using different approaches in the recent literature. In [16], an exact algorithm is proposed, but its complexity is too large to be applied to practical problems. A heuristic search is also proposed to deal with this issue. However, the formulation of the problem is quite different as the authors considered unidirectional CNOT, meaning that if the CNOT\((q_i, q_j)\) is feasible on the hardware, CNOT\((q_j, q_i)\) is not. To be exact, it is feasible but demands a “reversal” transformation. Unidirectional CNOTs were the norm at the time the paper was written (in 2018). On current quantum systems, CNOTs are bidirectional, so the added difficulty of unidirectionality is no more.

Other heuristic approaches to this problem have been studied. For instance, the SABRE (SWAP-based BidiREctional) heuristic search [11] is currently implemented in Qiskit [13], a Python library developed by IBM for quantum computing. This heuristic approach has been developed with the goal of being efficient while remaining fast to run. This is achieved through greatly reducing the search space and an efficient initial mapping finder. Indeed, heuristic approaches in the literature have shown to be significantly dependent on the initial mapping they start with.

Duostra (Dual-source Dijkstra) [4] is another fairly recent heuristic contribution designed with large circuits in mind. Their result compared fairly well with SABRE, but the article mentions using the optimization level 0 of Qiskit, i.e. the lowest one.

Metaheuristic approaches have also been developed, and coupled with heuristic search operator. The mapping is found using the metaheuristic while a heuristic search adds the necessary gates to the circuit according to the previously found mapping. Indeed, in the literature, heuristics often try to simultaneously find a mapping and, given that mapping, they find the necessary SWAPs to be performed on the circuit. On the contrary, metaheuristic approaches focus on finding the mapping, leaving the task of adding gates to a heuristic operator. Namely [17] proposed a simulated annealing algorithm to search for the mapping as well as a specific look-ahead heuristic search to add the necessary SWAPs.

---

\(^4\) Recall that SWAP gates are made of three consecutive CNOTs, which have high error rate.

\(^5\) https://quantum-computing.ibm.com/
A genetic algorithm has also been developed in [6]. In this work, the GA finds a good mapping and the SABRE heuristic adds the necessary gates to match the requirement. Our approach shares this principle but is actually quite different. While [6] focuses on one type of circuit of different sizes on different hardware, our study focuses on testing different large-scale circuits on one hardware. We also study the potential improvement parallel computing could provide. Indeed, [6] shows the promising results of the genetic algorithm but the execution gets much longer than the other approaches, and our contribution aims to lower that time.

Another completely different approach using Deep Neural Network (DNN) has been developed in [1]. In this paper, the qubit mapping problem is modeled as a classification task instead of an optimization problem. An important contribution of this work is that the execution time does not increase with the depth of the input circuit. However, the study was conducted on a 5-qubit hardware and fairly limited circuit sizes. Moreover, generating data to train the DNN for other architectures relies on an additional heuristic search.

3 The Proposed Parallel Genetic Algorithm : PGA-QM

GAs are a type of optimization algorithms inspired by nature, where a population of initially randomly selected individuals evolves throughout a number of generations to find a near-optimal solution to the problem. A pseudo-code of the parallel GA is given in Algorithm 1 and described below.

The population $P$ is initialized with randomly selected genes for each individual. In our application, a gene is simply the mapping of one qubit, i.e. there are $d$ genes which simply consist of one integer each. A generation starts with the selection of parents, which consists of randomly selecting individuals that will be subsequently recombined by a two-point crossover operator (see Figure 4). A mutation operator is then applied to the newly created population $P_2$, mimicking natural gene mutations. Here, each gene has a 0.1 probability of being changed to another randomly chosen value. Finally, the cost of each individual of $P_2$ is evaluated and based on that evaluation, the population is updated for the next generation. Here, an elitist replacement is considered, that is, only the best individuals are kept. Note that an early stopping criterion may be used, in particular, if for several generations, the best solution has not been improved. In this paper, we considered either no early stopping criterion or a stop after 10 generations without improvement.

Other variations of parameters (crossover type, mutation probability) have been tested to try and get the best possible results, but no significant differences have been observed.

![Illustration of the two-point crossover operator](image-url)
To select the best individuals for the next generation, one needs to know the cost of each individual, i.e. the depth of the quantum circuit transpiled to match the mapping. This evaluation phase is done in parallel and can significantly reduce the time required for this step, as shown in Section 5.4.

4 Performance Evaluation

4.1 Quantum hardware and circuits

In this section, we describe the benchmarks used in the experiments as well as the experimental protocol and goals. The operation that transforms the abstract circuit into one that fits the hardware requirements is known as the transpilation. Qiskit’s transpilation function relies on six steps: initialization, layout, routing, translation, optimization and scheduling. The layout step is the one responsible to find the best mapping, and our proposed PGA-QM aims to replace that step, leaving the other steps unchanged for a fair comparison.

An instance of the problem is partly defined by the architecture of the used hardware. Given our aim to consider large circuits, and the available resources built-in within Qiskit, we focused our study on a single architecture, namely, the IBMQ Washington system, whose connectivity graph is given in Figure 2. On such graph, each vertex designates one qubit, and an edge is present between two vertices if a CNOT gate can be applied between these two qubits.

The other element to choose to get one instance of the qubit mapping problem is the quantum circuit to efficiently and effectively deal with. We focused our study on using large circuits of 80 and 120 qubits. These sizes were not chosen randomly. Given that the largest quantum hardware currently provided in Qiskit is, at the time these experiments are performed, made of 127 qubits, we considered the mapping of a “medium-sized” circuit (≈ 2/3 of all 127 available qubits are used) and a “full” one (≈ all 127 available qubits are used). A medium mapping problem leads to a partial permutation problem while a full one leads to a true permutation problem (partial permutation problem with $d = m$).

We considered three different quantum circuits, two of which being taken from [14], the Greenberger–Horne–Zeilinger (GHZ) and Deutsch–Jozsa (DJ) circuits. The third one we used will be referred to as GHZALL. Both GHZ and GHZALL lead to the same quantum state at the end, precisely a state where qubits are either all in state $|0\rangle$ or all in state $|1\rangle$, with probability 0.5 for each state. However, they achieve this state through two different means. GHZALL connects all qubits to the same one through CNOT gates while GHZ makes a cascade of CNOT connections as shown in Figure 5. As we will see in Section 5, this leads to very different results.

---

6 https://docs.quantum.ibm.com/api/qiskit/transpiler
7 In this paper, we worked using Qiskit 0.43.3
The three circuits studied have been chosen for their scalability. Actually, since we aimed to study large circuits, we considered those from [14]. The scalability of the circuits limits the type of circuits one can use as not all circuits can be extended to large numbers of qubits.

![Circuits](image.png)

(a) GHZALL with 3 qubits  
(b) GHZ with 3 qubits

**Fig. 5.** GHZALL and GHZ circuits

### 4.2 Objectives of the performance evaluation

The first goal is to evaluate the efficiency and effectiveness of the PGA-QM compared to SABRE. Given the stochastic nature of the algorithms, each execution is done 30 times and the median of the best results found was considered. The median is favored over the average since it is not as sensible to extreme values.

To evaluate the efficiency of our approach compared to SABRE, we considered the execution times of both approaches and their ratio. As shown in Table 1, three variants of the PGA-QM are considered varying their three parameters (population size, number of offsprings per generation and maximum number of generations). These three variants have also been tested with two different stopping criteria: either all the generations are run, or the algorithm is allowed to stop earlier if the best solution did not evolve for 10 consecutive generations. The idea behind those various sets of parameters was to evaluate how impactful they are on both the execution time and quality of the solutions.

A third comparison one can make is based on the robustness of the approaches. Given their stochastic nature, they do not always lead to the same result from one execution to another one. To study the robustness, we considered two statistical metrics: the Inter Quartile Range (IQR) and Median Average Deviation (MAD). The lower these two values are, the less scattered the results are and the more robust the algorithm is.

Finally, we studied the speedup provided by the parallel computing part of the PGA-QM, questioning how much the parallelization actually improved the execution time and how does it scale.

**Table 1.** Different set of parameters of the GAs.

<table>
<thead>
<tr>
<th></th>
<th># Individuals</th>
<th># Offsprings per generation</th>
<th># Total generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>PGA 1</td>
<td>40</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>PGA 2</td>
<td>30</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>PGA 3</td>
<td>20</td>
<td>15</td>
<td>35</td>
</tr>
</tbody>
</table>

In this paper, we used the parallel GA implementation provided in the PYGAD [8] Python library, and compared its performances with SABRE heuristic [11] currently used in Qiskit. Experiments were run on the GRID5000 testbed [2], using a processing node of 2 AMD EPYC 7301 CPUs. Their characteristics are the following: 16 cores/CPU, 2.2GHz base frequency, 64MB total L3 memory. All codes are available at https://github.com/Jrouze/PGA-QM.

### 5 Results and Discussion

In this section, we report some experimental results and their discussion. Note that in the following tables the stop10 rows refer to the variants of PGA-QM with an early stop after 10 generations.
without improvement while the others refer to the variants without an early stopping criterion.

5.1 Depth analysis

Table 2. Median best found depth for all studied quantum circuits and algorithms

<table>
<thead>
<tr>
<th>Solver</th>
<th>GHZALL 80</th>
<th>GHZALL 120</th>
<th>DJ 80</th>
<th>DJ 120</th>
<th>GHZ 80</th>
<th>GHZ 120</th>
</tr>
</thead>
<tbody>
<tr>
<td>PGA1</td>
<td>292</td>
<td>446.5</td>
<td>362</td>
<td>564.5</td>
<td>790</td>
<td>1335.5</td>
</tr>
<tr>
<td>PGA2</td>
<td>290.5</td>
<td>451.5</td>
<td>363.5</td>
<td>565</td>
<td>797</td>
<td>1346</td>
</tr>
<tr>
<td>PGA3</td>
<td>292</td>
<td>457.5</td>
<td>365</td>
<td>569.5</td>
<td>814.5</td>
<td>1340</td>
</tr>
<tr>
<td>SABRE</td>
<td>344</td>
<td>537.5</td>
<td>389.5</td>
<td>618.5</td>
<td>83</td>
<td>844</td>
</tr>
<tr>
<td>PGA1 stop10</td>
<td>292.5</td>
<td>459.5</td>
<td>370</td>
<td>568</td>
<td>808.5</td>
<td>1326</td>
</tr>
<tr>
<td>PGA2 stop10</td>
<td>297</td>
<td>462</td>
<td>367</td>
<td>568</td>
<td>803.5</td>
<td>1355.5</td>
</tr>
<tr>
<td>PGA3 stop10</td>
<td>298</td>
<td>465</td>
<td>375.5</td>
<td>570.5</td>
<td>814</td>
<td>1356.5</td>
</tr>
</tbody>
</table>

Our first analysis is the study of the best mapping each algorithm has found. According to Table 2 one can observe that the PGA-QM variants can outperform SABRE and find a better mapping, but not for every type of quantum circuit. Indeed, the PGA-QM variants found better solutions for the GHZALL and DJ problems, but a worse one for the GHZ instances. One may also notice that the PGA-QM variants present rather similar results. That can be easily explained by the fact that they all rely on the same operators at each step. Recall that the differences between the three variants of PGA-QM are the population sizes and numbers of generations. The similarity in terms of best result found by each of these variants leads us to conclude that one can lower those parameters to improve the computational time (see next section) while maintaining the quality of the found solution. A study of the best parameters of the GAs (crossover, selection and mutation operators) could likely lead to improving those results.

One thing we have yet to mention is the differences between PGA-QM variant and SABRE and why one is better than the other depending on the circuit. To explain the differences, we studied the solutions they returned. Figure 6 illustrates the connectivity graph of the used hardware, where the red-colored qubits are the ones selected to be part of the solution found by either PGA1 or SABRE for each 80-qubit problems. One can notice that the PGA-QM’s solutions are scattered while SABRE ones are connected subgraphs of the connectivity graph. This is due to the SABRE routine that actively searches for connected solutions. It seems that such solutions are more adapted to the GHZ instances. Therefore, the PGA-QM struggles to outperform SABRE for those instances because it does not enforce connected solutions. However, the GHZALL and DJ instances seem to favor scattered solutions and therefore, the PGA-QM finds better solutions for those instances. The GHZALL circuit requires that all qubits are connected to one particular qubit (see Figure 5). That means there is a CNOT between qubit \( j \) and qubit 0 for all \( j > 0 \) while the GHZ circuit does not present such “overloaded” qubit. Our assumption to explain why the GHZALL seems to prefer scattered solutions is that the overloaded qubit is easier to connect to all the others in a scattered mapping, i.e. it required fewer gates to do so. On the other hand, for the GHZ circuits, a connected mapping works best.

It is especially true for the GHZ 80-qubit circuit since the displayed solution is the global minimum: a 80-qubit linear subgraph is found. For that particular instance, the PGA-QM leads to much worse result (10 times SABRE’s one). The reason is, for that instance, there actually is a solution that requires no circuit transformation and SABRE is able to find that solution because it looks for it first. Our metaheuristic approach struggles to find such a perfect solution. Our understanding of this is that the PGA-QM explores the search space reasonably well and finds a promising region but struggles to find the local minimum of said region. An hybridization of the PGA-QM with a local search operator could likely help in that regard.
5.2 Time analysis

From a computational time point of view, the execution of the PGA-QM is up to 65 times slower than SABRE for the larger (120-qubit) circuits (see Table 3). However, one can notice that reducing the population size and total number of generations with the early stopping criteria do not lead to significant worsening of the quality of the produced solutions while it greatly decreases the execution time. In particular, one can get similar results with a PGA-QM just 35 times slower than SABRE in the worst studied case. It is still a lot, but the execution time remains reasonable (a few minutes), and with finer tuned parameters, these results could likely be further improved. Note that in this section, we only compared PGA-QM with enough parallel threads to compute the cost of all individuals in parallel. The conclusion one can draw up here is that PGA-QM can outperform SABRE and find better mappings, and despite being quite slower, remains computationally acceptable.

5.3 Robustness analysis

Another comparison that can be made concerns the robustness of both approaches, defined as their sensibility to randomness. An algorithm is said to be more robust that another one if several executions return similar results despite stochastic operations. Looking at Table 4, one can notice that the PGA-QM variants are more robust. It is an important factor to take into account because a high sensibility to randomness means one can get an “unlucky” bad mapping, leading to a higher error rate of the corresponding quantum program. PGA-QM tends to fix that by being more robust. One can also notice that all PGA-QM variants are more robust than SABRE, but our fastest variant (PGA3 with an early stopping criterion of 10 generations without improvement) is less robust than our slowest one (PGA1 with a stopping after 30 generations). While the median result of both sets of parameters is very similar (see Section 5.1), the results with PGA3 are more split around that median, meaning that the time saved by the faster versions is lost dealing with robustness. However PGA3, even with early stopping after 10 generations without improvement, remains more robust than SABRE. Regarding the robustness scale, PGAs exhibit a better performance.
Table 3. Median time and Ratio (=PGA time/SABRE time) for all studied circuits and algorithms

<table>
<thead>
<tr>
<th>Solver</th>
<th>GHZALL 80</th>
<th>GHZALL 120</th>
<th>DJ 80</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time (s)</td>
<td>Ratio</td>
<td>Time (s)</td>
</tr>
<tr>
<td>PGA1</td>
<td>182.841</td>
<td>49.074</td>
<td>285.422</td>
</tr>
<tr>
<td>PGA2</td>
<td>143.095</td>
<td>38.568</td>
<td>225.280</td>
</tr>
<tr>
<td>PGA3</td>
<td>147.355</td>
<td>39.550</td>
<td>232.732</td>
</tr>
<tr>
<td>SABRE</td>
<td>3.725</td>
<td>4.445</td>
<td>3.856</td>
</tr>
<tr>
<td>PGA1 stop10</td>
<td>110.17</td>
<td>29.570</td>
<td>163.822</td>
</tr>
<tr>
<td>PGA2 stop10</td>
<td>78.825</td>
<td>21.156</td>
<td>111.972</td>
</tr>
<tr>
<td>PGA3 stop10</td>
<td>72.459</td>
<td>19.448</td>
<td>98.426</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Solver</th>
<th>DJ 120</th>
<th>GHZ 80</th>
<th>GHZ 120</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time (s)</td>
<td>Ratio</td>
<td>Time (s)</td>
</tr>
<tr>
<td>PGA1</td>
<td>289.968</td>
<td>61.093</td>
<td>339.560</td>
</tr>
<tr>
<td>PGA2</td>
<td>212.788</td>
<td>44.832</td>
<td>274.653</td>
</tr>
<tr>
<td>PGA3</td>
<td>212.439</td>
<td>44.759</td>
<td>284.734</td>
</tr>
<tr>
<td>SABRE</td>
<td>4.746</td>
<td>44.812</td>
<td></td>
</tr>
<tr>
<td>PGA1 stop10</td>
<td>151.263</td>
<td>31.869</td>
<td>189.291</td>
</tr>
<tr>
<td>PGA2 stop10</td>
<td>115.103</td>
<td>24.251</td>
<td>137.801</td>
</tr>
<tr>
<td>PGA3 stop10</td>
<td>96.942</td>
<td>20.425</td>
<td>152.789</td>
</tr>
</tbody>
</table>

Table 4. Inter Quartile Range and Median Average Difference for all studied circuits and algorithms

<table>
<thead>
<tr>
<th>Solver</th>
<th>GHZALL 80</th>
<th>GHZALL 120</th>
<th>DJ 80</th>
<th>DJ 120</th>
<th>GHZ 80</th>
<th>GHZ 120</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IQR</td>
<td>MAD</td>
<td>IQR</td>
<td>MAD</td>
<td>IQR</td>
<td>MAD</td>
</tr>
<tr>
<td>PGA1</td>
<td>11.5</td>
<td>5.5</td>
<td>14.75</td>
<td>8.5</td>
<td>8.75</td>
<td>6</td>
</tr>
<tr>
<td>PGA2</td>
<td>9.75</td>
<td>4.5</td>
<td>16.5</td>
<td>7.5</td>
<td>6.75</td>
<td>3.5</td>
</tr>
<tr>
<td>PGA3</td>
<td>10.25</td>
<td>5</td>
<td>14</td>
<td>7.5</td>
<td>10.25</td>
<td>5.5</td>
</tr>
<tr>
<td>SABRE</td>
<td>21</td>
<td>5</td>
<td>27.5</td>
<td>14.5</td>
<td>22.5</td>
<td>11</td>
</tr>
<tr>
<td>PGA1 stop10</td>
<td>9.75</td>
<td>5</td>
<td>18.5</td>
<td>9.5</td>
<td>13.75</td>
<td>6.5</td>
</tr>
<tr>
<td>PGA2 stop10</td>
<td>16.5</td>
<td>7.5</td>
<td>13</td>
<td>7</td>
<td>16.5</td>
<td>9</td>
</tr>
<tr>
<td>PGA3 stop10</td>
<td>15.5</td>
<td>8.5</td>
<td>11.75</td>
<td>7</td>
<td>13.75</td>
<td>7</td>
</tr>
</tbody>
</table>

5.4 Parallel scalability analysis

Lastly, let us discuss the scalability obtained with the parallelization of the PGA-QM for two different quantum circuits, the 80-qubit GHZ and the 40-qubit Quantum Fourier Transform (QFT) [14]. As one can notice in Figure 7, the speedup for GHZ reaches only 12.5 over 20 parallel threads. The two reasons behind this limited scalability are the following. First, the GA is not entirely parallel. As only the evaluation phase is done in parallel, the serial part negatively impacts the speedup. Secondly, the evaluation of the cost function is quite fast for the GHZ circuit (∼3 seconds). Therefore, parallel evaluations speed up the execution moderately as the serial parts remain important. Another important factor is the irregularity of the application. Given that the parallelization is done using the Master-Worker model, the workload is likely to be poorly balanced because of the irregularity of the application.

The QFT circuit is included as an additional problem instance to confirm the irregularity of the application. While using fewer qubits, this circuit is made of a lot more gates (80 for 80-qubit GHZ versus 840 for 40-qubit QFT), making the cost function more computationally expensive (∼40 seconds). Actually we can observe on the right side of Figure 7, the speedup is much better, reaching 16 over 20 parallel threads. Indeed, with an objective function more computationally expensive, the parallel part becomes significantly more important than the serial one, leading to a better speedup. This result indicates that the parallel GA is better suited for larger circuits (with many gates).
6 Conclusions and Future Works

In this paper, we have proposed a Parallel Genetic Algorithm to solve the Qubit Mapping problem (PGA-QM). This latter consists in mapping a quantum circuit on a graph of physical qubits composing a NISQ machine. This mapping is an important step in the transpilation of circuit-based quantum programs in NISQ computers. The objective is to transform the quantum circuit into another one which best matches the graph of physical qubits taking into account its limited connectivity constraint. The challenge is therefore to minimize the depth of the transformed circuit and the execution time and error rate consequently. The parallel GA has been integrated into the IBM Qiskit framework, experimented and compared to the SABRE heuristic provided in this latter. Medium-to-large circuits are considered as benchmarks in the experiments.

The reported results show that our proposed PGA-QM outperforms SABRE on some circuits in terms of error rate, while it is quite slower. Parallelism played a major role in speeding up the execution. The parallel scalability depends on the circuit and its size (number of gates). In addition, while for medium circuits, the parallel GA does not outperform SABRE, it is always more robust, meaning it more often leads to good mappings while it is not as heavily impacted by unlucky draws.

In the future, we plan to combine PGA-QM with single-solution metaheuristics in a two-level approach. At the high level, the hybridization will consist in applying a multi-start local search to the final population produced by the PGA-QM parallel algorithm. At a low level, a local search will be used as a mutation operator in the GA. A second perspective of this work will consist in extending the problem formulation to a multi-objective one considering additional cost functions than the circuit depth.

References


Machine Learning Applications in Road Safety: A Comprehensive Review

Ilyas Modni*, Zoubida Chorfi¹, and Abdelaziz Berrado¹

¹Ecole Mohammadia d’Ingénieurs – Morocco

Abstract

This comprehensive literature review explores the transformative potential of Machine Learning and Deep Learning methodologies in revolutionizing road safety across five fundamental pillars: Road safety management, safer roads, safer vehicles, safer road users, and post-accident response. As societies worldwide face increasing traffic challenges and encounter various road conditions, the study examines the diverse applications of ML across these critical dimensions. The review illuminates the versatility of ML techniques, ranging from traditional signal processing to advanced deep learning approaches. The findings underscore the profound impact of ML on road safety, offering actionable insights for accident prediction, infrastructure optimization, and a nuanced understanding of user behaviors. This study serves as an instructive roadmap, emphasizing the indispensable role of ML in shaping safer, adaptive, and efficient road environments.

*Speaker
Patient Visits Forecasting in the Post-Pandemic Era at Emergency Departments

Nicolas Haxaire¹,², Farah Mourad-Chehade¹, and Alice Yalaoui¹

¹ Computer Science and Digital Society Laboratory (LIST3N), University of Technology of Troyes, France
https://www.utt.fr
² Optacare
https://www.optacare.fr
✉Corresponding author: nicolas.haxaire@utt.fr

Abstract. COVID-19 disrupted time series trends in Emergency Departments (ED), causing a notable decrease in patient visits. Many studies focus on pre-pandemic or pandemic-specific data, raising concerns about models trained on altered reality. This paper proposes original strategies for patient visit post-pandemic forecasts. The proposed study couples deep learning models, such as LSTM or CNN, to pandemic-included preprocessing data to reduce accuracy loss. The first approach incorporates explicit COVID-19 features labeling the data in the deep model used for forecasting. In the second approach, the COVID anomaly period is manually excluded for the data and aggregated pre-COVID and post-COVID data are used. To illustrate the study, we assess the models using a dataset from a public French hospital, acknowledging challenges in time series forecasting. Results reveal that the proposed strategies achieve high performance, showcasing forecasting accuracies with MAPE values of 7.26% and 7.92% for the first strategy, and 8.55% and 8.24% for the second strategy.

Keywords: Forecast · Patient visits · AI · COVID-19 · Emergency Department

1 Introduction

Time series forecasting poses a significant challenge that scientists have grappled with for decades, spanning various domains. Understanding the future behavior of specific events holds the potential to benefit us all by reducing our ecological footprint and enhancing the daily lives of individuals. This perspective has spurred research in emergency services, aiming to predict patient visits for better upstream organization of human and material resources [2]. Anticipating these patterns helps avoid under- or over-staffed teams, directly impacting emergency service wait times and preventing unnecessary costs associated with resources that could be redirected within the hospital for other projects.

Numerous efforts have been undertaken over the decades [9] to enhance predictions for emergency services regarding patient visits. These endeavors have
witnessed significant evolution, starting with the exploration of exogenous and endogenous variables, such as incorporating weather conditions and calendar variables [16] into patient visit patterns, improving statistical methods, and the advent of machine learning and deep learning models.

These advancements have brought predictions closer to reality, empowering healthcare professionals to improve working conditions while enhancing the patient experience in emergency services. However, the landscape shifted dramatically in 2020 with the global arrival of COVID-19 [8, 11]. Lockdowns and the exclusive prioritization of COVID-19 cases disrupted the various patient arrival patterns in emergency services, to the extent that even highly performing models struggled to make accurate forecasts. Yet, it is precisely during these critical moments that hospitals should rely on reliable tools to assist them in emergencies and focus on what matters most – timely patient care. While the waves of COVID-19 are mostly behind us today, it underscores the vulnerability of algorithms to withstand such changes in the components of time series that we thought we knew so well.

In this conference paper, we delve into Deep Learning models associated with time series forecasting using various COVID-19-related data to construct a more robust architecture capable of adapting to changes in time series, should such events recur in the future. The study is based on a private dataset from public hospitals in France.

Section 2 provides an overview of relevant works, Section 3 delves into the approach, including dataset analysis, model implementation, and methods for enhancing accuracy, particularly during COVID period inclusion. Results are presented in Section 4, followed by discussion in Section 5. Section 6 distills findings, emphasizing strengths and limitations, practical implications for healthcare management, and potential future research avenues.

2 Related Work

Healthcare represents a crucial field of research, as advancements in this domain not only enhance the quality of life for patients but also significantly alleviate the demanding responsibilities faced by healthcare professionals. While the appeal of Deep Learning models in areas like computer vision for cancer diagnosis on radiography is undeniable, the forecasting of patient visits at Emergency Departments remains a vital and substantial field within healthcare research.

The inception of research in this area dates back to statistical models such as ARIMA, exploring the impact of calendar and meteorological factors on predictions [9, 4], particularly in regular and pediatric Emergency Departments. Over time, the landscape of models has evolved, introducing techniques like Exponential Smoothing (ETS), Random Forest [12], and Deep Learning models including Artificial Neural Networks (ANN), Long Short-Term Memory (LSTM), and Convolutional Neural Networks (CNN).

This subject matter has undergone scrutiny in the works of Wargon et al. [13] and Silva et al. [10]. Wargon’s review, spanning 1981 to 2007, predominantly
focused on statistical models. Conversely, Silva et al. [10] employed a PICO method, filtering papers based on specific criteria, analyzing a selected set primarily comprised of machine learning algorithms and delving into the features used for model training. In their comprehensive review, Silva et al. [10] suggested that an average of three years of historical data serves as a well-balanced training set, while cautioning about the potential mitigating effect of meteorological variables on predictions due to dataset size.

Recent publications have shifted the spotlight towards Deep Learning approaches, showcasing recurrent neural networks (RNN) like LSTM, Gated Recurrent Units, and convolutional models such as CNN, predominantly utilized for classification tasks. Zhao et al. [15] specifically focused on patient visit predictions across different acuity levels. Applying a triage method to classify patients into groups (P1, P2, P3, and PAll) based on acuity levels, they employed forecasting methods using a three-year period. Noteworthy features included calendar days, public holidays, meteorological variables like average temperature and rainfall, alongside a less conventional feature, Pollution Standard Index (PSI). Their LSTM/CNN hybrid model, with CNN layers for feature extraction and LSTM for sequence prediction, yielded impressive results with MAPE values of 17.37%, 7.19%, 6.11%, and 4.50% for P1, P2, P3, and PAll, respectively.

In the study by Harrou et al. [5], a comprehensive comparison of seven deep learning models was conducted to forecast patient flows at Emergency Departments. The models considered encompassed a diverse array, including Deep Belief Network (DBN), Restricted Boltzmann Machines (RBM), Gate Recurrent Unit (GRU), well-established models like LSTM and CNN, and hybrid models such as CNN-GRU, LSTM-CNN, and Generative Adversarial Network based on Recurrent Neural Networks (GAN-RNN). Patient flow, inclusive of arrivals at the Emergency Department, was categorized into two subgroups denoted as CCMU1 and CCMU2, reflecting the French classification of patient conditions as urgent or non-urgent upon arrival.

The performance metrics, evaluated using the MAPE metric, revealed that the DBN models outperformed others, achieving impressive results with 3.027% for CCMU1 and 1.833% for CCMU2. In stark contrast, LSTM, a well-established model in time series forecasting, exhibited comparatively higher MAPE values, achieving 12.47% for CCMU1 and 7.12% for CCMU2. These findings underscore the efficacy of DBN models in the specific context of forecasting patient flows, outshining established models like LSTM in terms of prediction accuracy.

3 Methodology

In this section, the dataset employed for the current investigation is delved into, and the architectural framework is elucidated. Additionally, the various models instantiated are exhaustively described, accompanied by a detailed exposition of their respective hyperparameters fine-tuned through the utilization of the Optuna optimization framework. The section concludes with an exploration of the metrics used to assess the efficiency of our models in the field of daily forecasting.
3.1 Dataset
The dataset is collected from a public hospital in France, from January 2018 to July 2023. This temporal scope facilitates a nuanced examination of the COVID-19 impact on forecasting models. Within this timeframe, a total of 256,843 visits transpired, yielding a mean of 130.78, a standard deviation of 22.75, and a distribution characterized by a minimum percentile of 43, a 25th percentile of 121, a median (50th percentile) of 133, a 75th percentile of 144, and a maximum of 195 patients per day.

3.2 Proposed Architecture and Features
To our knowledge, the literature lacks studies that integrate the COVID-19 period into the forecasting of daily patient visits to the Emergency Department, with most papers singularly concentrating on this specific timeframe [8, 11]. Recent contributions in the field, such as by Zhao et al.’s works [14, 15], primarily emphasize the development of novel architectures to enhance prediction accuracy. However, it is noteworthy that these endeavors predominantly rely on historical data predating the COVID-19 era, covering periods such as January 2015 to December 2019. This inclination may stem from the scarcity of data available for the post-COVID period at the time these papers were written. A similar trend is observable in other publications, exemplified by Harrou et al.’s [6], which utilizes data from January 2011 to November 2013. Another potential reason to this trend is the difficulty to access hospital data due to governing regulations. In this study, we consider data with pre-, in- and post-COVID history.

As for forecast horizon, hospitals employ predictive models for diverse purposes, ranging from short-term predictions to efficiently manage human and material resources, to long-term predictions geared towards strategic budget allocation. In the study, specific emphasis is given to short-term predictions, delineated by forecasting horizons of one day ahead.

Recognizing the dynamic nature of data relevance, it is acknowledged that recent data exerts a more pronounced influence on forecasting than information from five years ago. Consequently, in the pursuit of forecasting daily patient visits, priority is given to attributes with anticipatory characteristics, such as calendar variables. This strategic emphasis aligns with established findings in the literature, where attributes like meteorological variables have demonstrated substantial impact on predictions [12].

The primary goal is to leverage historical data spanning the recommended 3-4 year range, encompassing the COVID period, with the aim of minimizing accuracy loss attributed to the peculiar COVID pattern.

To achieve this objective, two architectural approaches are proposed. The first involves the use of the entire time series while incorporating the features "covid_period" and "lockdown" where periods of lockdown and COVID pandemic in France were identified based on information provided by the French government. A parallel experiment will be conducted without incorporating these features for result comparison. The second approach entails retaining only pre
and post-COVID data, as their visual similarity has been observed, and concatenating them for analysis as suggested by Fig.1. However, this strategy has a significant drawback: if a new pandemic occurs, the model will have never experienced such substantial variations during its training.

![Original dataset time series](image)

**Fig. 1: Original dataset time series**

### 3.3 Models and Hyperparameters Optimization

The exploration of models trained for specific tasks and the optimization of hyperparameters has been a focal point across diverse applications within the field of machine learning. In the context of patient visits forecasting, as well as time series forecasting more broadly, three principal approaches have been employed. The first involves the experimentation and development of various models, while the second entails the utilization of existing models with a focus on optimizing hyperparameter selection. The third approach involves the introduction of new features for models to learn from.

In this paper, the latter two approaches have been implemented. The primary objective is not to attain unparalleled performance, a feat seldom achieved in existing literature. Rather, our focus lies in addressing the fluctuations attributed to the COVID period within our historical dataset. Furthermore, the aim is to fortify the models to preemptively counteract potential variations stemming from future pandemics within the time series.

To achieve this, we have opted for well-established models in the literature, specifically Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM). While CNNs are traditionally applied to image processing, they can also discern patterns within time series data. LSTMs, renowned for their capacity to capture diverse temporal patterns through a gated system, adeptly discard extraneous information.

These models are poised to adeptly capture emerging patterns linked to designated features, specifically "lockdown" and "covid_period," showcasing their resilience and efficacy in the field of deep learning forecasting for time series data.
### Table 1: Grid of Hyperparameters to optimize with Optuna

<table>
<thead>
<tr>
<th>Commons</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>batch_size</td>
<td>[7, 16, 32, 64, 128]</td>
<td></td>
</tr>
<tr>
<td>sequence_length</td>
<td>[3, 7, 14, 21, 30, 60, 90]</td>
<td></td>
</tr>
<tr>
<td>epochs</td>
<td>2000</td>
<td></td>
</tr>
<tr>
<td>learning_rate_patience</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>patience</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>dropout</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>learning_rate</td>
<td>(1e-4, 1e-1, log=True)</td>
<td></td>
</tr>
<tr>
<td>optimizer</td>
<td>[&quot;Adam&quot;, &quot;RMSprop&quot;]</td>
<td></td>
</tr>
<tr>
<td>cost_function</td>
<td>[&quot;MSELoss&quot;, &quot;RMSLELoss&quot;, &quot;HuberLoss&quot;]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LSTM</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>n_layers</td>
<td>(1, 4, steps=1)</td>
<td></td>
</tr>
<tr>
<td>hidden_size</td>
<td>[7, 32, 64, 250, 500, 1000]</td>
<td></td>
</tr>
<tr>
<td>bidirectional</td>
<td>[True, False]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CNN</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>n_layers</td>
<td>(1, 3, steps=1)</td>
<td></td>
</tr>
<tr>
<td>conv_1_layer</td>
<td>[512, 128, 64]</td>
<td></td>
</tr>
<tr>
<td>conv_2_layer</td>
<td>[256, 128, 64]</td>
<td></td>
</tr>
<tr>
<td>conv_3_layer</td>
<td>[128, 64, 32]</td>
<td></td>
</tr>
<tr>
<td>kernels</td>
<td>(1, 3, steps=1)</td>
<td></td>
</tr>
</tbody>
</table>

In the context of the study, "lockdown" denotes periods where individuals were restricted to their homes, while "covid_period" encompasses the various waves of the COVID-19 pandemic. It’s important to note that the "covid_period" includes lockdown periods, but the distinction lies in the fact that people were not consistently confined to their homes throughout the entire COVID-19 period; there were instances when individuals could venture out while adhering to protective measures such as masks. These features are seamlessly integrated into the training dataset, alongside other variables like calendar and meteorological factors. The incorporation of these features stems from their observed impact on the time series. Notably, during the initial COVID-19 lockdown, only cases with prior COVID-19 diagnoses were admitted, diverting fewer cases to general practitioners compared to normal times. Providing an explanation for these features furnishes valuable context during model training, enabling a deeper understanding of time series behavior amidst the COVID-19 period. This enhanced contextual understanding contributes to the models’ accuracy in forecasting, ensuring they consider the specific contextual nuances prevalent during the forecasting period.

Nevertheless, the implementation of these models allows for myriad variations, particularly in terms of hyperparameters. Fully Connected models were adopted in the approach employed. The optimization of hyperparameters was facilitated through Optuna, a framework developed by Akiba et al. [1]. Noteworthy for its ease of implementation and familiarity among AI developers, Optuna has previously found utility in production settings. It incorporates compelling
optimization methods, including sampling and pruning mechanisms, based on an extensive study system subjected to numerous trials. Table 1 outlines the grid of hyperparameters we explored with Optuna’s implementation.

For the CNN model, the Optuna optimization process determines the number of layers in the model. In the case of a single layer, the CNN model consists of a Conv1D layer in PyTorch, followed by an Average Pooling layer, a ReLU layer, a Dropout layer set to 0.2, and a Linear layer. If Optuna selects multiple layers and 256 for output channels, the model will feature multiple blocks of convolutional layers, each followed by ReLU and pooling layers, such as Conv1D(n_features, 256), Conv1D(256, 128), and so forth. Optuna selects kernels for convolutional and pooling layers, with system verification ensuring the trial is dropped if the kernel exceeds the dimensions of the convolutional layer.

For each trial, training of our model occurs on the partitioned dataset, where the test size constitutes 30% of the original dataset, and the remaining 70% is further divided into an 80/20 split for the training and validation datasets, respectively. Implementation of Early Stopping is employed to optimize computational efficiency. Additionally, various batch sizes are experimented with, recognizing that while batches can expedite computations, they also influence the accuracy of predictions. The choice of batch size is crucial, as it dictates the patterns the model learns from the data in each batch wave.

The Sequence Length parameter is also incorporated to enhance data volume, providing the model with more contextual information for learning. For instance, with a sequence length of 7, the data fed to the network becomes a tensor of size (batch_size, sequence_length, number_features). This implies that for a value at time $t_0$, the tensor encompasses the features of the preceding 6 days.

3.4 Metrics

Various metrics were employed in this study, selected in accordance with the specific task at hand. For the training of the Deep Learning models, three distinct cost functions were chosen: Mean Squared Error (MSE), Root Mean Squared Log Error (RMSLE), and Huber Loss. These functions serve as the mechanisms through which models adjust their weights after each epoch, utilizing the cost function alongside predictions and true values during the training process.

Metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE) have been selected due to their recognition in the literature, providing default benchmarks for comparing our results. The incorporation of HuberLoss as a loss function is noteworthy, given its capacity to combine the favorable characteristics of both MAE and RMSE. Furthermore, the inclusion of Root Mean Squared Log Error (RMSLE) serves a crucial purpose, as it is a metric less prone to being influenced by outliers.
4 Results

In the initial experiment, LSTM and CNN models were executed within the Optuna optimization framework, utilizing Bayesian Optimization as a backend. For this specific experiment, two additional features, namely "lockdown" and "covid_period," were introduced, as detailed in Section 3.

4.1 Forecasting 1 day

On the test datasets, notable results were achieved for the 1-day forecast horizon on both LSTM and CNN models. The MAPE for this horizon was 7.26%, accompanied by an MAE of 9.77 for the LSTM model. Additionally, the CNN models exhibited a MAPE of 7.92% and 10.54%, respectively, for the best parameters identified using the Optuna framework.

Table 2: Metrics measured during the three phases training, validation, and test for LSTM and CNN models in one-day forecasting with the best hyperparameters found with optuna

<table>
<thead>
<tr>
<th></th>
<th>Training</th>
<th>Validation</th>
<th>Test</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RMSE</td>
<td>MAE</td>
<td>MAPE</td>
<td>RMSE</td>
<td>MAE</td>
<td>MAPE</td>
</tr>
<tr>
<td>LSTM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Without COVID Features</td>
<td>12.45</td>
<td>9.68</td>
<td>7.18</td>
<td>12.67</td>
<td>9.29</td>
<td>7.07</td>
</tr>
<tr>
<td>With COVID Features</td>
<td>11.60</td>
<td>8.18</td>
<td>6.03</td>
<td><strong>11.87</strong></td>
<td><strong>8.87</strong></td>
<td><strong>6.60</strong></td>
</tr>
<tr>
<td>Drop COVID Period</td>
<td>10.17</td>
<td>6.82</td>
<td>5.14</td>
<td>15.15</td>
<td>12.27</td>
<td>9.62</td>
</tr>
<tr>
<td></td>
<td>15.25</td>
<td>11.89</td>
<td>8.55</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CNN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Without COVID Features</td>
<td>13.81</td>
<td>10.79</td>
<td>8.43</td>
<td>15.11</td>
<td>12.11</td>
<td>9.84</td>
</tr>
<tr>
<td>With COVID Features</td>
<td>12.41</td>
<td>9.57</td>
<td>7.61</td>
<td>15.28</td>
<td>12.01</td>
<td>9.78</td>
</tr>
<tr>
<td>Drop COVID Period</td>
<td><strong>9.52</strong></td>
<td><strong>6.26</strong></td>
<td><strong>4.71</strong></td>
<td>13.87</td>
<td>10.85</td>
<td>8.69</td>
</tr>
</tbody>
</table>

Table 2 reveals that, in general, incorporating explicit features such as "covid_period" and "lockdown" during training has a subtle yet discernible impact on the model’s ability to learn from the COVID pattern. The improvement is nearly observable for the LSTM model, resulting in a MAE of 9.78, compared to 10.76 for the test dataset. The CNN model shows an even more noticeable benefit from these additional pieces of information, achieving a MAE of 10.54, significantly lower than the 13.68 obtained without the features. The other RMSE and MAPE metrics evolve in the same way. At the end, LSTM with COVID features lead to the best results.

Fig.2 does not explicitly highlight the models exhibiting optimal performance; instead, it depicts the model that strives to adapt most effectively to variations, thanks to the implementation of our new features. Despite the close scores, visual inspections of Fig.2 indicate that the additional features contribute to an overall better fit to the variations in the test dataset. Interestingly, our best models in this context did not prioritize fitting a wide range of variations.

Results of Table 2 show also that, in the second approach where COVID period is excluded, CNN achieve higher scores than the methodology taking all data without COVID features.
Patient Visits Forecasting in the Post-Pandemic Era at ED

(a) LSTM - Predictions on the test dataset including COVID and lockdown features.

(b) LSTM - Predictions on the test on the original dataset excluding COVID and lockdown features.

(c) CNN - Predictions on test dataset including COVID and lockdown features.

(d) CNN - Predictions on test on original dataset excluding COVID and lockdown features.

Fig. 2: LSTM and CNN performances on test dataset with and without COVID-19 features

5 Discussion

In this paper, we employed the LSTM model, a component of the well-known RNN architecture, and CNN models for our experiments. While Deep Learning methods excel in various tasks such as Computer Vision and Language Processing, their performance in forecasting domains is not as pronounced. Forecasting, being a task highly influenced by context and research field, has witnessed a progression from statistical models to machine learning models and, more recently, to deep learning models with the surge in their usage. Significant endeavors are dedicated to enhancing prediction accuracy by employing novel pre-processing techniques and introducing advanced models like the Temporal Fusion Transformer [7,3]. Yet, a prevailing challenge lies in achieving accurate results while accounting for the influence of the COVID period on the quality of our historical data, pivotal for training our models.
Handling the time series poses one of the primary challenges faced when training models on data that includes the COVID period. Adopting a 70/30 split for the training and test sets, and then further splitting the train set into an 80/20 ratio for training and validation sets, illustrates a problem in the training loop. The validation period does not accurately represent the training period, displaying a distinctive curve when the trend starts to increase, as depicted in Fig.1. This limitation can impact how effectively our model learns. An unexplored idea in this paper involves exploring time series classification and applying different preprocessing or model configurations to this particular pattern.

Models implemented demonstrate sufficient performance to handle the COVID period in historical data, particularly when explicitly incorporating lockdown periods and government-announced COVID waves. These features contribute to a slightly enhanced accuracy and more fitting prediction curves, as illustrated in Fig.2. It is crucial to acknowledge that these features may not be known in advance in a real-case scenario. However, considering how algorithms are integrated and utilized in decision support tools, it is likely possible to conduct predictions in virtual environments. By running predictions with and without these features, the prediction that includes the features could be set up as by the software user. This would allow for the input of information about incoming days, especially if there is a possibility of a lockdown situation or a resurgence of a COVID wave.

Simultaneously, there is a need to devise models capable of responding promptly to anomalous fluctuations in the emergency department’s patient visit flow. This becomes pivotal for cultivating robust algorithms tailored for hospital production environments. In literature, most papers, including this one, are conducted in a research context, with fewer considering the practical integration of algorithms into software for operational use. Future research could focus on methodologies simulating real-world scenarios, such as training an LSTM model on 3-4 years of data and subsequently running it in a loop with online learning. This approach would shed light on how models should be managed in hospitals, addressing questions related to updating model weights, frequency of updates, and whether to incorporate new data daily or concatenate it with historical data.

6 Conclusion

Forecasting patient visits at the Emergency Department is crucial for efficient resource management through decision tools. Accurate forecasts empower scheduling software to construct robust schedules, ensuring the right number of nurses are on duty for each day, optimizing working conditions, and improving patient experiences with faster and more efficient care.

The COVID period has significantly impacted data quality, prompting this paper to address challenges and propose solutions for forecasting on current data, in contrast to papers focusing solely on historical datasets. To address this, we employed well-known LSTM and CNN models from the literature, with the goal
of reducing accuracy loss attributed to the unique pattern of the COVID period in the time series.

Two distinct preprocessing methods were involved in the approach. First, utilizing the entire time series, models’ performances were compared with and without explicitly incorporating COVID periods and lockdowns as features. This exposed a potential 2% enhancement in MAPE for one-day forecasting with the CNN model, with a less pronounced impact on LSTM MAPE. Nevertheless, models incorporating new features, in general, exhibited a better fit to real data for both tested horizons.

Secondly, we explored dropping the COVID period and utilizing only pre and post-COVID data, as these two periods exhibited similar patterns. This experiment yielded comparable results to the first method.

In conclusion, these experiments demonstrate that COVID data is exploitable and can be effectively preprocessed for use in further research or software implementations for practical use.

Declarations

This paper employs ChatGPT 3.5 as a linguistic tool to enhance syntax and readability throughout the document. The authors affirm that ChatGPT has not been utilized for any other purposes, and the articles have undergone multiple revisions to ensure that the authors’ intent and results remain unchanged during syntactic corrections.

References

2. Alaouchiche, Y., Ouazene, Y., Yalaoui, F., Chehade, H.: Optimized nurse scheduling to improve efficiency and well-being in hospitals: a case study in a french hospital


Solving a shareable-setup time prize collection
VRP applied to an electrical maintenance sector

R. F. Tavares¹, A. Nogueira², M. O. Batalha³, R. A. Martins⁴, H. F. Moralles⁵, R. Seeberger⁶, G. Mendonca⁷, and M. Goncalves⁸

¹ Federal University of Sao Carlos
tavares@dep.ufscar.br
² Federal University of Sao Carlos
arineiana@gmail.com
³ Federal University of Sao Carlos
dmob@ufscar.br
⁴ Federal University of Sao Carlos
ram@dep.ufscar.br
⁵ Federal University of Sao Carlos
herickmoralles@dep.ufscar.br
⁶ EDP Espírito Santo Distribuição de Energia S.A.
rafael.seeberger@edp.com
⁷ EDP São Paulo Distribuição de Energia S.A.
gustavo.mendonca@edp.com
⁸ EDP Espírito Santo Distribuição de Energia S.A.
mauro.goncalves@edp.com

1 Introduction

The prize-collection vehicle routing problem (PCVRP) is a challenging NP-Hard problem with significant practical applications. It can be summarized as follows: let \( N^c \) be a set of available clients that can be visited by a set of \( k \) vehicles. Each client \( i \) generates a prize \( p_i \) when visited. The goal is to select a subset of \( N^c \) that maximizes the total profit and respects a set of problem-specific constraints.

As an example of the practical application of this problem, [2, 3] consider a variation of the prize-collection VRP, namely the Steiner Team Orienteering Problem (STOP), where there is a set of mandatory locations, and presents a cutting-plane approach to solve it.

[1] also uses the PCVRP aiming at the challenges faced during a disaster response, by modeling it as a multi-vehicle prize collecting, where the prizes are related to transversing arc, not reaching nodes. In this problem, the goal is to reconnect a set of network nodes maximizing the total profit within a specified time limit. They present a MILP formulation and a matheuristic method. [9] also uses the prize-collecting paradigm in disaster recovery. Besides a matheuristic approach, the authors provide a GRASP algorithm. [8] presents a software tool to support decision-making in disaster management, also facing this problem as a variant of the prize-collecting VRP.

Another practical application of the prize-collecting VRP is in the military field: e.g., [4] present a prize-collecting approach to plan reconnaissance missions.
of Unmanned Aerial Vehicles (UAV). In this research, the prize is composed of three objectives related to the information collected, duration of the mission, and safety. The prize is also time-dependent.

Beyond military applications, [7] modeled the problem of generating touristic itineraries by a prize-collecting approach. The authors use a firefly metaheuristic and combine it with game theory to accommodate the goals of heterogeneous groups of tourists. [12] also approach a tourist itinerary generation, presenting an interactive framework to solve it.

A further application of the Prize Collecting VRP is Green Logistics: [11] uses a firefly algorithm to solve the Environment Prize-Collecting Vehicle Routing Problem (E-PCVRP). This research aims to select a set of nodes to be visited, considering an objective function composed of the total CO2 emissions and the total prize earned.

Focusing on theoretical development, more approaches to solving the prize-collecting routing problems can be found in the literature, such as Iterated Greedy approach ([13]), Variable Neighborhood Search ([6]) and Iterated Local Search ([10]).

In this paper, we present a problem faced by a company that performs maintenance on electrical distribution networks. In the studied scenario, a set of homogeneous repairmen teams is assigned to perform a set of maintenance tasks on clients of a city. When a team arrives at a client, a set of common tasks must be performed, such as field delimitation and perform security checks. If two clients are close enough and scheduled in sequence, there is no need to perform those tasks again. The priority (prize) of each client is a function of the corresponding due date. The goal is to schedule a subset of clients to maximize the sum of prizes.

To present this research, this document is structured as follows: section 2 presents a formal definition of the problem; section 3 presents the mixed-integer linear formulation proposed by this paper; section 4 presents the iterated local search approach; the results are presented in section 5; the final remarks are presented in section 6.

2 Problem Definition

The problem can be stated as follows: a set \( N_c \) of \( n \) clients can be visited by \( k \) service teams. When visiting, there is a working time period composed of a service time \( s_i \) and a shared setup service time \( m_i \). If several clients are attended in a close location (e.g., several apartments in the same building), the shared service time is counted just once for these clients. To ensure that only similar activities can share the shared service time, the parameter \( b_{ij} \) is used. When a client is visited, the objective function increases according to the profit \( p_i \). In this specific case, the \( p_i \) is the priority given by the company to a service order, and it’s defined based on each due date. The travel time between clients and the base is given by \( t_{ij} \). The amount of working hours available for the service team is given by \( WH \).
In this paper, the solution space is represented as a graph $G$ with $n+2$ nodes. Node 0 is the departure point, nodes $\{1, \ldots, n\}$ represent clients, and node $n+1$ represents the final node of any route. When the service team visits node $j$ after node $i$, both the binary variables $y_j$ and $x_{ij}$ are set to "1". The shared time is not computed on two compatible clients thanks to variable $q_i$. The arrival time of the service team on node $j$ after visiting node $i$ is given by $Z_{ij}$. Table 1 summarizes the symbols adopted by this paper to define both the problem instance and a respective solution.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N^c = {1, 2, \ldots, n}$</td>
<td>Set</td>
<td>All client nodes from the graph</td>
</tr>
<tr>
<td>$N = {0} \cup N^c \cup {n + 1}$</td>
<td>Set</td>
<td>All nodes of the graph</td>
</tr>
<tr>
<td>$\mu_i$</td>
<td>Parameter</td>
<td>Profit earned by visiting client $i$</td>
</tr>
<tr>
<td>$t_{ij}$</td>
<td>Parameter</td>
<td>Transport time between nodes $i$ and $j$</td>
</tr>
<tr>
<td>$s_i$</td>
<td>Parameter</td>
<td>Service time of node $i$</td>
</tr>
<tr>
<td>$m_i$</td>
<td>Parameter</td>
<td>Shared service time regarding node $i$</td>
</tr>
<tr>
<td>$b_{ij}$</td>
<td>Parameter</td>
<td>Assumes &quot;1&quot; if clients $i$ and $j$ can share service time (does not include location checking)</td>
</tr>
<tr>
<td>$WH$</td>
<td>Parameter</td>
<td>The amount of working hours available to the service team</td>
</tr>
<tr>
<td>$M$</td>
<td>Parameter</td>
<td>A very large number</td>
</tr>
<tr>
<td>$y_i$</td>
<td>Binary variable</td>
<td>Assumes &quot;1&quot; if client $i$ is visited, 0 otherwise</td>
</tr>
<tr>
<td>$q_i$</td>
<td>Binary variable</td>
<td>Assumes &quot;1&quot; if the shared service time was previously computed at client $i$</td>
</tr>
<tr>
<td>$x_{ij}$</td>
<td>Binary variable</td>
<td>Assumes &quot;1&quot; if any vehicle uses arc $i \rightarrow j$</td>
</tr>
<tr>
<td>$Z_{ij}$</td>
<td>Real variable</td>
<td>Time when client $j$ is reached after arriving in client $i$</td>
</tr>
</tbody>
</table>

Table 1: Symbols used to define a problem and a solution.

3 A Mixed-Integer Formulation

The problem stated previously can be formally defined by the mixed-integer linear programming model (MILP) composed of Equations 1–11. This model, based on the work of [3], can be summarized as follows: Equation 1 if the objective function (to maximize the total prize collected). Equation 2 forces that the service team will depart from a node only if it arrives at the same node. Equation 3 binds the values of $x_{ij}$ and $y_j$. Equation 4 forces the following variables to be zero: the $y$ variables related to the initial and final nodes; the $x_{ij}$
variables that arrive at the initial node and depart from the final node; the $x_{ij}$ variables that arrive and depart from the same node; and the time arrived at node $i$ from itself, $Z_{ii}$. Equation 5 assures that a departure from the initial node has a corresponding arrival at the final node. Equation 6 set variable $q_{ij}$ to “1” if the service team does the route $i \rightarrow j$ and $i$ and $j$ can share the service time.

The moment when the service team arrives at node $i$ is given by Equations 7–10. Equation 11 performs subtour elimination.

4 Iterated Local Search approaches

Based on the work of [5], this paper presents two Iterated Local Search (ILS) algorithms to solve this problem. The core procedure is summarized on Algorithm 1, and relies on three operators: swap, destroy, and insert. Those operators update two sets, an ordered solution set, that contains the routes, and an unordered set containing all the unassigned jobs. For any corresponding pair of solution and unassigned jobs, one can state that $\text{solution} \cup \text{unassignedJobs} = N^c$. 

$$\max \sum_{i \in N^c} p_i \cdot y_i$$
(1)

$$\sum_{j \in N^c \setminus \{i\}} x_{ij} = \sum_{j \in N^c \setminus \{i\}} x_{ji} \quad \forall i \in N^c$$
(2)

$$\sum_{i \in N \setminus \{j\}} x_{ij} \leq y_j \quad \forall j \in N \setminus \{0, n+1\}$$
(3)

$$y_0 + y_{n+1} + \sum_{i \in N} x_{i,0} + \sum_{i \in N} x_{i,(n+1)}, i + \sum_{i \in N} (x_{ii} + Z_{ii}) = 0$$
(4)

$$\sum_{i \in N \setminus \{0\}} x_{0i} = \sum_{i \in N \setminus \{n+1\}} x_{i,(n+1)}$$
(5)

$$M \cdot q_i \geq \sum_{k \in N \setminus \{j\}} x_{ki} + x_{ij} - 1 \quad \forall \begin{cases} i \in N^c \\ j \in N^c \\ i \neq j \\ b_{ij} = 1 \end{cases}$$
(6)

$$Z_{0j} = t_{0j} \cdot y_j \quad \forall j \in N^c$$
(7)

$$Z_{ij} \leq M \cdot x_{ij} \quad \forall i, j \in N$$
(8)

$$\sum_{j \in N \setminus \{i\}} Z_{ij} - \sum_{j \in N \setminus \{i, i(n+1)\}} Z_{ji} = \left((t_{ij} + s_i) \cdot x_{ij} + m_i \cdot (1-q_i)\right) \quad \forall \begin{cases} i \in N^c \\ \{i, j\} \in N^c \end{cases}$$
(9)

$$0 \leq Z_{ij} \leq WH \quad \forall \begin{cases} \{i, j\} \in N^c \\ i \neq j \end{cases}$$
(10)

$$\sum_{(i,j) \in S} x_{ij} \leq |S| - 1 \quad \forall S \subseteq N, |S| \geq 2$$
(11)
Algorithm 1: The pseudo-code for the ILS Algorithm

**Input:** A problem, with the parameters defined in Table 1
**Input:** Algorithm parameters: nRuns, noImpRun
**Output:** A set of feasible routes

1. `BestSolution, UnassignedJobs \rightarrow \text{insert}(\emptyset, N)`
2. for $i \leftarrow 1$ to nRuns do
3.     `PartialSolution, UnassignedJobs \leftarrow \text{swap}(\text{bestSolution}, \text{UnassignedJobs})`;
4.     `Update UnassignedJobs, if necessary`;
5.     `PartialSolution, UnassignedJobs \leftarrow \text{destroy}(\text{bestSolution}, nd)`;
6.     `newSolution \rightarrow \text{insert}(\text{PartialSolution}, \text{UnassignedJobs})`; 
7.     if If there is an improvement on the objective function then
8.         `Update bestSolution`;
9.     `PartialSolution = bestSolution \setminus UnassignedJobs`;
10.    if The number of cycles without improvement is equal to noImpRun then
11.       return Current Solution

The first step of this algorithm is to create an initial (feasible) solution by applying the insertion operator. This results in two sets, the current (and so far, the best) solution and the corresponding unassigned jobs set. A second stage is designed to improve the solution. This is performed by (i) swapping the current unassigned jobs (swap operator), and (ii) removing elements from the current best solution (destroy operator) and applying the insert operator on the remaining solution and unassigned jobs set.

The swap operator tries to swap each one of the unassigned job $i$ with a currently assigned job $j$. This is done by calculating the score $\text{scoreSwap}(i, j)$ and then swapping where the maximum score is found (and not zero). The value of $\text{scoreSwap}(i, j)$ is given by Equation 12. In this equation, $\text{diffTime}(i, j)$ is the absolute value of the amount of time allocated to the route when $j$ is replaced by $i$.

$$ \text{scoreSwap} = \max \left\{ 0, \left( p_i - p_j \right) \cdot \frac{1}{1 + \text{diffTime}(i, j)} \right\} \quad (12) $$

Two different destroy operators are implementing, leading to two distinct algorithms: the destroy operator of the first ILS algorithm (ILS1) randomly transfers a set of $nd$ subsequent jobs from the current solution to the set of unassigned jobs. This is done by selecting a job from the current solution and removing it and the next $nd$ jobs. If the route ends before the $nd - ith$ job is found, the algorithm just removes the elements until the end of the route. The second ILS algorithm (ILS2) selects $nd$ random elements from the current solution at random.

On both ILS approaches, the insertion operator tries to insert each job $i$ present on the Unassigned Jobs set. It inserts after the job $j$ with a higher value greater than zero of $\text{scoreInsert}(i, j)$, as given by Equation 13. In this equation, $\text{addTime}(i, j)$ is the amount of time added to the route after inserting job $i$ after job $j$.

$$ \text{scoreInsert} = \max \left\{ 0, \left( p_i - p_j \right) \cdot \frac{1}{1 + \text{addTime}(i, j)} \right\} \quad (13) $$
The key idea of Equations 12 and 13 is to prioritize orders with higher profit and geographically closer. On these equations, the shareable setup times influence both $\text{diffTime}(i, j)$ and $\text{addTime}(i, j)$. Those values are composed of: (i) the transit time of the route; (ii) the shareable setup time (if needed); and (iii) the service time. If both $i$ and $j$ can share the setup time, it is not considered in one of the jobs.

5 Results and analysis

To analyze the behavior of those algorithms, we use a set of instances created inspired by the data supplied by a field maintenance department of a power distribution company. Those data use a real set of GPS coordinates of clients attended during the beginning of 2023, and we assume the distances as pure Euclidian distances between the points. Since this dataset comprehends mostly urban locations, we assume all vehicle speeds as 50km/h. The service times are the same as the ones reported by the company. The shareable service times are assumed to be 10$\%$ of the total activity time (this estimation was performed after analyzing the historical data provided by the company). By varying the number of available teams ([1, 2, 3, 4, 5]) and the number of possible clients ([10, 20, 30, 40, 50, 75, 100]), this paper uses a dataset of 35 problem instances.

The MILP formulation presented in section 3 was implemented in Python 3.10, using the CPLEX 22.1.1.0 library. During the implementation, the only change in the solver default parameters was the time limit (set to 3,600s). The ILS presented in section 4 was implemented in Rust 1.74.0, and called as a library by Python 3.10. Both solution approaches were run on an Ubuntu Linux machine with i7-1360P processor and 32GB of RAM. Both are set to run on a single processor core. Based on preliminary experiments, we set the parameters of the ILS approaches as: 10000 improvement cycles, removal of up to 30 jobs, and stop after 2000 non-improvement cycles. After running all the problem instances, three analyses were performed, aiming to analyze the quality of the solution and the computational time required to run each procedure.

The first analysis focuses on the solution status of the MILP approach. According to Figure 1, the MILP solver could achieve a solution on all problem instances, but faced difficulties in prove optimality within the time limit of 3,600s.
Fig. 1: Solution status obtained by the MILP solver

The next analysis performed focuses on the execution time of each algorithm. Two analyses are performed: Figure 2 presents the performance plots of each solving approach, and Table 2 presents the corresponding descriptive statistics. According to the data presented, it’s clear that the MILP approach faces difficulties in solving (i.e., obtain a solution and prove its optimality) medium and large instance sets. The ILS, on the other hand, returned the result in less than 2 seconds in all instances.

Table 2: Descriptive Statistics of the execution times of the three approaches

<table>
<thead>
<tr>
<th>Size</th>
<th>ILS1 Min</th>
<th>ILS1 Mean</th>
<th>ILS1 Std. Dev.</th>
<th>ILS2 Min</th>
<th>ILS2 Mean</th>
<th>ILS2 Std. Dev.</th>
<th>MILP Min</th>
<th>MILP Mean</th>
<th>MILP Std. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.01</td>
<td>0.02</td>
<td>0.02</td>
<td>0.01</td>
<td>0.02</td>
<td>0.02</td>
<td>0.02</td>
<td>0.02</td>
<td>0.02</td>
</tr>
<tr>
<td>20</td>
<td>0.06</td>
<td>0.13</td>
<td>0.16</td>
<td>0.09</td>
<td>0.13</td>
<td>0.09</td>
<td>0.07</td>
<td>0.13</td>
<td>0.09</td>
</tr>
<tr>
<td>30</td>
<td>0.27</td>
<td>0.25</td>
<td>0.20</td>
<td>0.28</td>
<td>0.18</td>
<td>0.13</td>
<td>0.18</td>
<td>0.60</td>
<td>7.34</td>
</tr>
<tr>
<td>40</td>
<td>0.44</td>
<td>0.53</td>
<td>4.23</td>
<td>7.34</td>
<td>0.34</td>
<td>1.42</td>
<td>1.34</td>
<td>2160.19</td>
<td>1971.56</td>
</tr>
<tr>
<td>50</td>
<td>1.08</td>
<td>0.83</td>
<td>18.04</td>
<td>0.15</td>
<td>0.34</td>
<td>1.42</td>
<td>0.09</td>
<td>6.29</td>
<td>1567.42</td>
</tr>
<tr>
<td>75</td>
<td>1.31</td>
<td>1.80</td>
<td>20.22</td>
<td>0.25</td>
<td>1.66</td>
<td>10.99</td>
<td>0.25</td>
<td>8.68</td>
<td>373.42</td>
</tr>
<tr>
<td>100</td>
<td>2.20</td>
<td>3.41</td>
<td>53.41</td>
<td>0.59</td>
<td>2.95</td>
<td>21.68</td>
<td>0.59</td>
<td>21.58</td>
<td>1868.35</td>
</tr>
</tbody>
</table>
Fig. 2: Performance Charts

(a) All approaches

(b) ILS approach
The major drawback of using a heuristic method is the lack of proof of optimality, and the lack of tools to assess the quality of the returned solution. To analyze the quality of the solutions, Table 3 presents the descriptive statistics of the relative deviation \( RD = (fitness - \min(fitness))/\min(fitness) \) related to the values of the fitness found by each method. According to the data presented, the MILP could not achieve the optimal solution (due to the limit of 3,600s running time), and the proposed ILS outperforms it. Moreover, when observing the data presented in Table 3 and Figure 2, it was clear that both ILS1 and ILS2 outperform the MILP approach regarding the computational times required to obtain the final solution. Regarding the value of the fitness, the solution found by the MILP solver was better than the heuristics, although requiring more computational resources. This lead us to two research opportunities: to derive better bounds for the MILP formulation (to allow a faster convergence) and to enhance the neighborhood structures of the ILS approaches.

Table 3: Descriptive Statistics of the relative deviation of the fitness found by each method

<table>
<thead>
<tr>
<th>Size</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Average</th>
<th>Std. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILS1</td>
<td>ILS2</td>
<td>MILP</td>
<td>ILS1</td>
<td>ILS2</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>0</td>
<td>0.10</td>
<td>0</td>
<td>0.02</td>
</tr>
<tr>
<td>40</td>
<td>0</td>
<td>0.19</td>
<td>0.05</td>
<td>0</td>
</tr>
<tr>
<td>50</td>
<td>0.13</td>
<td>0.05</td>
<td>0.17</td>
<td>0.08</td>
</tr>
<tr>
<td>75</td>
<td>0.09</td>
<td>0.05</td>
<td>0.16</td>
<td>0.09</td>
</tr>
<tr>
<td>100</td>
<td>0.05</td>
<td>0.03</td>
<td>0.23</td>
<td>0.14</td>
</tr>
</tbody>
</table>

Fig. 3: Graphical analysis of the relative deviation found by each method
6 Final remarks

This paper approached a variation of the well-known prize-collection vehicle routing problem, inspired by a real-world scenario found in the maintenance of a distribution of electricity network. In this case, some activities such as field delimitation and security checks should not be performed twice if two consecutive clients are close enough.

To solve this problem, we proposed two approaches: a MILP formulation and an ILS algorithm. Both strategies were able to solve small problem instances up to optimality. When the instance size increases, the computational time required by the MILP increases, frequently reaching the time limit of 3,600s. The ILS could return better solutions, both in the quality of the solution and in the computational time required.

As future works, we believe that there are some opportunities: the first one is to increase the instances dataset to obtain more insights regarding the behavior of the proposed procedures. A second opportunity is to enhance the MILP formulation and analyze the applicability of more advanced techniques such as column generation and Logic-based Benders, to assist the solver to achieve more suitable bounds for the problem. The third mapped opportunity is to design new neighborhood structures and derive new meta-heuristics to enhance the results given by the current ILS procedure.

Acknowledgements

We extend our sincere appreciation to the "Programa de Pesquisa, Desenvolvimento e Inovação da Agência Nacional de Energia Elétrica - ANEEL" for their invaluable support and funding of the project - PD-00391-0048/2023 - "Desenvolvimento de Sistemas de Decisão ao Apoio da Gestão dos Serviços de CCM e STC". Additionally, we wish to thank EDP São Paulo Distribuição de Energia S.A. and EDP Espírito Santo Distribuição de Energia S.A. for their generous provision of resources and facilities, and contributing to the successful execution of this study by Federal University of São Carlos.
Bibliography


Robust Models for Learning Languages

Tomasz Jastrząb\textsuperscript{1}[0000−0002−7854−9058], Frédéric Lardeux\textsuperscript{2}[0000−0001−8636−3870], and Eric Monfroy\textsuperscript{2}[0000−0001−7970−1368]

\textsuperscript{1} Silesian University of Technology, Poland
Tomasz.Jastrzab@polsl.pl
\textsuperscript{2} University of Angers, LERIA, France

Abstract. This paper focuses on learning probabilistic finite state automata in the context of grammatical inference. We introduce a two-step process: first, constructing a robust meta-model of 3-sort Non-deterministic Finite Automaton (3NFA); second, deriving a probabilistic automaton using a weighted-frequency approach. This allows for creating 3NFAs that are more or less accepting, respecting the sample classification. The proposed methodology offers robustness by adjusting weights for higher acceptance or rejection probabilities. Overall, the research extends the application of grammatical inference to probabilistic responses with potential implications in various domains.

Keywords: nondeterministic automata · SAT models · classification

1 Introduction

Numerous real-world phenomena can be conceptualized as sequences with syntactic structures, including DNA, natural language sentences, and electrocardiograms. Grammatical Inference involves the process of acquiring grammars and languages from data, specifically from such sequences. Machine learning of grammars holds diverse applications, spanning syntactic pattern recognition, adaptive intelligent agents, computational biology, prediction, and more. Our focus lies in learning grammars as finite state automata. This involves examining a language sample including positive sequences that must belong to the language and negative sequences that the automaton must reject.

Various approaches addressed the challenge of learning finite automata. These include ad-hoc methods like DeLeTe2 [2], which involves merging states from the prefix tree acceptor (PTA). Another perspective involves a range of algorithms for regular language inference, as discussed in [16]. Metaheuristic approaches (e.g., hill-climbing in [15]) have also been explored. Alternatively, the problem can be modeled as a Constraint Satisfaction Problem (CSP) and solved using generic tools like non-linear programming [17] or Boolean formulas [5, 7, 11, 8].

All of these studies focus on Deterministic or Non-deterministic Finite Automata (DFA or NFA). In both cases, the automata provide binary responses for a given a word, indicating that the word either is part of the language or it
is not part thereof. Given that samples are typically finite, with a maximum of hundreds of words, and considering that regular languages are infinite, this classification may be overly restrictive. Hence, the interest in exploring probabilistic responses, such as whether a word is part of the language with a certain probability (e.g., x%). This leads to the question: “How can one learn a probabilistic automaton from a sample comprising both positive and negative words?”

In this article, we focus on the robust modelization of probabilistic automata to provide probabilistic responses to the question of whether a word belongs to the learned language. We proceed in two steps. We first consider Non-deterministic Finite Automata with 3 sorts of states: accepting final states (+_states) which validate positive words, rejecting final states (−_states) which reject negative words, and whatever states (?)_states) that are not conclusive.

We use a property similar to the one for 2-sort NFA [7] to build a 3-sort automaton with only one accepting final state and one rejecting final state. Some extra constraints enable us to automatically reduce this size k + 2 automaton into a size k automaton. Moreover, we over-constrain this “meta” model in order to obtain models that maximize/minimize the number of accepting/rejecting states. Our meta-model is robust since we obtain k_3NFAs that are able to be more or less accepting, but always respecting the sample classification. The second phase is based on a methodology that we proposed recently to derive a 3-sort probabilistic automaton from a 3NFA and a sample of positive and negative words (see [9] or [10]). We first capture frequencies based on the sample, considering simple factors like “the number of positive words terminating in a specific +_state”, or more complex like “the number of times negative words passed through a given transition and terminated in a −_state versus any other state”. We outline the transformation process first from a 3-sort NFA to a 3-sort Weighted-Frequency NFA, which can subsequently be converted into a probabilistic NFA once weights are assigned. The probabilistic NFA enables the calculation of the likelihood that a word belongs to the language or not. Note that adjusting the weights allows for the generation of automata with a higher inclination toward acceptance or rejection, and this constitutes a second factor of robustness of our meta-model.

The paper is organized as follows: in Sect. 2 we state the problem and present a meta-model to solve it. In Sect. 3 we discuss several model instances. In Sect. 4 we present the weighted-frequency and probabilistic automata. Section 5 contains the description of conducted experiments. Section 6 concludes the paper.

2 The problem and our robust meta-model

To simplify models (by removing simple but special cases), we consider in the following that λ, i.e., the empty word, is not an element of the samples S.

3 These 3-sort NFAs appear to be suitable for our objective as discussed in [3].
2.1 The problem and some notations

Let $\Sigma = \{s_1, \ldots, s_n\}$ be an alphabet of $n$ symbols, $K$ be the set of integers $\{1, \ldots, k\}$, $Pref(w)$ (resp. $Suf(w)$) be the set of all prefixes (resp. suffixes) of the word $w$, that we extend to $Pref(W)$ (resp. $Suf(W)$) for a set of words $W$.

**Definition 1.** A 3-sort non-deterministic finite automaton (3NFA) is a 6-tuple $A = (Q, \Sigma, I, F_+, F_-, \delta)$ with: $Q = \{q_1, \ldots, q_k\}$ a finite set of states, $\Sigma$ a finite alphabet, $I$ the set of initial states, $F_+$ the set of accepting final states, $F_-$ the set of rejecting final states, and $\delta: Q \times \Sigma \to 2^Q$ the transition function. For simplification, note that in what follows, we will consider only one initial state, i.e., $I = \{q_1\}$. We also denote by $k\_3NFA$ a 3NFA with $k$ states.

A learning sample $S = S^+ \cup S^-$ comprises a set $S^+$ of “positive” words from $\Sigma^*$ that the inferred 3-sort NFA must accept, and a set $S^-$ of “negative” words that it must reject. The language recognized by $A$, denoted as $L(A)_A$, includes words for which there exists a sequence of transitions from $q_i$ to a state of $F^+$. On the other hand, the language rejected by $A$, $L(A)_R$, encompasses words for which there exists a sequence of transitions from $q_i$ to a state of $F_-$. We focus on non-ambiguous automata, i.e., $L(A)_A \cap L(A)_R = \emptyset$, i.e., no positive word terminates in a rejecting final state, and no negative word terminates in an accepting final state.

The NFA inference problem is intrinsically a Boolean problem, and thus, well suited for SAT solvers. Hence, we consider the following variables:

- $k$, the size (the number of states) of the 3NFA to be generated,
- a set of $k$ Boolean variables $F_+ = \{a_1, \ldots, a_k\}$ (resp. $F_- = \{r_1, \ldots, r_k\}$) determining whether state $i$ is a final accepting (resp. rejecting) state or not,
- $\Delta = \{\delta_{s, i, j, q} \mid s \in \Sigma$ and $(i, j) \in K^2\}$, a set of $nk^2$ Boolean variables representing the existence of transitions from state $q_i$ to state $q_j$ with the symbol $s \in \Sigma$, for each $i$, $j$, and $s$.
- we define $p_{w, q_i, q_m, q_1, q_2, \ldots, q_{m+1}}$ as the path $q_1, \ldots, q_{m+1}$ for a word $w = s_1 \ldots s_m$:
  $$p_{w, q_i, q_m, q_1, q_2, \ldots, q_{m+1}} = \delta_{s_1, q_i, q_2} \land \ldots \land \delta_{s_m, q_m, q_{m+1}}.$$  
Although the path is directed from $q_1$ to $q_{m+1}$ (it is a sequence of derivations), we will build it either starting from $q_1$, from $q_m+1$, or from both sides [8]. Thus, the notation $q_1, q_{m+1}$ has no direction/arrow. Paths will be built recursively, and we need at most $O(k^3)$ Boolean variables $p_{s, r_1, r_2, \ldots, r_k}$, with $\sigma = \sum_{w \in S} |w|$.

2.2 The meta-model

For lack of space, we consider directly the $(k+2)\_3NFA$ we use later in this work. Consider a sample $S$. If there is a $k\_3NFA$ for $S$, there is also a $(k+2)\_3NFA$ for $S$. This property is obvious and useless, but we can refine it to build what we call $k\_NFA^*$ extensions.

Let $A = (Q, \Sigma, \{q_1\}, F_+, F_-, \delta)$ be a $k\_NFA$. Then, there always exists a $(k+2)\_3NFA$, $A' = (Q \cup \{q_{k+1}, q_{k+2}\}, \Sigma, \{q_1\}, \{q_{k+1}\}, \{q_{k+2}\}, \delta')$, such that:

4 We discard models with 0/1 variables, either from INLP [17] or CSP [13]: we made some tests with various models with PyCSP [12] and obtained some very poor results.
there is only one + state $q_{k+1}$ and one − state $q_{k+2}$,
- each transition of $A$ is copied in $A'$,
- incoming transitions to accepting final state are duplicated to the new accepting final state $q_{k+1}$,
- the same transition duplication is made for rejecting final states to the new rejecting final state $q_{k+2}$,
- there is no outgoing transition from states $q_{k+1}$ and $q_{k+2}$,
- no negative (resp. positive) word terminates in the states from $F_+$ (resp. $F_-$). This is obvious in $A$, we have to make it effective in $A'$.

The interest of this $(k + 2)$-3NFA is that the complexity for building suffixes (which gives the complexity of the complete model) is now in $O((k + 2)^3)$ instead of $O(k^3)$ (see [7]). We now give the constraints of the $(k + 2)$-3NFA. Let $K_+ = \{1, 2, \ldots, k + 2\}$:

- For each $w \in S^+$, there must be a path to the + state $k + 1$, and there must not be one to the − state $k + 2$:
  \[
  p_u, q_{k} \rightarrow_1 ^{q_{k+1}} \quad \neg p_u, q_{k} \rightarrow_1 ^{q_{k+2}} \tag{1}
  \]

- It is the opposite for each word $w \in S^-$:
  \[
  p_u, q_{k} \rightarrow_2 ^{q_{k+1}} \quad \neg p_u, q_{k} \rightarrow_2 ^{q_{k+2}} \tag{2}
  \]

- For each word $uv \in S^+$ or $u'v' \in S^-$, a path is composed of a path for the prefix and one for the suffix of each word:
  \[
  \bigvee_{j \in K} p_u, q_{j} \rightarrow_1 ^{q_{k+1}} \wedge p_v, q_{j} \rightarrow_1 ^{q_{k+2}} \quad \bigvee_{j \in K} p_u', q_{j} \rightarrow_2 ^{q_{k+1}} \wedge p_v', q_{j} \rightarrow_2 ^{q_{k+2}} \tag{3}
  \]

- For each word of $S$, prefixes are built recursively, on the left for prefixes $u = s, s \in \Sigma$, on the right for prefixes $u = xs, s \in \Sigma$:
  \[
  \bigwedge_{i \in K} \delta_s, q_{i} \rightarrow q_{i}' \leftrightarrow p_s, q_{i} \rightarrow \left( \bigvee_{j \in K} p_x, q_{j} \rightarrow \wedge \delta_s, q_{j} \rightarrow q_{j}' \right) \tag{4}
  \]

- Similarly for suffixes (on the left for $v = s$, on the right $v = sx$), with $k^? = k + 1$ for positive words and $k + 2$ for negative ones:
  \[
  \bigwedge_{i \in K} \delta_s, q_{i} \rightarrow q_{i}' \leftrightarrow p_s, q_{i} \rightarrow \left( \bigvee_{j \in K} p_x, q_{j} \rightarrow \wedge \delta_s, q_{j} \rightarrow q_{j}' \right) \tag{5}
  \]

- There is no outgoing transition from the final states:
  \[
  \bigwedge_{s \in \Sigma} \bigwedge_{i \in K_+} \neg \delta_s, q_{i \rightarrow k+1} \rightarrow q_i' \wedge \neg \delta_s, q_{i \rightarrow k+2} \rightarrow q_i' \tag{6}
  \]
Each incoming transition of the accepting (resp. rejecting) final state $q_{k+1}$ (resp. $q_{k+2}$) must also terminate in another state (duplication), with $k' = k + 1$ for positive words and $k' = k + 2$ for negative ones:

$$\bigwedge_{s \in \Sigma} \left( \bigwedge_{i \in K} \left( \delta_{s,q_i,q_{k'}} \rightarrow \bigvee_{j \in K} \delta_{s,q_j,q_{k'}} \right) \right)$$

(7)

To be able to reduce the $(k + 2)_3NFA^*$ into a $k_3NFA$, we must consider what we call possibly rejecting (?−_state) and possibly accepting (?+_state) final states of the $k_3NFA$. We thus, add 2 corresponding sets of Boolean variables $\{a^*_1, \ldots, a^*_k\}$ and $\{r^*_1, \ldots, r^*_k\}$. The $(k+2)_3NFA^*$ is reduced to a $k_3NFA$ by just removing states $q_{k+1}$, $q_{k+2}$, and their incoming transitions. Then, the final states are fixed among the possible final states, i.e., determining the $a^*_i$ and the $r^*_i$ which are accepting/rejecting final states of the $k_3NFA$. To define these possible final states, we have to consider the following constraints:

• A negative (resp. positive) word cannot terminate in an accepting (resp. rejecting) possible final state:

$$\bigwedge_{i \in K} \left( a^*_i \rightarrow \bigwedge_{w \in S^-} \neg p_{w,q_i,q_{k'}} \right) \quad \bigwedge_{i \in K} \left( r^*_i \rightarrow \bigwedge_{w \in S^+} \neg p_{w,q_i,q_{k+2}} \right)$$

(8)

With Constraint (8), Constraints (1) and (2) right-part are no longer needed.

• Each accepting (resp. rejecting) possible final state validates at least one positive (resp. negative) word of $S$:

$$\bigwedge_{i \in K} \left( a^*_i \rightarrow \bigvee_{v \in S^+} \bigvee_{j \in K} \left( p_{v,q_i,q_j} \land \delta_{s,q_j,q_{k'}} \land \delta_{s,q_{j+1},q_{k+1}} \right) \right)$$

(9)

$$\bigwedge_{i \in K} \left( r^*_i \rightarrow \bigvee_{v \in S^-} \bigvee_{j \in K} \left( p_{v,q_i,q_j} \land \delta_{s,q_j,q_{k'}} \land \delta_{s,q_{j+1},q_{k+2}} \right) \right)$$

(10)

• Each positive (resp. negative) word terminates in at least one accepting (resp. rejecting) possible final state:

$$\bigwedge_{w \in S^+} \bigvee_{i \in K} \left( p_{w,q_i,q_{k'}} \land a^*_i \right) \quad \bigwedge_{w \in S^-} \bigvee_{i \in K} \left( p_{w,q_i,q_{k'}} \land r^*_i \right)$$

(11)

Although these constraints are redundant, we keep them since they are binary clauses, and thus very efficiently used by the SAT solver (polynomial 2-SAT).

• A state cannot be both accepting and rejecting possible final state:

$$\bigwedge_{i \in K} \neg \left( a^*_i \land r^*_i \right)$$

(12)
Several models can be considered, either building words starting with their prefixes, or with their suffixes, or from both sides, i.e., what we call hybrid models.

We thus have to determine where to split each word \( w \in S \) into a prefix \( u \) and a suffix \( v \). We then consider the set of prefixes \( S_u = \{ u | u.v \in S \} \) and \( S_v = \{ v | u.v \in S \} \) the set of suffixes. The model is the instantiation of the meta-model \( M_M \) which is the conjunction of Constraints (1–12). We consider here only one model, the ILS model, which is an hybrid model that optimizes each word splitting using iterated local search (ILS). The model \( ILS_{M_M}(Init) \), based on a local search optimization [14] of word splittings, starting with an initial configuration \( Init \), minimizes the fitness function \( f(S_p, S_s) = |Pref(S_p)| + k \cdot |Suf(S_s)| \) to instantiate the meta-model \( M_M \). As the initial configuration, we consider here the model \( S^* \), which optimizes suffixes building. The model resulting from the instantiation with the model \( ILS_{M_M}(S^*) \), together with a given sample \( S \), composes the SAT instance \( I(S) \) of which solutions are 3NFA.

**Reduction of \( ? \)-states** We now want to enforce that a \( ? \)-state cannot be final for a negative word. To this end, we consider the following constraint:

\[
\bigwedge_{i \in K} \bigwedge_{w \in S^-} (\neg \pi \cdot r_i^* \rightarrow a_i^*)
\]

(13)

The meta-model \( M_1 = M \wedge (13) \) is then used for building the model \( ILS_{M_1}(S^*) \).

Intuitively, it seems clear that \( M \wedge (13) \) is over constrained compared to \( M \). This means that if there is a solution, it is a correct solution. But in case of UNSAT answer, we cannot be sure that \( M \) has no solution. However, we made tests with 30 instances of the Stamina [6] benchmark, and we always obtained a SAT solution. Up to now, we do not have a proof whether \( M_1 \) is over constrained or not. Given a sample \( S \), the resulting SAT instance is thus \( I_1 = ILS_{M_1}(S^*)(S) \).

**Reduction of \( ? \)-states and increase of \( + \)-states** Intuitively, we want to obtain an automaton with as much as possible \( + \)-states (and thus a more “accepting” automaton) than \( M \) and \( M_1 \). To this end, we consider the meta-model \( M_2 = M_1 \wedge (14) \) with Constraints (14) defined by:

\[
\bigwedge_{i \in K} (\neg r_i^* \rightarrow a_i^*)
\]

(14)

We feel (but we still do not have a proof), as for the previous model, that this model is over constrained due to Constraints (13). The model \( ILS_{M_2}(S^*) \) is built from the meta model \( M_2 \). Together with a sample \( S \), we instantiate this model and \( S \) into the instance \( I_2 = ILS_{M_2}(S^*)(S) \).
Reduction of ?_states and reduction of +_states We want to obtain a model with fewer positive final states than before. To do so, we consider an automaton \( \mathcal{A} \) which is a solution of the SAT instance \( I_1 = ILS_{M_1}(S^*)(S) \) (generated as described in Sect. 3) and we apply to \( \mathcal{A} \) a post-treatment to reduce the number of +_states: it computes a minimal covering of the positive words by the +_states; it tests all subsets of states belonging to the set of +_states and returns the one of smallest size that still preserves the validation of positive words. The subsets are tested from smallest to largest to facilitate the search.

We call this SAT instance \( I_3 \). Note that we could have applied the post-treatment on \( I_2 \), but first, the process would be more costly, and second, we do not feel the difference would be significant.

4 Classifying words

Using a sample and a generated k_3NFA, we can build a weighted-frequency automaton. From it, we can create a probabilistic automaton to classify words.

4.1 Weighted-frequency automata

Usually, in a frequency automaton, the integer \( n \) attached to a transition \( \delta(q, a, q') \) means that this transition was used \( n \) times [3]. Here, we want to count differently positive (resp. negative) words terminating in a +_state (resp. −_state) from positive (resp. negative) words terminating in a ?_state. We thus need different weights, and thus, we obtain automata that reflect weighted frequencies.

We first need to define some weights \( \omega(x,w,s) \) where \( x \) defines if the weight is used for a final state (\( f \)) or a transition (\( \delta \)), \( w \) is “−” for negative words and “+” for positive ones, and \( s \) is the sort of state the word terminates in (+, −, ?):

- Weights to be used for weighting final states (\( f \)): \( \omega(f,+,+)(q) \) (resp. \( \omega(f,+,?)(q) \)) is a weight associated with positive words terminating in the +_state (resp. ?_state); the weights \( \omega(f,-,-) \) and \( \omega(f,-,?) \) are defined similarly for negative words terminating in a −_state and in a ?_state,

- Weights for transitions (\( \delta \)): \( \omega(\delta,+,+)(q) \) (resp. \( \omega(\delta,+,?)(q) \)) is a weight associated with positive words terminating in a +_state (resp. ?_state); the weights \( \omega(\delta,-,-) \) and \( \omega(\delta,-,?) \) are defined similarly for negative words terminating in a −_state and in a ?_state.

For the frequency, we also need to count how many times a word terminates in a given state, or passes through a given transition:

- Counters for ending in a given state \( q \): a counter \( \phi(f,+,+)(q) \) (resp. \( \phi(f,+,?)(q) \)) for counting the number of times (i.e., the numbers of physical paths for all positive words) a positive word terminates in the +_state \( q \) (resp. in the ?_state \( q \)). \( \phi(f,-,-)(q) \) and \( \phi(f,-,?)(q) \) are defined similarly for negative words and −_states.
– Counters for passing through a given transition \((q, s, q')\): a counter \(\phi(\delta, +, +)(q, s, q')\) (resp. \(\phi(\delta, +, -)(q, s, q')\)) for counting the number of times a positive word uses a given transition \((q, s, q')\) within a physical path that terminates in a \(+\) state (resp. in a \(-\) state). \(\phi(\delta, -)(q, s, q')\) and \(\phi(\delta, -)(q, s, q')\) are defined similarly for negative words and \(-\) states.

Note that for frequency we count each possible path for each word: for example, a positive word can have several paths terminating in the same \(+\) state. For more details and a possible implementation of these counters, please refer to [9].

**Definition 3 (3_NPFA).** A 3-sort non-deterministic probabilistic finite automaton (3_NPFA) is a 10-tuple \(A = (Q, \Sigma, I, \delta, \Gamma(f,+), \Gamma(f,-), \Omega(\delta,+), \Omega(\delta,-))\) where:

- \(Q, \Sigma, I = \{q_1\}, F_+, F_-\) and \(\delta\) have the same meaning as for a usual 3NFA,
- \(\Omega(f,+)(q)\) is a weighted frequency function \(Q \to \mathbb{N}\), i.e.,
  \[
  \Omega(f,+)(q) = \begin{cases} 
    \omega(f,+)(\cdot) \cdot \phi(f,+)(q) & \text{if } q \in F_+ \\
    \omega(f,+)(\cdot) \cdot \phi(f,+)(q) & \text{if } q \in Q \setminus (F_+ \cup F_-) \\
    0 & \text{otherwise}
  \end{cases}
  \]
- \(\Omega(f,-)(q)\) is a function \(Q \to \mathbb{N}\), \(\Omega(f,-)(q)\) defined as above but for negative words and rejecting states (i.e., \(\cdot\) is replaced by \(\cdot\)), and \(F_+\) by \(F_-\).
- \(\Omega(\delta,+)(q, s, q') = \omega(\delta,+)(\cdot) \cdot \phi(\delta,+)(q, s, q') + \omega(\delta,+)(\cdot) \cdot \phi(\delta,+)(q, s, q')\)
- \(\Omega(\delta,-)(q, s, q') = \omega(\delta,-)(\cdot) \cdot \phi(\delta,-)(q, s, q') + \omega(\delta,-)(\cdot) \cdot \phi(\delta,-)(q, s, q')\)

Remember that these automata are non-deterministic, and so, there can be several paths for a word, terminating in different states. Thus, for a given word, we are interested in all paths regardless of the sort of the terminating state.

### 4.2 Probabilistic automata

We can define 3-sort probabilistic automata (3_NPFA) with probabilities for transitions and probabilities for states to be \(+\) state or \(-\) state.

**Definition 4 (3_NPFA).** A 3-sort non-deterministic probabilistic finite automaton is an 8-tuple \(A = (Q, \Sigma, I, \delta, \Gamma(f,+), \Gamma(f,-), \Gamma(\delta,+), \Gamma(\delta,-))\) where:

- \(Q, \Sigma, I = \{q_1\}\), and \(\delta\) are defined as with 3NFA,
- \(\Gamma(f,+)(q)\) a function \(Q \to [0, 1]\), i.e., \(\Gamma(f,+)(q)\) is the probability of state \(q\) to be a \(+\) state,
- \(\Gamma(f,-)(q)\) a function \(Q \to [0, 1]\), i.e., \(\Gamma(f,-)(q)\) is the probability of state \(q\) to be a \(-\) state,
- \(\Gamma(\delta,+)(q, s, q')\) is the probability for a positive word to pass by the transition \(\delta(q, s, q')\).
– $\Gamma(\delta,-)$ – similar to $\Gamma(\delta,+)$ for negative words.

A 3_NPFA must respect the following constraint\(^5\):

$$\forall q \in Q, \left\{ \sum_{q' \in Q, s \in \Sigma} \left( \Gamma(\delta,+)(q, s, q') + \Gamma(f,+)(q) \right) = 1 \right\} \sum_{q' \in Q, s \in \Sigma} \left( \Gamma(\delta,-)(q, s, q') + \Gamma(f,-)(q) \right) = 1$$

### 4.3 From weighted-frequency to probabilistic automata

Consider a 3_NWFFA $A = (Q, \Sigma, I, F_+, F_-, \delta, \Omega(f,+), \Omega(f,-), \Omega(\delta,+), \Omega(\delta,-))$. We derive a 3_NPFA $A' = (Q', \Sigma', I', \delta', \Gamma(f,+), \Gamma(f,-), \Gamma(\delta,+), \Gamma(\delta,-))$ such that:

- $Q = Q'$, $\Sigma = \Sigma'$, $I = I'$, and $\delta = \delta'$
- the probability for $q$ to be an accepting final state is the weighted frequency of words of $S^+$ terminating in $q$, divided by the sum of the weighted frequencies of the positive words outgoing from $q$ plus the weighted-frequency of positive words ending in $q$:

$$\forall q \in Q, \Gamma(f,+)(q) = \frac{\Omega(f,+)(q)}{\Omega(f,+)(q) + \sum_{s \in \Sigma, q' \in Q} \Omega(\delta,+)(q, s, q')}$$

- the $\Gamma(\delta,-)$ are computed similarly for negative words replacing “+” by “−”.
- the probability for a positive word to follow transition $\delta(q, s, q')$ is computed similarly as the probability of ending in $q$:

$$\forall q, q' \in Q, \forall s \in \Sigma, \Gamma(\delta,+)(q, s, q') = \frac{\Omega(\delta,+)(q, s, q')}{\Omega(\delta,+)(q, s, q') + \sum_{s' \in \Sigma, q'' \in Q} \Omega(\delta,+)(q, s', q'')}$$

- the computation is similar for negative words replacing “+” by “−”.

These probabilities respect the properties of 3_NPFA.

### 5 Experimentation

#### 5.1 Dataset and Experimental Setup

Instances were generated utilizing regular expressions (regexp). For each sample, we specified a regexp that characterizes the words in $S^+$ and randomly produced a given number of words for this set, with lengths ranging from 1 to 15 symbols. Set $S^-$ was formed by randomly rearranging the positive words while ensuring they did not conform to the specified regular expression. The words were not necessarily unique within $S^+$ or $S^-$. The employed regexp were as follows:

- A: (0|1)(001|000|10)*0,
- B: [0–4][0–4][0–4](012|123|234)*0[1],

\(^5\) Remember that we consider only one initial state. In case of several initial states, some probabilities of being initial positive and initial negative can be added.
The dataset includes sub-samples with 20% and 60% of the initial words in each sample, maintaining a balance between the sizes of positive (S+) and negative (S−) examples through the generation of negative instances.

The models were implemented in Python using the PySAT library [4]. The experiments were carried out on a computing cluster with Intel-E5-2695 CPUs. Running times were limited to 15 minutes, including model generation and solving time. We used the Glucose [1] SAT solver with default options. Given its deterministic nature, a single execution was run for each experiment.

The classification of words was implemented in Java and run on a machine with Intel Core i7-7560U CPU. The aggregation of transition and state probabilities along the NFA paths was based on the multiply-max approach. Given word $w = s_1 \ldots s_m$, for each path $P_{w,q_{m+1}}^{(i)} = \delta_{s_1,q_2} \wedge \ldots \wedge \delta_{s_m,q_{m+1}}$, we:

1. Computed the probability that the word is positive $\Gamma_{w,+}(P_{w,q_{m+1}}^{(i)}) = \Gamma_{(+,+)}(q_1, s_1, q_2) \cdots \Gamma_{(+,+,+)}(q_m, s_m, q_{m+1}) \Gamma_{(+,+)}(q_{m+1})$, and similarly that the word is negative $\Gamma_{w,-}(P_{w,q_{m+1}}^{(i)}) = \Gamma_{(\delta,-)}(q_1, s_1, q_2) \cdots \Gamma_{(\delta,-)}(q_m, s_m, q_{m+1}).$

2. Selected the maximum probabilities $\Gamma_{w,+}^{\max} = \max_{i} \Gamma_{w,+}(P_{w,q_{m+1}}^{(i)})$ and $\Gamma_{w,-}^{\max} = \max_{i} \Gamma_{w,-}(P_{w,q_{m+1}}^{(i)})$ among all the paths for word $w$.

3. Considered the word positive if $\Gamma_{w,+}^{\max} > \Gamma_{w,-}^{\max}$, negative if $\Gamma_{w,+}^{\max} < \Gamma_{w,-}^{\max}$, and indefinite otherwise.

We selected the following set of weights: $\omega_{(f,+)} = \omega_{(f,-)} = 3$, $\omega_{(\delta,+)} = \omega_{(\delta,-)} = 1$, $\omega_{(\delta,+)} = \omega_{(\delta,-)} = 0.75$, and $\omega_{(\delta,+)} = \omega_{(\delta,-)} = 0.25$, to foster the states over the transitions and to promote the final states over non-conclusive states. Finally, to perform the classification we used the NFA generated for the 20% and 60% subsets, and classified the words included in the 100% samples, excluding the words already contained in the sub-samples.

### 5.2 Deriving 3NFA from a meta-3NFA

To experimentally assess the presence of the desired properties during the construction of the various models outlined in Section 3, we generated and examined corresponding 3NFA for all variations (20%, 60%, and 100%) of each dataset (A, B, and C). This investigation involved employing different instances of the models (I, I₁, I₂, and I₃). We constructed $k + 2$ 3NFA automata with 8, 13, and 18 states, transforming them into $k$ 3NFA automata with 6, 11, and 16 states.

Table 1 precisely displays the expected results. Instance $I₁$ generates an automaton with fewer ? states than in $I$. These states are predominantly transformed into − states. Instance $I₂$ converts all ? states from $I₁$ into + states. Finally, the anticipated reduction in + states by $I₃$ compared to $I₁$ is noticeable only for four instances (B-20%-11, C-20%-16, B-60%-16 and C-60%-16). This is likely due to the already minimal number of + states in $I₁$.
Table 1. Results for SAT instances coming from different models with more or less +_states and ?_states. '-' means that no automaton was found within the time limit.

<table>
<thead>
<tr>
<th>k</th>
<th>20%</th>
<th>60%</th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.3 Probabilistic 3NFA

The 3NFA were subsequently converted into probabilistic 3NFA applying the weights mentioned in Section 5.1. The quality of classification was evaluated based on the accuracy (15) and F1-score (16):

$$\text{Acc} = \frac{TP + TN}{TP + TN + FP + FN}$$ (15)

$$F1 = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$$ (16)

where $TP$ denotes true positives, $TN$ denotes true negatives, $FP$ denotes false positives, and $FN$ denotes false negatives.

The obtained metrics are shown in Tables 2 and 3. The metrics were computed excluding the indefinite classification results (columns marked 20% and 60%) and considering the indefinite results as false negatives (if the word is positive and $I_{(w,+)}^{(w,+)} \leq I_{(w,+)}^{(w,+)}$) or false positives (if the word is negative and $I_{(w,+)}^{(w,-)} \leq I_{(w,+)}^{(w,+)}$). Our idea is that, on the one hand, if the probabilities for a word to be positive and negative are equal, we may say that there is not enough evidence to make a definite decision. On the other hand, we may want to focus on the actual misclassifications thus removing the indefinite results.

We found that the differences in accuracy (Table 2) between the 20% and 60% subsets are statistically significant at $\alpha = 0.05$ (based on the t-test with $p = .0083$). The same conclusion applies to the pair 20% and 60%* ($p = .000673$). These results indicate, as expected, that building an NFA using a larger portion of the original sample allows us to achieve better classification results on unseen data coming from the same distribution. On the other hand, there are no statistically significant differences between 20% and 20%, as well as 60% and 60% pairs. This indicates that the number of indefinite classifications was relatively low. The results in Table 3 are in line with those shown in Table 2. Again, we observe statistically significant differences between 20% and 60% ($p = .000695$) as well as 20%* and 60%* ($p = .000424$) results, and no statistically significant results between 20% and 20%* as well as 60% and 60%* pairs.
Table 2. Classification accuracy. Values in the columns labelled 20%* and 60%* consider the non-conclusive results as false positives or false negatives.

<table>
<thead>
<tr>
<th>$k$</th>
<th>Instance</th>
<th>20%</th>
<th>60%</th>
<th>20%*</th>
<th>60%*</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>$I$</td>
<td>0.80</td>
<td>0.59</td>
<td>0.54</td>
<td>0.80</td>
</tr>
<tr>
<td></td>
<td>$I_1$</td>
<td>0.86</td>
<td>0.57</td>
<td>0.56</td>
<td>0.81</td>
</tr>
<tr>
<td></td>
<td>$I_2$</td>
<td>0.86</td>
<td>0.57</td>
<td>0.56</td>
<td>0.81</td>
</tr>
<tr>
<td></td>
<td>$I_3$</td>
<td>0.86</td>
<td>0.57</td>
<td>0.56</td>
<td>0.81</td>
</tr>
<tr>
<td>11</td>
<td>$I$</td>
<td>0.61</td>
<td>0.57</td>
<td>0.55</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td>$I_1$</td>
<td>0.91</td>
<td>0.67</td>
<td>0.49</td>
<td>0.96</td>
</tr>
<tr>
<td></td>
<td>$I_2$</td>
<td>0.91</td>
<td>0.68</td>
<td>0.49</td>
<td>0.96</td>
</tr>
<tr>
<td></td>
<td>$I_3$</td>
<td>0.91</td>
<td>0.67</td>
<td>0.49</td>
<td>0.96</td>
</tr>
<tr>
<td>16</td>
<td>$I$</td>
<td>0.58</td>
<td>0.58</td>
<td>0.48</td>
<td>0.63</td>
</tr>
<tr>
<td></td>
<td>$I_1$</td>
<td>0.68</td>
<td>0.68</td>
<td>0.51</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td>$I_2$</td>
<td>0.70</td>
<td>0.68</td>
<td>0.51</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td>$I_3$</td>
<td>0.68</td>
<td>0.68</td>
<td>0.51</td>
<td>0.98</td>
</tr>
</tbody>
</table>

Overall, the quality of classification is satisfactory, regardless of the regular expression considered (A, B, and C). As to the model instantiations, we usually can observe an improvement in the metrics achieved by instances $I_1$–$I_3$ over instance $I$, especially for the regular expressions B and C (an exception from this rule can only be observed for C-60%-6). As to regular expression A, for smaller values of $k$ and 60% subset, $I$ seems to perform better than the others.

Apart from the classification metrics, we also analyzed the distribution of absolute differences between the $\Gamma_{(w,+)}$ and $\Gamma_{(w,-)}$ values (cf. Fig. 1). This analysis allows us to define the decision margin ensuring definite classification results. The majority of differences (31%) falls within the first interval, $[0, 10^{-5}]$, which indicates that the differences are typically very small. It can be explained by the way the classifier aggregates the probabilities, producing small values in the first place. As to the extreme values, the indefinite results (0 difference) were obtained in 7% of cases, which proves that the NFA show good classification capabilities. On the other end, the difference of 1 occurred in 8% of cases and was solely related to the fact that no path for some words was found. Hence, the negative probability was set to 1, while the positive probability was set to 0.

6 Conclusions

We discussed some robust models for inferring 3-sort NFA capable of acting as probabilistic word classifiers. We proposed several instantiations of our meta-model differing in the way the final states are treated, yielding automata that are more accepting/rejecting. The conducted experiments have shown that the inferred 3NFA perform well in the classification task in terms of achieved metrics. In the future, we plan to employ some heuristics to fine-tune the weights and to find the best ways of aggregating the probabilities.
Table 3. F1-score values. Values in the columns labelled 20%* and 60%* consider the non-conclusive results as false positives or false negatives.

<table>
<thead>
<tr>
<th>k</th>
<th>Inst</th>
<th>20%</th>
<th>60%</th>
<th>20%*</th>
<th>60%*</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>I</td>
<td>0.82 0.60 0.55</td>
<td>0.98 0.72 0.68</td>
<td>0.82 0.60 0.54</td>
<td>0.98 0.72 0.68</td>
</tr>
<tr>
<td></td>
<td>I_1</td>
<td>0.83 0.61 0.52</td>
<td>0.76 0.77 0.55</td>
<td>0.70 0.57 0.40</td>
<td>0.65 0.69 0.45</td>
</tr>
<tr>
<td></td>
<td>I_2</td>
<td>0.83 0.61 0.52</td>
<td>0.76 0.77 0.55</td>
<td>0.70 0.57 0.40</td>
<td>0.65 0.69 0.45</td>
</tr>
<tr>
<td></td>
<td>I_3</td>
<td>0.83 0.61 0.52</td>
<td>0.76 0.77 0.55</td>
<td>0.70 0.57 0.40</td>
<td>0.65 0.69 0.45</td>
</tr>
<tr>
<td>11</td>
<td>I</td>
<td>0.61 0.60 0.52</td>
<td>0.98 0.69 0.54</td>
<td>0.61 0.60 0.51</td>
<td>0.98 0.69 0.54</td>
</tr>
<tr>
<td></td>
<td>I_1</td>
<td>0.91 0.68 0.24</td>
<td>0.95 0.82 0.54</td>
<td>0.78 0.67 0.22</td>
<td>0.91 0.77 0.50</td>
</tr>
<tr>
<td></td>
<td>I_2</td>
<td>0.91 0.69 0.24</td>
<td>0.95 0.84 0.54</td>
<td>0.78 0.67 0.22</td>
<td>0.91 0.79 0.50</td>
</tr>
<tr>
<td></td>
<td>I_3</td>
<td>0.91 0.68 0.24</td>
<td>0.95 0.82 0.54</td>
<td>0.78 0.67 0.22</td>
<td>0.91 0.77 0.50</td>
</tr>
<tr>
<td>16</td>
<td>I</td>
<td>0.53 0.57 0.36</td>
<td>0.64 0.69 0.55</td>
<td>0.53 0.51 0.36</td>
<td>0.64 0.69 0.55</td>
</tr>
<tr>
<td></td>
<td>I_1</td>
<td>0.64 0.61 0.33</td>
<td>0.98 0.67 0.55</td>
<td>0.58 0.57 0.30</td>
<td>0.98 0.52 0.55</td>
</tr>
<tr>
<td></td>
<td>I_2</td>
<td>0.65 0.61 0.33</td>
<td>0.98 0.67 0.52</td>
<td>0.59 0.57 0.30</td>
<td>0.98 0.52 0.52</td>
</tr>
<tr>
<td></td>
<td>I_3</td>
<td>0.64 0.61 0.33</td>
<td>0.98 0.65 0.55</td>
<td>0.58 0.57 0.30</td>
<td>0.98 0.52 0.55</td>
</tr>
</tbody>
</table>

Fig. 1. Histogram of the absolute differences between maximum positive and negative probabilities. The black discs represent the number of words for which the difference was either 0 (on the left) or 1.
References

Optimisation-based classification tree: A game theoretic approach to group fairness

Georgios I. Liapis and Lazaros G. Papageorgiou

The Sargent Centre for Process Systems Engineering, Department of Chemical Engineering, UCL (University College London), Torrington Place, London WC1E 7JE, UK
georgios.liapis.20@ucl.ac.uk, l.papageorgiou@ucl.ac.uk

Abstract. The growing use of machine learning algorithms in decisions that significantly affect people necessitate interpretable and fair approaches. Mathematical programming based machine learning models have attracted attention because of the flexibility they provide to integrate features like interpretability and fairness, combined with high accuracy. This work introduces a mathematical programming based classification tree that uses a game theoretic approach to address group fairness. The proposed mathematical formulation is a Mixed Integer Linear Programming model using a piecewise linearisation strategy based on special-ordered sets. The overall misclassification rate is the fairness metric examined and the Nash bargaining scheme is followed to balance the trade off between the misclassification error of the groups. The efficiency of the methodology is evaluated via three binary and multi-class literature datasets, which provide evidence for the fairness and accuracy of the predictions made by the model.

Keywords: Machine learning · Fairness · Mathematical programming · Data classification · Game theory.

1 Introduction

Society has been increasingly utilising machine learning to make data-driven decisions. This implies that high-stakes decisions, with a significant impact on people, are made by machine learning algorithms. The continuous research and enhancement of these algorithms has improved the accuracy of the decisions. However, concerns about the transparency and fairness of these algorithms and numerous ethical issues have emerged. More specifically, some controversial decisions made by machine learning models have been reported when assessing the risk of potential recidivism [4], [22], candidates for a hiring process [11] and credit reliability [10].

Classification constitutes one of the main tasks that fairness is a crucial feature. Meanwhile, classification trees are inherently considered an interpretable methodology because their decisions can be translated into IF-THEN rules. Motivated by the need to construct interpretable and fair classifiers, a game theoretic approach for classification tree training is proposed that accounts for group
fairness. The group fairness is examined using the Nash bargaining scheme to explore the fair trade-off between the misclassification error of the groups. Finally, the methodology is not limited to binary classification, but it is applicable to multi-class classification.

The rest of the paper is structured as follows: Section 2 describes some related works from literature. In Section 3, the proposed approach is presented. In Section 4, the performance of our proposed method is evaluated using a number of benchmark fairness classification datasets. Finally, conclusions are drawn in Section 5.

2 Related work

Decision trees are considered a very powerful machine learning method used for both regression [17] and classification tasks [5], which is considered an interpretable approach because of its simplicity. Commonly used heuristics, such as those found in CART [7] and C4.5 [27], tend to produce sub-optimal trees. Meanwhile, the remarkable improvement of algorithms for mixed integer optimisation and computer hardware has rendered mathematical programming based approaches feasible for defining various machine learning methods [23]. Thus, Bertsimas and Dunn [5] proposed Optimal Classification Trees (OCT), which is a Mixed Integer Linear Programming (MILP) formulation of the classification tree training. The model determines the split point for each branch node, assigns classes to individual leaf nodes, and dictates the path for each sample from the root node to a leaf node. The formulation aims to minimise the misclassification error and the tree complexity. Two versions of the model are presented, namely a univariate approach and a multivariate approach. Of course, the univariate approach is more interpretable, while the multivariate one can capture more complex patterns in the dataset. Lately, numerous research papers have proposed models that utilise mathematical programming to train an optimal classification tree [29], [2], [6].

Another important feature sought in decision trees is fairness in order to avoid unequal treatment of individuals of a certain group based solely on their affiliation to that particular group [20]. Motivated by the need for interpretable and fair classifiers, Aghaei et al. [1] presented an optimal and fair decision tree, which is a mixed integer optimisation approach that prevents disparate treatment and disparate impact. Then, FairOCT was proposed by Jo et al. [19], which is an optimal classification tree augmented with arbitrary fairness constraints accounting for statistical parity, conditional statistical parity and equalised odds. It is important to note that only a limited number of publications that examine fairness in classification address the problem of multi-class classification [12],[3], [28]. This happens because many challenging datasets in fairness literature have binary outcomes [11]. However, there are cases that the predicted variable is not binary, like predicting levels of drug use [14].
3 Methodology

3.1 Problem statement

This section aims to introduce an optimisation based classification tree that accounts for group fairness using Nash bargaining approach, namely GIFT. As mentioned earlier, the originally proposed MILP approach for the classification tree problem was presented by Bertsimas and Dunn [5]. In this work, a classification tree approach is developed, in which the training of the classification tree is formulated as a mixed integer optimisation problem and a game theoretic approach is considered to improve the group fairness of the solution. Group fairness notions aim to correct a prediction bias against distinct groups. Some of the most widely used fairness metrics are statistical parity, predictive equality, equal opportunity, equalised odds, overall misclassification rate [30], [8], [25], [9]. In this work, overall misclassification rate is used as the notion of fairness. Of course, mathematical programming gives the flexibility to the modeler to extend the proposed model and explore different fairness metrics. Overall, the problem studied can be stated as follows:

*Given:*

- Numerical values of $S$ training samples with $M$ attributes
- Classification of training samples into one of $C$ classes
- Maximum allowed depth of the tree

*Determine:*

- The split applied at every branch node
- The assignment of classes to leaf nodes
- The path that every sample follows to a leaf node

*So as to:*

- Maximise the misclassification error difference with the status quo of the groups

3.2 Mathematical formulation

The indices, sets, parameters and variables related to the model are outlined below:

**Indices**

| $s$ | Sample ($s = s_1, s_2, ..., S$) |
| $m$ | Attribute ($m = m_1, m_2, ..., M$) |
| $c$ | Class ($c = c_1, c_2, ..., C$) |
| $t, t'$ | Node ($t = t_1, t_2, ..., T$) |
| $r$ | Discretisation point ($r = r_1, r_2, ..., R$) |
| $g$ | Group ($g = Protected, Non-protected$) |
Sets
\( T_B \) Branch nodes
\( T_L \) Leaf nodes
\( P_t \) Parent node of node \( t \)
\( L_t \) Set of ancestors of \( t \) whose left child has been encountered along the path from the root node to \( t \)
\( R_t \) Set of ancestors of \( t \) whose right child has been encountered along the path from the root node to \( t \)
\( S_g \) Set of samples that belong to group \( g \)

Parameters
\( A_{sm} \) Value of sample \( s \) on attribute \( m \)
\( N_{\text{min}} \) Minimum number of samples at each active leaf node
\( \epsilon_m \) Smallest non-zero difference between two adjacent values on attribute \( m \)
\( U \) Suitably big number
\( Z^*_g \) Upper bound for the misclassification error of group \( g \)
\( Z_{gr} \) The misclassification error of group \( g \) is discretised into \( r \) number of points
\( \mu_{gr} \) Logarithm of the misclassification error difference of group \( g \) for point \( r \)

Continuous variables
\( b_t \) Split point of node \( t \)
\( \tau \) Non-linear objective function for Nash bargaining approach
\( \bar{\tau} \) Linear objective function for Nash bargaining approach

Binary variables
\( d_t \) 1, if a split is applied at node \( t \)
\( W_{mt} \) 1, if a split is applied at node \( t \) using attribute \( m \)
\( E_{st} \) 1, if sample \( s \) is correctly classified to leaf node \( t \)
\( Y_t \) 1, if leaf node \( t \) is active
\( O_{ct} \) 1, if the prediction of leaf node \( t \) is the class \( c \)

SOS variables
\( X_{gr} \) Variable used to describe which discrete points \( r \) will be used for the linear approximation of the misclassification error of group \( g \)

The first constraints are used to model the splits applied at branch nodes. Binary variables \( W_{mt} \) are introduced to model if attribute \( m \) is used to apply a split at branch node \( t \). Constraints (1) allow up to one attribute to be selected to apply split at every branch node. In this way, univariate splits are applied, which makes the decisions of the classification tree more interpretable. If no attribute is selected for a split at a branch node, then the node becomes inactive and constraints (1), (2) enforce binary variables \( d_t \) and \( b_t \) to zero. Constraints (3) prohibit a branch node from applying a split if its parent node does not apply a split.
Optimisation-based classification tree: A game theoretic approach to group fairness

\[
\sum_m W_{mt} = d_t \quad \forall t \in TB \tag{1}
\]
\[
0 \leq b_t \leq d_t \quad \forall t \in TB \tag{2}
\]
\[
d_t \leq d_{t'} \quad \forall t \in TB, t' \in P_t \tag{3}
\]

Next, constraints (4), (5) are used to enforce the splits applied by the branch nodes and make the samples follow the corresponding path to the leaf nodes. More specifically, in constraints (4), the path from branch node \( t' \) to leaf node \( t \) is modelled, where \( t' \) is an ancestor of \( t \) and its right child has been encountered along the path from the root node to \( t \). Thus, if sample value \( A_{sm} \) on the attribute \( m \), for which \( W_{mt'} = 1 \), is greater than \( b_{t'} \), then sample \( s \) follows the path of the right child of \( t' \) and \( E_{st} \) can become equal to 1. For example, in Figure 1, it is depicted that a sample \( s \) can end up in node \( t_6 \) only if it follows the right branch of node \( t_1 \), so it must respect the corresponding split: \( \sum_m W_{mt} \cdot A_{sm} \geq b_{t_1} \).

Accordingly, in constraints (5), the path from branch node \( t' \) to leaf node \( t \) is modelled, where \( t' \) is an ancestor of \( t \) and its left child has been encountered along the path from the root node to \( t \). The only difference is that a small constant \( \epsilon_m \) is added to the left-hand-side, which is the smallest non-zero difference between two adjacent values on the attribute \( m \). Meanwhile, if \( E_{st} = 0 \), the aforementioned constraints become redundant, since \( U \) is sufficiently large to satisfy the constraints.

\[
\sum_m W_{mt'} \cdot A_{sm} \geq b_{t'} - U \cdot (1 - E_{st}) \quad \forall s \in S, t \in TL, t' \in R_t \tag{4}
\]
\[
\sum_m (W_{mt'} \cdot A_{sm} + \epsilon_m) \leq b_{t'} + U \cdot (1 - E_{st}) \quad \forall s \in S, t \in TL, t' \in L_t \tag{5}
\]

Fig. 1. Tree visualisation
Binary variables \( O_{ct} \) are introduced to model the predicted class of each node. Constraints (6) formulate that the class of sample \( s \) has to be the predicted class of node \( t \) in order that sample to be correctly classified. Then, constraints (7) state that if \( Y_t = 1 \), then the leaf node makes a single class prediction. On the other hand, if \( Y_t = 0 \), then the leaf node is inactive. Furthermore, constraints (8) state that if \( Y_t = 0 \), then a leaf node \( t \) cannot contain correctly classified samples. While, constraints (9) set a minimum number of samples \( N_{\text{min}} \) to be correctly classified in a leaf node. Of course, every sample can be correctly classified only once, which is ensured by constraints (10).

\[
E_{st} \leq O_{ct} \quad \forall s \in S, c \in C_s, t \in T_L \tag{6}
\]

\[
\sum_c O_{ct} = Y_t \quad \forall t \in T_L \tag{7}
\]

\[
E_{st} \leq Y_t \quad \forall s \in S, t \in T_L \tag{8}
\]

\[
\sum_{s \in S} E_{st} \geq N_{\text{min}} \cdot Y_t \quad \forall t \in T_L \tag{9}
\]

\[
\sum_{t \in T_L} E_{st} \leq 1 \quad \forall s \in S \tag{10}
\]

Objective function (11) aims to minimise the total number of misclassified samples. This objective function ignores the fairness between the different groups and only considers the total misclassification error. As a result, in certain datasets, some groups are prioritised over others leading to unfair solutions. This objective function is part of the naive version of the proposed approach.

\[
\min 100 \cdot \left( \frac{\sum_{s \in S} (1 - \sum_{t \in T_L} E_{st})}{|S|} \right) \tag{11}
\]

In the light of the increasing significance of group fairness, the objective is no longer to only minimise the total number of misclassified samples, but also to strike a balance between the misclassification errors of the different groups \( (Z_g) \). A cooperative game is considered to achieve the best misclassification error for the groups through the Nash bargaining approach. The objective is to minimise each group misclassification error by maximising the difference between the status quo point and the optimisation variable. Hence, the Nash equilibrium is computed as the maximum value of the Nash product which is given by equation (12)

\[
\tau = \prod_g \left( Z_g^* - Z_g \right) \tag{12}
\]

The status quo point \( (Z_g^*) \) corresponds to the upper bound for the misclassification error of group \( g \) and it is set equal to the misclassification error of the group at the previous depth. While for depth \( D=1 \), the status quo point is considered to be the misclassification of all samples. Equation (12) results in a
non-convex and nonlinear formulation that can make the problem computationally intractable. However, various reformulation strategies have been proposed to approximate the Nash product \cite{16}, \cite{21}, \cite{24}. More specifically, a logarithmic transformation of equation (12) leads to a concave formulation.

\[
\ln \tau = \sum_g \ln(Z_g^* - Z_g)
\]  

(13)

Of course, equation (13) is still non-linear, which results to a MINLP. So, separable programming is used to discretise \( \ln \tau \) over \( r \) points. Parameter \( Z_{gr} \) is introduced to discretise \( Z_g \) variables into \( r \) points. While, a SOS2 variable \( X_{gr} \) acts as a weight factor that describes which two discrete points have been used for the approximation of the misclassification error of every group, as shown in equation (14). Equation (15) ensures that the active option \( r \) should be equal to 1.

\[
\sum_r Z_{gr} \cdot X_{gr} = 100 \cdot \left( \frac{\sum_{s \in S_g} (1 - \sum_{t \in T_L} E_{st})}{|S_g|} \right) \forall g
\]  

(14)

\[
\sum_r X_{gr} = 1 \quad \forall g
\]  

(15)

In equations (16), parameter \( \mu_{gr} \) discretises the logarithm of the misclassification error difference of group \( g \) into \( r \) points. Equations (16) constitute numerical relations and they are not part of the optimisation model.

\[
\mu_{gr} = \ln(Z_g^* - Z_{gr}) \quad \forall g,r
\]  

(16)

Finally, the objective function becomes the following:

\[
\text{max } \bar{\tau} = \sum_g \sum_r (\mu_{gr} \cdot X_{gr})
\]  

(17)

The developed MILP includes constraints (1) - (10),(14),(15) and (17) as objective function. It is named \textit{GIFT} (\textit{G}ame-theoretic using \textit{I}nteger optimisation \textit{F}air \textit{T}ree) and its goal is to maximise the misclassification error difference with the status quo of the groups.

4 Computational results

This section showcases the effectiveness of the proposed methodology through its application to benchmark datasets, shown in Table 1. All datasets are widely used in the fairness literature. The Diabetes dataset \cite{13} contains certain diagnostic measurements to predict whether or not a patient has diabetes. The protected group is the set of individuals with diabetes, which corresponds to 34.9 \% of the total patients. Then, the German dataset \cite{13} contains individuals who take credit by a bank. Each individual is classified as good or bad
credit risks according to the set of attributes, which include some personal information. The protected group is the set of individuals whose age is less than 25, which corresponds to 19% of the total people. Lastly, Drug dataset \[14\] includes demographic information such as age, gender, and education level, as well as measures of personality traits to predict the level of cannabis use. Unlike the previous datasets, Drug is a multi-class classification dataset. The protected group is the set of individuals, who do not have a college degree, which corresponds to 13.6% of the total people.

Table 1. Datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Samples</th>
<th>Attributes</th>
<th>Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diabetes</td>
<td>768</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>German</td>
<td>1000</td>
<td>24</td>
<td>2</td>
</tr>
<tr>
<td>Drug</td>
<td>1885</td>
<td>12</td>
<td>3</td>
</tr>
</tbody>
</table>

The implementation of \textit{GIFT} was conducted in GAMS (General Algebraic Modeling System) \[15\] and the selected solver was GUROBI \[18\]. The described model is iteratively solved starting from the first depth up to the maximum predetermined depth and the misclassification error of every group at the previous depth examined is used as status quo. The improvement of the training score is used as a terminating criterion to avoid overfitting. Thus, if the training score is not improved by 2% when compared to the training score of the previous depth examined, then the addition of a new layer is not justified. The time limit is set equal to 800 sec for every run and the solution of the previous depth is injected as warm start before starting the solver in order to boost its performance. It is worth noting that $N_{\text{min}}$ is set equal to 1% of the number of samples. The same implementation details were used for the naive version of the proposed model, which will be called \textit{Naive}. Lastly, \textit{CART} was implemented using Scikit-learn library \[26\]. The default parameters were used, while the maximum depth of the tree was controlled. All three classification trees were examined for depths $D=1,2,3,4$. The dataset is partitioned into training and testing subsets, with 70% of the entire dataset utilised for training and the remaining 30% for the testing phase. The allocation of samples to training or testing subset is conducted randomly and is repeated 15 times for every dataset. The mean prediction accuracy across these 15 iterations is then reported. It is important to note that all examined datasets undergo feature scaling within the range of [0,1]. Additionally, datasets containing categorical features are converted using one-hot encoding.

Table 2 presents a comparison of average prediction accuracy per dataset of \textit{CART, GIFT} and \textit{Naive} for depth $D=1,2,3,4$. The best performing method at every depth at every dataset is marked in bold. It is shown that \textit{Naive} outperforms \textit{GIFT} and \textit{CART} in 9 out of 12 cases, while in the rest of them, \textit{GIFT} exceeds the other methodologies. As expected, \textit{Naive} approach has better predictive accuracy than \textit{GIFT}, because its objective function is the minimisation of total misclassification error, without considering the unfairness that might arise by the solution.
Table 3 shows a comparison of average prediction accuracy of the protected group per dataset of CART and GIFT and Naive for depth $D = 1, 2, 3, 4$. Again, the best performing method at every depth at every dataset is marked in bold. It is demonstrated that GIFT outperforms Naive and CART in all datasets and for all depths examined. This showcases how GIFT constantly achieves superior predictive performance for the protected group.

Table 2. Average prediction accuracy (%) of CART, GIFT and Naive for depth $D=1,2,3,4$ per dataset.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CART-1</th>
<th>GIFT-1</th>
<th>Naive-1</th>
<th>CART-2</th>
<th>GIFT-2</th>
<th>Naive-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diabetes</td>
<td>72.38</td>
<td>72.12</td>
<td>73.30</td>
<td>73.74</td>
<td>73.10</td>
<td>74.66</td>
</tr>
<tr>
<td>German</td>
<td>69.62</td>
<td>69.78</td>
<td>70.13</td>
<td>69.64</td>
<td>70.78</td>
<td>70.62</td>
</tr>
<tr>
<td>Drug</td>
<td>53.49</td>
<td>53.36</td>
<td>54.06</td>
<td>53.70</td>
<td>54.28</td>
<td>54.15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CART-3</th>
<th>GIFT-3</th>
<th>Naive-3</th>
<th>CART-4</th>
<th>GIFT-4</th>
<th>Naive-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diabetes</td>
<td>72.78</td>
<td>73.16</td>
<td>74.55</td>
<td>73.28</td>
<td>73.33</td>
<td>74.55</td>
</tr>
<tr>
<td>German</td>
<td>70.73</td>
<td>70.87</td>
<td>71.13</td>
<td>70.47</td>
<td>71.18</td>
<td>71.23</td>
</tr>
<tr>
<td>Drug</td>
<td>54.06</td>
<td>54.72</td>
<td>54.05</td>
<td>54.15</td>
<td>54.77</td>
<td>54.80</td>
</tr>
</tbody>
</table>

Table 3. Average prediction accuracy of the protected group (%) of CART, GIFT and Naive for depth $D=1,2,3,4$ per dataset.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CART-1</th>
<th>GIFT-1</th>
<th>Naive-1</th>
<th>CART-2</th>
<th>GIFT-2</th>
<th>Naive-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diabetes</td>
<td>54.13</td>
<td>65.68</td>
<td>44.44</td>
<td>47.08</td>
<td>68.32</td>
<td>50.84</td>
</tr>
<tr>
<td>German</td>
<td>57.74</td>
<td>61.74</td>
<td>59.95</td>
<td>59.44</td>
<td>63.11</td>
<td>61.36</td>
</tr>
<tr>
<td>Drug</td>
<td>46.06</td>
<td>47.71</td>
<td>46.09</td>
<td>46.77</td>
<td>49.31</td>
<td>45.96</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CART-3</th>
<th>GIFT-3</th>
<th>Naive-3</th>
<th>CART-4</th>
<th>GIFT-4</th>
<th>Naive-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diabetes</td>
<td>48.82</td>
<td>68.15</td>
<td>50.76</td>
<td>48.73</td>
<td>68.24</td>
<td>50.76</td>
</tr>
<tr>
<td>German</td>
<td>62.01</td>
<td>62.31</td>
<td>62.22</td>
<td>62.55</td>
<td>63.24</td>
<td>62.22</td>
</tr>
<tr>
<td>Drug</td>
<td>47.14</td>
<td>48.83</td>
<td>46.46</td>
<td>47.58</td>
<td>48.65</td>
<td>47.15</td>
</tr>
</tbody>
</table>

Figure 2 depicts the average prediction accuracy across all datasets of CART, GIFT and Naive for depth $D=1,2,3,4$. It is clear that Naive surpasses the other methodologies for all depths, with GIFT being second and CART having the worst predictive performance. More specifically, it is observed that Naive has a higher accuracy of around 0.3–0.7% than GIFT and 0.6–0.9% than CART at all levels of complexity. Comparing GIFT and CART, it is observed that GIFT achieves higher prediction accuracy on average for 3 out of 4 depths examined, while for depth $D=1$, CART performs slightly better.

Figure 3 illustrates the average prediction accuracy of the protected group across all datasets of CART, GIFT and Naive for depth $D=1,2,3,4$. GIFT is shown to exhibit by far the highest average prediction accuracy of the protected group. More specifically, GIFT outperforms CART for all depths by 5.6–9.2% and Naive by 6.6–8.1% in terms of average prediction accuracy of the protected group. Considering both Figure 2 and Figure 3, it is indicated that
GIFT achieves to improve the prediction accuracy of the protected group without sacrificing the total prediction accuracy, when compared to CART. Comparing to Naive, the total prediction accuracy has slightly reduced, while GIFT has significantly better prediction accuracy of the protected group.

![Fig. 2. Average prediction accuracy (%) across all datasets of CART, GIFT and Naive for depth D=1,2,3,4.](image)

![Fig. 3. Average prediction accuracy of the protected group (%) across all datasets of CART, GIFT and Naive for depth D=1,2,3,4.](image)

5 Concluding Remarks

This work addresses the problem of training a mathematical programming based classification tree that accounts for group fairness. A game theoretic approach (GIFT) has been proposed to improve the group fairness using the misclassification error as fairness metric. More specifically, Nash bargaining scheme is followed to balance the trade off between the misclassification error of the groups.
This results into an MINLP model, which is approximated to an MILP model using SOS2 variables. The effectiveness of the proposed methodology has been demonstrated through three binary and multi-class literature datasets. Based on the computational results, our approach appears to outperform CART in prediction accuracy in most depths examined and in prediction accuracy of the protected group in all depths examined. Comparing GIFT with its naive version, we trade off some of the accuracy of the classification tree to enhance its fairness towards the protected group. Plans for future work involve the consideration of different fairness metrics and the improvement of the scalability of the approach in order to tackle bigger datasets. More specifically, clustering techniques can be applied to the data to reduce the dimensionality of the problem.

Acknowledgements Authors gratefully acknowledge the financial support from Engineering and Physical Sciences Research Council (EPSRC) under the project EP/V051008/1.

References

GAMS Development Corporation: General Algebraic Model System (GAMS) (2022), release 41.5.0, Washington, DC, USA.


Verwer, S., Zhang, Y.: Learning optimal classification trees using a binary linear program formulation. 33rd Conference on Artificial Intelligence (2019), https://doi.org/10.1609/aaai.v33i01.33011624

The Capacitated Vehicle Routing Problem with Pickup and Delivery using Suppliers for single and multiple commodities

D. Blanchard and H. Kheddouci
Université de Lyon - LIRIS UMR 5205
Bâtiment Nautibus 43, bd du 11 Novembre 1918, 69622 Villeurbanne, France
damien.blanchard@univ-lyon1.fr, hamamache.kheddouci@univ-lyon1.fr

Abstract. In this paper, we deal with a new kind of VRP based on a depot that holds no commodities and instead the commodities can be picked up at suppliers to then be delivered to the customers. It will be named Capacitated Vehicle Routing Problem with Pickup and Delivery using Suppliers (CVRPPD-S). We will modelise it, but focus our efforts on a simpler version of the problem, with one vehicle and one kind of commodity: the 1-commodity Pickup and Delivery Traveling Salesman Problem using Suppliers (1-PDTSP-S). We will explore an algorithm to give us an exact solution, then heuristics (in particular, a tabu search) to obtain a fast effective solution.

Keywords: Operations research · Vehicle routing · Pickup and delivery

1 Introduction

The Vehicle Routing Problem (VRP) is a notorious problem in Operations Research which has led to numerous papers on a very high number of variants. But in most of the works, the depot acts as a place where both vehicles and commodities are stored. However, we are interested in a setup where the depot only acts as a vehicle depot and holds no commodities. Instead, the commodities are picked up at suppliers to be then delivered to customers, and the depot’s only features are being the beginning and the end of a route for a vehicle.

The CVRPPD-S we are studying will have the following features: a depot with no commodities inside, \( n - p - 1 \) customers, \( p \) suppliers, \( m \) commodities, a set \( V \) of heterogeneous vehicles, the customers have a linear combination of commodities they need; and for each commodity, a supplier has an unlimited supply of it or nothing. The deliveries can be split, each vehicle has a maximum load, and a fixed cost per unit of distance travelled. The deliveries can be split amongst one or several vehicles, a vehicle can go several times through the same arc. No time windows constraints are assumed for the customers, but a vehicle has a maximum amount of time it can cover, time being defined by distance multiplied by a constant, plus a fixed loading time for each time a vehicle goes to a supplier and a fixed unloading time for each time a vehicle goes to a customer. The objective is to maximize gain minus costs for transportation.
The 1-PDTSP-S we are studying has a depot with no commodities, $n - p - 1$ customers, $p$ suppliers, one commodity, one vehicle. The deliveries will not be split, it is assumed that all customers have demands that are inferior in quantity to what the vehicle can handle. The objective is the same as the VRPPD-S.

Our goal in this paper is first (section 3) to suggest a modeling of the Vehicle Routing Problem with Pickup and Delivery using Suppliers (VRPPD-S), then (section 4) to analyze a simpler version of the problem with single-commodity, no split deliveries and a single vehicle, bringing us closer to the Traveling Salesman Problem: the One-commodity Pickup and Delivery Traveling Salesman Problem using Suppliers (1-PDTSP-S). We will (subsection 4.1) show an algorithm to obtain an exact solution and also (subsection 4.2) explore a naive heuristic to give us a quick good solution, which is then improved with a Tabu search to obtain an exact or near-optimal solution in near-instantaneous time (subsection 4.3).

2 Related works

Although we have found no articles dealing with this specific instance, it comes close to some variants of the Vehicle Routing Problem and the Traveling Salesman Problem (TSP) : the Vehicle Routing Problem with Pickup and Delivery (VRPPD), also called dial-a-ride problem (DARP), see Guy Desaulniers et al. [1] is vaguely similar since it contains pickup and delivery, but the pickup point and delivery point are predetermined, while in our problem choosing the good supplier is not done beforehand and is a part of solving the problem itself.

The Capacitated Vehicule Routing Problem (CVRP) includes the capacity constraint on vehicles: a vehicle can’t have more than a certain amount of commodities it can carry, a maximum weight so to speak.

The Split Delivery Vehicle Routing Problem (SDVRP) authorizes split deliveries [13], which leads to more optimized routes and is sometimes outright necessary if a customer requires more commodities than what a single vehicle could store.

The Multi-Depot Vehicle Routing Problem (MDVRP) also comes in mind when you consider a variant where you can use a depot more than once in your route, since it would be equivalent to what we call a supplier here. Papers explored describing MDVRP include [7], [8], [9], [10], [11] and [12].

In the context of a fleet having a single vehicle, which is quite common so it is of interest to us, the 1-PDTSP (see Hernández-Pérez et al. [2]) and m-PDTSP (see Hernández-Pérez et al. [3] and [4]) are very similar to our problem: customers with pickup components are re-used to get enough commodities to serve other customers. An extension to the VRP called 1-VRPPD (see Martinovic et al. [5]) solves the problem with multiple vehicles. In these works, the customers with pickup components are pre-defined, have fixed pickup amounts and have to be passed through, we however have suppliers that the vehicles do not necessarily have to go through. Indeed, in our problem, supplier stocks are assumed infinite in a subset of commodities: for instance, supplier A has an unlimited
supply of Commodity 1 and Commodity 3, but nothing else and supplier B has an unlimited supply of Commodity 3 and Commodity 4, and nothing else. We could even have a situation where we have a supplier C which is much better positioned than supplier A and has the same commodities at disposal, leading to supplier A never being used.

Figure 1 below summarizes the literature which is relevant to our problem. There are many other variations of TSP and VRP that are not included in the figure because they are not relevant here.

![Figure 1. Classifying the CVRPPD-S within the existing literature](image)

3 Modelizing the CVRPPD-S

Our first idea for the CVRPPD-S is the following: The depot, suppliers and graphs are generating a graph. The associated edges represent the distance between each vertex, the distance matrix $D_{ij}$ is euclidean.

We will consider there are $n$ vertices, $p$ suppliers and 1 depot, so $n - p - 1$ customers.

Vertices are denoted 1 to $n$, vertex 1 being the depot.

For each vertex, to know whether the vertex is the depot, a supplier or a customer we created the functions is_depot, is_supp and is_cust which are boolean (1 if it is, 0 if it isn’t).

The commodities will be noted 1, 2, ..., $m$. 

277
Vehicles belong to the set $V$ and a vehicle $v$ has a kilometric cost of $c(v)$ and a max charge of $W_{\text{max}}(v)$.

A customer has an order, for each commodity $k$ we will note $O(i, k)$ the needs for commodity $k$ for the customer present at vertex $i$.

The supplier disponibility matrix $SD_{ik}$, composed exclusively of bools, determines whether vertex $i$ has the commodity $k$ at disposal. If $is\_supp(i) = 0$, then $\forall j \in 1, ..., n; SD_{ik} = 0$

To know whether a customer has received his delivery, we will introduce $Served(i)$, another boolean function valued 1 if the customer $i$ has received the delivery and 0 if he didn’t. We will note $Gain(i)$ the amount of potential gain we obtain if customer $i$ were to receive his delivery.

In order to track where the vehicles are going and in which order, we will introduce the notion of slots and paths ($P$). Each vertex has slots associated to it. The maximum number of slots is chosen, having a low number will lead to faster results and deliveries that are "less split", however it could lead to an unsolvable problem, or a less optimal solution, having a high number of slots will noticeably slow down the algorithm. Slots will be named $1, 2, 3, ..., S_{\text{max}}$.

In order to track how many commodities are currently on a vehicle, we will name $Q(k, i, x, v)$ the amount of commodity $k$ remaining on the vehicle $v$ when it just passed vertex $i$ at slot $x$

We will assume loading times (resp. unloading times) have a fixed time for each time the vehicle goes to a point $LT_{fix}$ (resp. $UT_{fix}$), and a time per unit of commodity to load $LT_{unit}$ (resp. unload $UT_{unit}$). These times will be useful, as a vehicle can not be used more than a certain amount of time in the day (a driver’s day is 8 hours in our problem).

The slots are used to describe where a vehicle is and allows for a vehicle to go to the same vertex several times. A path links slots and vertices together.

$P(i, x, j, y, v) = 1$ if the vehicle $v$ goes to the vertex $i$, slot $x$ to the vertex $j$, slot $y$ (and 0 if it doesn’t). A slot can never be used twice by the same vehicle, but it can be used by different vehicles.

We also introduce $N(i, x, v)$ where $i$ is a vertex, $x$ a slot and $v$ a vehicle. $N$ determines the order of vertices visited. If for instance vertex 3 slot 4 is the third vertex not including the depot visited by vehicle $v, N(3, 4, v) = 3$

$N$ allows us to make sure any subtours are eliminated, without the need to write other subtour elimination constraints.

Here is the formulation of the problem:

$$\text{Max. } \sum_{i=1}^{n} Gain(i) * Served(i) - \sum_{i=1}^{n} \sum_{x=1}^{S_{\text{max}}} \sum_{j=1}^{n} \sum_{y=1}^{S_{\text{max}}} P(i, x, j, y, v) * D_{ij} * c(v)$$

s.t.

$$N(1, 1, v) = 0 \quad \forall v \in V \quad (1)$$
The CVRPPD-S for single and multiple commodities

\[ N(1, 2, v) = \sum_{i=1}^{n} \sum_{x=1}^{S_{max}} \sum_{j=1}^{n} \sum_{y=1}^{S_{max}} P(i, x, j, y, v) \quad \forall v \in V \]  

(2)

\[ \sum_{j=1}^{n} S_{max} \sum_{y=1}^{S_{max}} P(1, 1, j, y, z) = 1 \quad \forall v \in V \]  

(3)

\[ \sum_{j=1}^{n} S_{max} \sum_{x=2}^{n} \sum_{y=1}^{S_{max}} P(1, x, j, y, z) = 0 \quad \forall v \in V \]  

(4)

\[ \sum_{i=1}^{n} S_{max} \sum_{i=1}^{x=1} P(i, x, 1, 2, z) = 1 \quad \forall v \in V \]  

(5)

\[ \sum_{i=1}^{n} S_{max} \sum_{y=1}^{S_{max}} \sum_{z=1, y \neq 2}^{S_{max}} P(i, x, 1, y, z) = 0 \quad \forall v \in V \]  

(6)

\[ N(j, y, v) = \sum_{i=1}^{n} \sum_{x=1}^{S_{max}} P(i, x, j, y, v) \ast (N(i, x, v) + 1) \]  

(7)

\[ \forall v \in V, \forall j \in \{1, ..., n\}, \forall y \in \{1, ..., S_{max}\} \]

\[ \sum_{i=1}^{n} S_{max} \sum_{j=1}^{x=1} \sum_{y=1}^{S_{max}} P(i, x, j, y, v) \ast D_{ij} \]

\[ + \sum_{i=1}^{n} S_{max} \sum_{j=1}^{x=1} \sum_{k=1}^{m} ((Q(k, j, y, v) - Q(k, i, x, v)) \ast LT_{unit} + LT_{fix}) \ast is_supp(j) \]

(8)

\[ + \sum_{i=1}^{n} S_{max} \sum_{j=1}^{x=1} \sum_{k=1}^{m} ((Q(k, j, y, v) - Q(k, i, x, v)) \ast UT_{unit} + UT_{fix}) \ast Served(j) \]

\[ \leq 480 \quad \forall v \in V \]

\[ Served(j) \ast O(j, k) \ast is_cust(j) \leq \sum_{i=1}^{n} \sum_{x=1}^{S_{max}} \sum_{y=1}^{S_{max}} \sum_{v \in V} (Q(k, i, x, v) - Q(k, j, y, v)) \]

\[ \ast P(i, x, j, y, v) \ast is_cust(j) \]

(9)

\[ Served(i) \leq is_cust(i) \quad \forall i \in \{1, ..., n\} \]  

(10)
Blanchard and Kheddouci

\[ Q(k, 1, x, v) = 0 \quad \forall k \in \{1, \ldots, m\}, \forall x \in \{1, \ldots, S_{\text{max}}\}, \forall v \in V \]  \hfill (11)

\[ \sum_{k=1}^{m} Q(k, i, x, v) \leq W_{\text{max}}(v) \quad \forall i \in \{1, \ldots, n\}, \forall x \in \{1, \ldots, S_{\text{max}}\}, \forall v \in V \]  \hfill (12)

\[ \sum_{j=1}^{n} \sum_{y=1}^{S_{\text{max}}} P(i, x, j, y, v) \leq 1 \quad \forall i \in \{1, \ldots, n\}, \forall x \in \{1, \ldots, S_{\text{max}}\}, \forall v \in V \]  \hfill (13)

\[ \sum_{i=1}^{n} \sum_{j=1}^{S_{\text{max}}} P(i, x, j, y, v) \leq 1 \quad \forall j \in \{1, \ldots, n\}, \forall y \in \{1, \ldots, S_{\text{max}}\}, \forall v \in V \]  \hfill (14)

\[ P(i, x, j, y, v) \ast (Q(k, j, y, v) - Q(k, i, x, v)) \leq SD_{jk} \ast \sum_{h=1}^{n} O(h, k) \]  \hfill (15)

\[ \forall k \in \{1, \ldots, m\}, \forall (i, j) \in \{1, \ldots, n\}^2, \forall (x, y) \in \{1, \ldots, S_{\text{max}}\}^2, \forall v \in V \]

\[ (1 - \text{is\_depot}(i)) \ast P(i, x, j, y, v) \leq \sum_{h=1}^{m} \sum_{z=1}^{n} P(h, z, i, x, v) \]  \hfill (16)

\[ \forall v \in V, \forall (i, j) \in \{1, \ldots, n\}^2, \forall (x, y) \in \{1, \ldots, S_{\text{max}}\}^2 \]

\[ Gain(i), Served(i), is\_cust(i), is\_supp(i), is\_depot(i), P(i, x, j, y, v) \in \{0, 1\} \]

\[ Q(k, i, x, v) \in \mathbb{R}^+ \]

\[ \forall (i, j) \in \{1, \ldots, n\}^2, \forall (x, y) \in \{1, \ldots, S_{\text{max}}\}^2, \forall v \in V, \forall k \in \{1, \ldots, m\} \]  \hfill (17)

The objective function is to maximize the amount of gain generated by the customers: \( \sum_{i=1}^{n} \sum_{x=1}^{S_{\text{max}}} \sum_{j=1}^{n} \sum_{y=1}^{S_{\text{max}}} P(i, x, j, y, v) \ast D_{i,j} \ast c(v) \), which is basically the sum of "1 if the path from i slot x to j slot y is taken, 0 otherwise" * "distance from i to j" * "consumption of the vehicle per km". Generally speaking, Gain(i) is high enough that maximizing the objective function will also maximize \( \sum_{i=1}^{n} \text{Gain}(i) \ast \text{Served}(i) \). An alternative objective function would be to just minimize total consumption, it would require the additional constraint to serve every customer (\( \forall i \in \{1, \ldots, n\}; \text{Gained}(i) = 1 \)) and removing constraint (8). One could also include time as a cost and use the same sums used in constraint (8) directly in the objective function.

Constraints (3), (4) ensure a vehicle only leaves the depot once and (5), (6) ensure it returns the depot again only once.

Constraint (7) ensures paths are linked to each other and allow an easy overview of the route chosen by the vehicle: if its value is \( \lambda \) for \( N(i, x, v) \) it means the \( \lambda \)th step of the vehicle \( v \) is to go to vertex \( i \) (\( x \) does not matter here, and it is strictly impossible for \( N \) to have two or more strictly positive values that are
the same for the same $v$ but a different $i$ and $x$).
Constraints (1), (2) are there to ensure there are no isolated subtours in the
route chosen, with the help of constraint (7). Indeed, since $N$ increases by 1 for
every path taken, and $N(1, 2, v)$ is equal to the sum of all paths taken, when the
vehicle comes back to the depot it necessarily went through all paths.
Constraint (8) is a custom constraint: our vehicles can not be used more than
480 minutes (8 hours).
Constraint (9) defines when a customer is served: the amount of commodities
dropped at his vertex are superior or equal to the demand, for each kind of
commodity.
Constraint (10) ensures only customers are served: though it does not hold any
impact if you define correctly the gain of non-customers to be 0.
Constraint (11) ensures a vehicle does not start his trip with commodities and
therefore needs to fill up at suppliers.
Constraint (12) limits the weight a vehicle can hold at any given time, which is
the capacitated part of the problem.
Constraint (13) and (14) ensure that a vehicle currently at vertex $i$ slot $x$ only
has at most one destination and origin.
Constraint (15) allows to pickup commodities from supplier vertexes, exclusively
from those he has available.
Constraint (16) forbids a path from vertex $i$ to another vertex to exist if there
was no path leading to vertex $i$ in the first place. It excludes the depot which is
the starting point that does not need a previous path existing.
Constraint (17) simply defines what values our decision variables can take, which
are boolean, integer or real positive.

4 The 1-PDTSP-S, a simpler subproblem

Reducing to a simpler instance of the problem to obtain results especially on
problems that didn’t need to consider more than 1 vehicle and 1 commodity
rapidly became of the essence, so we decided to find an exact solution in a decent
amount of time and heuristics to get even faster results. To quickly reiterate, the
1-PDTSP-S is the same problem as above, except we only have 1 vehicle, 1
commodity and we no longer consider split deliveries. All suppliers have the
commodity as disposal (else they would simply be un-needed).

4.1 An exact solution to the 1-PDTSP-S

While analyzing the problem, we quickly realized that the route would be formed
by cycles containing 1 supplier then 1 or several customers.
Indeed, it is a waste of time to go to 2 different suppliers in a row since there
is only 1 commodity. That means, to optimize each cycle, we have to make sure
the supplier of the cycle is the optimal supplier considering where we come from
and the first customer, all other suppliers are simply worse. That will allow us
to limit our calculations.
The algorithm to find an optimal path is the following:

First we will define two recursive functions begin_cycle and complete_cycle. complete_cycle will take as parameters remaining_customers (an array), fitness (a double), route (an array), remaining_time (an array), load (a double) and initial_load (a double). begin_cycle will take the same parameters except for initial_load. We also define best_sols which gives the best found fitness yet for a certain combination of remaining_customers and last element of route.

begin_cycle chooses the optimal supplier for each element of remaining_customers, then calls complete_cycle with updated arguments (for each one). Then (still for each one) it checks if best_sols has a better solution with the same remaining_customers and last element of route. If it doesn’t, this becomes the new best_sols for these remaining_customers and this last element of route, and we also apply begin_cycle to it.

complete_cycle will, for each customer in remaining_customers, add it to the route if possible, update the other arguments and call itself with the new arguments. It then checks if best_sols has a better solution with the same remaining_customers and last element of route. If it doesn’t, this becomes the new best_sols for these remaining_customers and this last element of route, and we also apply begin_cycle to it.

We just call begin_cycle with arguments that correspond to our problem and the recursion will find the best solution.

This algorithm gives an exact solution in decent time (cf. section 5).

4.2 Greedy algorithm for a fast solution

In light of the increasing calculation times with the number of customers, obtaining a fast approximate solution is important. A greedy algorithm considering the customer giving the most gain from current position is therefore developed:

Initialisation: current_vertex = 1, load = max weight vehicle, remaining_customers = all customers, time_spent = 0

Step 1: Determine the supplier s and customer c for which the distance from current_vertex to s plus the distance from s to c is the lowest. If choosing that supplier s and customer c would lead to total time going over 480 if we add the time needed to travel back to the depot, stop the algorithm. Else add the distance from current_vertex to s plus the distance from s to c plus the loading time at s plus the unloading time at c to time_spent, add s then c to the route. Substract order quantity from load. Remove c from remaining customers.

current_vertex now is c

Step 2: While at least one customer has an order inferior to load, choose the closest customer c amongst those. If choosing that customer c would lead to total time going over 480 if we add the time needed to travel back to the depot, stop the algorithm. Else add the distance from current_vertex to c plus the unloading time at c to time_spent and add c to the route. Substract order quantity
from load. Remove c from remaining customers. 

current_vertex now is c

**Step 3**: Set load to max weight vehicle, go back to step 1.

While this algorithm gives a decent solution in instantaneous time, it is naive and we can do better. The real application is to use it as a starting solution to be improved. We have chosen a tabu heuristic to start from this solution, better it and get a near-optimal or optimal solution in quick times.

### 4.3 Tabu search heuristic

Tabu search (TS) is a metaheuristic search method employing local search methods used for mathematical optimization. It was created by Fred W. Glover in 1986. The idea is the following: during a certain number of iterations, find neighbour solutions in hope of finding a better solution, and previously visited solutions are "tabu", ergo they can’t be used again, so that we don’t get stuck in a local optimum.

In order to find neighbour solutions, we can’t just swap two vertexes at random because it is likely to make the solution infeasible: a cycle supplier-customers might have too many customers and the vehicle wouldn’t be able to carry all the orders. So we used the following process:

We choose at random a way in which the route is going to be changed, it can be:
- Removal of a customer. If the cycle it belongs to no longer has customers, remove the supplier of the cycle as well.
- Customer change. If the new customer already exists in the route, it is a swap: if they belong to different cycles, check if the change would lead to cycles with the sum of orders too big for the weight of the vehicle and do not proceed if it is.
- Supplier/supplier swap
- Customer addition. Check if it can be added at the end of a cycle. If it can’t (the sum of the orders is too big for the weight of the vehicle), add it at the end of the route, right after a supplier that is optimal for it (distance from last vertex to supplier + distance from supplier to customer is minimized).

Then a classic tabu search is applied, except that we allow solutions that go over 8 hours. We use them temporarily to try and get customers we would not easily have access to, but add an increasing penalty to fitness at each iteration as long as the solution remained over 8 hours so that it naturally went back to a solution under 8 hours. This helps find new solutions and not become blocked as all neighbours are either tabu or unfeasible, which stops the algorithm prematurely.
5 Performance evaluation

5.1 Exact solution

For reference, the machine used for the simulations was tested to have around 165 GFLOPS. For the instances of problems we had fixed time at loading (30min) and unloading (45min), 8 suppliers, 8 hours max route, the script execution times to get a solution were the following:

<table>
<thead>
<tr>
<th>Number of customers</th>
<th>Average script execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>230ms</td>
</tr>
<tr>
<td>10</td>
<td>675ms</td>
</tr>
<tr>
<td>12</td>
<td>3.63s</td>
</tr>
<tr>
<td>14</td>
<td>11.1s</td>
</tr>
<tr>
<td>15</td>
<td>26.2s</td>
</tr>
<tr>
<td>20</td>
<td>2min13s</td>
</tr>
</tbody>
</table>

The fact that our search is restricted to routes that can’t go over 8 hours of vehicle travelling largely restricts the execution time, especially since we have quite high fixed loading and unloading times. Tests for increased max duration of travelling revealed that the execution time skyrockets. For instance, for 12 customers and 8 hours max travel time, we have 3.63s execution time, but choosing 9 hours max travel time will bring it up to 18s, 10 hours max travel will bring it up to 75s.

5.2 Tabu search heuristic

The parameters chosen for the tabu search were 2000 iterations and 50 neighbour solutions at each iteration. Since the number of iterations is fixed, the calculation times are almost fixed. However the number of iterations needed to find the best solution we have is a good indication of how much iterations we would need to do if we could change the number of iterations. Here are the results of our algorithm tests, for scenarios with fixed time at loading (30min) / unloading (45min), 8 suppliers, 8 hours max route, same gain for every customer:

In the table 2 below, C stands for total Customers, CS for Customers Served, ES stands for Exact Solution, G for Greedy Solution, TS for Tabu Search solution, A/M/M stands for Average/Min/Max iterations to find the best solution (10 tries), AV stands for the Average Variance from the best solution found.

For reference, 1000 iterations roughly equates to 1 second of calculations with our 165 GFLOPS machine, so you can view the max number of iterations as a required execution time in ms. The time columns correspond to the time travelled by the vehicle in the route, it is not the execution time of the algorithm.
The results are somewhat hard to exploit: the number of iterations necessary to get the best solution have a high variance. It depends on how far our initial solution is from the optimal solution, to a larger extent than the number of customers here. Example 10 has variance, meaning at some tries 2000 iterations weren’t enough to get to the optimal solution.

Tests (not in the table) have shown that for a longer max travel time, there will be a higher number of iterations needed to find the exact solution, yet not dramatically higher. For the example 8, increasing max travel time to 9h will lead to average/min/max iterations of 682/27/1956. Increasing max travel time to 10h will lead to average/min/max iterations of 2637/234/5824 (we tried with more than 2000 iterations to get the right results).

All in all, it still remains far more scalable than the exact solution if we are interested in instances where the routes allowed are longer (or the loading/unloading times are lower): we obtain the optimal solution in less than 6 seconds in dozens of tests (expl 8, 9h/10h max travel time) when the exact solution takes 2min37 for 9h max travel time, 15min25s for 10h max travel time.

If however we stay with relatively small instances (around 6-7 customers maximum can be served in one day due to distance and loading/unloading times), the exact solution is affordable.

### 6 Conclusion and perspectives

We introduced a new dimension to the VRPPD by imagining a version where suppliers are included and the depot is no longer the source of the commodities. We suggested a modelisation for the Capacitated VRPPD using suppliers (CVRPPD-S) and studied the subproblem tied to the single-commodity PDTSP (1-PDTSP-S) by providing an exact solution algorithm and exploring a heuristic with a tabu search.

There needs to be more work on comparing different meta-heuristic to our tabu search (e.g. simulated annealing, genetic algorithms) to see if we can manage to obtain better results. The main problem has been left aside due to complexity but could be attacked with machine learning.

---

### Table 2. General results

<table>
<thead>
<tr>
<th>Expl</th>
<th>CS</th>
<th>CS (ES)</th>
<th>Time (ES)</th>
<th>CS (G)</th>
<th>Time (G)</th>
<th>CS (TS)</th>
<th>Time (TS)</th>
<th>A/M/M (TS)</th>
<th>AV (TS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
<td>4h9m17s</td>
<td>3</td>
<td>4h9m17s</td>
<td>3</td>
<td>4h9m17s</td>
<td>0/0/0</td>
<td>0%</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>5</td>
<td>6h13m16s</td>
<td>5</td>
<td>6h09m34s</td>
<td>4</td>
<td>6h13m16s</td>
<td>10/9/10</td>
<td>0%</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>5</td>
<td>7h58m12s</td>
<td>4</td>
<td>6h23m36s</td>
<td>5</td>
<td>7h58m12s</td>
<td>109/18/217</td>
<td>0%</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>5</td>
<td>7h54m14s</td>
<td>5</td>
<td>7h24m20s</td>
<td>5</td>
<td>7h54m14s</td>
<td>267/35/904</td>
<td>0%</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>6</td>
<td>7h42m51s</td>
<td>5</td>
<td>6h32m8s</td>
<td>6</td>
<td>7h42m51s</td>
<td>329/3/1224</td>
<td>0%</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>6</td>
<td>7h42m36s</td>
<td>6</td>
<td>7h56m12s</td>
<td>6</td>
<td>7h42m36s</td>
<td>382/31/988</td>
<td>0%</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>6</td>
<td>7h33m55s</td>
<td>6</td>
<td>7h57m39s</td>
<td>6</td>
<td>7h33m55s</td>
<td>135/14/345</td>
<td>0%</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
<td>7</td>
<td>7h35m13s</td>
<td>6</td>
<td>7h52m56s</td>
<td>7</td>
<td>7h35m13s</td>
<td>243/78/571</td>
<td>0%</td>
</tr>
<tr>
<td>9</td>
<td>15</td>
<td>6</td>
<td>7h4m49s</td>
<td>6</td>
<td>7h55m28s</td>
<td>6</td>
<td>7h4m49s</td>
<td>208/59/575</td>
<td>0%</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>7</td>
<td>7h24m20s</td>
<td>7</td>
<td>7h24m25s</td>
<td>7</td>
<td>7h24m20s</td>
<td>&gt;480/5/&gt;2000</td>
<td>0.001% (0.3s)</td>
</tr>
</tbody>
</table>

The CVRPPD-S for single and multiple commodities 11
Extending the 1-PDTSP-S to multiple commodities (m-PDTSP-S) or multiple vehicles (1-VRPPD-S) also sounds like a decent approach to start tackling the more complex problem. The works of Hernández-Pérez et al. [3] [4] on the m-PDTSP will help us generalize it to multiple commodities.

An alternative modelisation of the CVRPPD-S was considered, inspired from the works of S. Mitra [6] by considering the number of times a vehicle goes through an edge instead, but the isolated subtour elimination constraints have been problematic to implement. Subtours are technically possible as a customer can have a big order that needs the vehicle to come several times, but they should be connected to a route that start with the depot, hence the isolated subtour elimination. If we can manage to modelise these constraints without the number of constraints exploding with the number of customers, we will have a much faster solver solution.

References

Advancing Road Safety Metrics: Exploring Index construction

Ibtissam El Khalai¹ and Zoubida Chorfi¹ and Abdelaziz Berrado¹

¹Equipe AMIPS, Université Mohammed V de Rabat, Ecole Mohammadia d’ingénieurs, Rabat, Morocco
ibtissam-khalai@research.emi.ac.ma, chorfi@emi.ac.ma, berrado@emi.ac.ma

Abstract. Composite indicators stand as a powerful instrument for consolidating information from diverse domains into a single index. Their primary purpose is to offer a comprehensive overview of various factors or domains related to the subject under consideration. In recent years, the road safety community has increasingly acknowledged the pivotal role of composite indicators in enhancing decision-making processes. This study explores the specific motivations driving the development of these metrics within the road safety research field and analyzes the procedures and methods employed for this purpose. Through a careful examination of a selection of relevant studies, valuable insights into the significance and use of road safety indexes have been gained. This exploration unveils the reasons behind their widespread use and provides a deeper understanding of the techniques and methodologies applied in their construction. Consequently, our research establishes a robust foundation for future endeavors in the construction and refinement of road safety composite indicators.

Keywords: Road Safety Composite Indicator, Aggregated Metric, Decision-Making Support.

1 Introduction

Composite indicators, commonly used tools for evaluating and comparing countries performance, are increasingly recognized as invaluable instruments in policy analysis and public communication. These indicators facilitate straightforward comparisons across diverse fields, including the environment, economy, society, and technological development[1]. They serve as effective means to clarify complex and nuanced issues. Insights derived from numerous research studies have explored the development of these metrics, shedding light on various critical stages. The steps outlined in the OECD Handbook represent a comprehensive and widely adopted procedure for constructing composite indicators in research endeavors. This procedure commences with formulating a theoretical framework defining the multi-dimensional phenomenon to be measured, selecting relevant variables, and addressing missing data [1]. Subsequently, the systematic process includes applying multivariate analysis to check the underlying
structure of the data along various dimensions, normalizing data for consistency, and using weighting and aggregation to form the composite indicator. The assessment of robustness and sensitivity is then conducted, accompanied by a detailed exploration of the methodology. The consideration of the composite indicator connections with other variables follows, culminating in the final stages of presenting and visualizing the indicator's findings. Importantly, the decisions regarding the selection of indicators, the normalization of indicator values, and the weighting and aggregation methods can significantly influence the final results [2,3]. Over the past decades, the adoption of these aggregated metrics has gained popularity in the realm of road safety [3]. Given the multidisciplinary nature of road safety, policymakers in this field need to consider the numerous contributing factors when making decisions. A comprehensive examination of international literature on road safety evaluation reveals the efforts made by authors in establishing methodologies for calculating composite road safety indices within specific territories [4]. The well-known DaCoTa project, with the primary goal of equipping road safety policymakers with robust data, information, and tools for evidence-based decision-making, stands as an exemplary initiative. In addition to employing standard methods, the project incorporates research activities to generate innovative tools as the composite Index [5]. This index combines various indicators describing road safety outcomes and policy performance into a unified metric known as the Road Safety Index (RSI), using the target hierarchy for road safety as a theoretical framework[5]. For that, this project centered around seven key sub-objectives: selecting valid indicators, collecting reliable data from selected countries, devising a method to combine indicators, calculating composite indices, investigating the overall RSI, addressing structural and cultural differences between countries, and visualizing the results. Using Data Envelopment Analysis (DEA) for constructing the safety composite indices, this study benefits from the fact that weights are retrieved from observed data.

In our study, we aim to delve deeper into the development of composite indicators in the realm of road safety. Our goal is to synthetize and analyze the conclusions derived from high-quality research on this subject, with a specific focus on gathering more details about the techniques and methodologies employed. The subsequent sections of the paper are organized as follows: Section 2 outlines the research objectives and details the methodology employed in our investigation. The study findings are presented in Section 3, while Section 4 discusses these results drawing valuable insights. Concluding remarks and future directions for research are presented in the conclusion.

## 2 Research objectives and methodology

To conduct a thorough review of existing literature on the development of road safety performance indicators, we employed a systematic research process. We adhere to the methodology outlined by Kitchenham and Brereton for systematic literature review [6]. This process is structured into three key phases: planning, conducting, and reporting of results. During the planning stage, we set the objectives and establish the search strategy for the review. The execution phase involves selecting primary studies, assessing their quality, and extracting and synthesizing relevant information for our study. Finally, in a structured manner, we interpret the results based on the predetermined review objectives.
Our main objective in this research is to explore the construction of composite indicators in the road safety field. We aim to identify and analyze the findings of relevant and high-quality papers on this subject. To achieve this, we formulated five specific research questions: **RQ1** ‘What is the level of interest in this subject over the last decade in the research world?’ **RQ2** ‘What are the objectives and motivation behind constructing road safety composite metrics in selected studies?’ **RQ3** ‘How is the selection of indicators conducted, considering categories and sources of data?’ **RQ4** ‘Are there any references or frameworks adopted in selected studies to guide index construction?’ **RQ5** ‘What are the methods that are frequently employed in constructing these indexes?’. To conduct our review, we employed a systematic research process. Specifically, we utilized keywords such as “road safety" AND (“index" OR " composite indicator" OR "composite metric" OR "aggregated indicator" OR "aggregated metric") to identify relevant articles within ScienceDirect database.

We established inclusion and exclusion criteria for candidate papers, excluding those that did not align with our objectives. Initially, we considered only available research articles, written in English, and published between 2000 and 2023, meeting the search keywords in their title, abstract, or keywords. Subsequently, selected studies were required to concentrate on road safety performance, specifically addressing the construction of road safety indexes, presenting concrete methods and techniques, and offering an application case. Following the initial selection, a full-text review was conducted to assess the relevance of the chosen papers to our research objectives.

For data collection and analysis, we carried out a meticulous review and analysis of the selected articles, systematically extracting relevant data that directly addressed our predefined research questions and aligned with our research objectives.

## 3 Study Results

We firstly identified 434 papers through the systematic search process. After applying the selection process, our focus was refined to 22 articles deemed relevant to our study. In this section, we provide a detailed exploration of key aspects found in the selected studies, providing the basis for addressing our research questions.

The study results, comprehensively summarized in Table 1, cover various aspects. We begin by highlighting the publication years of the selected articles, crucial for investigating the evolving interest in this subject over the last decades. Additionally, we explore the motivations underpinning each study, a crucial element to understand the driving forces behind research efforts. We investigate the methods employed in developing road safety indexes, exploring specific categories of variables used for this purpose. This investigation provides valuable data for future works in selecting metric. Simultaneously, information about the sources of data is extracted. Furthermore, we explore whether these studies have adopted specific frameworks for analyzing road safety to characterize the sequences of facts, actions, causal relationships, and consequences within the context of road safety, with the aim of guiding the selection of metrics. Notably, since all selected studies involved case studies, we extract information regarding their application scope. This approach ensures a thorough understanding of the diverse contexts within which these methodologies have been applied.
<table>
<thead>
<tr>
<th>Article</th>
<th>Date</th>
<th>Motivation</th>
<th>Variables categories</th>
<th>Methods Used</th>
<th>Data sources</th>
<th>Framework /reference</th>
<th>Case study</th>
</tr>
</thead>
<tbody>
<tr>
<td>[2] 2009</td>
<td>Show that the road safety ranking of countries differs significantly according to the selected weighting method, the expert choice and the set of indicators</td>
<td>Alcohol and drugs, speed, protective systems, visibility, vehicle, infrastructure and trauma management, Fatalities per million inhabitants</td>
<td>Weighting methods (analytic hierarchy process (AHP), Budget allocation (BA)), Uncertainty and sensitivity analysis</td>
<td>WHO, ETSC</td>
<td>SafetyNet project</td>
<td>18 European countries</td>
<td></td>
</tr>
<tr>
<td>[3] 2013</td>
<td>Introduce an alternative and improved aggregation method used to estimate the composite road safety performance index</td>
<td>Collisions, fatalities, serious injuries, vehicles involved in collisions</td>
<td>Alternative aggregation approach (indicators pairwise comparison + development of marginal and composite performance functions)</td>
<td>Local databases</td>
<td>A structural framework is proposed with potential contributing factors and potential consequences</td>
<td>29 Northern Ireland policing areas</td>
<td></td>
</tr>
<tr>
<td>[4] 2018</td>
<td>Calculate performance index using a limited set of indicators for quality comparisons across as many countries as feasible.</td>
<td>Alcohol and drugs, speed, protective systems, daytime running lights, vehicle, roads and trauma management</td>
<td>Weighting method: DEA, Equal Weighting</td>
<td>SARTRE, WHO, ETSC, Eurostat, OECD</td>
<td>SafetyNet project</td>
<td>21 European countries</td>
<td></td>
</tr>
<tr>
<td>[7] 2023</td>
<td>Expand the list of weighting methodologies for constructing road safety index</td>
<td>Drivers’ behavior (speeding, driving under influence of alcohol or drug…), accidents and Fatalities per population</td>
<td>Data envelopment analysis (DEA), Equal Weighting (EW), Fuzzy experts’ opinion, Entropy, Grey Relational Analysis (GRA), Local databases</td>
<td>-</td>
<td></td>
<td>21 Montenegro Municipalities</td>
<td></td>
</tr>
</tbody>
</table>

*Ref: article reference*
<table>
<thead>
<tr>
<th>Article Ref</th>
<th>Date</th>
<th>Motivation</th>
<th>Variables categories</th>
<th>Methods Used</th>
<th>Data sources</th>
<th>Framework / reference</th>
<th>Case study</th>
</tr>
</thead>
<tbody>
<tr>
<td>[8] 2021</td>
<td>Assess road user behavior impact on road safety severity at country level by building a composite indicator</td>
<td>Road users’ behavior (drinking alcohol, speeding, not wearing seat belts…), Fatalities per 100,000 population</td>
<td>• Weighting (Simple weighting, Theoretical methods) • Aggregating variables (Simple additive)</td>
<td>WHO report, ETSC</td>
<td>-</td>
<td>12 European countries</td>
<td></td>
</tr>
<tr>
<td>[9] 2020</td>
<td>Estimate performance specifically, the relative efficiency of different regions by using DEA</td>
<td>Influence of drug and alcohol, speed, not wearing the seatbelt, registered vehicles, road density, number fatalities and injured</td>
<td>• DEA (Output-oriented model)</td>
<td>Local databases</td>
<td>-</td>
<td>21 Montenegro Municipalities</td>
<td></td>
</tr>
<tr>
<td>[10] 2016</td>
<td>Investigate a technique to combine multi-dimensional safety performance indicators (SPIs) into an overall index</td>
<td>Policy performance indicators, road safety performance indicators (final and intermediate outcomes)</td>
<td>• Entropy embedded RSR (Rank-sum ratio)</td>
<td>ETSC, European commission (EC), IRTAD SUNflower approach, Human Development Index</td>
<td>30 European countries</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[11] 2017</td>
<td>Present an MCDM model, for selecting an optimal composite index</td>
<td>Population and number of registered motor vehicles, number of fatalities and of seriously injured persons</td>
<td>• PROMETHEE-RS (MCDM model) • TOPSIS (MCDM) • DEA</td>
<td>Local databases</td>
<td>-</td>
<td>27 police departments in Serbia</td>
<td></td>
</tr>
<tr>
<td>[12] 2021</td>
<td>Analyze the performance indicators related to the percentage of helmet and seat-belt use versus the qualitative enforcement scores attributed by WHO</td>
<td>Helmet and seat-belt enforcement</td>
<td>• DEA-based Cis • Fuzzy DEA-based CIs</td>
<td>WHO</td>
<td>-</td>
<td>30-member states of WHO</td>
<td></td>
</tr>
</tbody>
</table>
Table 1. (continued)

<table>
<thead>
<tr>
<th>Article</th>
<th>Date</th>
<th>Motivation</th>
<th>Variables categories</th>
<th>Methods Used</th>
<th>Data sources</th>
<th>Framework/reference</th>
<th>Case study</th>
</tr>
</thead>
<tbody>
<tr>
<td>[13]</td>
<td>2015</td>
<td>Conduct a road safety risk evaluation from an overall perspective, based on a composite Road Safety Risk Index.</td>
<td>Human, vehicles, roads, environment and management factors, personal and traffic risk (fatalities)</td>
<td>Entropy TOPSIS - RSR methodology</td>
<td>Local databases</td>
<td>-</td>
<td>31 provinces of China</td>
</tr>
<tr>
<td>[14]</td>
<td>2015</td>
<td>Improve the application of DEA research in ranking DMUs in the road safety field.</td>
<td>Mortality rate (fatalities per capita), Fatality rate (fatalities per vehicle, per vehicle km traveled)</td>
<td>DEA, Cluster analysis, Bootstrapping method application.</td>
<td>Local databases, EC</td>
<td>-</td>
<td>27 Brazilian states</td>
</tr>
<tr>
<td>[15]</td>
<td>2012</td>
<td>Combine the multilayer SPIs into one overall index by incorporating experts’ knowledge.</td>
<td>Alcohol and drugs, speed, protective systems, vehicle, roads, and trauma management</td>
<td>Hierarchical fuzzy TOPSIS</td>
<td>SARTRE, ETSC, WHO, Eurostat</td>
<td>-</td>
<td>21 European countries</td>
</tr>
<tr>
<td>[16]</td>
<td>2008</td>
<td>Discuss five common methods for assigning weights to indicators in the construction process of composite road safety indicator</td>
<td>Alcohol and drugs, speed, protective systems, visibility, vehicle, infrastructure, and trauma care.</td>
<td>Weighing methods: DEA, Factor Analysis, AHP, BA, and Equal Weighing</td>
<td>IRTAD, Eurostat, WHO, SARTRE, SafetyNet project</td>
<td>21 European countries</td>
<td></td>
</tr>
<tr>
<td>[17]</td>
<td>2020</td>
<td>Propose a composite safety performance index that takes account (1) exposure, (2) measuring safety outcome, and (3) safety-related equipment and resources allocated.</td>
<td>Local databases</td>
<td>PROMETHEE; Fuzzy AHP (FAHP); Shannon Entropy method</td>
<td>-</td>
<td>The risk-exposure RE approaches 31 Iranian provinces</td>
<td></td>
</tr>
<tr>
<td>Ref</td>
<td>Date</td>
<td>Article</td>
<td>Motivation</td>
<td>Variables categories</td>
<td>Methods Used</td>
<td>Data sources</td>
<td>Framework /reference</td>
</tr>
<tr>
<td>-----</td>
<td>------</td>
<td>---------</td>
<td>------------</td>
<td>----------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>----------------------</td>
</tr>
</tbody>
</table>
| [18] | 2021 | Apply DEA on index construction in the context of road safety | Apply DEA on index construction in the context of road safety | Road user behavior as drink driving, speeding and protective systems | • DEA  
• New approach (composite index with optima common set of weights - CSW) | SafetyNet project | 13 European countries |
| [19] | 2016 | Define an assessment framework able to measure single country performances with reference to the efficiency and effectiveness of road safety funding and research | Research productivity, Quality, and international collaboration activity indicators | Research productivity, Quality, and international collaboration activity indicators | • Standardizing each indicator on the maximum score  
• Calculating the arithmetic mean of the three standardized indicators | ERSO | DACOTA research project 28 European countries |
| [20] | 2020 | Benchmark road safety performance for a set of European countries | Road user behavior, Vehicle, Infrastructure; with sub-indicators. | Multiple layer DEA-based composite indicator CI | • ETSC  
• EC  
• UNECE  
• ERF | ETSC, WHO | 28 European countries |
| [21] | 2013 | Explore the relationship between road safety management and road safety performance at country level | Road safety outcomes data, Management data, Background indicators: structure and culture; Intermediate outcomes data, safety performance indicators | Selection of indicators  
• Data preparation  
• Weighting: DEA  
• Aggregation: arithmetic averaging | European Commission, ETSC. | ETSC, WHO  
• SUNflower pyramid,  
• ETSC/PIN group  
• DaCoTA research project | 30 European countries |
<table>
<thead>
<tr>
<th>Article Ref</th>
<th>Date</th>
<th>Motivation</th>
<th>Variables categories</th>
<th>Methods Used</th>
<th>Data sources</th>
<th>Framework / reference</th>
<th>Case study</th>
</tr>
</thead>
<tbody>
<tr>
<td>[22] 2022</td>
<td></td>
<td>Provide government and policymakers with a framework for reporting road safety achievement of countries</td>
<td>Safer product, safer people, and safer system; with sub-indicators.</td>
<td>• Entropy weight (EWM) – grey relational analysis (GRA) – singular value decomposition (SVD) (EWM-GRA-SVD) • TOPSIS, RSR (MCDM)</td>
<td>WHO, IRTAD, World Bank, ITF</td>
<td>WHO</td>
<td>18 (APEC) Asia-Pacific Economic Cooperation countries</td>
</tr>
<tr>
<td>[23] 2010</td>
<td></td>
<td>Benchmark countries' road safety performance, combine the main layers of the road safety pyramid which describes the complex nature of road safety activities, performance and outcomes.</td>
<td>Road safety policy performance (safety measures and programs), accident fatalities/injuries (as the final outcomes) and safety performance indicators (as intermediate outcomes).</td>
<td>• Weights based on statistical models; Principal Component Analysis (PCA) and Common Factor Analysis (FA) weighting</td>
<td>ETSC • OECD • ITF • EC • UNECE</td>
<td>SUNflow approach</td>
<td>27 European countries.</td>
</tr>
<tr>
<td>[24] 2023</td>
<td></td>
<td>Monitor progress and calibrate interventions for road safety improvement using a regular performance rating</td>
<td>Safer products, safer behavior, and safer system, each containing multiple sub-indicators</td>
<td>• Principal Component Analysis of RIDIT scores (PSI-PRIDIT), MCDM</td>
<td>• ASEAN Stats, • WHO reports, • International Road Federation</td>
<td>-</td>
<td>13 East Asia Summit (EAS) countries</td>
</tr>
<tr>
<td>[25] 2020</td>
<td></td>
<td>Perform the benchmarking of road safety development in an integrative manner for OECD countries.</td>
<td>Outputs dimension: Personal risk, traffic risk, change trend Inputs dimension: Road user behavior, socioeconomic situation, vehicle safety, roads situation, enforcement</td>
<td>IV-VIKOR with FNBC (inverse variance weighting), VIKOR method (MCDM), Fisher’s Natural Breaks Classification</td>
<td>OECD, WHO, IRTAD, World Bank</td>
<td>-</td>
<td>36 OECD Member countries</td>
</tr>
</tbody>
</table>
4 Discussion

In this section, we thoroughly analyze and discuss the findings in response to the research questions that were posed:

**RQ1:** Examining the data reveals fluctuating interest in this subject over the last two decades, with notable peaks in 2013, 2020, and 2021, each recording four publications. This temporal pattern implies dynamic shifts in scholarly attention towards road safety composite indicators, reflecting potential changes in research priorities or emerging trends in this area. Additionally, the research interest in road safety composite indicators has been predominantly concentrated in Europe, Asia and America. Surprisingly, there is a notable absence of studies addressing this subject on the African continent. However, some studies including countries member states of the World Health Organization (WHO) and the Organization for Economic Co-operation and Development (OECD), do include certain African nations.

**RQ2:** Selected studies adopt different approaches, all aimed at improving understanding, guiding policy decisions, and ultimately enhancing road safety outcomes. The developed composite indexes present several advantages, serving as powerful tools for public communication, benchmarking and decision-making support. These metrics offer a standardized measure allowing comparison of safety levels across international territories and ranking regions for national road safety performance assessment. They provide a comprehensive evaluation of safety measures, highlighting areas in need of improvement. Additionally, some studies are motivated by the intent to apply particular methods for developing indexes in the context of road safety, while others contribute to refining existing methodologies by identifying specific weaknesses in current practices.

**RQ3:** The study findings reveal a diverse range of variables that contribute to the comprehensive assessment of road safety and are used in the formulation of composite indicators. These variables are frequently employed to reflect background indicators, policy performance indicators, final, and intermediate outcomes in the context of road safety. They can be categorized into distinct groups, each targeting specific dimensions of road safety. They cover elements related to road accidents and fatalities. These indicators extend to aspects such as road users' behavior, features related to vehicles, roads and infrastructure, and trauma management. Additionally, population-related factors are included. This diverse set of indicators reflects a comprehensive approach to assessing road safety, encompassing various factors and domains crucial for a holistic understanding of road safety dynamics.

On another side, the data sources used for the construction of road safety indexes vary among selected studies, encompassing a wide range of local and international databases and reports. The sources include local databases, and international databases and reports from different organizations, as the World Health Organization (WHO), the European Transport Safety Council (ETSC), the Organization for Economic Co-operation and Development (OECD), the World Bank, the International Road Traffic and Accident Database (IRTAD), the European Commission (EC), the statistical office Eurostat, the European Road Safety Observatory (ERSO) and the International Transport Forum (ITF). However, it is essential to highlight the issue of data availability that was underscored in the majority of selected studies, posing a notable
concern in the process of selecting road safety metrics. Efforts to enhance data collection and accessibility will contribute to the robustness of road safety metrics.

**RQ4:** Prominent frameworks and references were employed, providing a solid basis for selecting metrics. Valuable insights were thus obtained from existing road safety projects, including the SafetyNet project aiming to establish the framework of a European Road Safety Observatory, the SUNflower project conducting a comparative study on road safety development in Sweden, the United Kingdom, and the Netherlands, and the previously mentioned DACOTA research project. Additionally, publications and reports from the European Transport Safety Council (ETSC) and the World Health Organization (WHO) contributed to the wealth of knowledge in this area.

**RQ5:** Various methodologies have been employed to construct road safety indexes in selected studies, incorporating a spectrum of techniques ranging from traditional statistical methods to advanced decision-making models. The crucial steps outlined in the Handbook on constructing composite indicators, as discussed earlier, involve the selection of indicators, weighting techniques, and aggregating methods [1]. The aspects related to the selection of indicators were discussed in response to RQ3 et RQ4. In the context of weighting techniques, the assignment of weights is crucial as it represents the relative importance of indicators contributing to the final score. While various weighting techniques, including commonly used statistical methods like Factor Analysis (FA), Principal Component Analysis (PCA), and Rank-Sum Ratio (RSR), are employed, Data Envelopment Analysis (DEA) stands out as a widely adopted technique, notably prevalent in almost half of selected articles. DEA offers the advantage of not requiring external subjective judgment and is favored for determining weights, with the added benefit of integrating units with different measurements[7]. Other frequently used techniques include equal weights (EW), entropy, Grey Relational Analysis (GRA), and fuzzy expert opinions. In addition to statistical methods, multi-criteria decision methods (MCDM) have gained popularity in recent years, often employed for simultaneous weighting and aggregation[7]. Analytic Hierarchy Process (AHP) is one such method that incorporates the subjective opinions of experts to derive weights. Another frequently used weighting methods based on expert opinions is budget allocation (BA) [2]. The Shannon Entropy method was, however, used, representing the MCDM techniques that calculate the importance coefficient based on the differences of indicator values among the comparing units instead of expert opinions [17]. The PROMETHEE (MCDM), which is an intuitive method for decision-makers due to its simple comparison between each pair of alternatives, was also employed to make an optimal selection of a composite index [11,17]. Additionally, the TOPSIS method, a well-known mathematical model that can combine different factors into one unique measure, and functioning as a MCDM method relying on the relative distance from the ideal and anti-ideal solution, was employed [11,13,15,22]. Furthermore, the Ordered Weighted Averaging (OWA) operators were employed as an expert method for data aggregation. OWA functions represent the second type of common averaging aggregation functions [4]. Very common aggregation operators include also arithmetic averaging and simple additive [8,21].

**Conclusion**
This study delves into the evolution of road safety composite indicators in the literature, aiming to discover the motivations and examine the diverse methodologies used in their construction. Drawing insights from the analysis of high-quality papers, the study underscores the growing interest in this research field over the last decades, pinpointing the various objectives guiding the construction of these indicators. Primarily, the study provides a comprehensive overview of the techniques, methodologies, and frameworks commonly used within the realm of road safety composite indicators. Through this exploration, we aim to equip road safety decision-makers and researchers with an in-depth understanding of the significance of these metrics, providing guidance on their construction and offering a global benchmarking of accomplishments in this research domain. Future studies can leverage the techniques and tools uncovered in this study to develop composite indicators for analyzing road safety in African countries, addressing unique challenges, regional dynamics, and exploring potential solutions across the region.

Acknowledgements. This work was supported by the Ministry of Transport and Logistics, NARSA and CNRST.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References


1 Introduction

Urban traffic is among the main air pollution sources in our cities. Consequently, one of the main lines of action in the transition towards the future sustainable smart cities is reducing tailpipe emissions. The electrification of vehicles is a feasible solution that is currently being implemented, and it will allow achieving huge savings in traffic pollutant emissions. However, they have some drawbacks like the limited autonomy or the long recharging times. Plug-in Electric Hybrid (PEH) buses emerge as a more flexible solution while still highly reducing pollution levels. They can switch between electric motor and diesel engine at any time, allowing an efficient location of zero-emission zones (ZEZ) that not only provide environmental benefits but also social. The current strategies for managing the battery are very conservative and naïve, but more advance strategies can make use of their full potential.

This extended abstract presents our work [2], recently published in Sustainable Cities and Society where the Multi-objective Efficient PEH Bus Operation Problem, MEPBO for short, was proposed. It proposes that a tailored location of the ZEZ, i.e. a tailored battery management strategy can enlarge the electric range of the bus while at the same time reduce the pollution. A novel methodology for accurately estimating the energy consumption of the PEH bus is presented and two multi-objective optimization algorithms from the state of the art are used to solve this problem.

2 The Multi-objective Efficient PEH Bus Operation Problem, MEPBO

We define the Multi-objective Efficient PEH Bus Operation (MEPBO) problem as finding the most appropriate battery management strategy during the route operation of a PEH bus so that (i) the number of kilometers driven in electric mode is maximized, (ii) emissions are minimized, and (iii) the mandatory zero emissions zones (mZEZs), where buses must circulate using the electric motor, are satisfied.

A highly realistic definition of the problem is proposed, where highly accurate models for energy consumption and pollutant emissions are calculated according to the bus characteristics, road elevation profile, bus acceleration and speed, and the engine used [2]. Also, the energy recovery system when braking and downhill driving is taken into account.

MEPBO is modelled as follows. Lets assume that the bus route $S$ is divided into $n$ segments, $S = \{s_0, s_1, \ldots, s_{n-1}\}$, where each segment $s_i$ is defined by the tuple $(l_i, \alpha_i, z_i, b_{si})$, being $l_i$ its length (measured in kilometers), $\alpha_i$ its slope, $z_i$ an indicator that can take value 1 or 0 depending on whether the segment is a mZEZ or not, and $b_{si}$ another binary value indicating if the segment starts at a bus stop or not, respectively. The route is split into segments in points where the driving condition change or a bus stop (BS) exists, so consumption and emissions are constant within segments. MEPBO is a multi-objective optimization problem, where emissions are to be minimized while maximizing the distance traversed with the electric motor. Its mathematical formulation was first described in [2].

3 Methodology

The methodology followed to solve the proposed problem is divided into four steps, namely (i) accurate data acquisition, (ii) route segmentation, (iii) optimization process, and (iv) assessment simulations. The first step includes retrieving information about the bus route from multiple sources,
including the arrival times at the different bus stops, as well as their location, information on the route, as an accurate elevation profile and intersections, or characteristics of the bus. All these data is required to obtain highly accurate and realistic solutions. Second, it is required to split the route into segments in order to apply the mathematical bus consumption model. Segments are defined assuming that the bus consumption does not change within the segment, so they are split at intersections and bus stops, where the bus needs to break and accelerate, and at significant changes in the elevation profile (i.e., \( \geq 2\% \)). The route that the bus follows from every pair of bus stops is computed using OSMR API [3]. Third, the multi-objective combinatorial problem defined is solved with the well-known MOEA/D [4] and MOCell [1] algorithms. Furthermore, we design and implement GreenK, a simple yet effective heuristic to solve the problem, used in this work for comparison purposes. Finally, an assessment simulator is used in the fourth step to estimate the electric energy consumption and the pollutant emitted in every segment, as well as the total number of kilometers traversed in electric mode.

4 Results

The well-known MOEA/D and MOCell algorithms were used to solve two real problem instances, namely lines 18 in Grudziadz, Poland, and M6 in Badalona, Spain. In addition, we compare the results against those of GreenK heuristic, designed following the actual strategy used in the PEH buses operating in the studied lines. We consider four scenarios for every route, where 0%, 5%, 10% and 15% of the segments composing the routes are defined as mZEZs. In addition, we consider the cases when the bus is allowed to recharge at destination (before starting the trip back to the initial stop) or not.

Figure 1 shows a representative example of the results found. They correspond to Line 18, in Grudziadz. As it can be observed, there always exit solutions proposed by the multi-objective algorithms that clearly outperform the result of GreenK for both objectives.

It is observed in figures 1(b), 1(c) and 1(d) that the difference in pollutants among solutions in the front for Line 18 is lower than 72 gr of \( \text{CO}_2 \), while the variation on the distance traversed with the electric motor in the solutions exceeds 851.98 m. However, a higher diversity is found for instance with 0% mZEZ (see Figure 1(a)) where the top right solution (traversing longer in electric mode) pollutes over 1.35 kg of \( \text{CO}_2 \) for driving only less than 6m longer than the solution on the bottom right corner. Therefore, this latter solution would be better suited as a candidate solution to be deployed.

5 Conclusions

In this work, a new problem is presented to find sustainable battery management strategies for urban plug-in electric hybrid buses. The problem is called MEPBO. Additionally, a novel methodology to efficiently estimate the energy consumption of the PEH bus is proposed. Moreover, two well-know multi-objective algorithms from the state of the art are used to solve it. Two different bus routes are used to validate the proposal, lines 18 from Poland and M6 from Badalona. Additionally, four use cases with different degrees of mandatory zero emission zones are studied as well as the possibility of recharging the battery at the destination point.

Results show that tailoring the battery management strategy for each use case provides the best performance given the (topological) peculiarities of each route, and significant achievements are obtained in terms of pollutants emissions reduction and electric range of the vehicles. Moreover, insights of the benefit of including en-route recharging stations at the destination are also obtained.

Acknowledgment

This publication is part of projects eMob (PID2022-137858OB-I00) and eFracWare (TED2021-131880B-I00) both funded by MCIN/AEI10.13039/ 501100011033 and the European Union “NextGenerationEU”/PRTR. J.M. Aragón-Jurado would like to acknowledge the Spanish Ministerio de Ciencia, Innovación y Universidades for the support through FPU21/02026 grant.
Battery Management Strategy Optimization for Urban Plug-in Hybrid Buses

Fig. 1. Best Pareto fronts obtained by both MOEA/D (red symbols) and MOCell (gray symbols), and the solution provided by GreenK (green symbol) for bus line 18 (in Grudziadz, Poland) considering four different mZEZ percentages in the route, and that recharging at destination is allowed.

References


3. The OSMR project: Open source routing machine (2022), http://project-osrm.org/.

Automatic Software Performance Optimization using Genetic Algorithms

J. M. Aragón-Jurado, J. C. de la Torre, P. Ruiz, and B. Dorronsoro
Higher School of Engineering, University of Cádiz, Spain.
{josemiguel.aragon, juan.detorre, patricia.ruiz, bernabe.dorronsoro}@uca.es

1 Introduction

Software programs (SW) are who ultimately drives the underlying hardware (HW). Therefore SW should make an efficient use of all available resources in order to achieve high performance levels when executing.

Compilers typically perform a number of code transformations, also known as optimizations, before generating the executable file. The purpose of applying such optimizations is to modify the use of the underlying HW by the considered SW. These are generic sequences of optimizations that are known to generally perform well. However, it is clear that distinct SW programs and HW architectures offer different characteristics, and the same sequence of code transformations cannot fully exploit them for optimal performance.

Therefore, there is a need of finding ad hoc optimization sequences to accurately optimize the performance of every SW/HW pair. However, this is a titanic task, taking into account the immense variety of existing HW architectures, for which every SW must be optimized. In addition, the process of optimizing SW performance for a given HW architecture is a difficult, error prone, and expensive task, requiring the hands of expert programmers with high knowledge on the considered architecture. Automatizing this process would allow the generation of a new generation of smart compilers, able to perform ad hoc transformations for the considered SW and HW for optimal performance.

We propose in this work a novel approach to automatize the process of optimizing the performance of a particular SW program on a specific HW. For that, we define a novel combinatorial optimization problem, that we call SCOP (for Software Code Optimization Problem), consisting in finding a sequence of code transformations that minimizes the runtime of the resulting SW on the considered HW architecture. This extended abstract presents our work [2], recently published in IEEE Transactions on Sustainable Computing.

2 Software Code Optimization Problem

The SCOP problem is to find a sequence \( S = [s_1, \ldots, s_n] \), where \( n \) is its length and \( s_i \in [0, \ldots, t] \), being \( t \) the number of available code transformations, 0 representing the case when no transformation is applied. The goal is to find the sequence that minimizes the SW runtime execution file:

\[
\text{Minimize } F(P, S) = \text{Runtime}(P')
\]

where \( P' \) is the result of applying all transformations in \( S \) to the intended software \( P \), in the exact same order as they appear in the sequence \( S \).

In this work, we make use of the transformations provided by LLVM 9.0.1 [3], also called passes. LLVM\(^1\) is a well-known compiler infrastructure that supports several programming languages (e.g., C, C++, Objective-C, CUDA, or Julia, among others) and instruction sets (including IA-32, x86-64, ARM, Qualcomm Hexagon, MIPS, or NVIDIA). One of its outstanding features is its Intermediate Representation (IR), a pseudo assembly programming language that can be generated from all supported languages. This LLVM IR code is generated using the Clang compiler on the original source code. LLVM passes are applied on the IR code, allowing our methodology to be applied to

\(^1\) Since December 2011, “LLVM” is officially no longer an acronym and simply a brand that applies to the LLVM umbrella project. For more information, see https://www.llvm.org.
any programming language supported by LLVM. Additionally, all passes implemented in LLVM guarantee that the semantic of the code remains unaltered. LLVM 9.0.1 offers 87 different code transformations, and they can be applied several times in the same sequence. The order in which passes appear in the sequence is the order in which they are applied to \( P \), given that it has a considerable impact on the result [5]. In addition, the fact that the direct impact of applying any single pass on the SW performance is negligible, while it becomes notable when it is applied in combination with others [4], makes the problem hard to solve.

3 Methodology

We propose the use of a Cellular Genetic Algorithm (cGA) [1] to find a sequence of LLVM passes that minimizes the SW run time. The length of the sequence (i.e., the chromosome length) is set to 300, similar to the number of transformations applied by typical compiler optimization flags.

In order to deal with the uncertainty on the SW runtime, we study five different fitness function, each of them dealing with the uncertainty in a different way:

- **One execution** (1run): the quality of individual is the value of one single execution of the software program.
- **Median** (median): in each fitness evaluation the potential solution is executed five times and the quality of the individual is set to median value.
- **Worst case** (wcase): in each fitness evaluation the potential solution is executed five times and the quality of the individual is set to the worst measured value.
- **Interval** (interval): in each fitness evaluation the potential solution is executed five times and the quality of the individual is set to the confidence interval obtained using the bootstrap method.
- **Worst case interval from 15 executions** (wcase15): this is a new approach proposed in this work where in each fitness evaluation the potential solution is executed fifteen times and the quality of the individual is set to the confidence interval obtained using the bootstrap method to the five worst runs (from the 15 possible).

4 Results

We analyze the performance of the proposed methodology when optimizing the well-known Polybench benchmark SW on four distinct HW architectures: low-power and high-performance ARM-based (Raspberry Pi 3B+ and a Huawei TaiShan 2280 V2 server with 2 Kunpeng 920-4826 CPUs of 48 cores each) and Intel-based (Latte Panda Alpha with an Intel Core M3-8100Y Dual-Core processor and a server with an Intel Core i7-7700K 4.20 GHz processor) architectures.

Polybench was optimized using the cGA, and the overall best solution out of the 30 cGA runs is taken. Then, for every architecture, Polybench is optimized applying the sequence of passes in the solution, and the resulting program version is executed 1000 times on the corresponding architectures. The runtimes obtained in these experiments are shown in Figure 1, where the red dot is the fitness obtained by the algorithm for the single-valued fitness functions, and the green and blue dots represent the interval used as fitness for the interval-based functions.

It can be seen that most fitness functions obtain underestimations (i.e., lower runtime) with respect to the actual performance of the evaluated program. This can mislead the algorithm in its convergence towards better solutions. In contrast, \textit{wcase15} fitness function provides accurate estimations of the actual runtime (both green and blue dots fall within the box), at the cost of requiring a higher number of executions.

It must be highlighted that \textit{wcase15} is the fitness function that leads the algorithm towards better SW versions. It is also interesting that there are little differences between the different methods in the platforms with ARM architectures.

If we compare the solutions against the original SW and its optimized version using \texttt{-O3} Clang optimization flag, we can see that the solutions reported by the cGA are between 28.1\% and 63.2\% faster than the original non-optimized software, and between 13.9\% and 26.3\% with respect to the well-known \texttt{-O3} optimization flag.
Fig. 1. Comparison of the performance of the best result provided by the GA, after 1,000 runs, on the four considered architectures. Red dot is the fitness computed during the GA execution for that solution. Blue and green dots represent the fitness of the solutions for interval-based ones.

5 Conclusions

This paper describes a novel combinatorial optimization problem for automatically optimizing SW programs performance. The work was originally published in [2]. Five different fitness functions are considered to study their resilience against system uncertainty, being the interval-based ones the most accurate. All solutions were able to outperform the well-known `-O3 Clang optimization flag by up to 26.3% lower runtime.

Acknowledgments

This publication is part of projects eMob (PID2022-137858OB-I00) and eFracWare (TED2021-131880B-I00) both funded by MCIN/AEI10.13039/501100011033 and the European Union “NextGenerationEU”/PRTR. J.M. Aragón-Jurado would like to acknowledge the Spanish Ministerio de Ciencia, Innovación y Universidades for the support through FPU21/02026 grant.

References

A Multi-objective Clustering Algorithm Integrating Intra-clustering and Inter-clustering Measures *

Beatriz Flamia Azevedo1,2,3, Ana Maria A. C. Rocha2, and Ana I. Pereira1,2,3

1 Research Centre in Digitalization and Intelligent Robotics (CeDRI), Instituto Politécnico de Bragança, Bragança - 5300-253, Portugal
2 ALGORITMI Research Centre / LASI, University of Minho, Campus de Gualtar, 4710-057 Braga, Portugal
3 Laboratório Associado para a Sustentabilidade e Tecnologia em Regiões de Montanha (SusTEC), Instituto Politécnico de Bragança, Bragança - 5300-253, Portugal.

Abstract. This study delves into bio-inspired approaches and clustering methodologies to introduce an automated clustering algorithm named Multi-objective Clustering Algorithm (MCA). Using multi-objective strategies and several combination measures, this method calculates the optimal number of clusters and element partitioning by minimizing intra-clustering measures and maximizing inter-clustering ones. Through experimentation on three benchmark datasets, the results highlight the success of the MCA in obtaining a set of optimal solutions (Hybrid Pareto front) through the integration of multi-objective strategies and clustering measures. Moreover, the Dunn clustering validity index is used to support the decision maker in selecting the optimal solution among the ones presented in the Hybrid Pareto front. This approach allows decision-makers to choose the most suitable solution by incorporating additional insights beyond the model.

Keywords: Multi-objective · bio-inspired methods · partitioning-clustering

1 Introduction

Clustering problem intends to divides a dataset based on inherent characteristics, ensuring that elements within the same cluster are more similar than

* This work has been supported by FCT Fundação para a Ciência e Tecnologia within the R&D Units Project Scope UIDB/00319/2020, UIDB/05757/2020 (DOI: 10.5449/UIDB/05757/2020), UIDP/05757/2020 (DOI: 10.5449/UIDB/05757/2020) and Erasmus Plus KA2 within the project 2021-1-PT01-KA220-HED-000023288. Beatriz Flamia Azevedo is supported by FCT Grant Reference SFRH/BD/07427/2021.
those in different clusters [1,2]. The prominent clustering methods rely on distance measures, metrics, and similarity functions. Additionally, many clustering algorithms exhibit performance dependence on the initial values of cluster centers [2,4]. Partitioning clustering methods, which create distinct non-overlapping subsets in a dataset, allocate each data point to a single cluster. The objective is to achieve a partitioning quality by optimizing a specified criterion (single objective approach) or criteria (multi-objective approach). Although computationally efficient and suitable for large datasets, partitioning methods can be sensitive to initial centroid choices and may struggle with irregularly shaped or differently sized clusters [2,4].

Traditional clustering algorithms often fail in real-world scenarios due to the high-dimensional and inherently sparse nature of data sets. Single objective algorithms struggle when clusters have diverse shapes across different feature space regions, leading to a lack of fitting criteria for the entire data distribution. Moreover, there are algorithms that assume a uniform criterion, resulting in detected clusters with similar shapes and densities [3].

On the other hand, multi-objective clustering involves addressing scenarios with conflicting objectives that need simultaneous optimization. Unlike traditional clustering, where the goal is to group similar data elements based on specific criteria, multi-objective clustering aims to find clusters that represent a trade-off among multiple conflicting objectives. Thereby, selecting the appropriate clustering criteria is crucial yet challenging, as different properties may conflict, and the underlying data distribution is usually unknown. Multi-objective evolutionary clustering addresses these issues by simultaneously optimizing multiple objectives, allowing for the resolution of complex clustering problems and the identification of trade-offs between objectives. This approach leads to higher quality solutions and improved robustness against various data properties compared to traditional methods [3].

In this context, Liu et al. [6] introduce two multi-objective clustering approaches, a multi-objective evolutionary clustering that combines multiple distance measures with an intra-cluster distance that integrates Euclidean and Path distance measures together for partitioning the dataset with different structures; and a multi-objective evolutionary automatic clustering that uses multiple distance measures and also considered inter-cluster measures. In both cases, the algorithms must define the optimum number of centroid, that belong to a range defined by the user, and the optimum solution is not defined automatically, but by the analysis of several cluster index measures.

In turn, Dutta et al. [5] proposes a Multi-objective Genetic Algorithm for automatic clustering, considering numeric and categorical features, which aims to minimize intra-cluster distance and maximize inter-cluster distance. This approach combines the local search ability of $k$-means with the global search ability of a Multi-objective Genetic Algorithm to determine the optimal cluster number.

This work explores multi-objective optimization strategies and clustering measures to achieve a robust automatic clustering method named Multi-objective Clustering Algorithm. This algorithm uses an evolutionary process and a set of
measure combinations to define the optimal number of clusters and the distribution of the elements, minimizing an intra-clustering measure and maximizing an inter-clustering one. As we are considering a multi-objective approach, the results of each combination consist of a set of Pareto fronts. The solution of these Pareto fronts was analyzed in terms of dominance and combined in a Hybrid Pareto front, composed of solutions provided by different combinations of measures. Moreover, the Dunn cluster validity index is used to support the decision-maker solution choice, if necessary.

This paper is organized as follows: after introduction, the methodology is presented in Section 2; it is the development of the Multi-objective Clustering Algorithm, starting with the problem formulation, followed by the clustering measures and the algorithm steps. The Dunn cluster validity index is defined at the end of this section. The results and discussion are shown in Section 3. The main conclusion and future steps are presented in Section 4.

2 Multi-objective Clustering Method

The following sections present the bi-objective problem, the clustering measures used, and the Multi-objective Clustering Algorithm definition. In the end, the Dunn cluster validity index is defined.

2.1 Multi-objective Problem

This study explores a bi-objective programming problem, whose objective functions are based on intra-clustering and inter-clustering measures. The bi-objective programming problem is characterized by the simultaneous minimization of an intra-clustering measure and the maximization of an inter-clustering measure:

\[
\min \{ f_i, -g_j \} \tag{1}
\]

where \( f_i \) \((i = 1, 2)\) is a intra-clustering measure and \( g_j \) \((j = 1, ..., 6)\) is the inter-clustering measure.

2.2 Clustering Measures

To cluster the dataset into distinct sets, it is crucial to establish criteria for computing distances between individual elements [11,12]. This research investigates various conventional measures and examines potential modifications in both intra- and inter-clustering metrics, detailed below. Throughout this study, the Euclidean distance is denoted by \( D(\cdot, \cdot) \).

**Intra-clustering Measures** refer to the distance among elements within a specific cluster. Several methods exist for calculating the intra-clustering measure, and this paper examines two of them, detailed as follows:
\( S_{AxC} \): it is the average distance between the elements to their centroid, in terms of the number of elements belonging to each cluster set \( \#C_j \), where \( S_{xcj} \) represents the sum of the distance between the elements and the centroid \( j \) as defined in Equation (2).

\[
S_{AxC} = \sum_{j=1}^{k} \frac{S_{xcj}}{\#C_j}
\]  

(2)

\( F_{Nc} \): it is the sum of the furthest neighbor distance of each cluster \( C_j \), where \( x_i^j \) and \( x_l^j \) belong to the same cluster \( j \), as described in Equation (3).

\[
F_{Nc} = \sum_{j=1}^{k} \max \{ D(x_i^j, x_l^j) \} \text{ for } i = 1, \ldots, \#C_j, \ l = 1, \ldots, \#C_j, \ i \neq l
\]  

(3)

**Inter-clustering measures** define the distance between elements that belonging to different clusters or the distance between different centroids \( c_j \). In this instance, six inter-clustering measures were taken into consideration:

- \( S_{cc} \): it is sum of the distance between each centroid \( c_j \), as shown in Equation (4).

\[
S_{cc} = \sum_{t,j=1, t \neq j}^{k} D(c_t, c_j)
\]  

(4)

- \( A_{cc} \): it is the average distance between all centroids, as presented in Equation (5), which is also used in the centroid method [11].

\[
A_{cc} = \frac{S_{cc}}{k}
\]  

(5)

- \( F_{Ncc} \): it is the sum of the furthest neighbor distance between elements from different clusters \( C_j \), as define by Equation 6. This measure is known in the literature as complete linkage [12].

\[
F_{Ncc} = \sum_{j=1}^{k} \sum_{l>j}^{k} \max \{ D(x_i^j, x_l^j) \} \text{ for } i = 1, \ldots, \#C_j, \ l = 1, \ldots, \#C_l, \ i \neq j
\]  

(6)

- \( A_{FNcc} \): it is the average of the distances of the furthest neighbors among the different clusters in terms of the number of clusters, as described in Equation (7).

\[
A_{FNcc} = \frac{F_{Ncc}}{k}
\]  

(7)
Multi-objective Clustering Algorithm

- **NNcc**: it is the sum of the nearest neighbor distance between elements of different clusters, also known as single linkage [11], is defined in Equation (8).

\[
NNcc = \sum_{j=1}^{k} \sum_{l=j+1}^{k} \min D(x_j^i, x_l^i) \text{ for } i = 1, \ldots, \#C_j, l = 1, \ldots, \#C_l, i \neq l \quad (8)
\]

- **ANNcc**: it is the average nearest neighbor distance between elements of the different clusters, which is defined in Equation (9).

\[
ANNcc = \frac{NNcc}{k} \quad (9)
\]

2.3 Multi-objective Clustering Algorithm

The Multi-objective Clustering Algorithm (MCA) assesses intra- and inter-clustering measures to autonomously determine the optimal number of centroids and their optimal positions [9,10]. This is accomplished by simultaneously minimizing intra-clustering distances and maximizing inter-clustering distances, incorporating various pairs of intra- and inter-clustering measures throughout the process.

To elucidate the MCA, consider a dataset \( X = \{x_1, x_2, \ldots, x_m\} \) consisting of \( m \) elements, where \( x_i \in \mathbb{R}^d \) (with \( d \) being the number of variables in the dataset). The objective is to partition \( X \) into \( k \) optimal groups (clusters). As MCA can autonomously determine the optimal number of clusters, it requires the specification of the range of possible partitions. This involves setting the minimum \( k_{\min} \) and maximum \( k_{\max} \) number of centroids possibilities.

The algorithm starts by receiving the dataset as input. Subsequently, a pair of measures, comprising one intra-measure and one inter-measure, is automatically chosen from those outlined in Subsection 2.2. Leveraging this selection, the Multi-objective Clustering Algorithm employs Multi-objective Particle Swarm Optimization (MOPSO) [7] to generate the Pareto front associated with the bi-objective function of the problem. Thereby, \( f_i \) denotes the intra-clustering measure, and \( g_j \) denotes the inter-clustering measure. This process iterates until all combinations of measure pairs have been assessed. At the end of this process, we have a 12 different Pareto fronts.

Each Pareto front is derived for each pair of measures. Since the considered measures are based on sums and averages with varying magnitudes, normalization is necessary to ensure a fair comparison of solutions. Then, each Pareto front is normalized through the Min-Max scaling method [8]. That is, each point \( x \) of the Pareto front is individually normalized between \([0, 1]\), where \( x_{\min} \) and \( x_{\max} \) are respectively the smallest and the largest solution value belonging to the Pareto front considered. Subsequently, all normalized solutions from the Pareto fronts were assessed based on dominance, and the non-dominated solutions were chosen to form the Hybrid Pareto front (HPF). This HPF represents the output of the MCA algorithm, constituting the set of non-dominated solutions, considering all the (normalized) solutions derived from the Pareto fronts obtained for each pair of measures.
Thereby, all solutions within the Pareto fronts generated were evaluated regarding dominance, and the non-dominated solutions were chosen to constitute the HPF. The Hybrid Pareto front is the algorithmic output, representing the optimal set generated by the MCA.

2.4 Cluster Validity Index

Cluster Validity Indices (CVI) define a relation between intra-cluster cohesion and inter-cluster separation to assess the clustering separation quality. A CVI is expected to be able to distinguish between superior and inferior potential solutions to guarantee the efficiency of the clustering algorithm. There are several CVI available in the literature, each serving a specific purpose [9].

This paper uses the Dunn index [13] to evaluate the Hybrid Pareto front solutions. The Dunn index is a ratio-type index where the cohesion is estimated by the nearest neighbor distance and the separation by the maximum cluster diameter. Thus, a higher Dunn value will indicate compact, well-separated clusters, while a lower index will indicate less compact or less well-separated clusters [13]. Thus, the Dunn index is defined as the ratio between the minimum distance between elements of different clusters, it is $x_i^j$ and $x_i^t$, and the largest distance between elements of the same cluster, it is $x_i^j$ and $x_j^l$, as defined in Equation (10).

$$Dunm = \frac{\min_{j,t=1,\ldots,k} \{D(x_i^j, x_i^t)\}}{\max_{j=1,\ldots,k} \{D(x_i^j, x_i^j)\}}$$

(10)

3 Results and Discussion

This section presents the achieved results by the MCA with three benchmark datasets, named Mydata [14], Spheres6 [15], and Spheres4 [15].

The parameters used in the MCA are population size and maximum of iterations $= 100$, repository size $= 30$, minimum number of centroids $k_{min} = 2$, and maximum number of centroids $k_{max} = \lceil \sqrt{m} \rceil$, where $[a]$ represents the nearest integer of $a$.

3.1 Mydata dataset results

The dataset Mydata, is a benchmark dataset, defined by two variables, available in Matlab library [14], which is composed of 300 elements divided into 3 groups, according to the benchmark documentation. Therefore, considering the previously mentioned MCA parameters, the possible number of clusters can vary between $k_{min} = 2$ and $k_{max} = 17$. And $\zeta = 17$ is the minimum number of elements per cluster, as the algorithm default.

Figure 1a presents the HPF obtained for this dataset, in which the $x-axis$ represents the intra-clustering measure and the $y-axis$ are the inter-clustering measure. This HPF is made up of 49 solutions provided by 12 different pairs...
of measures, as detailed in Table 1. Although the number of clusters may be the same for different solutions presented in the HPF, they are still considered distinct since the centroids’ position and the elements’ distribution are different. Besides, in this work, if two solutions are equal in terms of centroid positions and element distribution, but they were generated by different pairs of measures, these solutions are considered different.

We are dealing with a benchmark dataset, where the optimal number of clusters is precisely known. In this specific situation, the optimal solution would be the one with 3 clusters, as illustrated in Figure 1b, that was generated by the pair of measures $SAxc - Sxc$. However, in real-world situations, it may be more advantageous for the decision maker to split the dataset, as exemplified in Figure 1c, which represents a solution with 8 clusters, generated by the measures $FNc - AFNcc$. This can occur when the decision maker has prior knowledge about the data, incorporating this information into the decision-making process.

For example, consider a real situation in which there is a group of patients affected by three different illnesses. An effective clustering algorithm will classify these patients into 3 distinct groups, enabling the decision maker to identify the
patients affected by each disease, as exemplified in Figure 1b. This is an excellent result when the decision maker does not have prior information about the data. However, in scenarios where the decision maker already knows that there are 3 sets of illnesses and seeks to understand which stage of the disease each patient is in, it becomes interesting to have the data partitioned into more groups. This allows for identifying the different stages that patients go through, as illustrated in Figure 1c.

On the other hand, another tool that can be employed to support decision-making is Cluster Validity Indices. When there is no prior information about the dataset, the decision maker can use this resource to determine the most appropriate solution from the Hybrid Pareto front. In this case, the Dunn index is used to support the decision, but other metrics can be used, including the combination of several ones, in a collaborative strategy [9].

Thus, the Dunn index of all solutions described in the Hybrid Pareto front was calculated. Remember that the higher the Dunn index, the better the clustering. The highest value found was 0.057 for the solution presented in Figure 1b, which shows 3 clusters, as suggested by the literature. In the case of the solution of Figure 1c, which considers 8 clusters, the Dunn value found was 0.021.

As can be seen, the best Dunn index is obtained by the solution considered optimal by the benchmark. Although the other solutions presented in the Pareto front are also considered optimal for the mathematical model, the Dunn index value will always have a lower quality than that indicated as optimal by the benchmark documentation. However, as already mentioned, in certain contexts, there are advantages to using a solution different from that shown in the benchmark documentation.

3.2 Spheres6 dataset results

The Spheres6 dataset, is a benchmark dataset defined by two variables, available in [15], with comprises 300 elements, divided into 6 clusters [15]. Therefore, the $k_{\text{min}}$ and $k_{\text{max}}$ parameters are respectively 2 and 17, and $\zeta = 17$.

The HPF obtained for this dataset is described in Figure 2a, where the $x$ - axis represents the intra-clustering measures and the $y$ - axis are the inter-clustering measures. This Hybrid Pareto front composes 37 solutions provided by 12 different pairs of measures, as detailed in Table 2.

As can be seen in Table 2, in terms of cluster number, there are 4 different possibilities, it is 6, 7, 8, or 9 groups. However, there are 37 different optimal ways to perform this division, as suggested by the MCA algorithm and represented in the Hybrid Pareto front illustrated in Figure 2a.

Unlike the Mydata dataset, where elements from one cluster to another are very close, in the Spheres6 set, we observe a clear separation between distinct groups. In these cases, it is clear that the MCA manages to obtain a predominance of these solutions. In this case, 20 of the 37 solutions have 6 clusters.

Figure 2b displays one of these solutions, generated by the $FNC - Acc$ pair of measures, considered one of the most appropriate by the benchmark, with 6 clusters. On the other hand, the solution in Figure 2c presents a configuration
with 9 clusters, generated by the measures $SAxc - FNcc$. Regarding the value of the Dunn index, the values 0.515 and 0.018 were found, respectively, with 0.515 being the highest possible among the HPF solutions. Thus, as already mentioned, it may be interesting in certain contexts, to partition a given set into more subsets. Thus, a solution like the one presented in Figure 2c, even though it is not the best for the CVI, may be attractive for the problem.

### 3.3 Spheres4 dataset results

Spheres4 dataset, is a tri-dimensional dataset [15]. This dataset involves 400 elements, being 4 the optimal number of centroids [15]. Figure 3a presents the HPF obtained for the Spheres4 dataset. This HPF comprises 45 solutions provided by 12 different pairs of measures, as detailed in Table 3.

Like the Sphere6 set, the Sphere4 presents a clear separation between the groups, resulting in a predominance of solutions with 4 clusters (18 out of 45). Figure 3b represents one of these solutions with 4 clusters (generated by the measures combinations $SAxc - Scc$), highlighted as the optimal number in the literature. On the other hand, Figure 3c represents the HPF solution with 8 clusters, also considered optimal by the MCA, that was generated by the measures combination $FNc - Scc$. When calculating the Dunn index, a values of 0.648 was obtained for the solution with 4 clusters, the highest Dunn index found
among the HPF solutions. The value obtained for the solution with 8 clusters was 0.050.

Table 3: Details Spheres4’ Hybrid Pareto front

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SAxc-Scc</td>
<td>2</td>
<td>4 and 9</td>
<td>FNc-Scc</td>
<td>10</td>
<td>4, 5, 6, 7, and 8</td>
</tr>
<tr>
<td>SAxc-Acc</td>
<td>4</td>
<td>4, 5, 6, and 9</td>
<td>FNc-Acc</td>
<td>3</td>
<td>4 and 8</td>
</tr>
<tr>
<td>SAxc-FNcc</td>
<td>1</td>
<td>4</td>
<td>FNc-FNcc</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>SAxc-AFNcc</td>
<td>2</td>
<td>4 and 9</td>
<td>FNc-AFNcc</td>
<td>2</td>
<td>4 and 8</td>
</tr>
<tr>
<td>SAxc-NNcc</td>
<td>10</td>
<td>4, 5, 6, and 7</td>
<td>FNc-NNcc</td>
<td>7</td>
<td>4, 5, 7, 8, and 9</td>
</tr>
<tr>
<td>SAxc-ANNcc</td>
<td>2</td>
<td>4 and 6</td>
<td>FNc-ANNcc</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

Figure 3b shows that the bottom cluster (green) is the most compact compared to the other three. For this reason, the solution represented in Figure 3c did not partition this cluster into smaller ones, contrary to what happens in the others. Once again, this type of solution can be interesting in real contexts where there is additional information about the data that is not included in the mathematical model.

4 Conclusion

This paper presented a novelty multi-objective clustering algorithm approach. This algorithm provides significant contributions, leveraging a multi-objective strategy and combining diverse solutions to determine the optimal number of cluster sets and their element partitioning. Unlike single-objective algorithms that minimize one measure at a time, the proposed approach overcomes limitations, offering a set of optimal solutions. This variety empowers decision-makers to select the most suitable solution based on their knowledge or preferences.
Hybrid Pareto front enhances diversity and robustness by considering different measures. The MCA stands out for its application of multi-objective optimization, providing a versatile set of optimal solutions. This flexibility proves valuable, especially in addressing challenges where encapsulating certain information within a mathematical model is difficult.

When a single-objective approach is used, only one criterion is considered, and the solution depends exclusively on it, and may not find the global optimum of the problem. Using the multi-objective approach is a significant advantage as it allows the resolution of complex grouping problems and the identification of trade-offs between objectives.

Many single-objective clustering algorithms use a CVI as their objective function. However, there are several CVIs, each suited to a specific purpose and more advisable in certain situations and models. For this reason, the use of multi-objective strategies becomes more interesting, as it allows different criteria to be combined.

Analysis of MCA results on benchmark sets reveals that the algorithm not only manages to find the optimal solution as indicated by the literature but also proposes other solutions that belong to the Pareto front set. These alternatives can be even more interesting for the decision maker, depending on the context in which they are working.

In benchmark datasets, solutions are generally validated and it is relatively easy to identify the optimal solution. However, the main contribution of MCA lies in real cases, in which the optimal solution is not trivial and can vary according to the context and the criteria that the decision-maker wants to fulfill. Therefore, through the results presented, it can be stated that MCA not only guarantees finding the optimal solution described in the literature but goes further by offering other optimal solutions to the decision maker. In this way, MCA contributes to the decision support process, allowing the decision maker to rethink the choice of the most appropriate solution for the problem in question.

Moreover, from the range and variability of the Hybrid Pareto front generated, it is possible to perceive the impact of combining different measures to solve a problem. In this way, if only one pair of measures was considered in the model, the solution would be restricted to the optimum provided by one combination of measures and could be inappropriate for the decision-maker. So, the Hybrid Pareto fronts strategy enriches the model’s final solution. Furthermore, the MCA does not require the prior indication of the cluster number, which is a common complaint in the literature regarding $k$-means and other partitioning clustering algorithms [2].

Since the MCA recognizes that the decision maker possesses vital information essential to addressing the issue effectively, the main advantages of MCA are more evident in real systems in which the decision maker has a greater ability to influence the choice of solution. Regarding future developments, it is intended to investigate split and strategies to improve the quality of solutions, mainly in datasets with complex geometries.
References


A Benchmark for Missing Data Imputation
Techniques: development perspectives and comparative of performance

Juan-Francisco Cabrera-Sánchez1, Carlos Cruz-Corona2, Andrés Yáñez Escolano1, and Esther-Lydia Silva-Ramírez1

1 Department of Computer Science and Engineering, University of Cadiz, Spain
{juanfrancisco.cabrera,andres.yaniez,esther.silva}@uca.es
2 Department of Computer Science and Artificial Intelligence, University of Granada, Spain carloscruz@decsai.ugr.es

Abstract. Knowledge extraction from information stored in databases is always subject to the presence of missing values. Missing data is an unavoidable problem that affects many disciplines of researchers and data scientists. Inasmuch as machine learning algorithms cannot work with incomplete data in the data sets, data imputation is an essential task to obtain quality data. This research approach provides an overview of the data missingness mechanism and the process of generating synthetic missing data, the imputation of all types of variables, and the performance assessment of several imputation methods. Traditional algorithms, Machine Learning methods and various Autoencoder-based deep learning architectures have been studied. An exhaustive analysis and comparison of 21 heterogeneous data sets in various areas has been proposed. They have been exposed to a perturbation procedure with different missingness mechanisms and various missingness rates, covering the different possibilities that can occur in real life. The experimental results show that deep learning models outperform the other methods studied. Furthermore, the performance of data imputation methods does not depend on the missingness mechanism or the synthetic missingness generation method used nor on the percentage of missing values.

Keywords: Missing data · Machine Learning · Deep Learning · Autoencoders

1 Introduction

Nowadays, a large amount of data is continuously being transferred and treated between individuals and entities. However, this data may be incomplete due to gathering errors or errors while being transmitted. This absence of values is called missing data and affects multiple disciplines. The quality of the data is necessary for the Machine Learning (ML) algorithms to work well, allowing the extraction of useful information that helps organizations in the decision-making process.
process. Therefore, tackling missing data throughout a data imputation process is an essential preprocessing step.

Traditionally, statistical methods such as mean or mode, regression, or even listwise deletion have been used. However, mean/mode substitution causes some problems in terms of bias for a given perturbed variable [7]. Therefore, machine learning-based methods and, by extension, deep learning models have also been frequently applied. In particular, Neural Networks-based models have been widely studied [2].

However, some of these works present certain limitations. Important aspects such as a comparison with the three mechanisms, a variety of missingness rates, diverse and heterogeneous data sets. The mechanisms explain the reason why the value is absent: completely at random (MCAR), not missing at random (MNAR) or missing at random (MAR). In most studies, only observed data is used for training, and they do not distinguish between the different types of variables.

Jerez et al. presented a thorough study of the imputation methods for a real breast cancer problem with continuous and categorical features only under the MAR mechanism. The authors showed how machine learning methods outperformed other studied techniques [9]. Also, a benchmark for the different imputation methods was presented in [10]. The three mechanisms were considered for heterogeneous data sets where the presence of 1%, 10%, 30%, and 50% missingness values was distributed. Machine learning-based models, such as k-Nearest Neighbours (k-NN), Random Forests, and Deep Learning approaches ranked over the first positions under MCAR, MNAR and MAR assumptions, demonstrating the superiority of these methods over statistical ones.

Denoising Autoencoders (DAE) was proposed for handling incomplete Electronic Health Records in [3] under MNAR and MCAR mechanisms. Results showed how the DAE model obtained better RMSE than other methods, such as k-NN or SoftImpute. The authors highlighted that computationally complex techniques were gaining popularity due to increased available resources. DAE was also studied for a multiple imputation scheme in [8], which produced better results than Multiple Imputation by Chained Equations (MICE) in about 85% of cases. The authors generated missingness under MCAR and MNAR mechanisms with only a fixed ratio of 20% in the training subset.

DAE and other Neural Network-based models require a previous codification to mark the gap to be filled, the missing values. This fact was approached along a deletion and compensation scheme in [16,17]. Instead of just coding with some constant value, a pre-imputation process estimates the value using some method such as Support Vector Machine (SVM), Decision Trees or Multilayer Perceptron (MLP). The authors generated missingness for data sets with only continuous features and only under MCAR mechanisms assumption for initial missingness rates of 10%, 20% and 30%.

Variational Autoencoder (VAE) is a robust method for tackling missingness under MNAR mechanisms, according to the study in [13]. Heterogeneous data sets with categorical and continuous features were exposed to a perturbation procedure where a range of missingness between 10% and 80% was generated only
under the MNAR mechanism. The authors concluded that their proposal, named Partial Multiple Imputation with VAE (PMIVAE), outperformed other imputation methods in most cases. They observed how Autoencoder-based methods maintained or improved their results as the missingness rate increased.

The present study describes an exhaustive study about the effects that the different missingness mechanisms (MCAR, MAR and MNAR) and rates (10%, 20%, 30%, 40% and 50%) produce in the imputation process, considering all the possible situations that occur in a real-world scenario. The performance of several models is measured: Mean/mode, k-NN, SoftImpute, MICE, DAE, and VAE. The results are analyzed to detect the sensitivity of the models to the perturbation mechanism, this is the process of generating synthetic missing data. This work provides guidelines to data scientists and practitioners in preprocessing tasks to select an appropriate data imputation method and improve the data quality.

A comprehensive set of experiments on 21 heterogeneous data sets of diverse areas is proposed. Data sets containing continuous, categorical, or both types of variables are imputed by deep learning approaches and classical ML-based imputation methods. The results are reviewed from various perspectives, which allows confirming the validity of the methods as useful alternative tools for data imputation. Thus, the imputation quality is evaluated with the degree of similarity between the imputed data set and the original data set through different metrics such as mean absolute error and error rate.

The rest of the paper is organized as follows: In section 2, the methodology used in this work describing aspects such as data sets, preprocessing or missing values generation is presented. The results and the discussion are described in section 3. Finally, in section 4 some conclusions about the obtained results are drawn.

2 Methodology

The present work performs an empirical analysis to impute missing data by applying several techniques from perturbed data sets. In this section, the data sets employed in this study and the design of the experiments carried out are described. In addition, the preprocessing tasks, the perturbation mechanisms, and the evaluation criteria are explained. Also, imputation methods are summarized.

2.1 Data sets

In this approach, the experiments have been conducted on 21 data sets of different sizes and dimensionality extracted from the UCI-Repository [5] or Kaggle [1]. These data sets cover diverse areas (medicine, business,...) and contain different numbers and types of variables. Thus, there are data sets with only quantitative variables, with only qualitative variables, and with both types, hereafter referred to as mixed data sets.

Table 1 describes the characteristics of the data sets used in this analysis. The columns correspond to the following:
Table 1: Data sets used in the experiments

<table>
<thead>
<tr>
<th>Name</th>
<th>Field</th>
<th>n</th>
<th>p</th>
<th>t</th>
<th>q</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abalone</td>
<td>Biology</td>
<td>4177</td>
<td>10</td>
<td>7</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>Cleveland</td>
<td>Medicine</td>
<td>303</td>
<td>25</td>
<td>7</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>Contraceptive</td>
<td>Medicine</td>
<td>1473</td>
<td>24</td>
<td>7</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>Credit</td>
<td>Business</td>
<td>690</td>
<td>51</td>
<td>6</td>
<td>9</td>
<td>-</td>
</tr>
<tr>
<td>Flag</td>
<td>Other</td>
<td>194</td>
<td>77</td>
<td>10</td>
<td>18</td>
<td>-</td>
</tr>
<tr>
<td>SKY</td>
<td>Other</td>
<td>10000</td>
<td>20</td>
<td>15</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>Zoo</td>
<td>Biology</td>
<td>101</td>
<td>31</td>
<td>1</td>
<td>15</td>
<td>-</td>
</tr>
<tr>
<td>Bupa</td>
<td>Medicine</td>
<td>345</td>
<td>6</td>
<td>6</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Image</td>
<td>Other</td>
<td>2310</td>
<td>18</td>
<td>18</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MAGIC</td>
<td>Physics</td>
<td>19020</td>
<td>10</td>
<td>10</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Pima</td>
<td>Medicine</td>
<td>768</td>
<td>8</td>
<td>8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Sonar</td>
<td>Physics</td>
<td>208</td>
<td>60</td>
<td>60</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Twonorm</td>
<td>Mathematics</td>
<td>7400</td>
<td>20</td>
<td>20</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Wine</td>
<td>Chemistry</td>
<td>1484</td>
<td>8</td>
<td>8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Breast</td>
<td>Medicine</td>
<td>683</td>
<td>89</td>
<td>-</td>
<td>9</td>
<td>-</td>
</tr>
<tr>
<td>Car</td>
<td>Other</td>
<td>1728</td>
<td>21</td>
<td>-</td>
<td>6</td>
<td>-</td>
</tr>
<tr>
<td>Led7</td>
<td>Computer Science</td>
<td>1236</td>
<td>14</td>
<td>-</td>
<td>7</td>
<td>-</td>
</tr>
<tr>
<td>Lymphography</td>
<td>Medicine</td>
<td>148</td>
<td>59</td>
<td>-</td>
<td>18</td>
<td>-</td>
</tr>
<tr>
<td>Promoters</td>
<td>Biology</td>
<td>106</td>
<td>228</td>
<td>-</td>
<td>57</td>
<td>-</td>
</tr>
<tr>
<td>Solar</td>
<td>Physics</td>
<td>1389</td>
<td>16</td>
<td>-</td>
<td>7</td>
<td>-</td>
</tr>
<tr>
<td>TicTacToe</td>
<td></td>
<td>958</td>
<td>22</td>
<td>-</td>
<td>9</td>
<td>-</td>
</tr>
</tbody>
</table>

- Name: name of the data set.
- Field: subject area into which the data set is categorized
- n: number of cases.
- p: number of total attributes in machine learning-based models after coding the categorical variables.
- q: number of quantitative variables
- c: number of qualitative variables

The first group represents mixed data sets. The second group contains the data sets with only quantitative attributes. The third group corresponds to data sets with only qualitative variables.

2.2 Data preprocessing

According to the habitual practical applications of ANN [4,18], several preprocessing tasks are necessary to obtain the data sets in a suitable format to be processed by the machine learning models.

The variables with the same value for all individuals do not contribute anything, so they were removed. The quantitative variables were normalized into the range [0,1]. In this manner, for a vector variable $X_j$ the value was computed by:

$$\beta_{ij} = \frac{x_{ij} - x_{i,min}}{x_{i,max} - x_{i,min}}$$

(1)
where the $x_{i,max}$ and $x_{i,min}$ are the maximum and the minimum of $X_j$, respectively.

Categorical features were transformed into binary variables using a 0/1 encoding (one-hot encoding) for each category. Thus, a vector with dichotomous variables was created for each categorical variable, a variable for each class. For a variable $X_j$, with $k$ categories, $k$ binary variables were generated: $E_1, \ldots, E_k$. Each variable $E_i$ takes the value 1 if the variable value $X_j$ is equal to the $i$-th category and 0 otherwise.

Therefore, the total number of inputs to the models is usually larger than the number of variables in the original data set.

2.3 Missing value generation

Missing data can arise by different mechanisms. [11] and [14] define three types of missing data mechanisms:

- **MCAR (Missing Completely at Random).** The probability that the value of an attribute $X_j$ is missing for any instance does not depend on any variable.

  The synthetic missing generation process used in this proposal is performed as described in [20]. Each feature will have missing values based on random positions derived from some distribution such as Bernoulli or binomial. In this case, a Bernoulli distribution is followed. So, for each feature, values are drawn from this distribution, using the missingness rate as the probability of success to decide if an instance is absent or not. To ensure the desired missingness rate, the process is repeated until the number of instances exposed to missingness is equal to the desired rate.

- **MAR (Missing at Random).** The probability that the value of a feature $X_j$ is missing for any case does not depend on the variable itself but depends on the value of other variables.

  In this study, the perturbation process is performed following the strategy proposed in [6]. This method randomly chooses one feature as a causative variable, while the others will be perturbed depending on the values of that feature. In the case of a categorical causative variable, the categories are ranked through their frequencies. So, the indexes of the less common categories are selected to have missingness for the rest of the features. On the other hand, the indexes of the lower values are chosen in the case of a continuous causative variable. Thus, extreme values are used to cause the missingness in the other features. This method requires adjusting the missing rate between $p−1$ features, as one will not have any missing value.

- **MNAR (Missing Not at Random).** The probability that the value of a variable $X_j$ is missing for any record depends on the value of this variable itself, being this value unknown.

  In this approach, generation is carried out through the method proposed in [21]. Each feature is divided into two equally sized sets. For continuous features, the split is accomplished through the median. For categorical variables, random sampling for each group is performed. After that, one of the
two splits is randomly chosen to have missing values according to the desired missing rate. However, this method cannot allow missing rates above 50%.

In most studies, the response mechanism is not specified, so data are assumed to be MCAR. However, we have wanted to study how sensitive the models are to the missing data mechanisms. Thus, experiments to evaluate the behaviour of the models have been carried out on distorted data according to these three perturbation schemes. All databases have been subjected to errors following the MCAR, MAR, and MNAR missingness mechanisms.

In this study, a non-monotone pattern was followed, where the lack of response occurred for any record and variable.

2.4 Performance evaluation criteria

Comparison based on statistical indexes was carried out between the observed and estimated values by the previously mentioned models. Indexes to measure how close predictions are to outcomes, such as the widely used scoring rule Mean Absolute Error (MAE) and Error Rate (ER), were implemented as:

\[ MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \]

where \( y_i \) is the observed value and \( \hat{y}_i \) the predicted value for the case \( i \)th.

\[ ER = \frac{1}{n} \sum_{i=1}^{n} I(\hat{y}_i \neq y_i) \]

being \( I(c) \) equal to 1 if \( c \) is true and 0 if \( c \) is false.

MAE is only measured on continuous features, whereas the error rate is applied to categorical ones. In the case of having both types of variables, both metrics are computed.

2.5 Missing data imputation methods

The proposed deep learning-based techniques and the traditional models used in the comparison are briefly described in this section.

The models employed in this study can be classified as attending to two types [7]: statistical methods and machine learning-based methods, which include deep learning methods. Some of them are considered state-of-the-art methods [12]. The models applied in this study are described as follows:

Mean/mode substitution. This method is simple and easy to use. The missing value is replaced with the mean for continuous features and the mode for categorical ones. However, it has the disadvantage of ignoring correlations between variables [19].
**k-Nearest Neighbours.** It replaces the missing value by estimating the $k$ closest instances according to a distance or similarity metric such as Euclidean distance. Thus, the mean or the most frequent category is computed considering $k$ neighbours to replace the missing value. This calculus can be computationally expensive if the data set has a large number of instances.

**SoftImpute.** The data set is considered a matrix in which missing values are replaced iteratively with a soft-thresholded Single Value Decomposition (SVD) using nuclear norm normalization. Thus, a unique single imputation is achieved through an approach of expectation maximization.

**Multiple Imputation Chained Equation.** Based on the multiple imputation approach, several regression models are applied consecutively to fill features with missing values. Once completed, each of them will be used as an additional independent variable for the remaining features. This process is repeated a certain amount of times to produce a unique data set.

**Denoising Autoencoders.** This model is usually associated with computer vision problems, due to its capacity to remove noise, but its presence in missing data problems has increased in recent years. In this context, training is carried out on a corrupted version of the data set with the objective of reconstructing the original data set. This type of neural network tries to reconstruct the input by encoding it to a lower dimensional representation called latent space. Then using it as an input, the information is decoded towards the output layer, increasing the number of nodes in each step. Over-complete approaches where latent space possesses higher dimensionality are also feasible [8].

**Variational Autoencoders.** Similarly to DAE, it is capable of reproducing its input to the output by mapping its input to a latent space and then decoding it. Although several distributions can be used, the normal distribution is the most widespread. In this manner, it is possible to draw samples from the latent space layer of VAE that follows the same distribution. Thus, it has generative capabilities as new instances are created as similar to training data as possible. The main differences with DAE are the probabilistic approach of VAE and how the network is trained. While DAE focuses on reconstructing, VAE learns the latent space to be able to generate new data.

### 2.6 Empirical experiments

As aforementioned, the missing data imputation has been carried out with deep learning-based models, statistical models, and state-of-the-art methods. The deep architecture, with the same number of neurons in the input layer and the output layer, allows all variables to be input variables and, at the same time, output variables. In order to deal with the missing data, an encoding of the missing value to 0 or $-1$ could be established, though a pre-imputation approach offers better performance [16]. In this approach, missing values were pre-imputed with the mean of the values for each feature.

The data set $T$, assumed as a complete set, is represented as a $n \times p$ matrix, being $p$ the attributes measured for each of the $n$ instances. These data were
subjected to the perturbation process detailed above, so the data set $T_d$ was obtained, where certain cells were missed. We proposed a procedure to identify an approximation $T^*$ to $T$ training the models with the pair $(T_d, T)$. $T^*$, obtained by models, is supposed to be a corrected and improved version of $T_d$, but the desired matrix $T$ is not guaranteed to be achieved.

Each correct data set $T$ and its associated perturbed version $T_d$ were randomly split into two separate subsets, a training subset (70%) and a test (30%) subset. Thus, two subsets, $T^1$ and $T^2$, were obtained with associated perturbed sets, $T_{d1}$ and $T_{d2}$, respectively. The rows of $T_{d1}$ defined the inputs to the models, while the rows of $T^1$ were the target records. This partitioning was randomly carried out 30 times to take into account the existent inherent variability in the random split. In each one of these splits, it was ensured that the train and test subsets preserved the same proportion of missing values. The rows of $T_{d2}$ were fed to the models, and the obtained output $T^{*2}$ was compared with the true records contained in $T^2$. Figure 1 displays the whole procedure for the fixed data set $T$.

![Diagram](image)

**Fig. 1:** Strategy used in the process of imputation for each data set $T$

As is known, in any application, an appropriate configuration of the model’s hyperparameters is necessary. Thus, ad-hoc experiments were designed to determine the influence of the particular hyperparameters on the resulting performance. Experiments to make decisions regarding the learning algorithm, the random initialization of the weights, the number of hidden units or the number of iterations (epochs) of the learning algorithms were conducted. Table 2 shows the values of the hyperparameters resulting from these experiments.

Euclidean distance and $k = 5$ were selected for $k$-NN model. MICE setup was based on Scikit-Learn library configuration as a similar approach to Miss Forest with 10 iterations since can handle categorical features. DAE and VAE were implemented with Pytorch Lightning library with two hidden layers in both the encoder and decoder. The number of hidden nodes is based on the approach of [15,16], with an increase factor $\theta$ of 1.75. So, given a layer $h_n$ it will have $\theta \cdot h_{n-1}$ nodes. For the training process, AdamW optimiser with a learning rate of 0.01 and a weight decay of 0.01 was chosen. Also, an Early Stopping procedure was implemented. Thus, if there are 10 epochs with no lower losses for the validation set, the training process is finished.
Table 2: Models configuration

<table>
<thead>
<tr>
<th>Model Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>k-NN</td>
<td>k</td>
</tr>
<tr>
<td>Distance</td>
<td>Euclidean</td>
</tr>
<tr>
<td>MICE</td>
<td>Estimator</td>
</tr>
<tr>
<td></td>
<td>Random Forest</td>
</tr>
<tr>
<td></td>
<td># Iterations</td>
</tr>
<tr>
<td>AE</td>
<td>Structure</td>
</tr>
<tr>
<td></td>
<td>2 layers</td>
</tr>
<tr>
<td></td>
<td>Hidden nodes</td>
</tr>
<tr>
<td></td>
<td>$\theta \cdot h_{n-1}$</td>
</tr>
<tr>
<td></td>
<td>Latent space</td>
</tr>
<tr>
<td></td>
<td>$\theta^2 \cdot p_t$</td>
</tr>
<tr>
<td></td>
<td>Optimizer</td>
</tr>
<tr>
<td></td>
<td>AdamW</td>
</tr>
<tr>
<td></td>
<td>Learning rate</td>
</tr>
<tr>
<td></td>
<td>Regularization</td>
</tr>
</tbody>
</table>

The described imputation process was performed for different missing rates aforementioned. The percentage range of missing values varied from 10% to 50%, increasing by 10 in 10.

3 Results and discussion

The results obtained from evaluating the data imputation models are shown in this section. A thorough analysis is carried out to identify how sensitive the models are to the error generation mechanisms. Therefore, methods under MCAR, MAR and MNAR mechanisms show insights about the behaviour of the imputation methods in different situations, such as large or small data sets and with heterogeneous types of variables. Thus, the results are presented with the mean and the standard deviation considering different scenarios. In each one of them, the best method is highlighted.

The results for MAE in data sets with continuous variables are presented in Table 3. For the MCAR mechanism, there is a slight worsening in the results starting at 40% for some models. VAE model presents an inferior performance to the rest of the compared models, except at higher rates, that surpasses k-NN and SoftImpute. On the other hand, DAE model achieves better results, especially at higher missingness rates. $k$-NN and SoftImpute obtain similar results at 40% and 50%, whereas $k$-NN clearly outperforms it at lower rates. Mean/Mode presents the worst performance in all scenarios. MICE method yields the best results for the MCAR mechanism in all missingness rates scenarios.

In MAR mechanism Mean/Mode, some models as $k$-NN and MICE present worse results as the missingness rates increase until 40%. These methods reduce the error at 50%. This behaviour is also observed for SoftImpute, VAE and DAE in which the improvement at 50% is even more significant. However, it can be noticed how SoftImpute achieves the worst results considering that it is also less stable given its standard deviation. Autoencoder-based methods obtain the best results, especially in higher missingness rates.

In MNAR mechanism, it can be observed at 10% of missingness MICE and $k$-NN achieved the best results. However, there is a degradation for these methods and SoftImpute, especially on higher rates such as 40% and 50%. On the other hand, Autoencoder-based methods are more stable to this
increase. Furthermore, VAE and DAE present robust behaviour except for the 50% of missingness, where there is a minimal increase. Mean/Mode obtained the worst results for the quantitative variables in all missingness rates.

<table>
<thead>
<tr>
<th>Mechanism</th>
<th>%</th>
<th>Mean/Mode</th>
<th>k-NN</th>
<th>SoftImpute</th>
<th>MICE</th>
<th>VAE</th>
<th>DAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCAR</td>
<td>10</td>
<td>0.145 ± 0.06</td>
<td>0.104 ± 0.06</td>
<td>0.112 ± 0.05</td>
<td><strong>0.104 ± 0.06</strong></td>
<td>0.126 ± 0.06</td>
<td>0.114 ± 0.06</td>
</tr>
<tr>
<td>MAR</td>
<td>20</td>
<td>0.139 ± 0.04</td>
<td>0.106 ± 0.05</td>
<td>0.114 ± 0.04</td>
<td><strong>0.101 ± 0.05</strong></td>
<td>0.123 ± 0.04</td>
<td>0.112 ± 0.04</td>
</tr>
<tr>
<td>MNAR</td>
<td>30</td>
<td>0.138 ± 0.04</td>
<td>0.109 ± 0.04</td>
<td>0.113 ± 0.03</td>
<td><strong>0.102 ± 0.05</strong></td>
<td>0.120 ± 0.04</td>
<td>0.111 ± 0.04</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>0.144 ± 0.05</td>
<td>0.126 ± 0.05</td>
<td>0.129 ± 0.05</td>
<td><strong>0.112 ± 0.05</strong></td>
<td>0.128 ± 0.04</td>
<td>0.118 ± 0.05</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.140 ± 0.04</td>
<td>0.129 ± 0.03</td>
<td>0.131 ± 0.04</td>
<td><strong>0.115 ± 0.04</strong></td>
<td>0.128 ± 0.04</td>
<td>0.118 ± 0.04</td>
</tr>
</tbody>
</table>

The results of the Error Rate for the categorical features are presented in Table 4. Under MCAR mechanism, a consistent deterioration in error rate with the increase of missingness rates is observed. Moreover, the difference between the methods also narrows as the missingness rates escalate. SoftImpute and k-NN exhibit the best results at 10%, whereas VAE, DAE and MICE present a relatively inferior performance. This situation changes at 20% where SoftImpute outperforms the other methods. The results for 30% are very similar, except for Mean/Mode, which presents the worst results in all scenarios. For the 40% and 50% missingness rates, DAE yields the best results.

Under the MAR mechanism, the comparison is straightforward. There are two groups visible with VAE and DAE and the other compared methods. The last group presents worse performance for all missingness rates. Also, in this case, Mean/Mode outperform k-NN and SoftImpute, achieving a similar error rate as MICE starting from 30%. It can be noticed that MICE ranks first in this group for all missingness rates. DAE and VAE achieve a significant improvement over the other group. Specifically, DAE presents slightly better results than VAE.

Regarding MNAR mechanism, Mean/Mode scores the worst results of all compared methods in all cases. k-NN, SoftImpute, and MICE achieve superior performance than Autoencoder-based methods at 10% and 20%. However, from 30% onwards the situation changes, inasmuch as DAE outperforms the other methods. On the other hand, VAE achieves worse results than k-NN, SoftImpute, and MICE, except at 50%, k-NN presents a pronounced deterioration with the increase of missingness rate. MICE produce consistently better performance than k-NN and SoftImpute in all comparisons.

Furthermore, it is interesting to compare not only how well they perform the imputation but also how much time they need for the imputation process. Thus, the execution time of the models was measured. Figure 2 shows the running times of all data sets grouped by mechanism and imputation method. It can be
Table 4: Error rate for data sets with categorical features

<table>
<thead>
<tr>
<th>Mechanism</th>
<th>%</th>
<th>Mean/Mode</th>
<th>k-NN</th>
<th>SoftImpute</th>
<th>MICE</th>
<th>VAE</th>
<th>DAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCAR</td>
<td>10</td>
<td>0.477 ± 0.18</td>
<td>0.397 ± 0.18</td>
<td>0.402 ± 0.19</td>
<td>0.416 ± 0.19</td>
<td>0.416 ± 0.19</td>
<td>0.433 ± 0.17</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.471 ± 0.18</td>
<td>0.411 ± 0.17</td>
<td>0.414 ± 0.19</td>
<td>0.414 ± 0.18</td>
<td>0.429 ± 0.17</td>
<td>0.418 ± 0.17</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>0.474 ± 0.18</td>
<td>0.432 ± 0.16</td>
<td>0.426 ± 0.19</td>
<td>0.429 ± 0.18</td>
<td>0.437 ± 0.17</td>
<td>0.429 ± 0.17</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>0.477 ± 0.17</td>
<td>0.446 ± 0.16</td>
<td>0.439 ± 0.18</td>
<td>0.438 ± 0.16</td>
<td>0.444 ± 0.17</td>
<td>0.437 ± 0.16</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.477 ± 0.17</td>
<td>0.454 ± 0.16</td>
<td>0.448 ± 0.18</td>
<td>0.447 ± 0.16</td>
<td>0.450 ± 0.17</td>
<td>0.446 ± 0.16</td>
</tr>
<tr>
<td>MAR</td>
<td>10</td>
<td>0.564 ± 0.19</td>
<td>0.561 ± 0.20</td>
<td>0.546 ± 0.19</td>
<td>0.522 ± 0.19</td>
<td>0.498 ± 0.17</td>
<td>0.486 ± 0.20</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.536 ± 0.16</td>
<td>0.525 ± 0.20</td>
<td>0.527 ± 0.20</td>
<td>0.512 ± 0.16</td>
<td>0.437 ± 0.18</td>
<td>0.432 ± 0.19</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>0.509 ± 0.20</td>
<td>0.530 ± 0.19</td>
<td>0.514 ± 0.19</td>
<td>0.508 ± 0.19</td>
<td>0.476 ± 0.19</td>
<td>0.473 ± 0.20</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>0.524 ± 0.19</td>
<td>0.540 ± 0.18</td>
<td>0.532 ± 0.18</td>
<td>0.522 ± 0.19</td>
<td>0.485 ± 0.19</td>
<td>0.487 ± 0.20</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.493 ± 0.19</td>
<td>0.506 ± 0.17</td>
<td>0.497 ± 0.18</td>
<td>0.493 ± 0.19</td>
<td>0.456 ± 0.20</td>
<td>0.450 ± 0.19</td>
</tr>
<tr>
<td>MNAR</td>
<td>10</td>
<td>0.468 ± 0.18</td>
<td>0.405 ± 0.17</td>
<td><strong>0.403 ± 0.19</strong></td>
<td>0.407 ± 0.19</td>
<td>0.435 ± 0.17</td>
<td>0.424 ± 0.17</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.471 ± 0.18</td>
<td>0.415 ± 0.16</td>
<td>0.412 ± 0.19</td>
<td><strong>0.410 ± 0.18</strong></td>
<td>0.430 ± 0.17</td>
<td>0.419 ± 0.16</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>0.473 ± 0.17</td>
<td>0.427 ± 0.16</td>
<td>0.423 ± 0.19</td>
<td>0.422 ± 0.17</td>
<td>0.432 ± 0.17</td>
<td><strong>0.421 ± 0.16</strong></td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>0.474 ± 0.17</td>
<td>0.448 ± 0.17</td>
<td>0.435 ± 0.18</td>
<td>0.427 ± 0.17</td>
<td>0.437 ± 0.17</td>
<td><strong>0.422 ± 0.16</strong></td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.478 ± 0.17</td>
<td>0.463 ± 0.17</td>
<td>0.448 ± 0.18</td>
<td>0.435 ± 0.16</td>
<td>0.449 ± 0.17</td>
<td><strong>0.422 ± 0.15</strong></td>
</tr>
</tbody>
</table>

observed that the most time-demanding methods are MICE and Autoencoder-based methods, being statistical methods less time-demanding. In this Letter-Value boxplot, the width of each box represents the number of data points within a quantile. Thus, wider boxes indicate that there is a higher concentration of values i.e. more

![Time vs Mechanism](image)

Fig. 2: Running times for each imputation method

Although statistical methods such as mean/mode take less time, they produce the worst results in almost all situations. Therefore, they can be disregarded in favour of k-NN. This method performs relatively well in nearly all cases. However, it does not achieve as good results as deep learning models. Autoencoder-based models have obtained the best results in two of the three studied mechanisms, MAR and MNAR. For MCAR they achieved similar results as MICE and k – NN, particularly for data sets with categorical features.

4 Conclusions

This work presents a benchmark between several imputation methods and the impact of the missingness mechanisms and rates on them. Deep learning ap-
proaches, such as Autoencoder-based models, are compared to several state-of-the-art methods using 21 heterogeneous data sets. Therefore, data sets with only continuous variables, data sets with only categorical and also mixed data sets are exposed to a perturbation procedure to generate missingness under MCAR, MAR and MNAR mechanisms with missing rates from 10% to 50% increasing by 10 at each step. These incomplete data sets are split into training and test sets where missingness is equally distributed. Each proposed method is trained and subsequently applied to predict on the test set evaluating the results.

The Autoencoder-based models are well-fitted to deal with different missing data mechanisms, which can be a bottleneck for the performance of other imputation methods. The experimental results show that DAE and VAE models outperform the other studied methods under varying data sets and missingness generation processes, particularly under MAR and MNAR mechanisms. This performance is not equivalent under MCAR mechanism, where MICE outperforms the other compared methods. Thus, the experimental results are aligned with the conclusions extracted from previous works about the superiority of MICE and Autoencoder-based methods.

In future works, other approaches for the perturbation procedure in each mechanism can be studied with higher missingness rates. Furthermore, an analysis of the impact that the missingness generation schemes could present on specific domains or other variants of deep learning models will be studied.

Acknowledgments. This publication is part of project eMob (PID2022-137858OB-I00) funded by MCIN/AEI/10.13039/501100011033 and the European Union “NextGenerationEU”/PRTR. This work is supported by the grant PID2022-137646OB-C33 funded by CIN/AEI/10.13039/501100011033, and by ERDF A way of making Europe. Also, this work is partially funded by PLAN PROPIO UCA 2022-2023.

Disclosure of Interests. The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Selecting the Most Relevant Objectives Using Data Minning: A Real-World Case Study

António Gaspar-Cunha\textsuperscript{1}, Wagner Galuppo\textsuperscript{2}, Miguel Nóbrega\textsuperscript{2}, Francisco Monaco\textsuperscript{3}, and Alexandre Delbem\textsuperscript{3}

\textsuperscript{1}IPC - Instituto de Polímeros e Compósitos, Universidade de Minho, Guimarães, Portugal – Portugal
\textsuperscript{2}IPC-Institute for Polymers and Composites, University of Minho, Guimarães, – Portugal
\textsuperscript{3}Institute of Mathematics and Computer Science, University of São Paulo – Brazil

Abstract

Often in real-world optimization problems is difficult to define the objectives that are important to effectively optimize the process. In the phase of problem definition, the decision maker, who knows the process, is faced with many aims or objectives, but he/she does not know if they are relevant and will help in optimizing the problem. This work proposes a method based on data mining to identify the relations between the data, decision variables and objectives, which enables the determination of the redundant objectives and the ones that are less important to the optimization. This method, together with a multi-objective evolutionary algorithm is used to optimize a problem with a previous definition of twenty-one objectives allowing to identify the set of relevant objectives. The results obtained allow to conclude about the effectiveness of the approach proposed.
On The Organizational Network Analysis

Mohammed LALOU¹ and Hamamache KHEDDOUCI²

¹ Université de Bourgogne, Laboratoire LIB, 21000 Dijon, France, mohammed.lalou@u-bourgogne.fr
² Université Lyon 1, Laboratoire LIRIS, 69100, Villeurbanne, France, hamamache.kheddouci@univ-lyon1.fr

Abstract. To improve working conditions, increase productivity, and optimize performance, organizations, whatever their activity area, are more and more interested in studying, understanding, and taking advantage of their informal networks existing between employees through an Organizational Network Analysis (ONA). ONA is the process that leverages data from informal relationships between different actors within the organization to fit these objectives. It consists of analyzing how the inner work is actually performed and then determining the most appropriate actions and measures to take for performance purposes. The main goal of this paper is to discuss the general framework of ONA through reviewing, analyzing, and classifying different recent works on the issue. Also, to highlight the relationships among different ONA objectives considered in the literature and identify and discuss different research gaps worthy of being explored.

Mots-Clefs. network analysis, graph metrics, centrality, SNA, CNA.

1 Introduction

The human factor in an organization of any nature whatsoever, a healthcare system, a commercial enterprise, an industrial factory, or even a school or a small sports club, is probably the most important driver and key to success (or failure). The human value within the organization is assessed by two main factors, generally referred to as human capital and social capital [9]. The first one depends on what the human actor has as skills, aptitudes, and experience to accomplish his mission within the system, while the second one, namely the social capital, consists of his relationships, collaborations, and links with the other actors within the system.

For many years ago, human resources management considered only the human capital and used for management the classical people analytics methods based on human skills³. However, considering the human relational aspect can lead to entirely different, and even opposite, results, which is strongly observed in many life fields. For instance, in nature, oxygen, when considered independently, has a combustion effect on the fire, as well as hydrogen, which is highly flammable. However, considering the relationships between the two, we get a new molecule, water, that has absolutely the opposite effect; it puts out fire. Thus, the capital social, i.e., human relationships, has a much broader impact on the added value for the organization and even the dominant effect on system productivity compared to the human capital [4]. This is usually proved by the case of two employees, a highly qualified one with many skills but not good at communicating with others, and another who is not qualified enough but with good abilities to build relationships with others, so many studies showed that the company benefits from the second more than the first one. This has also been confirmed long before through the idiomatic saying “It is not what you know, it is whom you know that matters most.”

Therefore, studying the employee’s relationships within the organization is very important to maximize its potential. These relationships form a social network between employees, like relationships among individuals in a society. The Organizational Network Analysis (ONA) consists of studying this network of relationships, understanding patterns, and deriving insights to improve how the organization works. In other words, it involves leveraging data from employees’ relationships to get a holistic view of the inner work of the organization and then improve organization performance and efficiency, based on the obtained view, through enforcing collaboration [5, 14, 15, 21, 31], promoting innovation [3, 35, 36], ensuring interconnectedness [7, 10, 19], etc. Compared to

³ https://www.linkedin.com/pulse/history-people-analytics-five-ages-david-green/
the classical people analytics methods, which represent the static aspect of managing human resources, ONA represents the dynamic aspect as it helps analyze how communications, decisions, and information flow internally. Indeed, rather than evaluate the employee’s value based only on his skills, we also assess the influence of transferring these skills on the organization’s performance.

Many studies proved the usefulness of ONA for tracking the organization’s evolution and providing precious information for future decisions. It is a powerful means that makes different patterns of connections and information flow within the organization visible and allows for their display and visualization using graphs. This helps identify the strong and weak connection points, hence taking the necessary measures to improve the current situation and elaborating on the appropriate plan to maintain the ongoing improvement. Thus, each organization expresses the ONA importance from its perspective and according to its appropriate needs and interests. However, its importance can be generally summarized in the following.

- **A comprehensive and holistic view of the organization’s work.** We can visualize and know how work actually gets done within the organization and how communications, information, and decisions flow through it. That allows understanding of the interconnectedness between all organization entities, how employees interact and communicate, how knowledge is transferred, how projects are collaborated on, etc., and thus identifies the strength of relationships and effectiveness with both internal and external collaborators. Accordingly, we will have an assessment of the current state, based on which we can take the necessary measures.

- **Data-driven-decisions.** Deriving patterns and insights needed for developing appropriate solutions, not only for the current situation but also for the future, where we can predict and avoid obstacles and enhance success factors. Several issues can be concerned, including talent management and levers identifying leading to greater success and optimal outcomes, silos determination for a possible reintegration, collaboration level assessment for promoting innovation, etc.

The main goal of this paper is to discuss the general framework of ONA through reviewing, analyzing, and classifying different recent works on the issue. Also, to highlight the relationships among different ONA objectives considered in the literature and identify and discuss different research gaps worthy of being explored.

### 1.1 ONA, SNA and CNA

Complex networks are used to model real-world systems, representing elements and their interactions using graphs, and CNA, for Complex Network Analysis, is the study of complex networks — their structure, properties, and dynamics. Note that “complex” refers to any collection of interrelated things [34]. Complex networks can be found in diverse areas such as computer science, sociology, chemistry, biology, geography, etc. For the latter, studying social networks has attracted significant attention for a long and is known as Social Network Analysis (SNA) [6]. More specifically, it evaluates and measures the relationships among actors in a social network. If the network concerns social relationships within an organization, it is known as Organization Network Analysis (ONA). Thus, ONA is a study of relationship patterns between actors sharing common interests, may be personal (e.g., Sports) or professional (e.g. common employer), and where actors can be individuals [11, 16, 22, 29], groups within organization [3, 13, 10], or organizations themselves [14, 15].

Given that SNA has already long been studied [6], ONA uses all what SNA has provided as methods, techniques, results, etc. [13, 15, 27, 39]. Accordingly, ONA can be seen as the application of SNA on organizations [7], and thereby, it is a particular case of SNA. However, ONA potential can go beyond social relationships analysis. In fact, if we consider all organizational entities, not only employees, ONA becomes a generalization of SNA [8]. Indeed, an employee, as the main actor within the organization, usually performs a knowledge to carry out a task using some resources. Thus, one can map relationships between employees, resources, knowledge, and tasks used to perform work. Hence, we are modeling not only the social system but also the technical system of the organization [7, 8], which is more complex. Examples of questions tackled by analyzing both social and technical networks may be whether or not the employee has access to the knowledge he needs to accomplish his mission, or where resources are inadequate for employees to perform their work, etc. Note that studying the socio-technical network ensures a multimodal analysis approach to the organization.
1.2 Formal and Informal networks

We generally distinguish two kinds of networks in an organization, namely Formal network and Informal network [33]. The first one represents the pre-established relationships that should be between employees according to the organization chart, while the second one represents the ties forged between employees by collaboration, acquaintance, and friendships. ONA is particularly interested in studying the informal network, which forms the social relationships between people within the organization [5]. The informal network can include relations of formal network, but this is not always the case. Unlike a formal network, an informal network is self-organizing. It stays the same even when a formal one changes⁴, which makes it a source of stability for the organization. Also, it is hard to perceive and see from the outside, but it is essential and forms the backbone of the organization [10].

Analyzing informal networks helps understand how communications, information, and decisions actually flow through the organization, as almost all work occurs through them rather than connections prescribed by formal structures. For instance, the informal network can stand out that an employee plays an essential role within the organization, which is not prescribed in the formal network, or conversely, that a leader is not as involved in decision-making as it should be. Also, informal relationships usually differ from formal and hierarchical ones, showing the groups tightly connected and more collaborative. This is essential for promoting innovation, for example, [3], and is not recognizable in formal networks.

2 Organization Network Analysis process

An ONA project can be conducted according to two main approaches, namely Reactive and Proactive approach. In the first one, the study is usually triggered to deal with an existing problem, understand it, and develop appropriate solutions. In this case, the objective is usually set beforehand, and we can decide, from the outset, the analysis techniques, the metrics, and the tools to be used. The second one is when we generally create and study a use case while making a general analysis with the goal of identifying possible troubleshooting and/or predicting a concern likely to occur in the ongoing evolution of the organization [8, 11, 12, 15].

The ONA process goes through several stages, from collecting data to interpreting analysis results. From all conducted studies in the literature, we can summarize the ONA process in the following five steps (two of which are optional).

Data collection. The first step is to gather information and data about informal connections within the organization. This can be either by asking people specific and carefully formulated questions, known as the active method [3, 5, 7, 8, 15], or by retrieving needed information from data stored in log files of any communication platform, such as email, chat, etc. [11, 16, 17, 38], as well as administrative archive [12, 14]. This method is known as the passive method. It has become possible through technological progress and the computerization of the work environment. In contrast, the active method is considered classical and has been used for a long time, especially in SNA [6]. Each method has its advantages and limitations. The active approach allows getting data about the connections and nature of the relationship, even through a qualitative face-to-face interview with employees [33]. However, it depends critically on the response rate, which must be of a high rate to be relied upon [8, 10]. People usually abstain from giving answers, more severely, if confidentiality is not guaranteed. Also, it needs laborious work to select pertinent questions [8, 15]. A nice study that discusses and overviews the main factors crucial to collecting survey-based data is available in [32]. For the passive method, there is no problem with data availability. However, they may be context-less and require pre-processing [16, 17]. Combining both methods is also possible and even likely necessary to take advantage of both methods’ benefits [3, 10]. Attention must be paid to the data collection period, which may impact the reliability of the data. This is the case of academic organizations (Universities, schools, etc.) where gathering data during the holidays differs from school days [10].

Network uncovering and visualization. Collected data, after pre-processing, are woven and stitched together to uncover the relationship network, which can be with or without graphical visualization.

⁴ https://www.robcross.org/what-is-organizational-network-analysis/
Visualizing data, displayed as a graph or sociogram, using powerful tools while monitoring several parameters allows many important analyses and insights to be easily derived [3, 5, 11, 14], such as identifying isolated and disconnected actors, determining communities, finding cohesive groups, etc. For instance, in Figure 1 the size (or color, shape, etc.) of the circle, representing the employee, is used to highlight the employee’s centrality and influence within the organization according, for example, to the number of connections, represented by links. For more information about graphical representation, refer to [18]. Also, one can either deal with all informal relationships, i.e. professional, social, administrative, etc., together and display them in only one network, or use different informal networks, each one for a specific category of relationships [8, 10, 27]. Note that the studies without graphical visualization usually utilize meta-matrices and vector computing [8, 12, 16, 15].

**Fig. 1.** Visualizing graphical representation of informal organizational network

*Analysis and measurement.* The different network properties are assessed and measured using metrics mainly used in SNA. Like any social network, organizational networks are analyzed with a 3-level analysis approach, according to three levels of abstraction. The high level deals with the whole network, the middle level concerns the community, and the trim level considers the node. Several metrics already used for analyzing social networks [6, 18] are reused in ONA, each for a specific purpose. For instance, we find the *average shortest path length* is mainly used to assess how connectivity is between two employees [10], and how quickly knowledge flows through the organization [7]. The *redundancy* is generally used to determine the number of employees who have access to the same resources, do the same task, or need the same knowledge [8]. The *Clustering coefficient* measures the extent to which the nodes clump up and group together in the network. In [27] and [10], it was used to assess the grouping of employees in the organization. The *Centrality* with its different variants generally indicate the number of connections an employee has or how likely he is to receive any flow through the network. In [7], degree centrality is used to identify influential employees, while in [15], decay centrality is used to assess the employee’s accessibility to knowledge, i.e. the ease with which an employee can communicate and exchange, transmit and receive information.

*Table 1* reviews the main metrics considered for ONA studies, sorted by abstraction level.

**Table 1.** Metrics for analyzing and measuring organizational network properties according to three abstraction levels: network, group and individual.

<table>
<thead>
<tr>
<th>Node (individual or employee)</th>
<th>Community (group or team)</th>
<th>Whole network (organization)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decay centrality [15], in-out degree [12], Congruence [8], PageRank [10, 23]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Results interpretation and Feedback. The critical step of interpreting results and deriving insights comes once the informal network is analyzed and patterns are defined. We analyze and interpret insights deduced from the evaluation obtained by metrics, based on which plans and measures to be taken are developed. It is essential to correctly interpret the results to propose adequate solutions. For doing so, interpretation has to accurately distinguish patterns from random noise, which can be difficult mainly due to the lack of necessary information about the organization [7]. Thereby, collaboration with the organization’s staff is strongly recommended for a meaningful analysis.

After applying the measures released from the interpretation, and even with solutions expected to be the most adequate, performance may not improve. This may be due to an error in deriving insights, analysis interpretation, or other possible reasons. For that, feedback may be needed to assess and monitor the proposed solution. Generally, feedback data are collected using the passive method, as it is the most suitable method for measuring changes in network behaviors after an initiative.

3 ONA key issues and applications

ONA has been used for studying all types of organizations, ranging from small project teams [21] to very large networks of management organizations [15, 17], and going through organizations with particular properties such as healthcare systems [19, 29]. In the literature, we find many studies carried out on industry companies [3, 5], service business and commercialization organizations [16, 17, 39], management companies [13, 15], service delivery agencies [5, 11], and sport [22]. Healthcare [7, 29, 19] and academic organization; universities [10], schools [14], libraries [8], etc. are considered for a socio-technical study, where entities task, resources and knowledge, besides employees, are taken into account.

Also, ONA has been used to deal with different organizational issues, including assessing interconnectedness among entities [7, 10, 19], evaluating collaboration [15, 14, 5, 21, 31], determining influencers [11, 17, 16], promoting innovation [3, 35, 36], studying job mobility and governance [12, 38, 37]. All the issues are interrelated and generally refer to the study of connectivity within the organization. In what follows, we discuss each issue dealt with by ONA and present recent related works. We try to exhibit, for each one, the relationship with the others.

Interconnectedness. System productivity is generally related to how the necessary tasks are completed, which depends on the ability to access the needed knowledge to achieve the task. This is ensured by a good flow between employees to exchange this knowledge. Krebs [30] proved that communication extent can be assessed by path length (connectivity). Accordingly, all ONA studies, considered interconnectedness issues, are based on the idea that the more connectivity among employees, the better the flow and the better the performance. In [10], authors studied communication between both academic and administrative staff to assess how well information is exchanged within the university. Similarly, for public health agencies, connectivity was studied to understand information flow within the healthcare department [7, 19]. Note that interconnectedness is the central issue on which other issues focus. In fact, it helps identify more internally cohesive groups, which helps promote innovation based on cohesive groups qualified as innovative [3]. Also, it helps understand how projects are collaborated on [5, 15], and identifying the role importance of employees through deriving patterns of connectivity and determining employees holding the network together [16, 17].

Collaboration. Promoting effective collaboration between employees is probably the most studied question of ONA [5, 14, 15, 21, 31]. By "effective" we mean collaboration without lack nor excess (overload). Both lead to performance issues; the first is because of misusing organization potential, as a paid employee who does not collaborate, neither to help nor to be helped, is an inert potential. The second is because of employees’ departure and burnout, as overloaded employees cannot carry out their tasks and are likely at risk of burnout [5]. Therefore, studying collaboration helps identify underused employees and take necessary measures to increase their participation (related to the integration issue). As well, overloaded employees, once identified, benefit from unloading

\[ A \text{ real information exchange takes place in small network horizon for an average path length } \leq 3, \text{ otherwise the knowledge is assumed unavailable (no communication). } \]
and reducing their participation before it is too late. In [15], after uncovering collaboration links among all organizations managing the LMB\(^6\) natural resource, authors identified organizations well positioned to serve as information hubs and hence increase collaboration. As well, for a school group [14], to ensure a high teaching quality, teachers’ collaboration was assessed and improved. For promoting sustainable management of transportation networks, the collaboration level between transport agencies was identified in [31]. Similarly, authors in [33] studied collaboration to improve partnerships among more than 80 organizations. As the interconnectedness issue, studying collaboration is strongly related to other issues. According to Cross et al. [5], the overused employees are those too connected to others; hence, identifying them may amount to a connectivity issue. Also, they are generally the organization’s key players, which strongly relates to determining employees’ roles and identifying influencers.

*Role importance.* The employee’s importance within the organization differs, and some are more important and more solicited than others (3 – 5% of employees account for 20 – 35% of the value-adding in the organization [5]). These employees are known as key players or influencers and are usually those controlling both knowledge [11] and decision flow [13]. Their influence may be positive, creating energy within the organization, or negative enough to cause employee burnout. Identifying informal influencers is of great importance for the organization as it helps recognize hidden talents considered peripheral in the formal network and whose contributions go untapped. In [17], centrality measures analysis were used on the online social network of employees to detect leadership roles inside the organization. Similarly, in [16], leaders and different employees’ positions were inferred using node centralities on the informal networks. In [11], authors identified influencer knowledge diffusion within an insurance company based on a model that classifies employees into different kinds of streamers. Identifying influencers is related to other issues of ONA. Indeed, these talents are generally stars who may be overlooked and make ideal leaders and succession candidates, which is needed for any organization restructuring. Also, determining positive and negative influence is crucial for driving change as it helps understand the behaviors of the high-performing groups that may conduct and incubate the innovation process.

*Innovation.* The organizational network structure significantly impacts its ability to innovate, and new ideas are usually created and developed into products through informal social relationships in organizations. One of the several ways to promote innovation is to identify the group that adopts and performs it, which is called innovative. An innovative group must satisfy two main properties, namely creativity, and execution [35]. The first one ensures the group’s ability to generate innovative ideas. For that, it should have as many external connections as possible and hence be connected to many perspectives [36]. While the execution needs a group to be dense and tightly linked, this helps bring the new ideas to market, *i.e.* the more dense the group, the more productive it is. The challenge is that a network is usually good at either creativity or execution but not both [3]. Indeed, a network optimized for creativity is generally very different from the one optimized for execution, which is the main thing that makes innovation difficult. In [3], authors stimulate innovation within a pharmaceutical company and identify which divisions are better structured for innovation.

*Restructuring and mobility.* Any change in organizational network structure impacts the organization’s performance. This impact can be assessed after restructuring initiatives, such as identifying key players and succession candidates, burnout, job mobility, etc. Such a study helps understand the impact of informal networks in building new relations and take necessary measures to reduce the integration and time-to-productivity of new employees. In [12], a detailed analysis was conducted on large firms (~ 7000 employees worked in ~ 400 jobs) to capture job mobility patterns within organizations. In [37], predicting sub-optimal patterns across the organization in a software development community called community smells was tackled. The authors focused on assessing the power of socio-technical metrics, such as congruence, in predicting such communities. In [38], employee turnover has been studied, and a method for predicting it has been presented based on a comparison of e-mails sent (before turnover) by who chooses to leave and who decides to stay.

*Table 2* summarizes different works dealing with the above-mentioned issues.

\(^6\) One of the largest rivers in the world that flows through six Asian countries.
Table 2. The literature overview of the Organizational Network Analysis. The used metrics are: (a) Degree (Avg degree, in-degree, etc.), (p) Congruence, (b) Diameter, (i) Density, (c) Avg Path Length, (j) Component, (d) Clustering Coefficient, (k) # isolate, (e) Modularity, (l) Centralities (degree, betweenness, etc.), (f) Node Degree, (m) # edges, (g) PageRank, (n) Godfather index, (h) Size, (o) Redundancy, (q) vertex-connectivity.

<table>
<thead>
<tr>
<th>Issue</th>
<th>Work Area</th>
<th>Metric</th>
<th>Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interconnectedness</td>
<td>University (a) (b) (c) (d) (e) (f) (g)</td>
<td>Gephi</td>
<td></td>
</tr>
<tr>
<td></td>
<td>School (a) (b) (l) (j) (k)</td>
<td>UCINET</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Insurance company</td>
<td></td>
<td>UCINET</td>
</tr>
<tr>
<td></td>
<td>Healthcare department</td>
<td></td>
<td>ORA</td>
</tr>
<tr>
<td></td>
<td>University library</td>
<td>(o) (p)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pyramid scheme-organization (c) (d) (e) (l) (k) (m)</td>
<td>R (igraph)</td>
<td></td>
</tr>
<tr>
<td>Collaboration</td>
<td>Management organization (l) (m) (j) (k) (a) (c) (n)</td>
<td>UCINET</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Transportation (a) (l) (c) (d)</td>
<td>Gephi</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Research team (ITService) (l) (j) (d)</td>
<td>UCINET</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cluster organization</td>
<td></td>
<td>ERG</td>
</tr>
<tr>
<td>Innovation</td>
<td>Pharmaceutical company</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Role importance</td>
<td>Comercial company (l) (g)</td>
<td>Cytoscape</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Multinational companies</td>
<td>(l)</td>
<td>Networks</td>
</tr>
<tr>
<td></td>
<td>Production company (l)</td>
<td>Cytoscape</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IT Enterprise (fedIT LTD)</td>
<td>(l)</td>
<td></td>
</tr>
<tr>
<td>Restructuring &amp; mobility</td>
<td>Network of water organizations (l)</td>
<td>R (igraph)</td>
<td></td>
</tr>
<tr>
<td>10 Centers (Large firm)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software development team</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service company (l)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Management consulting company (a) (l) (l) (q)</td>
<td>UCINET, R</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4 Example of discussion and open questions

This paper presented the general approach for conducting an ONA project. In such a project, analyzing the organizational network can be considered for different objectives, which a priory determine which graph metrics should be used to analyze the informal relationships among actors in the organization. Almost all studies presented positive results to answer the question raised at the beginning of the study, with some limitations to be subsequently considered in extended research works.

To illustrate the main idea, let us consider the widely used datasets on organizational network analysis, containing the data collected from 21 managers of a high-tech company [41]. Figure 2 and 3 show, respectively, the organization chart of the company and the informal relationships between the employees. In literature, we can find a detailed study dealing with this data, but we want to draw attention to

Fig. 2. Friendship network within the Hi-tech enterprise [41]
Comparing the two networks, we can observe the effect of informal relationships and identify cliques within the company. The friendship network is far more dense, and one can easily see that the request for advice follows friendships more than formal relationships. For instance, Hal is in the largest work group, Ev, but he collaborates with employees from other groups. Also, some employees, such as Chris, Rick, Ken, and Irv, play an essential role within the organization, which is not prescribed in the formal network. Conversely, some leaders, such as Ev, are not as involved in decision-making as they should be. Also, the informal network shows that Alex’s group is tightly connected and more collaborative than Fred’s group (for instance), which is essential for promoting innovation, for example, [3], which is not recognizable in formal networks. Furthermore, it is interesting to observe that employees like Tom and Hugh may be overloaded, which risks causing burnout, as mentioned above.

Using ONA can provide interesting insights to improve work within organizations, and many studies have been carried out to take advantage of this and develop efficient solutions for many problems. However, some general issues about ONA have arisen from our analysis and can be summarized in the following:

Almost all studies focused on the informal network and only consider the hidden relationship, which makes complete sense as the work within the organization occurs through this network. But how about the formal network? Does it have any impact on the informal network? More precisely, one can take advantage of the formal structure to understand some behaviors in the informal interactions between employees and/or propose a more efficient solution to the question after studying the relationship between the two networks.

Another question worthy of special attention is the following. Does the organization’s informal network have a well-defined structure? We generally consider the informal network modeled by a general graph, but if there are some particular structural properties, then proposing solutions that exploit these properties might be more efficient. Thus, a structural study must be carried out. For instance, considering collaboration between groups, which are usually heterogeneous, one can look at which nodes belong to different groups. Also, considering the hierarchical formal network, what impact does the hierarchy have on the objective at hand? We can go deeper by taking into consideration the particularities of the network.
As the importance and real meaning of ONA comes from what we can do with the analysis, which mainly depends on its interpretation, we can consider more precision in interpreting measurements. For instance, for employee churn, the turnover of a director is different from that of a simple employee and must be highlighted for a more accurate interpretation.

References

18. Hanneman, R. & Riddle, M. Introduction to social network methods. (University of California Riverside, 2005)


GPU Computing in Chapel: Application to Tree-Search Algorithms

G. Helbecque\textsuperscript{1,2}, E. Krishnasamy\textsuperscript{1}, N. Melab\textsuperscript{2} and P. Bouvry\textsuperscript{1}

\textsuperscript{1} Université du Luxembourg, DCS-FSTM/SnT, Luxembourg
\textsuperscript{2} Université de Lille, CNRS/CRIStAL UMR 9189, Centre Inria de l’Université de Lille, France

Keywords: GPU Programming, Tree-Search, Chapel, Backtracking, N-Queens.

1 Introduction

Graphics Processing Units (GPUs) have emerged as integral components in modern supercomputers\textsuperscript{1}, reshaping the landscape of high-performance computing (HPC). Their parallel processing capabilities accelerate computations, making them invaluable in addressing the complex nature of optimization challenges in areas like logistics, finance, and scientific research. In the context of combinatorial optimization, the focus of this work, the incorporation of GPUs in tree-search algorithms like backtracking or Branch-and-Bound (B&B), plays a crucial role in expediting decision-making processes, making them a valuable asset in solving complex problems. It also raises multiple challenges related to the irregular workloads, dynamic memory requirements, and data exchanges of those methods.

Some works proposed efficient GPU-enhanced multi-core B&B algorithms to solve challenging combinatorial optimization problems (COPs), such as the Permutation Flowshop Scheduling Problem (PFSP) \cite{1,2}. They demonstrate significant improvements compared to CPU-only approaches, as well as a high scalability in terms of GPUs count.

While these implementations rely on a combination of low-level programming environments, such as C, POSIX standard, and CUDA, this work investigates an alternative approach based on the Chapel programming language \cite{3}. It is a versatile parallel programming language specifically designed for high-level, productive development across various architectures, seamlessly supporting multi-core, distributed and GPU computing environments.

In \cite{4}, the authors revisited the design and implementation of tree-search algorithms dealing with multiple GPUs using Chapel. The proposed algorithm exploits Chapel’s distributed iterators by combining a partial search strategy with pre-compiled CUDA kernels, and relies on an efficient data structure specific to permutation-based problems. Extensive experimentation on big N-Queens problem instances shows that up to 90\% of the linear speed-up can be achieved.

In contrast to \cite{4}, this work presents the design of a GPU-accelerated tree-search algorithm implemented using the native GPU support of Chapel, meaning that no additional programming environment is needed. In addition, our approach relies on a multi-pool strategy, avoiding a problem-specific implementation. The latter is equipped with a static load balancing mechanism in order to face the irregularity of tree-search techniques. The N-Queens problem is considered as a proof-of-concept that motivates further improvements in solving related COPs, and the algorithm is evaluated and compared to a CUDA-based baseline implementation using up to 8 GPU devices. Finally, this work considers both Nvidia and AMD GPU architectures, which are rarely addressed in the literature.

2 Design and implementation in Chapel

Our algorithm relies on the parallel tree exploration model, in which several disjoint subtrees are explored in parallel. The Depth-First Search (DFS) exploration order is considered for memory usage reasons, and the set of generated but not yet evaluated nodes is stored in a pool data structure. More precisely, the algorithm is based on a multi-pool, and each task manages its privatized pool. Since tree-search methods for optimization generally produce irregular trees, the multi-pool is equipped with a load balancing mechanism. It is based on a static distribution of the workload evenly between tasks before the parallel execution.

\textsuperscript{1} Top500 ranking of supercomputers worldwide (11/2023): https://www.top500.org/lists/top500/2023/11/.
The exploration of the tree is then performed as follows: (1) take a node from the pool, (2) evaluate it, (3) prune it if applicable, (4) branch it, and (5) repeat until the pool is empty.

In our approach, the GPU devices are used to accelerate the evaluation of nodes, which is generally the most time consuming part of the algorithm. In particular, when the pool contains a sufficiently large number of nodes, we offload them on the GPU and the latter evaluates them in parallel. The non promising nodes are labeled and the result is sent back to the CPU for pruning and branching. Figures 1 and 2 show the flowchart and the Chapel pseudo-code of the proposed algorithm, respectively, in the single-GPU version.

In Chapel, code is deployed on GPU devices using the **on**-clause, which provides control over locality and affinity. By default, the array data is directly stored on the GPU device, while other data is stored on the host in a page-locked manner. Inside a GPU region, Chapel transparently generates and launches associated kernels for all eligible loops, i.e., order-independent loops that do not involve external function calls or network communication. Code outside of these eligible loops will be executed on the CPU. In contrast to the widespread CUDA programming model, Chapel avoids all explicit data (de)allocations, data transfers, and synchronization barriers between the CPU host and the GPU device.

3 Experimental results

Our algorithm is evaluated on the N-Queens problem, which consists of placing \( N \) non-attacking queens on an \( N \times N \) chessboard. In order to study the effect of the granularity on our approach, we introduce a parameter \( g \) that controls the number of evaluation(s) per tree-node. For comparison purposes, we also implemented a CUDA-based baseline of the algorithm presented in the previous section.

In this work, two different systems are used: one equipped with Nvidia Tesla V100 GPUs and the other with AMD Radeon Instinct MI50 GPUs. The implementation is based on Chapel 1.33.0 and uses LLVM 15.0.7 as a back-end compiler. In the following, we solve the N-Queens instances from 14 to 17, and only the results obtained on the Nvidia-powered system are shown. Similar results have been obtained using AMD GPUs.

Figure 3 shows the normalized execution time of our Chapel single-GPU code compared to the CUDA baseline. We can see that the Chapel implementation is between 3% and 8% slower than the baseline solving the biggest instances. Chapel’s high-level expressive and user-friendly syntax results in a more readable and maintainable code but incurs a performance cost for computational tasks with fine-grained parallelism, where the more explicit control provided by CUDA leads to optimizations not easily achieved in Chapel.

Figure 4 shows the speed-up achieved by our Chapel multi-GPU implementation over the CUDA baseline executed on one GPU. Considering the finest granularity for the N-Queens problem \( (g = 1) \), we can note that the Chapel’s performance is quite limited due to the overheads generated by the high level of abstraction. 70% of the linear speed-up is achieved using 4 GPUs, and no performance gains are observed using more than 4 GPUs. When the granularity is artificially high \( (g = 10,000) \), we see that the performance results are much more better: we achieve between 62% and 75% of the linear speed-up using 8 GPUs.
4 Conclusion and future works

We presented the design and implementation of a GPU-accelerated tree-search algorithm in Chapel. The latter is based on a general multi-pool approach equipped with a load balancing mechanism. The proposed algorithm has been experimented on the N-Queens problem and compared to a CUDA baseline on different GPU architectures. While Chapel facilitates software development via its high level of abstraction, it suffers from a performance loss of up to 8% in our experiments. In addition, the scalability in terms of GPUs count is limited in fine-grained scenarios. In contrast, we demonstrate its ability to achieve up to 75% of the linear speed-up using 8 GPUs in coarse-grained scenarios. In conclusion, we can expect the resolution of large COPs to be highly efficient using GPU-accelerated tree-search techniques in Chapel, in addition to the ease of programming.

In future works, we plan to instantiate our approach to the B&B tree-search method to solve large COPs, such as PFSP. Some of their best-known instances have remained unsolved for 25 years and still represent a serious challenge for HPC. In order to face such a large computational workload, we plan to extend our approach to distributed multi-GPU systems involving many more GPU devices. Finally, we plan to investigate the use of a pre-defined Chapel data structure, DistBag [5], implementing a parallel-safe distributed dynamic multi-pool for large scale applications.

Acknowledgments

The experiments presented in this paper were carried out using the Grid’5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER, and several Universities as well as other organizations. The second author is supported by FNR CORE (ref. U-AGR-7213-00-V), while the others are supported by the Agence Nationale de la Recherche (ref. ANR-22-CE46-0011) and the Luxembourg National Research Fund (ref. INTER/ANR/22/1713848).

References

S$^3$EA: A Self-Supervised Stacked Ensemble Framework for Robust Anomaly Detection to Reduce False Alarms

Sarala M Naidu$^1$ and Ning Xiong$^1$

1 Mälardalen University, Universitetsplan 1, 722 20 Västerås, Sweden.

Abstract. Anomaly detection plays a critical role in safeguarding critical industrial applications, such as cooling systems. This study proposes a novel self-supervised stacked ensemble framework (S$^3$EA) for robust anomaly detection to reduced false alarms in cooling systems. This framework utilizes five base models, including convolutional autoencoders (CAEs), variational autoencoders (VAEs), hybrid models with LSTM, and conventional autoencoders, that are trained on unsupervised data. Using the output of the base-models the ensemble meta-model identifies the final anomalies. S$^3$EA significantly outperforms individual base models and achieves improvement of model performance - accuracy, false positive rate and mean absolute error. The effectiveness lies in its ability to combine the strengths of multiple models and thus, it reduces false alarms by 20.8%. S$^3$EA's superior performance and adaptability make it a promising tool for proactive risk mitigation in cooling systems.

Keywords: ensemble, anomaly detection, stacking, meta model

1 Introduction

1.1 Unsupervised anomaly detection

Unsupervised learning methods are gaining traction for anomaly detection in various domains, including predictive maintenance. [1] applied, both supervised and semi-supervised models to address the challenges of an HVAC system domain. Here, the poor performance from traditional methods like Vector Autoregression (VAR) and Random Forest led to the choice of Artificial Neural Networks (ANNs) to detect faults. Mean Absolute Error (MAE) and Balanced Accuracy Score (BAS) were used as metrics. Deep learning (DL) techniques have proven effective than the classic methods in industrial applications where there is abundance of data from the Inter of things (IoT) and the lack of ground truth labels. However, individual learning models may miss or detect false anomalies that may lead to either increased false alarms or poor performance. To deal with unsupervised data, the reconstruction-based dimensionality technique is common among other. Here autoencoder algorithm is promising wherein the normal and abnormal samples are separated using the reconstruction error. Application of autoencoders has proven to outperform traditional approaches in several recent research works [8-15]. [8] presents the effectiveness of autoencoder based unsupervised anomaly detection method in real-life operation data of gas turbines. [9] is a comprehensive survey on autoencoders application for unsupervised data. [11] employs long short-term
memory with autoencoder (LSTM-AE) to detect and localize faults in an industrial refrigeration system. Ground truth labels were generated automatically by the PLC system, and ROC, F1, precision, recall, and Jaccard index were metrics to evaluate. In [12] LSTM-AE model detects cookware overheating and faults in the inverter side using only inverter temperature. In [13] anomaly detection using Bi-LSTM-AE on wind power data is done, while [14] uses LSTM-Variational autoencoder (LSTM-VAE). In [15] the relationships between autoencoders, shallow models and other DL models is discussed. Nearly 300 studies included indicates the rising popularity and potential of influential research prototype using autoencoders. In power system applications, models like LSTM with and without autoencoders, convolutional neural network (CNN) are used to detect faults in the overall station[16]. While [2] gives a comprehensive review related to 84 articles of DL in electrical power systems in terms of the input data, fields of application, proposed methods, and the level of accuracy.

Unlike traditional machine learning (ML) algorithms, that focus on building a single predictive model, ensemble techniques involve training multiple learners on the same task and combining their predictions to produce a more robust and accurate outcome. Ensemble learning (EL) stands out as an advanced learning approach that harnesses the power of collective intelligence and often surpasses the performance of individual models, offering enhanced generalization capability, better escape from local optima, and superior abilities. [5] show that ensemble framework improved the sensitivity of the collective contextual anomaly detection using sliding window on real-world building energy consumption data. It uses pattern-based and prediction-based learners with majority voting. There are couple of application of EL for-intrusion detection system in Power domain [18-19]. [18] presents the ensemble approach with traditional ML models (decision tree and random forest) to identify the anomaly in SCADA traffic on an in-house developed industrial compliant test bench. [19] also present unsupervised ensemble using traditional ML (isolation forest) in PLC-based process control. [5] presents an ensemble approach based on Hybrid DL for intrusion detection in Smart Grid Networks. [17] presents a systematic literature review on state-of-the-art of various EL methods for application in DL. From this, it is evident that the traditional ensemble learning (TEL) approaches primarily focus on generating base learners and forming ensemble learner, and the choice of the base learner is mostly the traditional tree-based algorithms, with bagging, and voting mechanism being the method of ensembles applied in the survey. Among the deep fast ensemble methods (FEDL) various approaches to reduce the time and space overheads of deep EL (EDL) is proposed. Several methods, including Snapshot and SWA, have been proposed to accelerate EDL by finding multiple base deep learners more efficiently. However, the definition of FEDL remains unclear, and existing methods often produce base deep learners that are too similar. More diverse base learners may improve ensemble performance.

1.2 Industrial Cooling system

HVDC (High-Voltage Direct Curren) systems are crucial for long-distance electricity transmission, but they generate substantial heat, demanding efficient cooling systems. In a power transmission system, the overall HVDC converter system produced more than 95% of the heat loss [3]. A robust cooling system is crucial for maintaining the integrity of HVDC stations by efficiently dissipating heat and ensuring safe operating conditions for sensitive power electronics components. The existing SCADA (supervisory control and data acquisition) system takes care of the monitoring, control and
A Self-Supervised Stacked Ensemble Framework for Robust Anomaly Detection

protection of the overall system. These systems often rely on fixed thresholds and preprogrammed empirical rule-based logics. Then the current practice has the Failure Mode and Effect Analysis (FMEA)/Failure Mode Effect and Criticality Analysis (FMECA) which is a structured approach to investigate how an item's function might fail, the repercussions of such failures, and the mitigation methods employed to minimize their impact, and it adheres to IEC60812: 2018 standard [6].

1.3 Motivation and Contributions

All the previous, related studies on anomaly detection in general have shown that autoencoders, its variants and its hybrid combined models like LSTM and CNN, have a great potential on the unsupervised time series data. The main approach in this paper is to use the benefits of autoencoder based unsupervised learning and offer an ensemble of heterogeneous base models that are stacked to detect anomalies.

Thus, to address the unsupervised data challenge in the domain, a novel framework that merges the learning of five individual hybrid models like LSTM-AE, LSTM-VAE, CNN with AE (CAE), and other variants of AE into a stacked ensemble learning for efficient detection of anomalies to reduce the false alarms. This approach leverages the strengths of individual unsupervised strong learners without requiring extensive labelled data. A real-HVDC station data is used for the training and evaluation of the proposed framework. The evaluation is performed on probability of false alarms (FPR), probability of detection and misdetection (TPR and FNR respectively).

There is no previous study that ensembles individual base and hybrid models for anomaly detection on the industrial cooling system data, to the best of our knowledge. The key contributions are summarized as follows:

C1. A real-industrial cooling system sensor measurement data of 3 years is leveraged for multivariate model learning task
C2. Semi-automated process of label generation through Anomaly (risk) detection metrics from five individual base-learner models on unsupervised data
C3. Stacking of the five model outputs with generated labels to form a new training dataset for the ensemble model
C4. Training of self-supervised stacked ensemble (S3EA) meta-model on the produced intermediate labelled dataset to make final anomaly predictions.
C5. Evaluation of S3EA performance using TPR, FPR, FNR and accuracy and comparing it with the best of the base-learner model. Evaluate the impact of the improvement in ensemble performance on the reduction of FPR.

2 Methodology

2.1 Data aggregation and preprocessing

The experiment utilizes operational data from a real HVDC bipolar station spanning three years from 2020 to 2022. The data is collected from the SCADA system on a per-minute basis and resampled to hourly averages. This multivariate time series dataset comprises one environmental feature (ambient temperature) and several cooling system parameters, including supply temperature, return temperature, expansion vessel level, pressure to redundant pumps, and liquid conductivity. To ensure data quality and facilitate effective learning, aggregated dataset is preprocessed that includes data imputation...
to replace missing values and remove noise, data resampling into hourly windowed data, and data normalization using standard scalar to rescale features to unit variance.

### 2.2 Base-Model in the Experiment

This section describes the adopted framework by first giving the background with details of the individual base models that independently perform on unsupervised time series data, followed by the ensemble model for learning from the intermediate supervised data, to finalize the anomalies.

**Autoencoder (AE):** Autoencoders are a type of ML model that operates in a self-supervised manner. Its primary purpose is to compress input data by reconstructing it. Initially trained as supervised models by suing only Normal data (which is majority), autoencoders function as unsupervised models during inference, which is why they are referred to as self-supervised. It consists of two main components – Encoder that works on compressing the input data and the Decoder that works on reconstruction of the compressed input, as shown in Fig. 1 [A]. So basically, it has a three-layered network structure comprised of input, hidden and output sections.

![Fig. 1. [A] Autoencoder model; [B] variational Autoencoder model](image)

The AE converts the input vector into a latent space (also called as code vector or the bottle neck layer) using a set of recognition weights. The by applying a set of generative weights the code vector is reconstructed back to an approximation of the input vector. If zi is the prediction of xi the input vector, that is mapped to its corresponding hidden representation hi in the bottleneck layer, then the final mapped reconstructed output zi, is such that zi~xi,. The hidden unit that is used to represent low dimensional space and the mapping from the hidden layer into output layer (z) is given by “Eq. (1)” and “Eq. (2)” and the error is calculated as in “Eq. (3)”.

\[
h_i = f(x_i) \rightarrow h = s(w \times x + b) \tag{1} \\
z = g(h) = s(w'x + b') \tag{2} \\
\text{error} = x_i - z_i \tag{3}
\]

Where, hi is the hidden representation of the feature vector, encoder weight matrix is W and b is bias vector, while decoder weight matrix is W’ and b’ is bias vector. S is the activation function (sigmoid and tanh or ReLU). With the best selection of these values through training, the reconstruction error is minimized. To ensure that the output is equal to the input and calculate the parameters in this minimization process, Stochastic gradient descent (SGD) is used. Equation "(4)" indicates the minimized average reconstruction function through optimization of the model variables. This encoder provides training with an error value that minimizes the reconstruction error, \( L(x, z) \)

\[
j(M) = \sum_i L \left( x^i, g(f(x^i)) \right) \tag{4}
\]
Here, M is the model, for a given total number of datapoints n, L is the loss function which is here the mean absolute error (MAE) that calculates the non-negative difference between the actual and prediction and is given by “Eq. (5)”.

\[ L(x,z) = MAE = \frac{\sum_{i=1}^{n}|x_i-z_i|}{n} \]  

**(Variational Autoencoder (VAE):** As shown in Fig. 1 [B], the variational autoencoders (VAE) are similar in architectural structure with conventional autoencoders but differ significantly in their purpose and mathematical formulation. VAEs are probabilistic generative models that rely on neural networks as their core components. The encoder part transforms the input data into a latent space, which represents the underlying probability distribution of the data. This latent space captures the essential characteristics of the data while discarding irrelevant noise. The decoder part reconstructs the original input data from the latent space. This reconstruction process allows VAEs to generate new data points that are similar to the training data but not exact replicas. The encoder and decoder are trained jointly using a technique called the reparameterization trick, which enables the model to efficiently optimize its parameters. The unique ability of VAEs to learn a latent representation and generate new data makes them versatile tools for various applications, including data compression, and anomaly detection.

The combined distribution “p” of the latent and observable variables is given by “Eq. (6)”.

\[ p(x,z) = p(x|z)p(z) \]  

A joint probability distribution presents the probability distribution for multiple variables is accomplished by employing an encoder network to approximate the genuine posterior distribution with a learned approximation q(z|x). VAE learns the model parameters by maximizing the Evidence Lower Bound (ELBO) that is given by “Eq. (7)”.

\[ \text{ELBO} = E[\log(p(x|z))] - KL(q(z|x)||p(z)) \]  

The first part is the reconstruction part, that calculates the ability to recover the input data. The second part is the KL variance, that defines the difference between the estimated posterior distribution (q(z|x)) and the prior distribution (p(z)).

**Convolutional Autoencoder (CAE):** Another hybrid model using the autoencoders is the combination of convolutional neural network (CNN) and the autoencoders, to form the Convolutional autoencoders (CAE). In this hybrid model, the spatial features are learnt using the CNN layers and the key differences with respect to the traditional AE is the utilization of this convolutional layers that are characterized to extract knowledge and learn the internal representation of the data. Thus, CAE can create compressed representation of the input data, removing noise while parallelly keeping all useful information and extracting features. CAE is composed of two CNN models, the encoder, and the decoder. The full proposed algorithm for CAE is illustrated in Algorithm 1, as an example for one base learner model. The loss is calculated using the same function as in “(3)” and “(4)”.

**Algorithm 1: Convolutional Autoencoder**

*Input:* Train dataset T_a, test data T_e, hyperparameters { epochs, batch_size, learning_rate, activation function}

*Stage 1: Encoding:*

for each epoch with the batch_size and Train dataset T_a:

- perform convolutional encoder f_p =: x->z
- initialize weights W as per equation 1 and 2
- compress data to a low dimensional space ->h
Stage 2: Decoding:

With the latent representation $h$, perform deconvolutional decoder $g = z \rightarrow x$.

Model $M$ combines encoding and decoding for each $T_d$:

- Compute the MAE as per equation 5.

Compute threshold $T_h = \text{max} (\text{MAE})$.

For $K$ in $T_{ed}$ dataset:

- $K' = \text{predict}(M(T_{ed}))$.

Compute $T_{mae}$ using $K$ and $K'$ as per equation 5.

If $T_{mae} > T_h$ then

$T_{ed} \rightarrow \text{Anomaly}$

Else

$T_{ed} \rightarrow \text{Normal}$

End if

Generated labels on $T_{ed}$: Normal (as 0) and Anomaly (as 1)

**Hybrid model - LSTM with Autoencoders (LSTM-AE):** Yet another hybrid model using the autoencoders is the combination of autoencoder with the Long Short-Term Memory (LSTM) that learn the temporal dependencies. LSTMs can learn the complex dynamics within the temporal ordering of input sequences as well as use an internal memory to remember or use information across long input sequences. The LSTM-AE supports variable length input sequences as well as to predict or output variable length output sequences. Once the LSTM-AE model sequentially processes the input sequence, the hidden state or output of the encoder model encapsulates a learned representation of the entire input sequence in the form of a fixed-length vector. This vector is then fed into decoder model, which interprets it to generate each step of the output sequence. The loss is calculated using the same function as in “(3)” and “(4)”.

**Hybrid model - LSTM with Variational Autoencoders (LSTM-VAE):** The hybrid model of LSTM and VAE combines the strengths of LSTM networks and VAEs to improve anomaly detection performance. LSTMs are well-suited for capturing long-term dependencies in time series data, while VAEs excel at learning the underlying distribution of the data. By combining these two approaches, the hybrid model can effectively identify anomalies that deviate from both the temporal patterns and the overall distribution of the data. While the LSTM encoder serves as a sequence folding layer, transforming features into a batch of time-based feature sequences. It operates similarly to convolution operations on individual timesteps of the feature sequences. The loss is calculated using the same function as in “(3)” and “(4)”.

349
A Self-Supervised Stacked Ensemble Framework for Robust Anomaly Detection

Table 1. Hyperparameters used in experiment for cross-fold validation.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value range used for best results</th>
</tr>
</thead>
<tbody>
<tr>
<td>TimeSteps</td>
<td>1 hr to 24 hr</td>
</tr>
<tr>
<td>Batch size</td>
<td>16 to 64</td>
</tr>
<tr>
<td>Learning rate</td>
<td>0.01 to 0.0001</td>
</tr>
<tr>
<td>Epochs</td>
<td>50 to 500 with callback to monitoring the loss, with patience=5, on a validation split of 10%</td>
</tr>
<tr>
<td>Optimizer</td>
<td>Adam</td>
</tr>
<tr>
<td>Activation functions</td>
<td>Relu, LeakyReLU, tanh &amp; sigmoid</td>
</tr>
<tr>
<td>Data</td>
<td>Hourly sample of 26280 used of which 25% for test and remaining for training &amp; validation (10% of the training for validation)</td>
</tr>
</tbody>
</table>

**Evaluation:** To assess the effectiveness of framework, apart from mae, the other metrics used are:

1. **Probability of Detection (TPR):** also called the true positive rate (TPR) measures the model's ability to correctly identify anomaly given by “Eq. (8)”.

   \[ TPR = \frac{TP}{TP+FN} \times 100 \]  

2. **False Positive Rate (FPR):** metric assesses the model's tendency to label normal flows as malicious and is given by “Eq. (9)”.

   \[ FPR = \frac{FP}{FP+TN} \times 100 \]  

3. **Misdetection Rate (FNR):** also called as the False negative rate (FNR) measures the model's inability to correctly identify normal cases and is given by “Eq.(10)”.

   \[ FNR = \frac{FN}{TN+FN} \times 100 \]  

4. **Accuracy:** is an overall assessment of model's performance given by “Eq.(11)”.

   \[ \text{Accuracy} = \frac{TP+TN}{(TP+TN+FP+FN)} \times 100 \]  

Where, TP is the number of correct predictions, FN is number of incorrect predictions, FP is the number of incorrect predictions, and TN is number of correct predictions.

2.3 **Stacking based Ensemble learning**

Ensemble learning techniques combines multiple base models to achieve better performance than any individual model. In this work, the described individual five base model’s output is stacked together to form the input to the ensemble learning model as in Fig. 2. Integrating multiple ‘weak learning’ models that are trained to solve the problem independently is the key technique in ensemble learning, to improve results. A few individually performing models are stacked together and the output of these models is fed to a meta learner model that predicts the final anomaly. This is a unique approach where multiple sub-models contribute to the combined prediction of an anomaly. The stacked model is generalized to learn how best to combine the multiple existing models. This approach leads to a synergistic effect, leveraging the strengths of each individual algorithm to improve overall performance. Stacking-based ensemble learning is a popular approach in which the predictions of base-level models are used as inputs to a meta-
level model that learns to combine them effectively. The meta model is a supervised neural network that is trained on the stacked data from the base-model. Each of the base-model and the meta model has its own set of hyperparameters optimized based on the cross-fold validations given in Table 1.

![Stacked Ensemble model](image)

**Fig. 2.** Stacked Ensemble model

### 2.4 Proposed Framework

Fig. 3, shows the overall proposed $S^3$EA framework. The measured data is preprocessed as per details in section 2.1. The processed data is then split into train and validation datasets for use with the individual base models, which individually identifies the anomalies using the reconstruction error on the derived threshold. Individual mae is captured for comparing performance.

![The overall $S^3$EA Framework](image)

**Fig. 3.** The overall $S^3$EA Framework

The output of each of the base model that includes the derived label of the anomaly on the test data, is stacked to be fed into the meta model. The meta-model now has a supervised dataset for training. The output of the meta model gives the final set of indexes of the anomalies identified and the mae score. The subject matter experts (SME) of cooling system domain, evaluates the results for the acceptability. This $S^3$EA approach has a human-in-the-loop for evaluating the final model predicted results. Another approach to this to enhance explainability of the model’s output apply the technique from explainable artificial intelligence (XAI). Thus, an extension of this work of $S^3$EA is presented in our [7] work, where in XAI based method...
using additive shapley values is experimented to enhance trust and reliability of the model’s prediction, to the domain experts. Here, each anomaly identified by the model is explained using the shapley values associated to the feature contributing to the anomaly score. And this approach reduces the effort to diagnose the occurrence of an anomaly which currently is domain knowledge dependent.

3 Results and Discussions

3.1 Individual base-model Results

The result of each base model is given in Table 2.

<table>
<thead>
<tr>
<th>Model Name</th>
<th>Test Data mae score</th>
<th>Loss Plots</th>
</tr>
</thead>
<tbody>
<tr>
<td>AE</td>
<td>mae=0.43</td>
<td>![Loss Plot AE]</td>
</tr>
<tr>
<td>VAE</td>
<td>mae=0.453</td>
<td>![Loss Plot VAE]</td>
</tr>
<tr>
<td>CAE</td>
<td>mae=0.007</td>
<td>![Loss Plot CAE]</td>
</tr>
<tr>
<td>LSTM-AE</td>
<td>mae=0.116</td>
<td>![Loss Plot LSTM-AE]</td>
</tr>
<tr>
<td>LSTM-VAE</td>
<td>mae=0.484</td>
<td>![Loss Plot LSTM-VAE]</td>
</tr>
</tbody>
</table>

The train and validation loss plot visually represents the model’s loss during training. Ideally, the training loss decreases while the validation loss remains stable, indicating effective generalization. Deviations from this pattern suggest overfitting, underfitting.
suboptimal learning rate, or overparameterization. Mae is a measure of how close a set of predictions is to the actual values. The lower the mae, the better the model’s predictions. A mae of 0 would indicate that the model’s predictions are perfect. From Table 2., it is observed that the best performing model is the CAE, based on mae score.

### 3.2 Ensemble Meta model Results

The Table 3 shows that the mae metric of the ensemble meta model has a slightly improved performance of 20.8% compared to the mae score of the best performing individual base model- the CAE with mae of 0.007 from Table 2.

<table>
<thead>
<tr>
<th>Model Name</th>
<th>Mae score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ensemble Meta-model</td>
<td>0.0056</td>
</tr>
</tbody>
</table>

In case of the applications where a tradition approach as in cases of [8, 9] is used with either a single best performing model or an individual combined (hybrid) model. Thus, with proposed approach, the performance is enhanced and the impact of this is shown on the false positive and false negative in the domain, described next.

### 3.3 Discussing performance

Now mae metric is one way to compare the direct scores, while the confusion matrix summarizes the performance of a prediction model with evaluation of the probability of the detection of the anomalies, the false positives, misdetection rate and the false negatives, as given by Eq. 8 to 11 in section 2.2. The confusion matrix shows how often the model correctly identifies anomalies and how often it incorrectly identifies the normal data as anomalous. For the test data used in the experiment, the confusion matrix (CM) for the best base model CAE and the final meta model is shown in Fig. 4 and the corresponding individual performance metrics in Table 4.

![Fig. 4](image)

**Fig. 4.** [A] CAE model ‘s confusion matrix ; [B] Meta-model ‘s confusion matrix

<table>
<thead>
<tr>
<th>Model Name</th>
<th>TPR</th>
<th>FPR</th>
<th>FNR</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ensemble Meta-model</td>
<td>0.868</td>
<td>0.0298</td>
<td>0.0082</td>
<td>0.9644</td>
</tr>
<tr>
<td>CAE model</td>
<td>0.846</td>
<td>0.0377</td>
<td>0.00999</td>
<td>0.9554</td>
</tr>
</tbody>
</table>
S$^3$EA achieved an FPR of 0.029, an FNR of 0.008, and an accuracy of 0.964, demonstrating its effectiveness in identifying anomalies while minimizing false alarms by 20.8%. This reduction of FPR, can be evaluated with the FMEA [6] for the project data, to validate the need for maintenance based on the risk of the anomaly identified. These results favorably compare to individual base models, demonstrating the superior performance of the ensemble approach. S$^3$EA's ability to combine the strengths of multiple models leads to a more comprehensive and robust representation of the data distribution, enabling it to effectively identify anomalies avoid unnecessary alerts.

As a result, the proposed S$^3$EA is a powerful tool for anomaly detection, and they can be used to improve the performance of anomaly detection in this domain of industrial cooling system. While the computational cost of training each individual base model and then the meta-model is higher than the traditional hybrid mode or single model approach, the potential benefits, especially in reducing false positives or nuisance alarms, may warrant the additional effort. An extended work with an approach to skip the meta model and propose a novel method to combine the individual outputs from base learners is presented in [20].

4 Conclusion

In this study, a novel framework of self-supervised stacked ensemble (S$^3$EA) for detecting anomalies in a cooling system domain is proposed. The ensemble meta-model is fed by the output of the five base-models that are based on autoencoder architectures and trained on unsupervised data. The ensemble meta-model effectively captures the underlying patterns and anomalies inherent in the data, leading to improved performance in terms of TPR, FNR, and accuracy compared to individual base models. The findings demonstrate the effectiveness of combining multiple architectures into an ensemble meta-model for anomaly detection. Future research directions focus on bringing clarity on the definition for FEDL to reduce time and space or enhancing ensemble diversity or the base-model diversity, developing interpretability on the detected anomaly, and deploying the ensemble meta-model in real-time systems. By pursuing these avenues, further revolutionization of anomaly detection enabling proactive risk mitigation in critical industrial application domain is possible.

Acknowledgments. Hereby we wish to acknowledge the support provided by the Swedish Foundation for Strategic Research Project: ID20-0019 towards Innovation, Design and Technology. The valuable comments and suggestions from anonymous reviewers are highly appreciated.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References


6. IEC STANDARD · IEC 60812:2018


A Parallel Surrogate Approach for High-Dimensional Constrained Optimization with Discrete Variables

Rommel G. Regis
Saint Joseph’s University, Philadelphia PA 19131, USA
rregis@sju.edu

Abstract. This paper presents a parallel surrogate approach for high dimensional constrained black-box optimization where the decision variables are discrete. The motivation is to solve highly constrained simulation-based optimization problems with a large number of discrete variables such as the Mazda benchmark problem that has 222 ordinal discrete variables and 54 black-box constraints. The proposed algorithm is a parallel implementation of the Two-Phase CONDOR algorithm that begins by using surrogates of the constraints to find a feasible point, and then uses surrogates of the objective and constraint functions to improve on this feasible point. Computer experiments on the Mazda problem and on test problems suggest that Parallel CONDOR yields good speedups in terms of the number of parallel evaluations to find a feasible point. Moreover, it provides a significant improvement in the quality of solution over the original (serial) CONDOR given a fixed number of parallel evaluations.

Keywords: Simulation-based optimization · ordinal discrete variables · high-dimensional optimization · surrogates · parallel optimization.

1 Introduction

High-dimensional and computationally expensive simulation-based optimization problems typically require a large number of function evaluations to find a good feasible solution. While surrogate models of the objective and constraint functions have been shown to be effective in reducing the number of function evaluations (and hence the time) to find a high-quality feasible solution, parallelization strategies (e.g., [2]) are expected to further reduce the wall clock time to find such a solution.

This paper presents a parallel surrogate approach for constrained discrete black-box optimization with discrete decision variables of the form:

\[
\begin{align*}
\min_{x \in \mathbb{R}^d} & \quad f(x) \\
\text{s.t.} & \quad G(x) = (g_1(x), \ldots, g_m(x)) \leq 0 \\
& \quad x(i) \in D_i \subset \mathbb{R} \quad i = 1, \ldots, d
\end{align*}
\]

where \(x^{(i)}\) is the \(i\)th entry of \(x \in \mathbb{R}^d\) and \(D_i\) is an ordered finite set of possible settings of \(x^{(i)}\). The proposed parallel algorithm is applied to a couple of test problems involving 7 and 15 discrete variables, and to the Mazda benchmark problem [1, 3–5] that has 222 discrete variables and 54 black-box inequality constraints.

2 Parallel Two-Phase CONDOR Algorithm

This paper develops a parallel implementation of the Two-Phase CONDOR (CONstrained Discrete Optimization using Response surfaces) algorithm [6] that can be used for high-dimensional constrained black-box optimization with discrete decision variables and many black-box inequality constraints. Two-Phase CONDOR [6], or simply CONDOR, uses a two-phase approach where the first phase searches for a feasible point while the second phase improves on this feasible point. In each phase, surrogates of the objective and constraint functions are built or updated and are used to guide the selection of function evaluation points.

In each iteration of Phase I, three criteria (number of constraint violations, maximum constraint violation, and sum of squares of constraint violations) are used to identify nondominated infeasible points one of which...
Table 1. Number of parallel evaluations to feasibility for 5 trials of ParCONDOR using 4 and 8 processors on the Mazda Problem. The number inside the parenthesis is the first feasible objective value obtained. The approximate feasibility speedups are also given for each trial.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Trial 1</th>
<th>Trial 2</th>
<th>Trial 3</th>
<th>Trial 4</th>
<th>Trial 5</th>
<th>Mean</th>
<th>Std Err</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONDOR (serial)</td>
<td>334 (3.1450)</td>
<td>339 (3.0542)</td>
<td>328 (3.1052)</td>
<td>335 (3.1059)</td>
<td>340 (3.0485)</td>
<td>335.2</td>
<td>2.1</td>
</tr>
<tr>
<td>ParCONDOR (4 proc)</td>
<td>91 (3.0974)</td>
<td>88 (3.1071)</td>
<td>93 (3.0511)</td>
<td>89 (3.0653)</td>
<td>88 (3.0835)</td>
<td>89.8</td>
<td>1.0</td>
</tr>
<tr>
<td>ParCONDOR (8 proc)</td>
<td>47 (3.0164)</td>
<td>45 (3.0736)</td>
<td>48 (3.1250)</td>
<td>48 (3.0949)</td>
<td>48 (3.1463)</td>
<td>47.2</td>
<td>0.6</td>
</tr>
<tr>
<td>Feas Speedup (4 proc)</td>
<td>3.7</td>
<td>3.9</td>
<td>3.5</td>
<td>3.8</td>
<td>3.9</td>
<td>3.7</td>
<td>0.1</td>
</tr>
<tr>
<td>Feas Speedup (8 proc)</td>
<td>7.1</td>
<td>7.5</td>
<td>6.8</td>
<td>7.0</td>
<td>7.1</td>
<td>7.1</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table 2. Mean number of parallel evaluations to feasibility of ParCONDOR (over 30 trials) on test problems. The number inside the parenthesis is the standard error of the sample mean. The mean feasibility speedup is also given.

<table>
<thead>
<tr>
<th>Test Problem</th>
<th>CONDOR (serial)</th>
<th>ParCONDOR (4 proc)</th>
<th>ParCONDOR (8 proc)</th>
<th>Mean Feas Speedup (4 proc)</th>
<th>Mean Feas Speedup (8 proc)</th>
</tr>
</thead>
<tbody>
<tr>
<td>G9-Discrete</td>
<td>42.97 (1.90)</td>
<td>11.73 (0.55)</td>
<td>6.10 (0.22)</td>
<td>3.71 (0.08)</td>
<td>7.03 (0.16)</td>
</tr>
<tr>
<td>G19-Discrete</td>
<td>82.30 (0.15)</td>
<td>21.67 (0.11)</td>
<td>11.60 (0.10)</td>
<td>3.80 (0.02)</td>
<td>7.11 (0.06)</td>
</tr>
</tbody>
</table>

is chosen at random. The point where the objective and constraint functions are evaluated is chosen from trial points generated in some neighborhood of the chosen nondominated point according to the predicted values of the above criteria. In each iteration of Phase II, multiple trial points are generated in some neighborhood of the current best feasible solution and the trial points with the minimum number of predicted constraint violations are identified. The function evaluation point is chosen from these eligible trial points according to the predicted objective value and distance from previous sample points.

In the proposed Parallel CONDOR (ParCONDOR) using $k$ processors, $k$ function evaluation points are generated in each iteration. This is achieved by using multiple nondominated points in Phase I that serve as centers for the generation of trial points and by considering various sizes of neighborhoods where the trial points are generated. Moreover, in each parallel iteration, the selection of each function evaluation point uses some information from previously evaluated points and also from the previously selected points within the same parallel iteration before the expensive function evaluations will take place in parallel.

3 Numerical Experiments

ParCONDOR with RBF surrogates is run on a single-objective version of the Mazda 3-car structure design problem [1, 3–5] using Matlab. This problem involves 222 ordinal discrete variables (thicknesses of structural parts) and 54 black-box inequality constraints, including collision safety requirements. The objective to be minimized is the total weight of three types of Mazda cars that use common parts. The number of settings of each decision variable ranged from 4 to 18, and the size of the discrete search space is $4.4427 \times 10^{498}$, which is beyond astronomical. ParCONDOR is also applied to the test problems G9-Discrete and G19-Discrete [6], which are discrete modifications of the well-known benchmark problems G9 ($d = 7$, $m = 4$) and G19 ($d = 15$, $m = 5$). The number of discrete settings of each variable in these problems ranged from 6 to 18, and the search spaces consist of $1.30 \times 10^7$ points for G9-Discrete and $1.12 \times 10^{17}$ points for G19-Discrete.

Tables 1 and 2 show the number of parallel evaluations to find a feasible point and the first feasible objective value obtained by ParCONDOR on the Mazda Problem and on the test problems. For expensive optimization problems, the number of parallel evaluations is roughly proportional to the wall clock time since the overhead time is much smaller than the time for function evaluations. Given a parallel algorithm on a constrained optimization problem, define the feasibility speedup to be the ratio of the wall clock time of the serial version to find a feasible point to the wall clock time of the parallel version to also find a feasible point, which may be different from the one obtained by the serial algorithm. The feasibility speedup of ParCONDOR is approximated by the ratio of the number of function evaluations to feasibility of the serial version to the number of parallel function evaluations to feasibility and this is shown in Tables 1 and 2. The
Table 3. Mean of the best feasible objective value obtained ParCONDOR (over 5 trials) on the Mazda problem after a fixed computational budget. The number in the parenthesis is the standard error of the sample mean.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>350 par eval</th>
<th>400 par eval</th>
<th>450 par eval</th>
<th>500 par eval</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONDOR (serial)</td>
<td>2.9983 (0.0074)</td>
<td>2.8692 (0.0090)</td>
<td>2.8244 (0.0048)</td>
<td>2.7855 (0.0042)</td>
</tr>
<tr>
<td>ParCONDOR (4 proc)</td>
<td>2.7112 (0.0069)</td>
<td>2.7020 (0.0086)</td>
<td>2.6906 (0.0092)</td>
<td>2.6844 (0.0081)</td>
</tr>
<tr>
<td>ParCONDOR (8 proc)</td>
<td>2.6748 (0.0039)</td>
<td>2.6699 (0.0040)</td>
<td>2.6662 (0.0045)</td>
<td>2.6638 (0.0043)</td>
</tr>
</tbody>
</table>

Fig. 1. Mean of the best feasible objective value found by ParCONDOR (over 5 trials) at different computational budgets. The plot also shows 95% t confidence intervals for the true mean.

Feasibility speedups of ParCONDOR on the given problems are good, around 3.7 when using 4 processors and around 7.1 when using 8 processors. The resulting parallel efficiencies on the given problems are around 0.9 and decreases slightly from 4 to 8 processors. Further study is needed to assess the speedup and efficiency of ParCONDOR on large-scale parallel architectures with many processors or cores.

Next, Table 3 and Figure 1 show the mean of the best feasible objective value obtained by ParCONDOR on the Mazda Problem given a fixed computational budget (number of parallel evaluations). These results show that ParCONDOR also yields a significant improvement in the best feasible objective value over the serial version given a fixed number of parallel evaluations. However, the improvement from 4 to 8 processors appears to be much smaller than the improvement over the serial version when using 4 processors.

References

Feature Selection Based on Membrane Clustering

Sharmin Sultana Sheuly\(^1\) Johannes Deivard\(^2\) Tiberiu Seceleanu\(^1\) Ning Xiong\(^1\)

\(^1\) Mälardalen University, Västerås, Sweden
\(^2\) Volvo CE, Eskilstuna, Sweden

Abstract. Given increased complex data with high dimensions, feature selection aims to select a subset of features to increase the efficiency of machine learning. This paper proposes a new feature selection method based on membrane computing. The proposed method has two main advantages. First, it provides a new solution to search for feature combinations while requiring no model construction (which is time-consuming) to evaluate a feature subset. Second, feature selection is embedded in a membrane clustering algorithm, which is designed to enable searching for the best feature subset and finding active cluster centres at the same time. The designed clustering algorithm mimics the behavior of multiple cells and it has stronger global search ability than existing evolutionary algorithms. The efficacy of the proposed method has been shown by the evaluation of a set of benchmark data sets.

Keywords: feature selection · membrane computing · clustering.

1 Introduction

The amount of data produced by real-world applications is increasing with the advancement of sensors, and data storage systems. However, the increase in data amount does not always result in an increase in performance. The real-world data may contain redundant, irrelevant, and noisy features leading to very large and poor search space and degraded performance of Machine Learning (ML) algorithms. Pattern recognition and natural language processing are two domains where the amount of data is enormous, which provides extra challenges for computational resources. In the case of comprehensive modeling, a complex model can degrade the explainability of the model.

Dhal et al. [1] defined feature selection as "the process of selecting a subset of features from the original set to increase classifier’s performance, degrade space complexity, time complexity, and storage requirements". Feature selection aims at reducing the number of features by considering information content, correlation, or classification accuracy. It is a widely used technique for improving the performance in process modeling and classification [2], regarding complexity, accuracy, the risk of over-fitting, and computational expense.

Feature selection methods are divided into three categories: filter, wrapper and embedding [3]. In the filter methods a feature subset is chosen based on
the information content of the feature subset without considering model performance [4]. The intrinsic property of the feature subset is measured based on correlation, or consistency in a filter method. The limitation of intrinsic property measurement is allowing for the acceptance of redundant features while ignoring complementary features [5]. On the other hand, the filter methods are advantageous in reducing the time complexity of the feature selection process. In contrast, the wrapper methods attempt to find the best combination of features with the assistance of a predefined learning algorithm. This means that many models using different feature subsets have to be constructed for evaluations of possible solutions in the search process. It leads to very high computational expense of the wrapper methods. Additionally, the selected feature subset is very dependent on the learning model and algorithm used in wrapper [6].

Evolutionary algorithms have been widely used as a search mechanism for feature subsets in both filter and wrapper methods [7], [8]. In this case, a feature subset that optimizes an objective function is selected. Certain objective functions comprise feature number, classification accuracy, and feature interaction. Multiple objectives can also be handled simultaneously in feature selection by using a multimodal and multiobjective differential evolution algorithm [9]. Unfortunately, traditional evolutionary algorithms still have the problem of easily getting stuck with a local optimum [10].

This paper proposes a new method of feature selection based on membrane clustering. The main idea is to tackle feature selection by resorting to supervised clustering. We aim to find a feature subset that leads to optimal clustering performance in terms of both natural groupings and the impurity of objects in clusters. As the effect of feature combination is considered without requiring repeated model building, our approach saves the computational overhead of wrapper while still delivering results benefiting classification. Further, we designed an algorithm of membrane clustering with embedded feature search, which enables solving the combined feature selection and clustering problems at the same time. Since our membrane clustering algorithm is inspired by the behaviors of multiple cells in a tissue-like membrane system, it has a stronger global search ability than existing evolutionary algorithms employed in feature selection. Preliminary evaluation results also demonstrated the superiority of the proposed method in comparison to some traditional feature selection methods in regards to model accuracy and identifying feature subsets containing complementary features.

In summary, the novelty and contribution of the paper is highlighted as follows:

– We propose a new method of feature selection via supervised clustering. It exhibits the benefits of both wrapper and filter methods.
– We designed a membrane clustering algorithm that is capable of handling feature search and clustering simultaneously. Also, the number of active clusters can be flexibly adjusted by this membrane algorithm.
– Our membrane algorithm for feature selection runs on multiple-cell evolution in a tissue-like membrane system. It is stronger than existing evolutionary
algorithms that are based on the evolution of a single population of objects in solving feature selection problems.

The remainder of the paper is organized as follows. A review of the related work is given in Section 2, followed by the presentation of the proposed method in Section 3. The results of the experiments are discussed in Section 4. Finally, the paper is concluded in Section 5.

2 Related work

A review of related work is presented in this section. In literature, feature selection methods are divided into three types: (a) wrapper, (b) filter, (c) embedding, as mentioned in the introduction. Certain examples of wrapper-based feature selection include: ensemble wrapper feature selection model that combines multiple classifiers to reduce the risk of over-fitting [11], differential evolution wrapper feature selection that uses differential evolution for feature selection in combination with accuracy evaluation in Extreme Learning Machine [12], hybrid ensemble-filter wrapper feature selection that searches for feature subset with filter method followed by optimization with wrapper method [13]. The limitation of these methods is the computationally intensive search process which is time-consuming. The proposed method solves the feature selection problem through membrane clustering, which is less computationally intensive and requires less time.

Mutual information (MI) has been widely used in the filter approaches of feature selection. Yin, Li, et al [14] improved normalized MI to achieve better accuracy considering redundancy, Vinh et. al [15] used normalized MI to remove the dominance of redundancy and relevance in feature subset, Hoque et. al [16] proposed feature–feature MI and feature–class MI to increase relevance and decrease redundancy. Handling the trade-offs between relevance and redundancy seems to be an interesting issue attracting a number of researches.

Genetic algorithms were found useful in selecting feature subsets that contribute to discovery of the information about feature importance as well as the interaction among features [17], [18]. A hierarchical memetic algorithm was developed for combined feature selection and similarity modeling in case-based reasoning [19]. Particle swarm optimization was employed for finding features to optimize the classification accuracy of a classifier while minimizing the feature number [10], [20]. The proposed algorithm in this paper is based on membrane computing. It runs on multiple cells in a tissue-like membrane system and hence it has a stronger global search ability than existing evolutionary algorithms.

Recently feature selection has also been conducted through clustering. The quality of the selected features depends on the clustering mechanism. Certain examples of clustering mechanisms include: separation degree between classes [21], clusters of features with strong correlation [22], maximized cluster quality, and minimized feature number [23]. These aforementioned works have limitations: clustering without considering the impurity of clusters [21], correlation within
feature group [22], and predefined feature size [23]. The proposed feature selection method overcomes these limitations by integrating the impurity measure of clusters and allowing for flexible numbers of selected features and active clusters.

We realize that the recent practices of feature selection through clustering cannot be classified into any current types of methods: wrapper, filter, and embedding. We therefore propose two new categories: a) extrinsic feature selection and b) intrinsic feature selection. In an intrinsic technique, model performance is incorporated into the feature selection process while an extrinsic technique does not incorporate model performance. Wrapper belongs to intrinsic feature selection techniques while filter belongs to extrinsic techniques. As our algorithm searches for feature subsets via clustering without considering model performance, it belongs to the category of extrinsic techniques.

3 The Proposed Method

We propose here a new feature selection method which involves no model construction while still considering the impact of feature combinations. The key to our method lies in supervised clustering with selected features. We seek a subset of features that not only discriminate the data with interesting natural groupings (clusters) but also promote data purity in the groups in terms of classes. An algorithm of Membrane Clustering with Embedded Feature Search (MCFS) is developed to enable identifying the optimal feature subset and data clusters at the same time.

3.1 Supervised Data Clustering

Assume that \( \{P_1, \ldots, P_n\} \) is a set of labelled data points (instances) in a feature space. A supervised clustering algorithm attempts to partition this data set into \( k \) disjoint clusters \( \{C_1, \ldots, C_k\} \) by optimizing the performance in terms of the two evaluation criteria as described in the following.

The first criterion concerns natural grouping and requires that instances in the same cluster be close to each other while different clusters are far from each other. The proximity inside the cluster can be evaluated upon the distances from each instance to other members of the same cluster. The extent to which two clusters are separated can be reflected by the distance between the cluster centres. Taking into account both intra-cluster proximity and inter-cluster separation, we adopt the CS function [24] to assess the quality of the clusters \( C_1, \ldots, C_k \) in the spatial perspective.

\[
CS(C_1, \ldots, C_k) = \frac{\sum_{i=1}^{k} \frac{1}{N_i} \sum_{P \in C_i} \max_{Q \in C_i} \ Dist(P,Q)}{\sum_{i=1}^{k} \min_{j \neq i} \ Dist(m_i, m_j)},
\]

\[
m_i = \frac{\sum_{P \in C_i} P}{N_i}
\]
where $Dist()$ denotes the Euclidean distance metric for two points, $m_i$ is the mean of all data in cluster $C_i$, and $N_i$ the size of cluster $C_i$.

The second criterion is posed for classification purposes and desires that each cluster be possibly pure in terms of its member classes. The number of instances in cluster $C_i$ that have class $h$ is calculated as

$$L(C_i, h) = \sum_{P \in C_i} [\text{class}(P) = h]$$

(2)

Note that the sign “=” in (2) corresponds to a logic function returning 1 or 0 depending on whether both sides of the sign are identical or not. The winning class of cluster $C_i$ is then defined as

$$\text{Win}(C_i) = \arg\max_{h'} L(C_i, h')$$

Further, let’s consider the number of instances in each cluster that are not belonging to the cluster’s winning class. The total portion of such instances reflects how impure clusters are generated. Hence the overall impurity of the generated clusters $C_1, \ldots, C_k$ is measured as

$$IP(C_1, \ldots, C_k) = \frac{\sum_{i=1}^{k} \sum_{h \neq \text{Win}(C_i)} L(C_i, h)}{\sum_{i=1}^{k} N_i}$$

(3)

Since the CS measure in (1) involves the measure of distance, which features to use in distance calculation plays an important role in determining clustering results. There is a strong coupling between the feature selection and clustering problems. This paper develops an algorithm based on membrane computing to tackle combined feature selection and clustering for identifying a subset of the most relevant features.

### 3.2 MCFS Algorithm

The task of clustering is solved by locating the optimal centres of clusters for obtaining a good separation of data into groups. It is an NP-hard optimization problem. The traditional K-means algorithm and its variants are inherently local search methods that easily get stuck with local optima solutions. The proposed MCFS algorithm utilizes membrane clustering [25] [26] for feature selection. It is based on the principle of membrane computing and it aims to better handle the global optimization problem of finding optimal data partitions in conjunction with simultaneous search of relevant features. Membrane algorithms have shown some merits such as better convergence and stronger robustness compared to other evolutionary algorithms with exploration in high dimensional spaces.

Membrane computing [27] is a class of biologically inspired computing models emulating the functioning and cooperation of living cells in tissues, organs and populations of cells. More specifically, our MCFS algorithm mimics the behaviour of a tissue-like membrane system, which consists of multiple membrane cells evolving in the same environment. Each cell contains a number of objects. The
tasks of a cell are 1) controlling the evolution of its objects; 2) communicating
with other cells by announcing its best object in the environment. A tissue-like
membrane system with three cells is shown in Fig. 1.

![Fig. 1. A tissue-like membrane system.](image)

An object in a cell represents a possible solution to the underlying problem. In this research, for the purpose of combined feature selection and clustering, the object is represented as a composite vector of real numbers as shown in Fig. 2. It is concatenated with three segments. The first segment \((f_1, \ldots, f_D)\) comprises feature control bits, where \(f_i \geq 0.5\) means that the \(i\)-th feature is selected and vice versa and \(D\) is the total number of candidate features. The second segment \((g_1, \ldots, g_M)\) decides the active or non-active states of clusters, where \(g_i \geq 0.5\) indicates that cluster \(i\) is active (existing) and vice versa, and \(M\) denotes the maximum number of clusters. This means that we don’t need to specify the exact number of clusters in advance. The number of valid clusters will be automatically decided by the MCFS algorithm in the evolution process. The third segment \((S_1, \ldots, S_M)\) represents the centres of all possible clusters. Note that a cluster centre \(S_i\) in this segment will be ignored if the cluster control bit decides its cluster is not active.

Given an object as shown in Fig. 2, data points can be assigned to clusters based on their distances to the cluster centres. The clustering performance is evaluated based on both the CS measure for nature grouping as defined in (1) and the cluster impurity as assessed in (3). The overall objective function for object evaluation is built by combining the CS and impurity measures as follows:

\[
J(C_1, \ldots, C_k) = CS(C_1, \ldots, C_k) + \alpha IP(C_1, \ldots, C_k)
\]

where the coefficient \(\alpha\) is used to balance finding well-separated clusters and reducing the impurity inside the clusters. Since the distance calculation in data assignment only uses the selected features, the objective function in (4) also reflects the quality of the feature subset represented by the object.
Let an object be represented by a vector $Z$, it is evolved in iteration $t$ by the extended velocity position rule as follows:

$$V_t = wV_{t-1} + c_1 r_1(Z_{ob}(t) - Z_t) + c_2 r_2(Z_{lb}(t) - Z_t) + c_3 r_3(Z_{ob}(t) - Z_t)$$ (5)$$Z_{t+1} = Z_t + V_t$$ (6)

where $c_1, c_2, c_3$ are the learning factors, $r_1, r_2, r_3$ are the random numbers in the interval $[0, 1]$. $Z_{ob}$ denotes the best position identified for the object until now. $Z_{lb}$ is the local best object in the cell and $Z_{ob}$ is termed as the external best, which is randomly picked from the local best positions that are communicated by the cells to the environment. The inertial weight, $w$, is also used in the velocity updating.

The algorithmic description of the proposed MCFS algorithm is given in Algorithm 1. The tissue-like membrane system is initialized with random positions of objects in the cells. It is followed by evaluation of all objects in the cells with the local best being identified for each cell and communicated to the environment. Then the objects are evolved in their respective cells following the extended velocity position rule as described in (5). This procedure is repeated in several iterations before the global best object is returned as the final solution.

4 Experiments and Evaluation

The proposed method was evaluated on seven benchmark datasets from the UCI\(^3\) and Kaggle\(^4\) repositories. The data sets used in our experiments are: Arrhythmia, Breast Cancer, Spectf, Waveform, Ovarian cancer, Electric Grid, and Ionosphere. We compared the proposed MCFS algorithm with the widely used MI (mutual information) based feature selection method in the evaluation.

More specifically, the evaluation was performed via supervised learning of classification models that employ the selected features as inputs. Arguably higher

\(^3\) https://archive.ics.uci.edu/datasets
\(^4\) https://www.kaggle.com/
Algorithm 1 Membrane clustering with feature selection

Read the labelled training data and hyper-parameters;

Begin
Initialize the tissue-like membrane system;
for $t = 0; t < I_{\text{max}}; t++$ do
  for each cell $c$ in the tissue do
    for each object $o$ in $c$ do
      Assign data to clusters using the cluster centres given in $o$;
      Evaluate the data partitions according to the clusters;
      Update the individual best position $Z_{ob}$ found for $o$
    end for
    Update the local best $Z_{lb}$ for cell $c$;
    Communicate the local best $Z_{lb}$ to the environment;
  end for
  for each cell $c$ in the tissue do
    for each object $o$ in $c$ do
      Randomly pick a local best from the environment as external best $Z_{eb}$
      Evolve the object $o$ using the position velocity rule
    end for
  end for
end for
Identify the tissue’s global best $Z_{gb}$ from the local best of all cells;
Return the set of selected features according to the global best $Z_{gb}$
End

quality of a feature subset leads to higher accuracy of the trained models and vice versa. Hence we utilize the accuracy of models as an important indicator to measure the performance of both the proposed MCFS algorithm and the MI based feature selection method. The three learning models used here for comparison are: *k*-nearest neighbor (*kNN*), *Naïve Bayes*, and *Support Vector Classifier* (*SVC*), which were trained on the selected features. The value of $k$ in *kNN* was 7. Gaussian density functions were used to represent the probability for features in *Naïve Bayes*. The kernel used for SVC was the Radial Basis Function (*RBF*).

In the experiments with the MI-based method, we set the threshold as 20% for accepting features. This means that if the mutual information between a feature and the class variable is lower than 20%, the feature will be excluded. The calculation of the mutual information is based on the method proposed by Kraskov et. al [28] and Ross et. al [29].

Figs. 3, 4, and 5 depict the accuracy of KNN, Naïve Bayes, and SVC respectively, using the features selected by the proposed MCFS algorithm versus the features selected by the MI-based method. Our MIFC algorithm outperforms the MI-based method in 13 cases (62%) and shows equal performance in 6 cases (29%). The proposed method is inferior in only 2 models (9%) on the *Ionosphere* dataset.
Fig. 3. Comparing MCFS and MI on KNN

Fig. 4. Comparing MCFS and MI on Naive Bayes
The MCFS algorithm attempts to find the best combination of features to optimize the clustering results. Hence features that complement each other can be included even if a single alone appears irrelevant. Moreover, redundant features may be excluded with this algorithm since including a redundant one will not contribute to reducing the objective function but increasing the distance quantities.

In contrast, the MI-based method calculates the one-to-one relevance of features to the target. This leads to the possible inclusion of relevant but redundant features in the final feature subset. On the other hand, the MI method fails to identify complementary features that exhibit low mutual information with the target. For instance, no feature was selected for the Spectf data set according to MI. However, this data set contains a group of features which complement each other for classification. Such complementary features were detected by our proposed algorithm.

Our method was slightly inferior to MI on the Ionosphere dataset. This can be attributed to the imbalanced distribution of the data, in which the number of samples of the minority class is extremely raw. The rare examples of the minority class challenge the impurity measure in (3).
5 Conclusion

We propose a novel method for feature selection based on membrane clustering. The proposed method outperforms the traditional MI-based method regarding accuracy, finding complementary features and removing redundant features.

The main spirit of our work is to find the best feature combination without repeated model construction. This is realized by the proposed MCFS algorithm that enables simultaneous feature searching and clustering via membrane computing. Moreover, the result of MCFS is independent on any classification model.

Acknowledgement

The research has been partly supported by Vinnova’s Advanced digitalization programme in the project D-RODS (ID: 2023-00244), as well as the Swedish Knowledge Foundation in the project PerFlex (ID: DNR 20220033).

References


370
Collaborative Smart Agriculture: Leveraging Technology and Community Partnerships to Improve Crop Yields and Sustainability- A Case Study

Dua Weraikat, Rochester Institute of Technology of Dubai
Martin Žagar, Kristina Šorić, RIT CRoatia

Abstract

In recent times, we have been witnessing the effects of climate change, escalating issues related to energy and food supply, and disruptions in the supply chain. These factors serve as the primary inspiration for our project, which focuses on supporting small family farms engaged in agricultural production.

Collaboration within the agricultural supply chain has garnered considerable attention in response to the growing global demand for increased crop yields and sustainable farming practices. Small family farms, often situated in rural areas, face challenges due to their limited resources, geographical dispersion, and isolation from larger centers and each other. This makes it challenging to exchange experiences, provide education, and establish a robust network.

Typically constrained by limited access to resources such as land, capital, machinery, and technology, small family farms may encounter difficulties in reaching markets and distribution networks. The smaller scale may also make it challenging for these farms to specialize in a particular crop or livestock production, potentially limiting their competitiveness and reliability.

Although the potential benefits of smart collaboration in farming are recognized, its widespread implementation is hindered by high costs and time consumption, especially among individual farmers lacking proper knowledge. Effective collaboration necessitates the sharing of equipment, data, and expertise, but farmers may be hesitant to share confidential information.

In the first part of our project, we focused on implementing technological solutions to enhance agricultural production. During this phase, we installed sensors and collected data. The entire process is still ongoing. Based on the gathered data, we will develop a smart farming model based on IoT for vegetable cultivation management and optimization.

Throughout the process, our host, Darko Hrnčević, who provided his greenhouse for the project, learned alongside us. Additionally, two more owners of small family farms, Zeljka and Predrag, visited him during the process, expressing interest in participating in the project. From their discussions, we can infer that they are open to collaboration and the implementation of new technologies, with Darko's project inspiring them. Interestingly, they independently concluded that, besides advanced technological solutions, their mutual collaboration is essential for success.
This leads us to the second phase of our project, where we anticipate providing education and encouraging more owners to collaborate and develop advanced strategies for managing the agri-food supply chain.
Ants’ detection and tracking based on Convolution neural networks: a comparative study

Amir Nakib*1

1Laboratoire Image, Signaux et Systèmes Intelligents (LISSI - EA 3956) – Université Paris-Est Créteil Val-de-Marne (UPEC) – 61, avenue du Général de Gaulle 94010 Créteil cedex, France

Abstract

In last few years, deep learning has achieved a lot of success in visual tracking. The purpose of this paper is to review the different tracking methods based on deep learning and traditional tracking algorithms. First, we introduce different traditional and deep visual tracking algorithms, Then we take 4 different trackers and explain its network perspective and workings, to conduct an extensive study to compare the different trackers on our ants dataset. Finally, we give a comparative insight on all the trackers and summarize our findings to point out the future trends for visual tracking.

*Speaker
Through Optimal Vision Transformer architectures based on metaheuristics

Amir Nakib∗1

1Laboratoire Image, Signaux et Systèmes Intelligents (LISSI - EA 3956) – Université Paris-Est Créteil Val-de-Marne (UPEC) – 61, avenue du Général de Gaulle 94010 Créteil cedex, France

Abstract

In recent years, classical neural architectures have shown great potential for computer vision tasks, such as image classification and detection. Simultaneously, the exploration of neural architectures has increased, aiming to reduce human efforts. From convolutional neural networks to recent studies on Computer Vision Transformers (ViT) networks, the significance of exploring hybrid architectures with diverse building blocks has become evident. However, manually designing neural network architectures and optimizing their hyperparameters pose significant challenges. It has been observed that the performance of ViT networks is intrinsically linked to factors such as depth, embedding dimension, and the number of heads, which can significantly affect network performance. In this work, we propose a new automatic search technique for a pure ViT network architecture and a hybrid cell-based CNN architecture combined with a Vision Transformers network. The algorithm, based on fractal decomposition, is referred to as Fractal Decomposition Algorithm (FDA)

∗Speaker
Neural Architecture Tuning: A BO-Powered NAS Tool

Houssem Ouertatani∗1, Cristian Maxim, Smail Niar, and El-Ghazali Talbi

1IRT SystemX – IRT SystemX, Bonus, Inria Lille - Nord Europe – France

Abstract

Neural Architecture Search (NAS) consists of applying an optimization technique to find the best performing architecture(s) in a defined search space, with regard to an objective function. The practical implementation of NAS currently carries certain limitations, including prohibitive costs with the need for a large number of evaluations, an inflexibility in defining the search space by often having to select from a limited set of possible design components, and a difficulty of integrating existing architecture code by requiring a specialized design language for search space specification. We propose a simplified search tool, with efficiency in the number of evaluations needed to achieve good results, and flexibility by design, allowing for an easy and open definition of the search space and objective function. Interoperability with existing code or newly released architectures from the literature allows the user to quickly and easily tune architectures to produce well-performing solutions tailor-made for particular use cases. We practically apply this tool to certain vision search spaces, and showcase its effectiveness.
Adaptive greedy heuristic algorithm for rubber tyred gantry crane scheduling problem at container terminal

Saber Ben Zina 1, Abdelkarim Elloumi2, Alain Quilliot3

1-2 Laboratory of Modeling and Optimization for Decisional, Industrial and Logistic Systems (MODILS), University of Sfax. (FSEGS), Airport Road km 4, 3018, Sfax, Tunisia.
saber.benzina@fsegs.usf.tn
abdelkarim.elloumi@fsegs.usf.tn

1-3 Université Clermont Auvergne, LIMOS UMR CNRS 6158, 63178 Aubière Cedex, France. alain.quilliot@uca.fr

Abstract. This paper deals with the gantry cranes problem scheduling at a container terminal. The scheduling of this equipment so as to minimize their container assignment time when executing two different operations. A mixed integer linear programming model is developed to minimize the execution time. An optimized greedy heuristic algorithm (GHA) is also applied to solve this problem, with a new best-scheduling update strategy included to improve the stability of the solution. For small-scale problems, the proposed GHA solutions were compared with the optimal solutions obtained from the model using CPLEX software. For large-scale problems, the proposed GHA solutions are compared with large instances due to its inability CPLEX to solve the problem optimally in a reasonable time. Modeling is developed to handle this problem in the context of container traffic at the import or export level. The results of this study confirm that the GHA method proposed in this article is capable of efficiently solving the RTG cranes scheduling problem on the container terminal.

Keywords: Container terminal, Yard crane scheduling problem, Optimization, Greedy heuristic algorithm

1 Introduction

Container transport services have grown exceptionally in recent years, and this trend shows no signs of abating, as there is a growing demand for container transport services. In a port container terminal, a large number of vessels are attached to each other, and each vessel relies on a certain amount of equipment, supported by a number of berths and grues mobile unloading/loading. In a second step, the unloaded containers are moved to the storage area by a conveyor via internal trucks and placed in suitable positions by crane RTGs. Finally, the containers are shipped from the storage area to their customers by external trucks. For export operations, import containers are moved to the terminal’s storage area by internal trucks and unloaded from trucks and several blocks using RTG cranes in the terminal. Then the internal trucks transport the containers to the storage areas, placing them on the yard blocks that need to be scheduled. When operating container terminals, RTG crane gantry cranes play an essential role in storage areas. There are also two models of gantry terminal handling resources, RTG crane and resarche staker (RS).

However, the role played by de yard crane exceeds the performance of the (RS), especially when the height of the container blocks has exceeded their characteristics. Because of the importance of this type, we have focused in this study on the optimal scheduling of the RTG crane as a function of location set in block container. When the RTG crane is ready to be assigned, the containers in the storage area move with difficulty and take up a lot of space on the container block for a long time when turning in the yard, as they are very bulky and difficult to turn. For this reason, a competitive container terminal needs to improve RTG crane synchronization and enhance the operating profitability of its rubber-tired gantry cranes (RTGCs), are therefore of enormous value to the running of container terminal operations. Recently, the planning of gantry crane yards in a container storage area has been studied by several researchers.

The objective is to minimize the duration of all these operations for a gantry crane waiting at a container block. Each container block slot a maximum of 3 gantry crane containers, is directly connected to the storage area and
features yard blocks with a total length of 30 containers and 6 stacks in width with a height of 5 containers. This document deals with the problem of minimizing the duration of all these operations in relation to the resources required in terms of gantry cranes and storage areas, while taking into consideration the risk factors of gantry crane interference in storage spaces. Consequently, the aim of this study is to determine the processing time for each container using a gantry crane and a storage location, in order to complete all operations in the shortest possible time. Fig. 1 shows the gantry crane from yard to block container, while Fig. 2 is an example of the terminal operation of containers and gantry crane, and, finally, Flowchart of the (GHAGCCSP) greedy heuristic algorithm GCSP are given in Fig. 3. In recent research, we have proposed exact methods for solving the GCSP with non-interference constraints and container location capacities at the container block in the port of Sfax-Tunisia. However, building large instances takes a long time and is difficult due to memory saturation.

The objective of this paper is to reduce execution time using a GHA. There are many researchers who have focused on (GCSP) in the literature. In Section 2, the authors review the literature on the (GCSP) in container storage areas. Section 3 explains the definition of the principles of numerical examples and the mathematical model of the container terminal gantry cranes scheduling problem (GCSP). The structure of the approximate optimality solution, based on the greedy heuristic algorithm GHA, and fig. 2 shows the process of assigning the gantry crane to the block container. is shown in Section 4, and Sect. 5 contains illustrations and comparisons between CPLEX and the proposed GHA. This section also deals with a large-scale problem. The last section 6, then contains the most essential conclusions of this document.

2 Related Works

Research into gantry crane scheduling problem (GCSP) has focused on improving the performance of container terminals. In [1] The researcher has identified this objective in the context of scheduling and organizing the transport of individual loads, and in a generalization of the routing of a container transporter in a storage terminal. Let's analyze the operational problem of loading and unloading trains at a container terminal. The approach is based on MIP for scheduling the deployment of gantry straddle carriers and gantry cranes for loading and handling container, with the aim of minimizing total train delay time. Consequently, they show a solution based on a branch-and-bound (B&B) method and a simulated annealing heuristic to optimize the container terminal.

In [2] proposes an integrated optimization approach to determine crane yard schedules and truck placement positions for moving through storage areas, a mixed integer programming (MILP) model is formulated to improve the problem, and a two-stage heuristic algorithm is realized to optimally solve the problem.

For container import and export activists, data on the various destinations, boarding vessels and collection times are normally known by the terminal manager at the time of arrival. In [3] examines the problem of space allocation for export containers in automated container terminals, applying optimization integrated with simulation. Part of the objective function of an optimization problem corresponds to a simulation module for minimizing vehicle waiting time. Then, to anneal the target a MILP proposes with genetic algorithm adopted from the simulation model that shows computational speed performance resolution time.

In [4] the authors compared automated QC and gantry crane (GC) scheduling during ship loading and unloading. They had put forward a MILP that included precedence rules and time windows, and aimed to improve the efficiency of container port operations scheduling, as well as a heuristic algorithm adapted to wait for the total objective of minimizing the overall time of all seaside and landside resources. They propose a metaheuristic genetic algorithm (GA) to address this problem. The genetic algorithm and MIP to provide an optimal solution to the GCSP problem. On the other hand, in [5] have proposed a MILP to minimize the transport time of problematic automated guided vehicles (AGVs). With a strategy based on graph theory solved by exact method and finally to optimize their result a heuristic algorithm adapts and shows the efficiency of solution compared to other defined approaches. Subsequently, numerous heuristic approaches have also been proposed to deal with the various types of GCSP problems, with the aim of improving the quality of gantry crane scheduling.

In [6] develop a two-level stochastic model to optimize storage space operations according to constraints linked to increasing container traffic. By applying the model, they are able to determine how to minimize total transport distances, although stochastic models provide a good measure of terminal performance. The first model level is calculated so as to assign the ships in each block, without considering the physical characteristics of the container block location, and the second is a genetic algorithm-based solving system applied to solve the first-level model. A satisfactory solution result from the equilibrium coefficient share will be obtained based on the practices between
the two levels in a container terminal. Concerning GCSP this is a topical issue, in [7] the authors have developed a gantry crane (GC) displacement problem with the aim of finding ways to improve the operational efficiency of container terminals and to consider the integration of various automated container handling methods. This integration of resources minimizes container loading and unloading times, both in the storage area and on the ship. To this end, two high-performance solution methods, inspired by the PIM of a mathematical model proposed for the purpose of validating a small-size instance and an illustrious large-size genetic algorithm (GA), are developed for finding a solution to a problem. In all cases, both approaches provide an efficient result within an acceptable computing time. For container import and export activists, data on the various destinations, boarding vessels and collection times are normally known by the terminal manager at the time of arrival. In another sense [8] develop a scheduling strategy for the yard crane workload balance and resolution (MILP) proposed in the case study that shows the operational optimization of terminals.

In [9] have proposed a MILP to minimize the transport time of problematic automated guided vehicles (AGVs). With a strategy based on graph theory solved by exact method and finally to optimize their result a heuristic algorithm adapts and shows the efficiency of solution compared to other defined approaches. Subsequently, numerous heuristic approaches have also been proposed to deal with the various types of GCSP problems, with the aim of improving the quality of gantry crane scheduling. In a second phase, a genetic algorithm (GA) combined with a heuristic algorithm was implemented as a solution to the problem. The performance obtained is better than that of the exact method for achieving the fixed objective of minimizing loading and unloading operations in the container storage area [10].

Consequently, they show a solution based on a branch-and-bound (B&B) method and a simulated annealing heuristic to optimize the container terminal, and a two-stage heuristic algorithm is realized to optimally solve the problem of scheduling site cranes to non-interference constraints and container location capacity for single container blocks. Section 3 provides a mathematical model based on the model proposed by [13] has a mixed integer programming (MILP) model solved using CPLEX, and the exact method we propose for determining all possible choices for assigning the storage area gantry cranes to the container blocks while considering the non-interference constraints between the RTG cranes and the container location capacity. It is capable of calculating the optimum completion time for a container block. We found that the exact MILP method outperformed the port identification method in terms of calculation time and completion time for a small number of container locations. However, the difficulty with the MILP method is that for large instances, such as 25 slots, it may take 2 hours to complete the operation. For these situations, we wish to propose in this study a GHA to achieve results closer to the optimum, with fast execution for the yard gantry crane scheduling problem as well as with capacity in each container location follow the dimension of the container block to storage area. Besides, in [14] to examine the yard crane scheduling problem (YCSP), with the objective of specifying the order in which containers are processed in the storage and retrieval areas. To this end, they have developed a mixed integer programming (MIP) formulation for handling gantry crane movements in the tank farm and a heuristic level that handles the optimal sequence based on local search. The results show the performance of the solution in large real instances obtained from a container port.

Fig. 1. Rubber-tired gantry cranes for container terminals
3 Mathematical Models for the GCSP

The problem of the yard gantry crane that we have studied meets the elements mentioned, container scheduling is affected on board a single container block, moving time between slots for the gantry crane is negligible, with the length of the container block location is determined by the standard container size (20-40 feet). Thereafter, the gantry crane work is interrupted if a storage slot is not available for the slot it has been allocated. Finally, the intermediate trucks arrive at the assigned location once the gantry crane has finished loading or unloading, so there is always a sufficient number of intermediate trucks available.

This feature of the greedy heuristic algorithm GHA can considerably optimize the inconvenience of mixed integer linear programming in earlier studies by [13].

3.1 Mathematical Model

The mathematical model GCSP is represented as follows:

Parameters:
- $S$: sets of slots (storage location per TEUs), indexed by $k, g = 1, 2, \ldots, S$
- $K$: sets of gantry crane, indexed by $k, g = 1, 2, \ldots, K$
- $S_i$: sets of stacks in slot $i$, indexed by $s = 1, 2, \ldots, S_i$
- $p_i$: the processing time of slot $i$ by a gantry crane
- $n_i$: number of containers loading or unloading
- $t_i$: a size to the set stack $S_i$ for slot $i$, indexed by $s' = 1, 2, \ldots, t_i$
- $h$: the time taken to assigned containers in slot $i$
- $f_i$: the storage capacity of slot $i$ per TEUs
- $M$: big number integer used to infinity

Decision Variables:
- $c_i$: an integer variable that represents the current completion time of stack $s$ in slot $i$

$max C_i : \text{Makespan}$
the maximum of workload of container operations in the storage area, which is the final completion time from all slots. Constraints (2) and (3) define the value properties the final completion time for variable decision and the completion time at the beginning of the handling operation. Constraint (4) ensures that each slot must be assigned to at least one gantry crane. Constraint (5) ensures that the processing time must be greater than or equal to the workload (workload = number of containers × time to assign a container in the slot per gantry crane). Constraints (6) and (7) definition of the linear of the decision variable $y$: constraint (6) shows that $y_{ij} = 1$ if the (laste completion time less than or equal to completion ti - processing time, if only if means that $y_{ij} = 1$ when slot $i$ ends before slot $j$ begins, constraint (7) indicates the contrary. Constraint (8) ensures that each slot capacity $i$ does not exceed the stack tier (height), indicates that the two gantry cranes do not process the same slot at the same time. And finally, constraint (9) and (10) represent the interference constraint, that is, when slot $i$ is assigned to crane $k$ and slot $j$ is assigned to crane $g$ at the same time and under the condition
that slot \( i \) that the processing time is lower than slot \( j \) than the processing time if \( i < j \) & \( k < g \). Constraint (11) ensure the nature of the used decision variables.

4 A Greedy heuristic algorithm

The greedy heuristic algorithm is specialized in assignment problems to solve specific problems using a few simple steps. In this section, we develop a heuristic glutton algorithm for solving the problem of RTG crane scheduling and slot assignment in block container to container terminal. The principle of this heuristic is to assign an RTG crane to the slot in the block container that can take the least amount of processing time. This algorithm can compute the makespan of \( S \) slot as a function of \( k \) RTG crane and respecting the following steps in \( S \) iterations:

— Step 1. Start by initializing all parameters, such as the number of RTG cranes, the assignment times in each slot, and the number of stacks (length and height).
— Step 2. Sort the execution times and number of containers for each slot in descending order.
— Step 3. Generate all possible combinations of RTG crane slot assignments using a combinatorial approach. Create an assignment function that calculates the total time needed to complete all assignments.
— Step 4. For each assignment generated, check that it respects the conditions of non-interference between RTG cranes. You can do this by checking that each slot allocated to the same RTG crane does not overlap in time. Go to step 5 if the container assignment is less than or equal to the slot capacity; otherwise, increase the slot value and go on to step 2.
— Step 5. For the assignments that satisfy the non-interference conditions, calculate the total time required to complete all these assignments.
— Step 6. Choose the solution that offers optimum handling with minimum makespan.

A flowchart visualizes an algorithm, a process of tasks in the terminal port. In this section, we propose a flowchart for the greedy heuristic algorithm of the Gantry Cranes Scheduling problem (GHAGCSP), which allows optimal solutions to be built from a single block container (Fig.3).
In this paper, we use a greedy heuristic algorithm (GHA) to find solutions close to reality, in order to simplify the problem-solving process. The greedy heuristic algorithm for obtaining a near-optimal solution is generally used for solving difficult problems and with easy sets of steps used. Initialization, at each stage, an initial local optimal choice is made. This involves choosing the best existing option at that stage, based on certain criteria specific to the problem, solution update, stop criterion, execution.

<table>
<thead>
<tr>
<th>Greedy heuristic algorithm</th>
<th>Rubber-Tired Gantry Cranes Scheduling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize the parameters:</td>
<td></td>
</tr>
<tr>
<td>sort: PTsloti, NCsloti, STsloti, in descending order</td>
<td></td>
</tr>
<tr>
<td>for i to sloti-1 do</td>
<td></td>
</tr>
<tr>
<td>PTsloti, NCsloti ← i + 1</td>
<td></td>
</tr>
<tr>
<td>i = 0</td>
<td></td>
</tr>
<tr>
<td>while (i &lt; j, k &lt; g &amp; &amp; NC + ST (sloti) ≤ CAT slot i = false)</td>
<td></td>
</tr>
<tr>
<td>Cmax ← CCTi</td>
<td></td>
</tr>
<tr>
<td>i = i + 1</td>
<td></td>
</tr>
<tr>
<td>// update execution time</td>
<td></td>
</tr>
<tr>
<td>for k to RTG do</td>
<td></td>
</tr>
<tr>
<td>RTGk ← 1</td>
<td></td>
</tr>
<tr>
<td>min CCTi ← CCTi (i−1)</td>
<td></td>
</tr>
<tr>
<td>for k to RTG-1 do</td>
<td></td>
</tr>
<tr>
<td>if (CCTi ≤ Cmax ) then</td>
<td></td>
</tr>
<tr>
<td>Cmax ← CCTi</td>
<td></td>
</tr>
<tr>
<td>RTGk ← k</td>
<td></td>
</tr>
<tr>
<td>end if</td>
<td></td>
</tr>
<tr>
<td>Cmax ← min CCTi</td>
<td></td>
</tr>
<tr>
<td>end for</td>
<td></td>
</tr>
<tr>
<td>end for</td>
<td></td>
</tr>
<tr>
<td>// Calculates maximum completion time (Cmax )</td>
<td></td>
</tr>
<tr>
<td>Cmax ← CCTi</td>
<td></td>
</tr>
<tr>
<td>for k to RTG-1 do</td>
<td></td>
</tr>
<tr>
<td>if CCTi ≥ Cmax then</td>
<td></td>
</tr>
<tr>
<td>Cmax = CCTi</td>
<td></td>
</tr>
<tr>
<td>end if</td>
<td></td>
</tr>
<tr>
<td>end for</td>
<td></td>
</tr>
<tr>
<td>end while</td>
<td></td>
</tr>
<tr>
<td>Execute min Cmax</td>
<td></td>
</tr>
</tbody>
</table>

RTGk : gantry crane, bay indexed k, sloti : bay indexed i, CCTi : current completion time of the gantry crane, CATsloti : capacity of slot i, PTsloti : the processing time of slot i by a gantry crane, STsloti : number of stacks in slot i, NCsloti : number of container in sloti, min Cmax : Makespan

5 Experimental Results

In this section, small-scale examples are given to compare the performance of greedy heuristic algorithms (GHA) with the optimal solution obtained using CPLEX. Due to the complexity of CPLEX in solving large-scale problems, evaluations for large-scale instances are only made using the optimal solution computed by CPLEX, comparisons for large-scale instances are only made between them and the proposed heuristic algorithm. Several tested instances are solved using CPLEX 12.4.1 (a commercial software package for exact integer programming solving) and the greedy heuristic algorithm (GHA) which relates to the reduction of a complex problem.
5.1 Small and large instance comparison between MILP and GHA

Table 1 illustrates the \( \min C_{\max} \) values and computation time for different tests by applying the proposed heuristic and CPLEX to the MIP. Instances that are optimally solved by CPLEX in acceptable time. Processing times for container locations are given between \([5, 60]\).

<table>
<thead>
<tr>
<th>Instance no.</th>
<th>Size (K × S × Si)</th>
<th>CPLEX</th>
<th>HAG</th>
<th>GAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Objectif</td>
<td>CPU(s)</td>
<td>Objectif</td>
</tr>
<tr>
<td>1</td>
<td>2×3×6</td>
<td>82</td>
<td>0.047</td>
<td>81</td>
</tr>
<tr>
<td>2</td>
<td>2×4×6</td>
<td>104</td>
<td>0.109</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>2×6×5</td>
<td>190</td>
<td>0.141</td>
<td>187</td>
</tr>
<tr>
<td>4</td>
<td>2×7×6</td>
<td>216</td>
<td>0.250</td>
<td>213</td>
</tr>
<tr>
<td>5</td>
<td>2×8×5</td>
<td>380</td>
<td>4.813</td>
<td>170</td>
</tr>
<tr>
<td>6</td>
<td>2×7×6</td>
<td>494</td>
<td>8.156</td>
<td>297</td>
</tr>
<tr>
<td>7</td>
<td>3×9×6</td>
<td>658</td>
<td>17.828</td>
<td>199</td>
</tr>
<tr>
<td>8</td>
<td>2×10×6</td>
<td>232</td>
<td>49.45</td>
<td>237</td>
</tr>
<tr>
<td>9</td>
<td>3×10×6</td>
<td>156</td>
<td>10.53</td>
<td>161</td>
</tr>
<tr>
<td>10</td>
<td>2×11×6</td>
<td>198</td>
<td>120.02</td>
<td>247</td>
</tr>
<tr>
<td>11</td>
<td>3×11×4</td>
<td>176</td>
<td>71.07</td>
<td>166</td>
</tr>
<tr>
<td>12</td>
<td>2×12×5</td>
<td>226</td>
<td>392.24</td>
<td>253</td>
</tr>
<tr>
<td>13</td>
<td>3×12×5</td>
<td>188</td>
<td>148.11</td>
<td>170</td>
</tr>
<tr>
<td>14</td>
<td>2×13×5</td>
<td>228</td>
<td>448.47</td>
<td>262</td>
</tr>
<tr>
<td>15</td>
<td>3×13×2</td>
<td>206</td>
<td>1230.09</td>
<td>176</td>
</tr>
<tr>
<td>16</td>
<td>2×14×6</td>
<td>187</td>
<td>926.82</td>
<td>283</td>
</tr>
<tr>
<td>17</td>
<td>3×14×4</td>
<td>212</td>
<td>363.54</td>
<td>190</td>
</tr>
<tr>
<td>18</td>
<td>3×15×6</td>
<td>131</td>
<td>194.67</td>
<td>297</td>
</tr>
<tr>
<td>19</td>
<td>3×16×5</td>
<td>260</td>
<td>569.14</td>
<td>212</td>
</tr>
<tr>
<td>20</td>
<td>2×18×6</td>
<td>314</td>
<td>&gt;703.19</td>
<td>346</td>
</tr>
<tr>
<td>21</td>
<td>2×19×5</td>
<td>130</td>
<td>&gt;21600</td>
<td>361</td>
</tr>
<tr>
<td>22</td>
<td>3×19×5</td>
<td>MS</td>
<td>MS</td>
<td>361</td>
</tr>
<tr>
<td>23</td>
<td>2×20×4</td>
<td>MS</td>
<td>MS</td>
<td>410</td>
</tr>
<tr>
<td>24</td>
<td>2×22×5</td>
<td>MS</td>
<td>MS</td>
<td>410</td>
</tr>
<tr>
<td>25</td>
<td>3×25×6</td>
<td>MS</td>
<td>MS</td>
<td>312</td>
</tr>
</tbody>
</table>

\( K \times S \times Si \): number of RTGs crane \times number of slots \times number of stacks

\( MS \): Cut-off execution time after 5 hours, saturated memory

\( GAP \) = \( \frac{(CPLEX \ CPU - HAG \ CPU)}{CPLEX \ CPU} \) × 100

According to Table 1, since instance 21, the CPLEX has been unable to produce a solution after more than 6 hours of machine processing, while the glutton algorithm provides a better solution in an extremely short time, but instance 25 is the longest is no solution up to more than 5 hours of executions. The difference in performance (GAP) between the optimum and near-optimum solutions lies between 40.70% and 100%, meaning that the near-optimum solutions are more efficient than the optimum solutions.

5.2 Large random instances

Table 1 shows the combined results. To check the results of our gluttonous algorithm in detail, we carry out evaluations using large sets of instances. We choose three K values (the number of RTGs), 2, 3, 6 for each K value, we propose the S values (the number of slots), 5, 15, 25, and each K value as a function of S. 25 instances are formed, the processing time for each slot assignment is calculated randomly in the interval \([5, 60]\). The number of containers in each slot is calculated in the interval \([5, 30]\). Finally, the number of stacks in each slot is calculated randomly in the interval \([1, 6]\).
5.3 CPLEX performance analysis and the Greedy heuristic algorithm

As part of the analysis of the results, the scope is evaluated as a function of the activities, as shown in the tables. For each S × K slot, the scope value is reduced as the number of gantry crane is increased in all the medium and large instances tested. A short objective value reflects good gantry crane and slot utilization, as shown by the results of the table experiments. The results show that the objective function values obtained through the proposed MILP and the resolution using CPLEX for the small-sized port are very good. Due to the NP hard of the problem, it cannot be solved in an acceptable time for large-size problems, in a limited time is not possible. However, the proposed model is capable of solving small instances computed using CPLEX.

The problem parameters for each instance are set as a function of the number of RTGs, slots and container processing times. In order to clarify the quality of the strategies for improving the proposed (HAG), on container scheduling in the storage area. We first carry out a comparison between the improved GHA and the MILP. Tables 2 to 3 show the observations made on larger instances of three and four RTGs. It's worth mentioning that CPLEX computations are time-limited and that the best solution obtained in more than 4 hours is the best possible solution.

5.4 Example of a gantry crane scheduling sequence for MILP and HAG

Table 2. MILP results for assigning an RTG to each slot. 3x10

<table>
<thead>
<tr>
<th>i</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>k</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(c_i)</td>
<td>102</td>
<td>156</td>
<td>156</td>
<td>156</td>
<td>58</td>
<td>100</td>
<td>50</td>
<td>22</td>
<td>60</td>
<td>118</td>
</tr>
<tr>
<td>(C_{max})</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>156</td>
</tr>
</tbody>
</table>

— Sequence on RTG 1 (slot 1 will be assigned first, then slot 2, 5):

\(RTG_1: 1 \rightarrow 2 \rightarrow 5\)

\(RTG_2: 4 \rightarrow 7 \rightarrow 8\)

\(RTG_3: 1 \rightarrow 5 \rightarrow 9 \rightarrow 10\)

Table 3. HAG results for assigning an RTG to each slot. 3x10

<table>
<thead>
<tr>
<th>No. of slots</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>(RTG_1)</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(RTG_2)</td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(RTG_3)</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

— Sequence on \(RTG_1\) (slot 4 will be assigned first, then slot 6):

\(RTG_1: 4 \rightarrow 6\)

384
We present here an example of processing for 10 slots and 3 RTG crane with number of containers between [5,25] and processing time between [5, 60]. This results in an RTG assignment of \( 1 \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 3 \rightarrow 3 \), the first RTG crane works in the first, second and fifth slot and RTG1, RTG2 and RTG3 crane is worked in the remaining slot. The Makespan corresponds to the largest value between the completion times of all the RTGs, so the most recent completion time between the whole slot is the highest of all the completion times for the crane RTGs. In all cases, we find no interference between RTG cranes when assigning them to slots. Processing time is identical for CPLEX and the glutton algorithm, except for instances 20 to 25, where CPLEX required a long runtime with no results, while the (HAG) achieved results. The proposed (HAG) is able to find practically optimal solutions for small instances, as opposed to the CPLEX algorithm.

6 Conclusion and future work

It is generally accepted that container terminals play a central role in the world. Play a central role in the world
Indeed, the performance of container terminals will depend on their ability to meet the needs of their customers and to satisfy them. The operation of container terminals depends on the allocation of resources within the terminal. Storage RTGs are the most important equipment in the storage area. This paper proposes a greedy heuristic algorithm for the RTG crane storage yard scheduling problem is essential in container port terminal operations. In this study, the problem of scheduling multiple storage yard crane for container handling is treated as an important problem in port terminals. Since it has been shown that the scheduling problem is an NP-hard problem, a gluttonous heuristic algorithm based on real decisions has been proposed to find optimal schedules for this problem.

To further develop the performance of the solutions in terms of execution time and approximation to the optimal solution, a meta-heuristic approach was tested.

References


Author Index

A.c.c. Fontes Fernando, 64
Abdelhakim Aitzai, 197–208
Ablanedo-Rosas Jose Humberto, 94–96
Adel Dabah, 197–208
Aggarwal Swati, 49–51
Aghelinejad Mohammadmohsen, 97–99
Alizadeh Sina, 185–196
Amodeo Lionel, 97–99
Aragon-Jurado Jose Miguel, 299–304
Barrera Jose, 65–76
Batalha Mario, 237–248
Becerra-Rozas Marcelo, 3–12
Ben Zina Saber, 376–386
Benhra Jamal, 112–140
Berrado Abdelaziz, 224, 287–298
Blanchard Damien, 275–286
Bouvy Pascal, 341–343
C.a.R.m. Fernandes Manuel, 64
Cabrera Juan Francisco, 299–301
Cabrera-Sánchez Juan-Francisco, 317–329
Carvalho Da Costa Rui, 64
Chorfi Zoubida, 224, 287–298
Cisternas-Caneo Felipe, 65–76
Cody-Kenny Brendan, 52–63
Cork Jordan, 171–184
Crawford Broderick, 3–12, 65–76
Cruz-Corona Carlos, 317–329
De La Torre Juan Carlos, 299–304
Deivard Johannes, 359–370
Delbem Alexandre, 330
Dorronsoro Bernabe, 299–304
Du Xingbang, 13–24
Dutta Pramit, 155–158
Ejjanfi Mohamed-Ali, 112–140
El Kassimi El Alaoui Hamza, 128–140
Feillet Dominique, 100–111
Ferreira Isabelle, 97–99
Filipic Bogdan, 171–184
Firmin Thomas, 3–12
Flamia Azevedo Beatriz, 305–316
Fonlupt Cyril, 209–211
Fontes Dalila B.m.m., 64
Galuppo Wagner, 330
Galvan Edgar, 52–63, 155–158
Garcés Alberto, 65–76
Gaspar-Cunha António, 330
Giacchetti Giovanni, 3–12
Gomez-Pulido José, 65–76
Goncalves Mauro, 237–248
Guibadj Rym Nesrine, 209–211
Gómez Sánchez Mariam, 65–76
Hamamache Khedouci, 331–340
Haxaire Nicolas, 225–236
Helbecque Guillaume, 341–343
Hosseinkhan Boucher Rémy, 77–93
Hvattum Lars Magnus, 49–51
Ibtissam El Khalai, 287–298
Jahan Sifat, 159–170
Jastrzab Tomasz, 249–262
Karima Rihane, 197–208
Khedouci Hamamache, 275–286
Konovalenko Anna, 49–51
Krishnasamy Ezhilmathi, 341–343
Lalou Mohammed, 331–340
Lardeux Frédéric, 249–262
Leclercq Etienne, 25–36
Liapis Georgios I., 263–274
Liu Chijia, 100–111
M Naidu Sarala, 344–355
Martins Roberto, 237–248
Mathelin Lionel, 77–93
Maxim Cristian, 375
McDonald John, 155–158
Melab Nouredine, 341–343
Mendonca Gustavo, 237–248
Mezatio Eric Papain, 97–99
Modni Ilyas, 224
Monaco Francisco, 330
Monfroy Eric, 249–262

387
Moralles Herick, 237–248  
Mouhoub Malek, 159–170, 185–196  
Mourad-Chehade Farah, 225–236  
Nakib Amir, 373, 374  
Nargeot Melvine, 37–48  
Niar Smail, 375  
Nogueira Arineia, 237–248  
Nouredine Melab, 212–223  
Nóbrega Miguel, 330  
Oualhbi Aymane, 37–48  
Ouertatani Houssem, 375  
Paiva Luís Tiago, 64  
Papageorgiou Lazaros G., 263–274  
Pereira Ana I., 305–316  
Perotto Filipo, 37–48  
Peña Javier, 3–12  
Peña-Arenas Iván Guillermo, 209–211  
Pochelu Pierrick, 2  
Quilliot Alain, 100–111  
Regis Rommel, 356–358  
Rivalan Jonathan, 25–36  
Rocha Ana Maria A. C., 305–316  
Roque Luís A.c., 64  
Rouzé Jérôme, 212–223  
Ruiz Patricia, 299–304  
Sadeghilalimi Mehdi, 159–170  
Seceleanu Tiberiu, 359–370  
Seeberger Rafael, 237–248  
Semeraro Onofrio, 77–93  
Sezer-Uzol Nilay, 141–154  
Sheuly Sharmín Sultana, 359–370  
Silva-Ramírez Esther-Lydia, 317–329  
Soric Kristina, 371, 372  
Soto Ricardo, 3–12, 65–76  
Stapleton Fergal, 52–63  
Sunay Yunus Emre, 141–154  
Talbi El-Ghazali, 3–12, 375  
Tavares Roberto, 237–248  
Toussaint Hélène, 100–111  
Tuyttens Daniel, 212–223  
Weraikat Dua, 371, 372  
Xiong Ning, 344–355, 359–370  
Yalaouï Alice, 225–236  
Yáñez Escolano Andrés, 317–329  
Zagar Martina, 371, 372  
Zhang Enzhi, 13–24