University of Concepción
Graduate Studies Office
Faculty of Engineering - Master's in Computer Science

AUTOMATIC ALGORITHM SELECTION FOR THE CAPACITATED VEHICLE ROUTING PROBLEM WITH GIVEN COMPUTATIONAL TIME LIMITS

Thesis proposal to obtain the degree of
MASTER'S IN COMPUTER SCIENCE

BY

Alexis Esteban Espinoza Rebolledo

CONCEPCIÓN, CHILE

March, 2024

Thesis advisor: Roberto Javier Asín Achá
Department of Computer Engineering and Computer Science
Faculty of Engineering
University of Concepción

Contents

List of Figures

Chapter 1

INTRODUCTION

1.1 Background and Motivation

The Capacitated Vehicle Routing Problem (CVRP) stands as one of the most explored combinatorial optimization problems within the fields of operations research and logistics. CVRP seeks to determine the most cost-effective routes for a fleet of vehicles to deliver goods to various customers, subject to capacity constraints. This problem is not only academically intriguing but also possesses substantial practical importance, manifesting in applications ranging from supply chain management to public transportation and beyond.

The complexity and variability inherent in CVRP instances make finding optimal solutions within reasonable time frames challenging. Traditional approaches, such as exact algorithms, heuristics, and metaheuristics, have been extensively studied to tackle this issue. However, the "one-size-fits-all" strategy often falls short in addressing the dynamic and diverse nature of real-world problems. This has led to the exploration of Algorithm Selection (AS) models, aiming to predict the most suitable solving method for any given CVRP instance based on its characteristics.

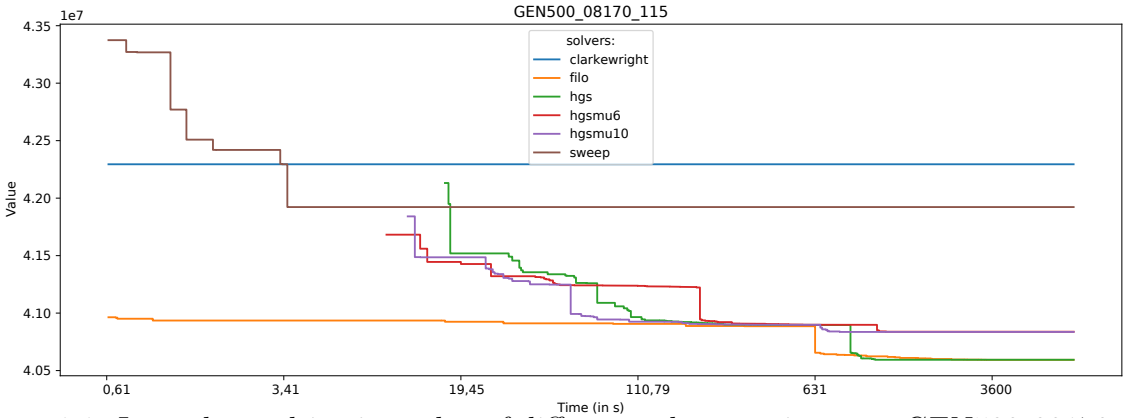


Figure 1.1: Incumbent objective value of different solvers on instance GEN500_08170_115 for an execution time of 4000 seconds.

For example, Figure 1.1 shows the objective value of the best solution found by different CVRP solvers across time for a given instance. We can observe that the solver of choice depends on the available running time. The concept of "Anytime Algorithm Selection" (AAS) has recently emerged as a significant advancement considering scenarios where there exist constraints on the running time of the algorithms. Unlike conventional AS models that provide a single recommendation based on the characteristics of the instance to solve, AAS allows for the selection based on both, the runtime limit and the characteristics of the instance, adapting to the available computational resources. This approach has the potential to significantly enhance the efficiency and robustness of solving strategies, especially in time-constrained scenarios.

The ensuing AAS model is assessed against two baseline solvers: the Single Best Solver (SBS), which is the solver within the portfolio demonstrating the best average performance, and the Virtual Best Solver (VBS), which is the solver achieving the best performance for each problem instance at every timestep. We propose a performance metric $\hat{m}$, which assesses the model's performance against these two baselines. For detailed insights into these performance metrics and their definitions, please refer to Section 2.2.

Despite the promising advantages of AAS, its application to CVRP has not been done yet, with numerous challenges and questions remaining unaddressed. The primary issue revolves around the development and integration of effective AAS models capable of real-time decision-making and adaptation. This involves the selection of a diverse set of algorithms, each with different strengths and weaknesses, under varying conditions and constraints. Most importantly, it also requires to find an efficient characterization of the instances, so that Machine Learning models can learn from them. Additionally, the lack of standardized frameworks and benchmarks for evaluating AAS approaches in the context of CVRP poses significant hurdles to progress in this field.

1.2 Hypothesis

A significant and computationally efficient characterization can be found for CVRP instances, allowing the generation of an algorithm selector model that considers time constraints and performs better than the Single Best Solver, and close to the Virtual Best Solver, under the metric $\hat{m}$.

1.3 Objectives

The main objective of this thesis is to develop an Anytime Automatic Algorithm Selection model for the Capacitated Vehicle Routing Problem that outperforms the Single Best Solver in the portfolio and approximates the behavior of the Virtual Best Solver, obtaining an overall $\hat{m}$ value less than one. For this, the following specific objectives should be achieved:

- Investigate the current state-of-the-art in Algorithm Selection and Anytime Algorithm Selection, particularly within the context of CVRP.
- Develop a comprehensive framework for implementing and evaluating AAS strategies for CVRP, considering factors such as algorithm characteristics, problem instance features, and runtime dynamics.
- Propose and test novel AAS methodologies tailored to the CVRP, assessing their performance against traditional solving approaches under various conditions.
- Analyze the implications of the findings for both theoretical research and practical applications.

1.4 Thesis Structure

The remainder of this thesis is organized as follows:

Chapter 2: Preliminaries provides an overview of the Capacitated Vehicle Routing Problem, Machine Learning, the concept of Algorithm Selection, Anytime Algorithm Selection, and evaluation metrics.

Chapter 3: Related Work describes current state-of-the-art work related to solving strategies for the CVRP as well as current work related to Algorithm Selection for the CVRP.

Chapter 4: Methodology describes the proposed framework for implementing and evaluating Anytime Automatic Algorithm Selection for CVRP, including data collection, feature extraction, model development, and performance metrics. the proposed

framework for implementing and evaluating Anytime Automatic Algorithm Selection for CVRP, including data collection, feature extraction, model development, and performance metrics.

Chapter 5: Experiments and Results presents the experimental setup, discusses the results of the AAS strategies compared to traditional approaches, and provides an in-depth analysis of the findings.

Chapter 6: Conclusions summarizes the key findings of the research, its contributions to the field, and the overall conclusions drawn from the study.

Chapter 2

PRELIMINARIES

2.1 Capacitated Vehicle Routing Problem

The Capacitated Vehicle Routing Problem (CVRP) [11] is a classical combinatorial optimization problem within the domain of logistics and operations research. It involves determining an optimal set of routes for a fleet of capacitated vehicles to deliver goods from a central depot to a set of geographically dispersed customers. Each customer must be visited exactly once, and the goal is to minimize the total transportation cost while adhering to the capacity constraints of each vehicle.

Mathematical Formulation: To formally define the problem, here we present one possible mathematical formulation of the CVRP problem, taken from [7]. Consider a complete graph $G =< V, E >$, where V represents the set of nodes comprising the central depot and customers, and E is the set of edges connecting these nodes. The central depot is assigned node 0.

- Let $N = 1, 2, 3, ..., n$ denote the set of customer nodes.
- Q represents the maximum capacity of each vehicle.
- d_i is the demand of customer $i \in N$.
- K represents the number of vehicles.
- c_{ij} denotes the cost (e.g., distance, travel time) of traversing from node i to node j.
- x_{rij} is a binary decision variable equal to 1 if the vehicle $r \in 1, 2, \ldots, K$ travels directly from node i to node j, and 0 otherwise.

The CVRP problem consists of finding an assignment for variables x_{rij}, so that:

$$\text{Minimize:} \quad \sum_{r=1}^{K} \sum_{i \in V} \sum_{j \in V, i \neq j} c_{ij} x_{rij}$$

Subject to:

$$\sum_{r=1}^{K} \sum_{i \in V, i \neq j} x_{rij} = 1, \quad \forall j \in N$$

$$\sum_{j \in N} x_{r0j} = 1, \quad \forall r \in \{1, 2, ..., K\}$$

$$\sum_{i \in V, i \neq j} x_{rij} = \sum_{i \in V} x_{rji}, \quad \forall j \in V, r \in \{1, 2, ..., K\}$$

$$\sum_{i \in V} \sum_{j \in N, i \neq j} d_j x_{rij} \leq Q, \quad \forall r \in \{1, 2, ..., K\}$$

$$\sum_{r=1}^{K} \sum_{i \in S} \sum_{j \in S, i \neq j} x_{rij} \leq |S| - 1, \quad \forall S \subseteq \{1, 2, ..., n\}$$

$$x_{r,i,j} \in \{0, 1\}, \quad \forall r \in \{1, 2, ..., K\}, i, j \in V, i \neq j$$

Here, the objective function minimizes the total travel cost, and the constraints ensure that each customer is visited exactly once by a vehicle, each vehicle leave the depot once, and the number of vehicles leaving is the same as the ones returning to it. They also guarantee that the sum of the demands of the customers visited by a vehicle is less or equal to its capacity and that each route is connected to the depot.

2.2 Algorithm Selection and Anytime Algorithm Selection

Algorithm selection is a crucial methodology for the automatic identification and choice of the most appropriate algorithm or solver from a predefined set, tailored for solving specific computational problems. The objective is to optimize the efficiency and performance of computational tasks by selecting the algorithm that is most likely to yield optimal results for a given problem instance. This process takes into account various factors such as problem characteristics, instance size, and available resources.

1. **Problem Representation:** Involves the definition of a comprehensive representation of problem instances, capturing relevant features influencing algorithm performance.
2. **Algorithm Portfolio:** Involves assembling of a portfolio of candidate algorithms, each designed to address the specific problem. Algorithms may vary in terms of strategies, heuristics, or optimization techniques.
3. **Feature Extraction:** Deals with the extraction of relevant features from problem instances to characterize their complexities and requirements. These features serve as input for the algorithm selection process.
4. **Training Data:** Involves the process of compiling a dataset of instances and the performance of algorithms on such instances. This dataset is used to train machine learning models.
5. **Model Training:** Refers to the use of machine learning techniques to train models predicting algorithm performance based on instance features. Common techniques include regression, classification, or ensemble methods.
6. **Model Evaluation:** Involves evaluating trained models using validation datasets to ensure generalization and predictive capabilities. Metrics such as accuracy, precision, and recall are considered.
7. **Algorithm Selection:** Refers to the application of the trained models to new instances, predicting the most suitable algorithm. This step involves using learned relationships between instance features and algorithm performance.

Addressing challenging optimization problems requires navigating the trade-off between the swift execution of efficient heuristics, which may sacrifice solution quality, and the application of exact or enumerative methods that demand considerable computational time but generally yield superior solutions. [19] provides a review of the Algorithm Selection research field, as well as points out its challenges. The research community on Algorithm Selection maintains algorithm selection competitions [24] which greatly influenced the research in the field. Another source of collaborative research is the maintenance and updating of the Algorithm Selection Library (ASLib) [6].

The Anytime Automatic Algorithm Selection (AAAS) framework introduces a strategic mapping of user-specified computational time limits to the selection of solvers predicted to deliver the best-possible solutions within these temporal constraints. Noteworthy contributions to this paradigm are the development of meta-solvers, as exemplified in problems such as the Knapsack problem [16] and the Traveling Salesperson problem [17].

In the work by [16], the authors propose an AAAS-based meta-solver for the Knapsack problem. Their findings reveal that, when accounting for time considerations, the meta-solver adeptly predicted the most effective algorithm from a set of 9 solvers across the majority of problem instances. Similarly, in [17], a new AAAS meta-heuristic for the Traveling Salesperson problem is presented. The model, capable of selecting from 5 state-of-the-art TSP algorithms, demonstrated a precision of 79.8% in its predictions.

Automatic Algorithm Selection and Anytime Automatic Algorithm Selection differ in their approach to handling computational problems, particularly in the consideration of time constraints and adaptability during the solving process.

Dynamic Time Considerations

- **Algorithm Selection (AS):** Traditional AS typically focuses on choosing the most appropriate algorithm based on static features of the problem instance. It does not explicitly consider the dynamic nature of the solution process over time.
- **Anytime Automatic Algorithm Selection (AAAS):** In AAAS, the model is trained to predict the best algorithm not just based on static features of the problem instance but also taking into account the evolving anytime behavior. The model considers the trade-off between computation time and solution quality, making predictions at different points during the execution.

Training Data Perspective

- **Algorithm Selection (AS):** In AS, the model is trained on a dataset where each instance is associated with the best algorithm for that instance, considering a fixed time limit or no time limit.
- **Anytime Automatic Algorithm Selection (AAAS):** AAAS involves training the model using data points for each $<$ instance, time $>$ pair. These pairs are

associated with labels indicating the best solver for that specific instance at the given time. This training approach enables the model to learn the evolving behavior of algorithms over time.

Adaptability and Decision Points

- **Algorithm Selection (AS):** AS makes a single decision on the most suitable algorithm for the entire instance based on static features, without considering different stages of the solution process.
- **Anytime Automatic Algorithm Selection (AAAS):** AAAS allows for adaptability during the solving process. It predicts the best algorithm at different time points, providing a sequence of decisions that adapt to the evolving solution landscape.

Use Cases

- **Algorithm Selection (AS):** AS is suitable for scenarios where the optimal algorithm remains relatively constant over the course of solving the problem, and where the primary concern is selecting the most appropriate algorithm at the beginning of the computation.
- **Anytime Automatic Algorithm Selection (AAAS):** AAAS is beneficial in situations where algorithms may exhibit different performances over time or where users require preliminary results quickly and then desire to refine them as computation progresses.

In essence, Anytime Automatic Algorithm Selection extends the traditional Automatic Algorithm Selection framework by incorporating the dynamic aspect of the solution process, making predictions at various points during computation.

Performance metric for Anytime Automatic Algorithm Selection

Given a problem P with a set of instances I, and a set of solvers $A = \{s_1, s_2, ..., s_n\}$ designed for P, an evaluation metric m_s, designed for problem P, is defined. Such metric m_s is used to evaluate the average performance of any solver $s \in A$ on I. Per-instance

Algorithm Selection (AAS) consists of the development of an algorithm selector S that can be seen as a function that assigns each instance $i \in I$ to a solver $s_i \in A$. The objective is to optimize the overall performance of a meta-solver ms using S on I according to the specified metric m_{ms}.

To evaluate the performance of an anytime solver $s \in A$ over a given time interval, we divide such interval into discrete *timesteps*. Let I represent a set of instances, and T denote a set of timesteps. For a specific instance-timestep pair $(i,t) \in I \times T$, $o_s(i,t)$ denotes the objective value achieved by solver s for instance i at timestep t. Due to the considerable gaps in magnitudes of $o_s(i,t)$ across instances and timesteps, we employ a normalization function $n(o_s(i,t),i,t)$ to standardize $o_s(i,t)$ values, ensuring equitable consideration of each data point in an average metric.
In the case of CVRP, we adopt a normalization approach that involves scaling the objective value between 0 and 1, based on the best value found by an algorithm and the worst value encountered by an algorithm up to timestep t:

$$n(o_s(i,t),i,t) = \frac{o_s(i,t) - o_{best}(i,t)}{o_{worst}(i,t) - o_{best}(i,t)} \tag{2.1}$$

The evaluation metric m_s, for a given anytime solver $s \in A$, is then defined as:

$$m_s = \sum_{(i,t)\in I\times T} n(o_s(i,t),i,t) \tag{2.2}$$

This represents the normalized performance of solver s across all instance-timestep pairs. For an anytime meta-solver ms that selects solver $S(i,t)$ for each instance-timestep pair (i,t), its performance metric is defined as:

$$m_{ms} = \sum_{(i,t)\in I\times T} n(o_{S(i,t)}(i,t),i) \tag{2.3}$$

This metric encapsulates the normalized performance of the meta-solver ms over all instance-timestep pairs $(i,t) \in I \times T$. A Meta-solver assessment typically involves comparisons with both the single best solver and the virtual best solver, each defined as follows:

Single Best Solver (SBS): This is the individual algorithm that achieves the best average performance across all instances, i.e. $SBS = \arg\min_{s \in A} m_s$.

Virtual Best Solver (VBS): The VBS is an idealized solver that makes optimal decisions, matching the performance of the best algorithm for each problem instance, without incurring any additional computational overhead.

For an algorithm selector meta-solver ms, the metric $\hat{m}_{ms}$ was introduced in Algorithm Selection Competitions [24]. It utilizes, for each solver s, the performance metric m_s. We extend this metric to anytime algorithm selector meta-solvers with the following generalization:

$$\hat{m}_{ms} = \frac{m_{ms} - m_{VBS}}{m_{SBS} - m_{VBS}} \tag{2.4}$$

Here, m_{ms} is the performance of meta-solver ms, m_{VBS} is the performance of the VBS, and m_{SBS} is the performance of the SBS.

Key observations:

- The closer $\hat{m}_{ms}$ is to 0, the more similar the meta-solver is to the VBS.
- If $\hat{m}_{ms} > 1$, then the meta-solver is inferior to the SBS and, consequently, is deemed not useful.

2.3 Machine learning

Machine Learning (ML) stands as a transformative field within computer science, pioneering approaches that enable computational systems to autonomously learn and improve from experience. At its core, ML involves the development of algorithms and models designed to discern patterns, glean insights, and make informed decisions without explicit programming for each task.

One of the defining characteristics of ML is its reliance on data-driven methodologies. Large datasets serve as the training ground for ML algorithms, allowing them to discern underlying patterns and relationships. This iterative learning process enhances the algorithm's ability to generalize and make accurate predictions or decisions on unseen data. Supervised learning [10] involves training models on labeled datasets, where the

algorithm learns to map input data to corresponding output labels. Unsupervised learning [5], on the other hand, explores patterns within unlabeled data, uncovering hidden structures and relationships. Reinforcement learning [18] introduces an interactive dimension, with algorithms learning through trial and error based on feedback from a designed environment.

ML models can also undergo classification based on the nature of their output. When the output involves discrete values employed to categorize inputs into distinct classes, the model is categorized as a classification model [21]. Conversely, if the output aligns with real values, the model is designated as a regression model [23].

The supervised ML algorithms for classification used here are:

Random Forest [8]:

Model Type: Ensemble Learning.

Algorithm Overview: Random Forest builds multiple decision trees during training and outputs the mode of the classes for classification problems. Each tree is constructed using a random subset of features and training data, and predictions are aggregated.

Key Features:

- Ensemble of Decision Trees: Combines predictions from multiple trees.
- Reduced Overfitting: Random subsets mitigate overfitting.
- Variable Importance: Provides a measure of feature importance.

AdaBoost [14]:

Model Type: Ensemble Learning.

Algorithm Overview: AdaBoost is a boosting algorithm that combines weak learners to create a strong classifier. It assigns weights to misclassified instances, emphasizing learning from errors, and iteratively adjusts weights.

Key Features:

- Weighted Instances: Assigns weights to focus on misclassified samples.
- Sequential Learning: Builds weak learners sequentially.
- Model Aggregation: Combines weak classifiers into a strong model.

Gradient Boost [15]:

Model Type: Ensemble Learning.

Algorithm Overview: Gradient Boosting builds a series of weak learners sequentially, with each correcting errors made by the previous one. It focuses on reducing errors through additive model building and uses gradient descent optimization.

Key Features:

- Sequential Model Building: Constructs models sequentially.
- Gradient Descent Optimization: Minimizes a specified loss function.
- High Accuracy: Generally yields high accuracy and is less prone to overfitting.

Convolutional Neural Network (CNN) [22]:

Model Type: Deep Learning.

Algorithm Overview: CNNs are deep neural networks designed for processing structured grid data, especially images. They utilize convolutional layers for feature extraction, hierarchical structures for learning features, and pooling layers for downsampling.

Key Features:

- Convolutional Layers: Employed for local feature extraction.
- Hierarchical Structure: Utilizes multiple layers for learning hierarchical features.
- Pooling Layers: Incorporates layers for spatial downsampling.

Chapter 3

RELATED WORK

This chapter provides detailed explanations of the algorithms used to solve the CVRP , and discusses previous work on algorithm selection for this problem.

3.1 CVRP solvers

CVRP solvers can be classified depending on the algorithm and approach they use to address the problem. The main approaches are:

- Exact Algorithms:
  - Branch and Bound [29]: Enumerates all possible solutions and systematically prunes the search space to find the optimal solution.
  - Branch and Cut [25]: Extends branch and bound by incorporating cutting planes to tighten the feasible region.
- Heuristic Algorithms:
  - Insertion Heuristics (e.g. Clarke-Wright Savings) [9]: Builds routes by iteratively inserting customers into existing routes based on certain criteria.
  - Sweep Algorithm [26]: Divides the customers into sectors and constructs routes by "sweeping" through each sector
- Metaheuristic Algorithms:
  - Genetic Algorithms [3]: Evolves a population of solutions over multiple generations through crossover, mutation, and selection operators.
  - Simulated Annealing [12]: Uses a probabilistic approach to accept or reject solutions based on temperature and energy concepts.
  - Tabu Search [4]: Iteratively explores the neighborhood of solutions while avoiding revisiting previously explored solutions (tabu list).

Following this, we enumerate the CVRP solvers under consideration for integration into the meta-solver portfolio. The selection process involved choosing solvers representing diverse approaches.

Clarke-Wright [9]: The Clarke-Wright Savings Algorithm is a heuristic method for solving the CVRP. Developed by Clarke and Wright, it efficiently constructs routes by identifying potential cost savings through the combination of individual routes. The algorithm calculates savings for each pair of customers and the depot, progressively merging routes with the highest savings until all customers are served. While known for its simplicity and effectiveness, it does not guarantee optimality, yet it provides practical and scalable solutions to the CVRP.

Sweep [26]: The Sweep Algorithm is a heuristic approach designed for solving the CVRP. It initiates by dividing customers into sectors based on their polar coordinates relative to the depot. The algorithm then "sweeps" through these sectors, creating routes by sequentially adding customers within each sector to the existing route until the vehicle's capacity is reached. In this implementation the Sweep Algorithm uses TSP solvers to optimize the route of each vehicle.

Hybrid genetic search [30]: The HGS algorithm is a hybrid optimization technique introduced by T. Vidal. Integrating a metaheuristic approach with local search, this algorithm employs a genetic algorithm to incorporate local search techniques, thereby improving the overall quality of solutions within the population.

Fast Iterated - Local - Search Localized Optimization [2]: The FILO algorithm is a metaheuristic approach designed for solving the CVRP. Initially, it generates an initial solution using the Clarke & Wright algorithm. Subsequently, it defines various local search operators for the CVRP, from which it selects one based on a Simulated Annealing criterion to enhance the solution. This iterative process allows the algorithm to achieve favorable outcomes, particularly for large-scale problem instances.

More details about the implementation of these algorithms can be found on section 4.1

3.2 Algorithm Selection for CVRP

To our knowledge, in the literature, only one previous work on algorithm selection for the CVRP exists, [13]. In this work, various machine learning models are tested, using a 23-feature vector as input, aiming to capture size, spatial attributes, and capacity constraints for each instance. The objective is to select the best algorithm from a portfolio of four algorithms. However, this approach does not consider the anytime behavior of algorithms or temporal constraints. This work was built around the following characteristics:

Portfolio: The algorithms in the portfolio include the Clarke & Wright algorithm [9], the Sweep Algorithm [26], a genetic algorithm implemented on a GPU [1], and a Self-Organized-Map implementation for CVRP [28]. The algorithms in this portfolio cannot be considered state-of-the-art, since the first two are very well-known heuristics, on top of which, current state-of-the-art solvers built, and the latest two algorithms in this portfolio showed to not be competitive with Clarke & Wright and Sweep Algorithm.

Features: The feature vector encompasses information related to the spatial distribution of customers represented by clusters. After executing the clustering algorithm, the feature vector is composed of various characteristics such as the number of customers, standard deviation of distance matrix, coordinates of the center of the plane, and more, capturing spatial attributes and demand-related metrics.

Models: Three classification models were trained using the feature vector:

- Decision Tree
- K-Nearest Neighbors (KNN)
- Single Layer Perceptron (SLP)

Results: The precision of selecting the best algorithm for the models was as follows:

- Decision Tree: 76.4%
- KNN: 75.1%
- SLP: 79.4%

While these models exhibit good precision, they do not account for the anytime aspect of algorithms. The manually selected features used for characterization can be used as a basis for creating models. The clustering algorithm used for feature calculation, while providing information about the spatial distribution of customers, involves calculating the distance matrix, incurring quadratic complexity. Some algorithms, like the Sweep Algorithm, may not require the complete distance matrix calculation, potentially affecting the effectiveness of clustering. Since Clarke & Wright is the dominant solver in this portfolio, and it is considered a classic heuristic for CVRP, the algorithms in this portfolio cannot be considered state-of-the-art, which hinders the impact of this work.

Chapter 4

METHODOLOGY

4.1 CVRP solvers

Selecting appropriate training and testing datasets is paramount when developing machine learning models for Combinatorial Optimization problems, ensuring their relevance and utility for Operations Research practitioners. In this study, we adhere to a guiding principle wherein the training set is crafted from synthetically generated data with the primary aim of encompassing a comprehensive spectrum of variations within the instances. The successful generation of such synthetic training data is anticipated to yield well-performing models when tested against public benchmarks designed to emulate real-world instances.

The solvers in our portfolio are:

Clarke-Wright: Implemented in C++. The implementation that uses the FILO algorithm is used.

Sweep: We implemented an anytime version of the Sweep algorithm in C++. It consists on various iterations of the algorithm, considering different values for the radius parameter. In each iteration, a radius between 0 and 1 is selected (there are a total of 100 radii, with a difference of 0.01 between them), representing the distance from the depot to the farthest client where a division will occur. Once the division is made, a sweep is conducted (as explained in section 3.1) with the points within that division and one for those outside of it, thus assigning clients to each truck. Subsequently, using a TSP algorithm, the route for each truck is optimized.

Hybrid genetic search: Available at `http://github.com/vidalt/HGS-CVRP`, it is implementated in C++. Three versions of this algorithm were utilized, varying parameters determining the minimum (μ) and maximum (λ) size of the genetic algorithm population. These three versions are `hgsmu6`, which has $\mu = 6$ and $\lambda = 12$, `hgsmu10`, which has $\mu = 10$ and $\lambda = 20$, and `hgs` which uses default values of $\mu = 25$ and

$\lambda = 40$. The purpose of this is for versions with smaller population sizes to achieve faster solutions, while those with larger populations yield better results at higher timesteps.

FILO: Available at https://github.com/acco93/filo, it is implemented in C++, it was modified so that whenever it finds a better solution, it prints the objective value and the time at which it was found.

Our selection criteria focused on state-of-the-art solvers with accessible source codes, allowing for modification to capture their anytime behavior, or those inherently equipped with this capability.

To assess the anytime behavior of the solvers, we partition a one-hour time span into 500 timesteps using a logarithmic scale, mirroring the methodology outlined in [17]. Whenever there is an improvement in the objective function value, we log the corresponding timestep along with the updated solution.

The anytime behavior of each solver is documented by tracking the continuously updated best objective value for each of the 500 timesteps. Figure 4.1 visually presents the anytime behaviors of the solvers for a specific instance. It's noteworthy that the solver producing the solution with the lowest value at a given timestep ts is deemed the optimal choice for any prescribed time limit between ts and the subsequent timestep $ts+1$. In this specific case, FILO establishes itself as the leading solver initially, until HGS surpasses it in the later timesteps.

For each instance-timestep pair, a solver is deemed to "win" if it computes the best-found solution. In case of ties, preference is given to the solver that initially achieved the best incumbent.

4.2 Training dataset generation

The training set was generated through the utilization of TSP instance generators, with subsequent conversion of TSP instances into CVRP instances. Specifically, the TSP generators employed were NETGEN, NETGENM [20], and TSPGEN. These generators have demonstrated efficacy in automatic algorithm selection for TSP problems [17].

The instances vary in size, ranging from 11 to 1000, with an average instance size

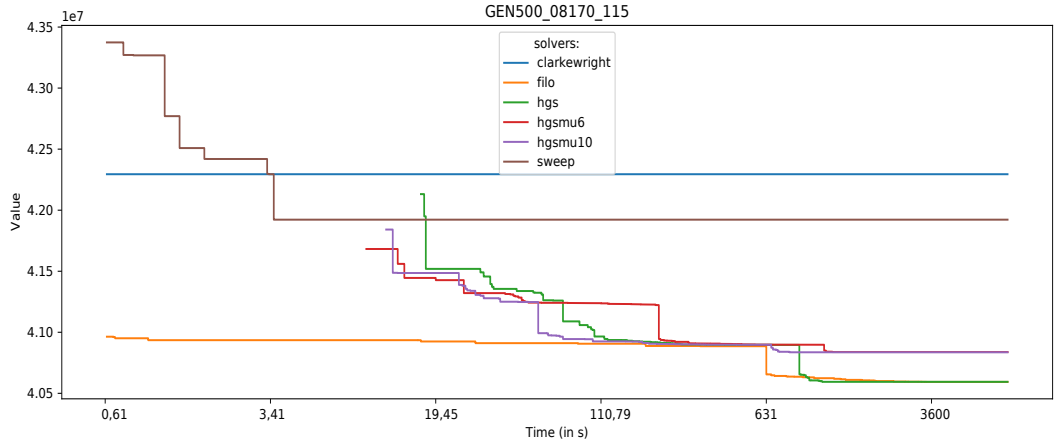


Figure 4.1: Results of each algorithm over time.

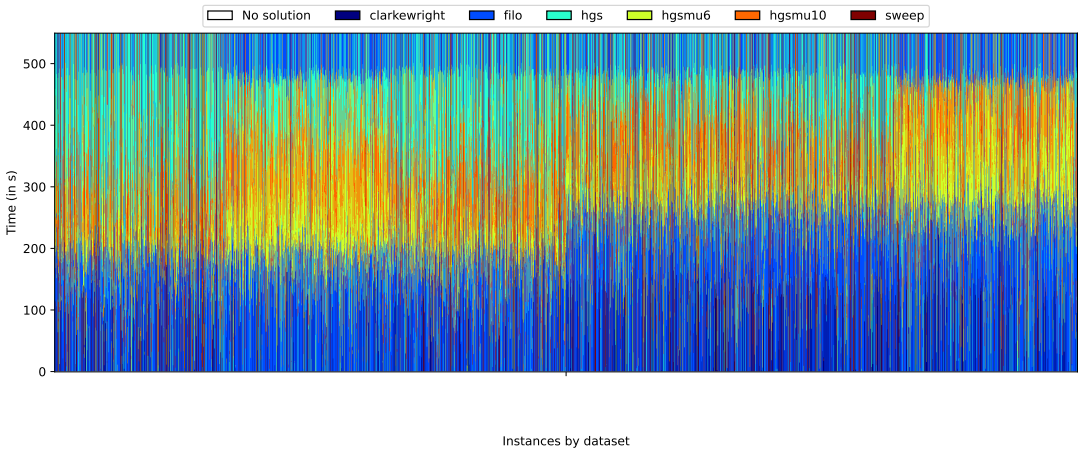


Figure 4.2: Best solver for each instance at every timestep for instances generated from NETGEN.

of 714. The CVRP instance generator operates by taking the coordinates of TSP instances, introducing a depot (with three potential depot positions), assigning demands to each client from seven possible demand categories, and specifying a capacity for the vehicles. This comprehensive approach ensures the generation of diverse and representative instances for training purposes.

The dataset we used for building our ML oracles was generated from running all the solvers on the portfolio over all the instances we generated.

Figure 4.2, 4.3, and 4.4 provide a comprehensive overview of the optimal solver performance for each instance and timestep. The figures highlight that the effectiveness

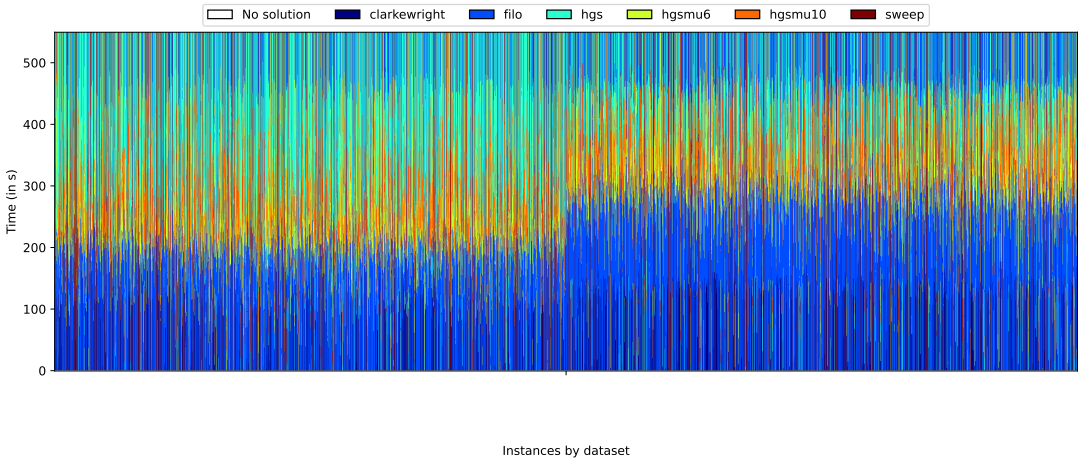


Figure 4.3: Best solver for each instance at every timestep for instances generated from NETGENM.

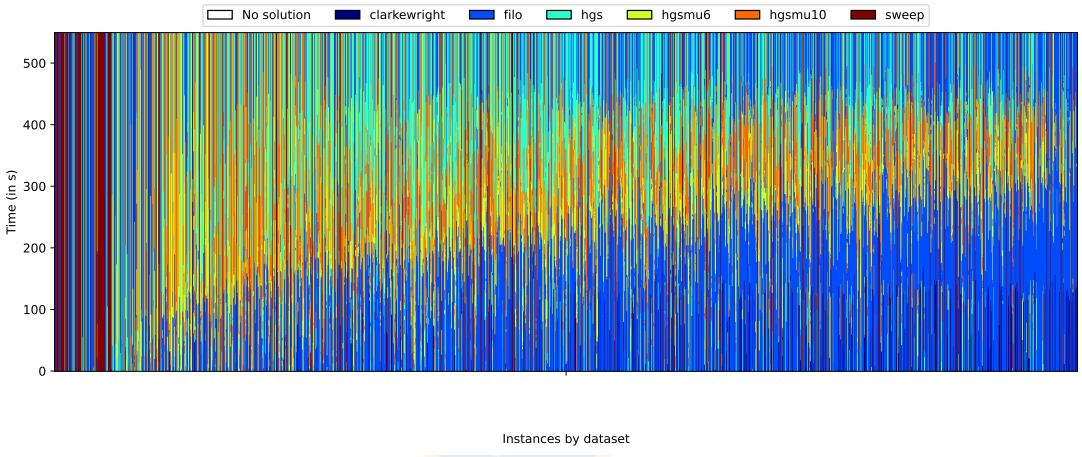


Figure 4.4: Best solver for each instance at every timestep for instances generated from TSPGEN.

of the best solver is contingent upon factors such as the instance size, the allotted time limit for the solver, and the inherent characteristics of the instance itself. Consequently, there is no singularly superior solver applicable across all instances and timesteps.

Notably, Clarke-Wright and Sweep excel in swiftly identifying solutions during the initial timesteps, with FILO frequently surpassing their performance afterward. Subsequently, the various adaptations of HGS consistently yield improved solutions. Variants with a smaller population demonstrate quicker solution discovery, whereas those with a larger population tend to achieve superior solutions overall. Over a prolonged timeframe,

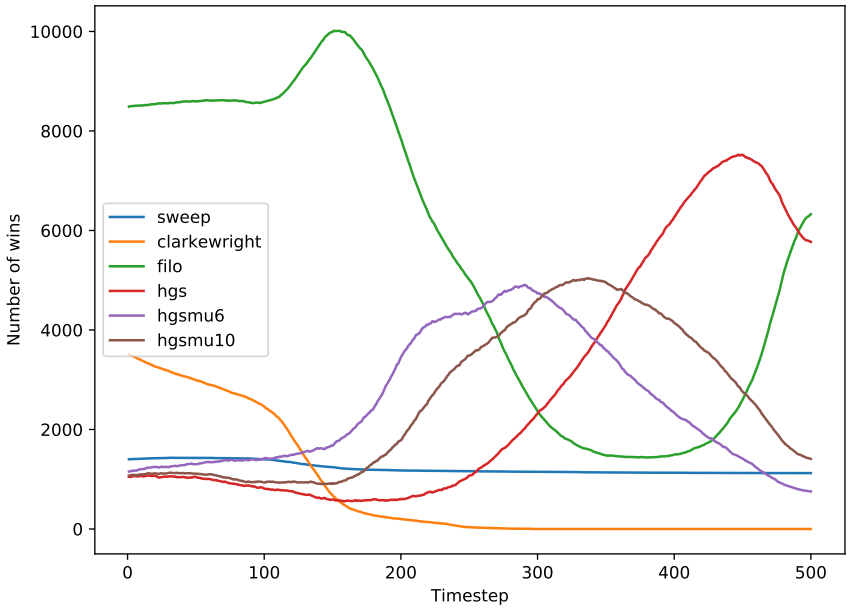


Figure 4.5: Number of times each solver in the portfolio find the best solution up to timestep t for the training set.

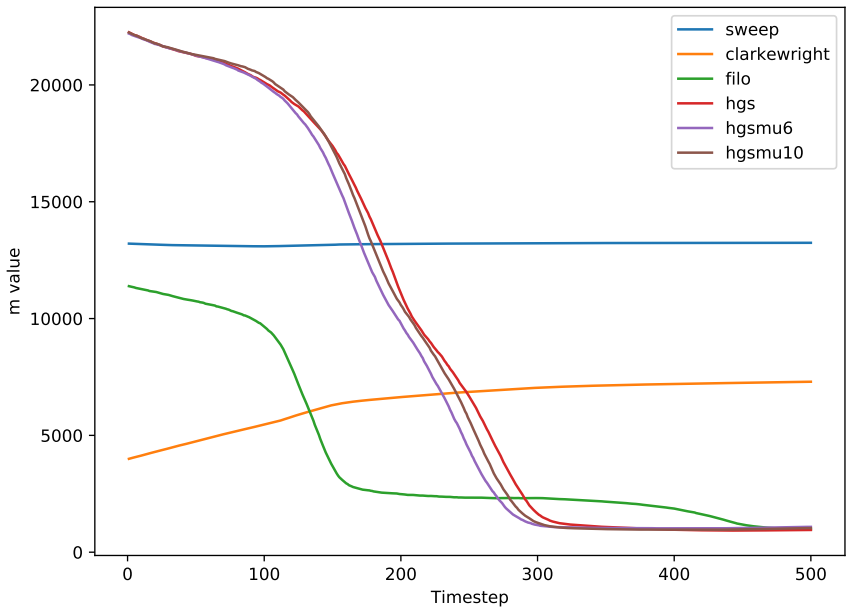


Figure 4.6: Value of m_s up to timestep t for each solver in the portfolio for the training set.

FILO emerges as the top-performing solver in the majority of instances, underscoring its efficacy with ample computational time.

In Figure 4.5, we can observe the number of times each algorithm finds the best solution among all algorithms for each timestep. We can see that `FILO` is the dominating solver up to timestep 260. Subsequently, `hgsmu6`, `hgsmu10`, and `hgs` manage to become the dominant solvers until the final timesteps where `FILO` ends up being the solver with the highest number of wins, narrowly surpassing `HGS`.

In Figure 4.6, we can observe the performance of each solver concerning the metric m_s defined in Equation 2.3. In this graph, a lower value of m_s represents better performance. Initially, up to timestep 130, the `CW` algorithm achieves the best performance according to this metric. Then, `FILO` is on average the best algorithm until timestep 280. Afterward, the three versions of `hgs` manage to obtain a lower value of m_s, until timestep 455, where `FILO` also manages to compete with them.

4.3 Testing dataset

The instances in the testing set are drawn from publicly available benchmarks sourced from CVRPLIB. The incorporated benchmarks encompass sets A, B, P, E, F M, CMT, Golden, Li, Uchoa, and AGS, totaling 264 instances.

Dataset	# Instances	Avg. Customers	Avg. Demands	Avg. Truck Capacities
A	27	49.44	13.27	100.0
AGS	10	12201.0	1.7	96.0
B	23	51.04	12.91	100.0
CMT	14	114.43	15.24	185.71
E	11	60.45	126.25	2192.0
F	3	84.0	542.38	11406.67
Golden	20	348.4	37.93	777.5
Li	12	844.33	19.98	1600.0
M	5	154.6	15.25	200.0
P	24	48.21	36.63	269.38
tai	13	130.62	132.85	1535.23
Uchoa	100	413.22	29.36	324.57

Table 4.1: Summary Statistics of Testing Dataset: Average Number of Customers, Demands, and Truck Capacities per Instance

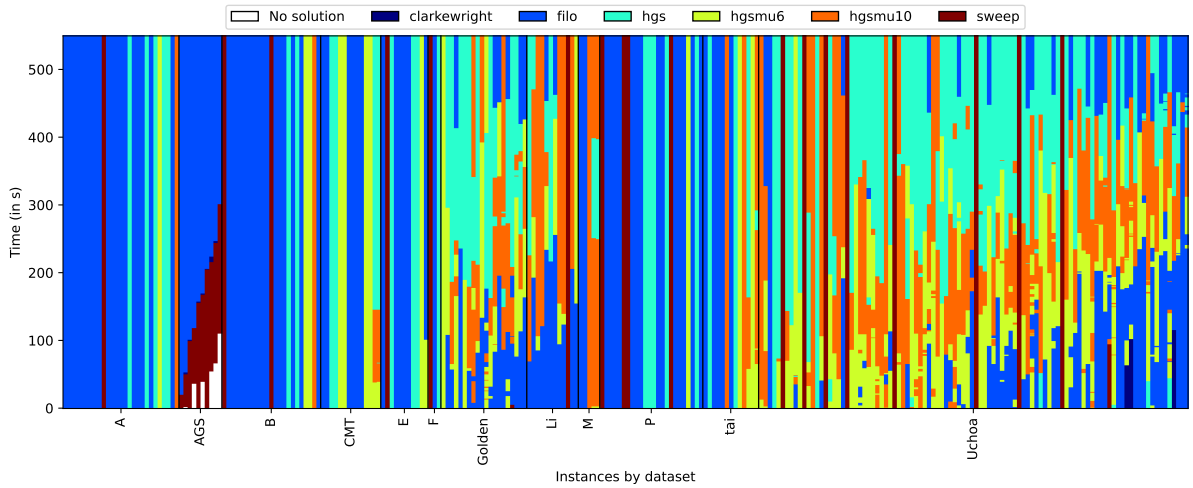


Figure 4.7: Best solver for each instance at every timestep for instances collected from CVRPLIB.

4.4 Characterization and labeling

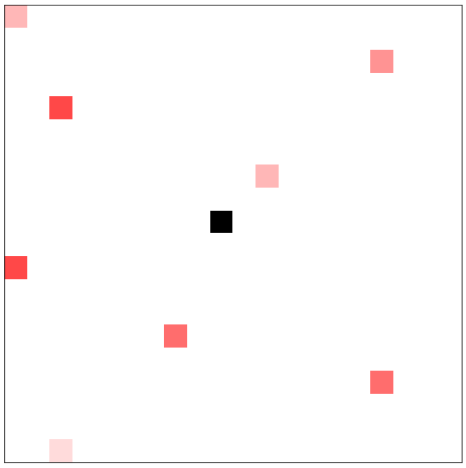
We present two possible characterizations for a CVRP instance. The first characterization entails a single matrix designed for its utilization in a Convolutional Neural Network (CNN), encapsulating the entirety of the instance. The second characterization consists of a handcrafted feature vector employed as input for Random Forest (RF), AdaBoost (AB), and Gradient Boosting (GB) models.

4.4.1 Matrix characterization

The first characterization involves the creation of a matrix designed to serve as a two-dimensional "image" of the CVRP instance. In this matrix, a distinct cell containing a value of -1 designates the location of the depot, serving as a spatial reference point. Each remaining cell in the matrix corresponds to a specific geographical position and reflects the demand of the respective clients in that cell. Notably, the demand values are normalized by dividing them by the capacity of the vehicles, providing a scaled representation of client demands relative to vehicle capabilities within the CVRP instance. This matrix-based approach enables the utilization of CNNs for effective pattern recognition and feature extraction from the visualized CVRP instance.

For example, the following instance would be represented by the next image:

```
NAME: CVRP Example
TYPE: CVRP
DIMENSION: 10
EDGE_WEIGHT_TYPE: EUC_2D
CAPACITY: 40
NODE_COORD_SECTION
1 50 50
2 40 60
3 60 70
4 70 40
5 30 20
6 20 80
7 90 20
8 10 10
9 80 80
10 60 10
DEMAND_SECTION
1 0
2 20
3 30
4 10
5 5
6 15
7 25
8 20
9 10
10 5
DEPOT_SECTION
1
-1
EOF
```



4.4.2 Domain-specific features for CVRP

Based on previous work on CVRP algorithm selection [13], we defined a set of features for our problem. For this selection, considering the anytime nature of our meta-solver, we focus on informative fast-to-compute features. These features are the following:

Timestep: Signifies the designated time limit within which algorithms are expected to compute.

Capacity: Denotes the capacity of the vehicles.

Minimum Demand: Represents the minimum demand value among all clients.

Maximum Demand: Represents the maximum demand value among all clients.

Dimension: Refers to the total number of clients within the instance.

SD of Demand: the standard deviation of the demands, normalized by the capacity.

Distance of the Depot to the Center: Measures the distance from the depot to the central coordinates of the instance plane, divided by the maximum distance between any client and the depot.

Min. Number of Vehicles: Is the ratio of the total demand and the vehicle capacity.

Avg. Clients per Vehicle: Is the ratio of the Dimension and the number of vehicles.

Max. Demand Normalized: Is the maximum demand normalized by the capacity.

Average Demand Normalized: Is the average demand, normalized by the capacity.

k-Circular Convolution of Clients: A mechanism capturing customer distribution using k-concentric circles centered at the depot. The radius of the last circle equals the distance from the depot to the farthest customer, r_{max}. For each of the $k-1$ preceding circles with radii r_i, $i = 1...k-1$, the radius is computed as $\frac{i}{k} \cdot r_{max}$. The percentage of customers within the ring between circles i and $i-1$ is determined for each circle $i > 0$. Here, we set $k = 10$ for this computation.

k-Circular Convolution of Demands: Similar to the k-circular convolution of clients, this computation involves determining the percentage of the total demand within each ring.

Underlined features were found in the previous work of [13].

4.4.3 Ground truth labeling for the models

Various solvers were employed in constructing the meta-solver, serving as labels to identify the most suitable solver for a given instance-time pair. A "no solution" label is incorporated to signify instances where no solver achieves a feasible solution within a specific timestep. This designation proves valuable, particularly for challenging instances where solvers necessitate extended durations to compute the initial feasible solution. In practical terms, a "no solution" label can alert users to allocate additional computational resources for problem resolution.

Despite the potential disparity between accuracy and $\hat{m}$ metrics, our strategy involves training multi-label classification models that prioritize accuracy. This choice is motivated by the imperative of deploying straightforward and swift machine learning models for anytime scenarios. Utilizing a multi-class classification model confers the advantage of considering all solvers simultaneously, enabling a singular oracle call to inform decision-making. This contrasts with more intricate Algorithm Selection Systems, which often entail multiple machine learning oracles, such as numerous binary classification models for each pair of alternatives or regression models for individual solvers. Implementing such intricate systems would lead to excessively prolonged prediction times, incompatible with the expeditious nature required for our anytime scenario.

4.5 Machine Learning Models

4.5.1 Models using vector of features

In our investigation, we carried out experiments employing three machine learning algorithms. The implementation of these algorithms was executed using the Scikit-learn library [27]. We conducted hyperparameter tuning to ascertain the optimal architecture for each model, and also employed weighting strategies to address the inherent bias stemming from the dominating classes within the portfolio.

The tested hyperparameters for each algorithm can be observed in Table 4.2. Any other hyperparameter not mentioned maintains its default value.

Table 4.2: Hyperparameters for Different Algorithms

Algorithm	**Hyperparameters**
Random Forest	n_estimators: 50, 100, 150 class_weight: None, balanced, balanced_subsample
Gradient Boosting	learning_rate: 0.05, 0.1 max_features: log2, sqrt subsample: 0.8, 1.0
AdaBoost	learning_rate: 0.05, 0.1 algorithm: SAMME, SAMME.R n_estimators: 30, 50, 100

RF: The hyperparameters used were: `n_estimators = 100`, `max_features = "sqrt"`, `class_weight = "balanced"` .

AB: The hyperparameters used were: `algorithm = "SAMME"`, `learning_rate = 1.0`, `n_estimators = 50`.

GB: The hyperparameters used were: `learning_rate = 0.1`, `max_features = "sqrt"`, `sumsample = 0.8`.

4.5.2 CNN using matrix representation

The matrix representation detailed in Section 4.4.1 serves as the input for the CNN, generating the optimal solver output for every timestep of the instance. To accommodate

anytime scenarios, we modify the methodology to train the network to predict a label for each of the 500 possible timesteps. Consequently, when a prediction for a specific time is required, we examine the network's output corresponding to the closest (smaller or equal) timestep prediction. The architecture employed for the CNN is based on AlexNet.

Chapter 5

RESULTS

5.1 Meta-solver's performance

Table 5.1 shows the accuracy and $\hat{m}$ metric values for our four ML models. These values correspond to the performance of the models on the testing set. Larger values for accuracy indicate better performance and lower values for the $\hat{m}$ metric are better.

ML Oracle	**Accuracy**	**Metric** $\hat{m}$
Alexnet	0.3855	0.6254
GB	0.4195	0.6009
AB	0.3809	0.8977
RF	0.4445	0.8023

Table 5.1: Accuracy and $\hat{m}$ values for different Machine Learning Models. For accuracy, higher values are better, and for $\hat{m}$, lower values are better.

The Random Forest model achieves higher accuracy, whereas the Gradient Boosting model exhibits a superior value for m-hat, ranking second in accuracy. This illustrates that, in this particular case, accuracy (number of wins) is not necessarily fully correlated with m-hat. This is so because for a given timestep, two or more solver may have found the same best-value solution, but the win is assigned to the solver that found it first. Also, it may be the case that even if a solver finds the best-solution, the second-best solver finds a solution that is very close to it. These cases are not reflected in the accuracy metric, but somehow influence the $\hat{m}$ value.

Since we privilege performance on the $\hat{m}$ metric, we select, for further comparison, the top 2 best-performing models with respect to this metric (Alexnet and GB), and compare their results with the individual outcomes of each solver.

In Figures 5.1 and 5.2, we compare the individual performance of each algorithm, as well as the performance of the two selected algorithms mentioned earlier, disregarding the time taken to make predictions. In Figure 5.1, we observe that both selection models exhibit superior initial performance in terms of the number of wins up to timestep 100. From timestep 100 to 160, `hgsmu10` outperforms all others individually. Subsequently, starting from timestep 160, the CNN achieves a higher number of wins over all solvers, and at timestep 300, GB manages to obtain more wins than each individual algorithm. From timestep 350 to 420, both selection models exhibit similar performance, and in the final timesteps, GB achieves a higher number of victories.

In Figure 5.2, we observe that all three versions of HGS achieve a low initial m value compared to the rest of the solvers and algorithms. Around timestep 150, both prediction algorithms manage to achieve a lower m value than each algorithm individually. Between timesteps 220 and 320, CNN is the algorithm with the best m metric, and from timestep 320 onwards, GB has the lowest m value followed by CNN.

In Figures 5.3 and 5.4, we can observe the comparison between the solvers and the prediction models that consider prediction time. The algorithm selected by each model only has an available execution time equal to the total available time minus the prediction time, which includes instance characterization and the prediction itself. The results are mainly altered in the initial timesteps, where CNN is more affected due to its longer prediction time compared to GB. However, for timesteps larger than 10, the results are similar to those in the graphs that do not consider prediction time.

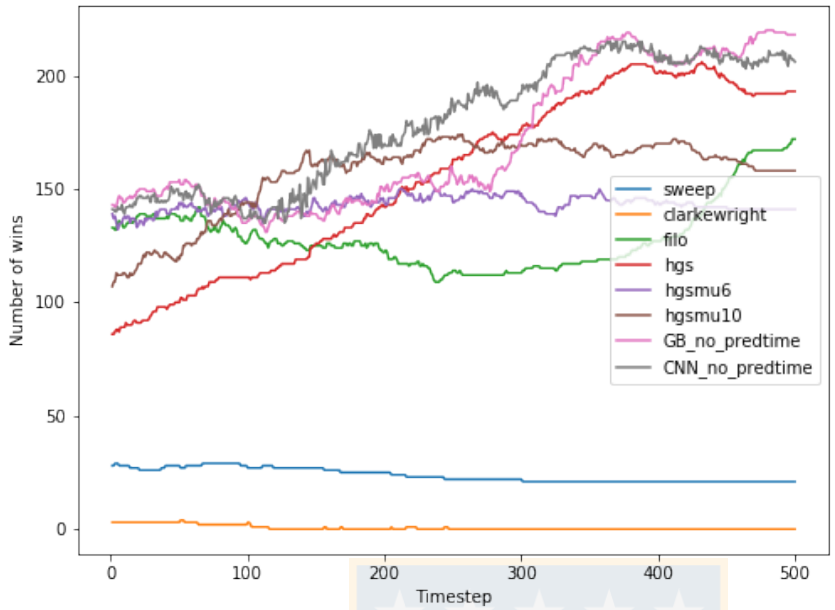


Figure 5.1: Number of times each solver in the portfolio and the two meta-solvers find the best solution up to timestep t, without considering prediction time.

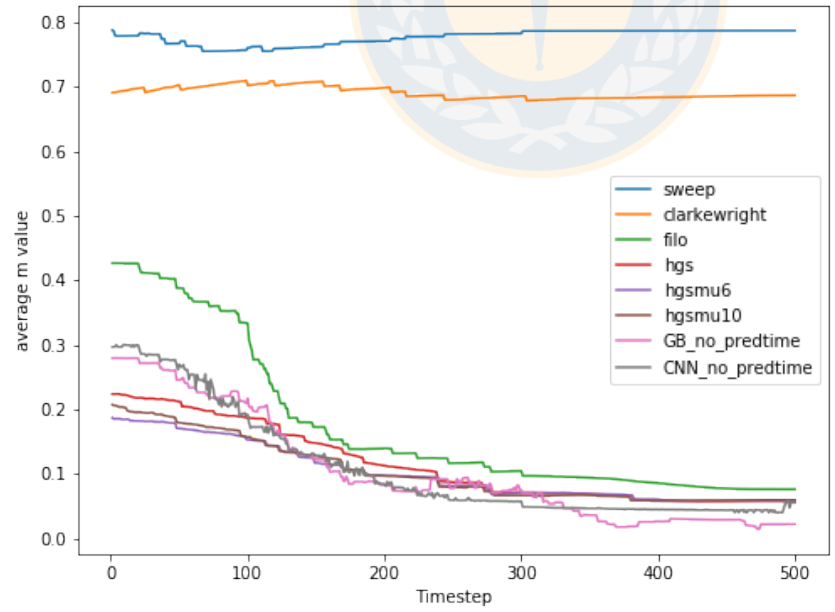


Figure 5.2: Value of m_s up to timestep t for each solver in the portfolio and meta-solvers based on GB and CNN, without considering prediction time.

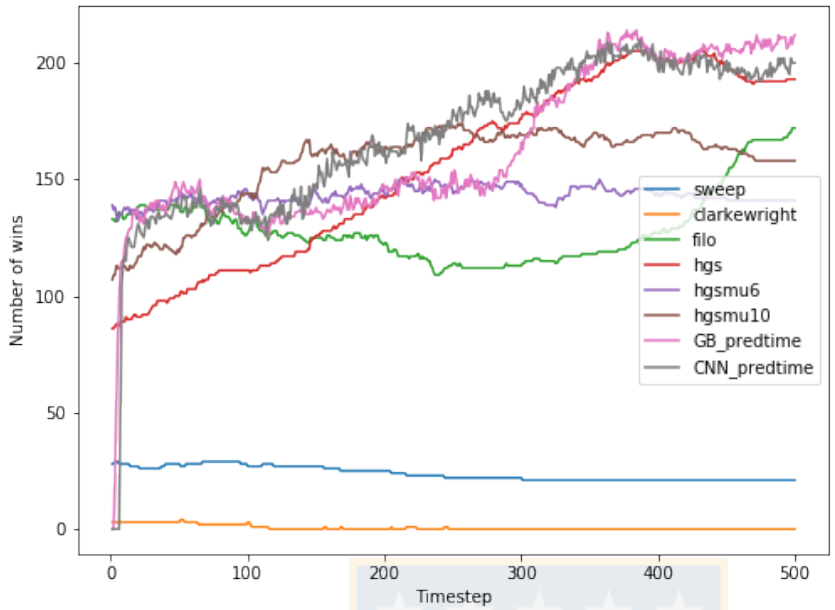


Figure 5.3: Number of times the solver finds the best solution up to timestep t, including the prediction time for the algorithm selector.

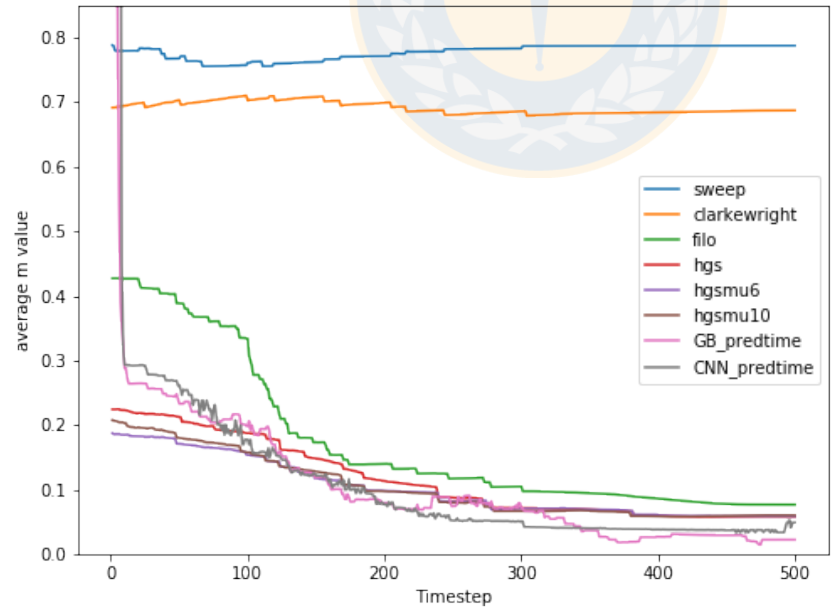


Figure 5.4: Value of m_s up to timestep t, including the prediction time for the algorithm selector.

Chapter 6

CONCLUSIONS AND FUTURE WORK

This study successfully implements an AAAS model tailored for CVRP, highlighting its potential to improve solution quality within given computational time limits. The model demonstrates an ability to select the most appropriate solver based on the characteristics of the problem instance and the available computation time, effectively outperforming the Single Best Solver in the portfolio. This research innovatively employs various machine learning techniques, including Convolutional Neural Networks and ensemble methods like Random Forest, Gradient Boosting, and AdaBoost. These techniques effectively predict the best solver for specific CVRP instances, considering the constraints on computational resources.

A significant achievement of this study is the generation of a diverse set of CVRP instances and the meticulous evaluation of solvers across these instances. This approach not only ensures the robustness of the AAAS model but also contributes valuable datasets to the CVRP community for further research.

The generalization of $\hat{m}$ evaluation metric specifically designed for assessing the performance of AAAS models in the context of CVRP can also be considered as a notable contribution. This metric facilitates a nuanced understanding of the models' effectiveness in selecting algorithms that can provide the best possible solutions within time constraints.

Overall, we consider that the work developed here supports our hypothesis and evidence that all the goals of the project were accomplished.

For future work, we plan to incorporate a broader array of CVRP solvers into the portfolio, including both traditional and newly developed algorithms. This could enhance the AAAS model's adaptability and performance across a wider range of problem instances. We plan to further investigating novel methods for CVRP instance characterization that could uncover more informative features, potentially improving the prediction accuracy of the machine learning models used within the AAAS framework.

Applying the developed AAAS model to real-world logistics and routing challenges would validate its practical utility and identify areas for further refinement and optimization. Future research could explore the integration of the AAAS model with dynamic problem environments where problem parameters and constraints change over time, necessitating real-time solver selection and adaptation.

Bibliography

[1] Marwan F Abdelatti and Manbir S Sodhi. An improved gpu-accelerated heuristic technique applied to the capacitated vehicle routing problem. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pages 663–671, 2020.

[2] Luca Accorsi and Daniele Vigo. A fast and scalable heuristic for the solution of large-scale capacitated vehicle routing problems. *Transportation Science*, 55(4):832–856, 2021.

[3] Barrie M Baker and MA1951066 Ayechew. A genetic algorithm for the vehicle routing problem. *Computers & Operations Research*, 30(5):787–800, 2003.

[4] Gulay Barbarosoglu and Demet Ozgur. A tabu search algorithm for the vehicle routing problem. *Computers & Operations Research*, 26(3):255–270, 1999.

[5] Horace B Barlow. Unsupervised learning. *Neural computation*, 1(3):295–311, 1989.

[6] Bernd Bischl, Pascal Kerschke, Lars Kotthoff, Marius Lindauer, Yuri Malitsky, Alexandre Fréchette, Holger H. Hoos, Frank Hutter, Kevin Leyton-Brown, Kevin Tierney, and Joaquin Vanschoren. ASlib: A Benchmark Library for Algorithm Selection. *Artificial Intelligence Journal (AIJ)*, (237):41–58, 2016.

[7] Zuzana Borcinova. Two models of the capacitated vehicle routing problem. *Croatian Operational Research Review*, pages 463–469, 2017.

[8] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[9] Geoff Clarke and John W Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, 12(4):568–581, 1964.

[10] Pádraig Cunningham, Matthieu Cord, and Sarah Jane Delany. Supervised learning. In *Machine learning techniques for multimedia: case studies on organization and retrieval*, pages 21–49. Springer, 2008.

[11] George B Dantzig and John H Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.

[12] Richard W Eglese. Simulated annealing: a tool for operational research. *European journal of operational research*, 46(3):271–281, 1990.

[13] Justin C Fellers. *Algorithm Selection for the Capacitated Vehicle Routing Problem Using Machine Learning Classifiers*. PhD thesis, University of Rhode Island, 2021.

[14] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[15] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[16] Isaías I Huerta, Daniel A Neira, Daniel A Ortega, Vicente Varas, Julio Godoy, and Roberto Asín-Achá. Anytime automatic algorithm selection for knapsack. *Expert Systems with Applications*, 158:113613, 2020.

[17] Isaías I Huerta, Daniel A Neira, Daniel A Ortega, Vicente Varas, Julio Godoy, and Roberto Asín-Achá. Improving the state-of-the-art in the traveling salesman problem: An anytime automatic algorithm selection. *Expert Systems with Applications*, 187:115948, 2022.

[18] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[19] Pascal Kerschke, Holger H Hoos, Frank Neumann, and Heike Trautmann. Automated algorithm selection: Survey and perspectives. *Evolutionary computation*, 27(1):3–45, 2019.

[20] Pascal Kerschke, Lars Kotthoff, Jakob Bossek, Holger H Hoos, and Heike Trautmann. Leveraging tsp solver complementarity through machine learning. *Evolutionary computation*, 26(4):597–620, 2018.

[21] Sotiris B Kotsiantis, Ioannis Zaharakis, P Pintelas, et al. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160(1):3–24, 2007.

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[23] Max Kuhn, Kjell Johnson, et al. *Applied predictive modeling*, volume 26. Springer, 2013.

[24] Marius Lindauer, Jan N van Rijn, and Lars Kotthoff. The algorithm selection competitions 2015 and 2017. *Artificial Intelligence*, 272:86–100, 2019.

[25] Denis Naddef and Giovanni Rinaldi. Branch-and-cut algorithms for the capacitated vrp. In *The vehicle routing problem*, pages 53–84. SIAM, 2002.

[26] Gunadi W Nurcahyo, Rose Alinda Alias, Siti Mariyam Shamsuddin, and Mohd Noor Mohd Sap. Sweep algorithm in vehicle routing problem for public transport. *Jurnal Antarabangsa Teknologi Maklumat*, 2:51–64, 2002.

[27] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[28] Meghan Steinhaus, Arash Nasrolahi Shirazi, and Manbir Sodhi. Modified self organizing neural network algorithm for solving the vehicle routing problem. In *2015 IEEE 18th International Conference on Computational Science and Engineering*, pages 246–252. IEEE, 2015.

[29] Paolo Toth and Daniele Vigo. Branch-and-bound algorithms for the capacitated vrp. In *The vehicle routing problem*, pages 29–51. SIAM, 2002.

[30] Thibaut Vidal. Hybrid genetic search for the cvrp: Open-source implementation and swap* neighborhood. *Computers & Operations Research*, 140:105643, 2022.