Metaheuristic Enhanced with Feature-Based Guidance and Diversity Management for Solving the Capacitated Vehicle Routing Problem

Bachtiar Herdianto, Romain Billot, Flavien Lucas, Marc Sevaux

Metaheuristic Enhanced with Feature-Based Guidance and Diversity Management for Solving the Capacitated Vehicle Routing Problem

Bachtiar Herdianto*[1], Romain Billot[1], Flavien Lucas[2], and Marc Sevaux[3]

[1] *IMT Atlantique, Lab-STICC (UMR 6285, CNRS), Brest, France*
[2] *IMT Nord Europe, CERI Systèmes Numériques, Douai, France*
[3] *Université Bretagne Sud, Lab-STICC (UMR 6285, CNRS), Lorient, France*

bachtiar.herdianto@imt-atlantique.fr*[1]
romain.billot@imt-atlantique.fr[1]
flavien.lucas@imt-nord-europe.fr[2]
marc.sevaux@univ-ubs.fr[3]

Abstract

We propose a metaheuristic algorithm enhanced with feature-based guidance that is designed to solve the Capacitated Vehicle Routing Problem (CVRP). To formulate the proposed guidance, we developed and explained a supervised Machine Learning (ML) model, that is used to formulate the guidance and control the diversity of the solution during the optimization process. We propose a metaheuristic algorithm combining neighborhood search and a novel mechanism of hybrid split and path relinking to implement the proposed guidance. The proposed guidance has proven to give a statistically significant improvement to the proposed metaheuristic algorithm when solving CVRP. Moreover, the proposed guided metaheuristic is also capable of producing competitive solutions among state-of-the-art metaheuristic algorithms.

Keywords— Metaheuristic Algorithm, Supervised Machine Learning, Explainable Artificial Intelligence (XAI), Capacitated Vehicle Routing Problem

1 Introduction

Routing represents a significant activity in logistics and supply chains, involving the movement of products or goods, from one location to another. This process is crucial for driving economic and social activities, impacting various aspects of daily lives (Arnold and Sörensen, 2019b). The main challenge arises from the increasing delivery costs, that may directly influence the pricing of goods. Optimizing the delivery routes becomes one significant solution for mitigating this problem (Simchi-Levi, Kaminsky, and Simchi-Levi, 2002). Various routing problems exist, with one of the most studied being the Capacitated Vehicle Routing Problem (CVRP) (Toth and Vigo, 2014; Sörensen and Schittekat, 2013; Prodhon and Prins, 2016; Arnold and Sörensen, 2019a; Accorsi and Vigo, 2021). The first research that attempted to solve the CVRP was performed by Dantzig, Fulkerson, and Johnson, 1954. Yet, despite decades of study, the CVRP remains a challenging problem in laboratory and industrial applications (Prins, 2004; Laporte, 2009).

*Corresponding Author

Recently, there has been a growing interest in using Machine Learning (ML) to enhance optimization algorithms (Bengio, Lodi, and Prouvost, 2021). However, many optimization algorithms start from scratch for similar problem types, ignoring valuable insights from previous solutions. Utilizing historical data could offer efficient and effective ways to improve optimization algorithms Arnold and Sörensen, 2019b. Furthermore, the optimization algorithm can learn from its own decisions, adapting its behavior for improved performance. In parallel, Explainable Artificial intelligence (XAI) offers techniques that can identify the strongest features, as well as investigate how these features behave for making a decision (Lundberg and Lee, 2017; Arrieta et al., 2020; Lundberg et al., 2020).

In this research, we aim to solve CVRP by developing a hybrid ML and metaheuristic algorithm. Our approach involves developing a learning model that can learn how to achieve an optimal quality solution, based on the problem features. Subsequently, we try to interpret the developed learning model and use these insights to formulate a guidance able to boost the performance of the metaheuristic algorithm.

1.1 Related Work

The CVRP can be characterized as an undirected graph $G = (V, E)$. The set of nodes V is composed of a depot D and a collection of customer nodes C, where $C = c_1, c_2, ..., c_N$ and $N = V - 1$ represents the set of customers (excluding the depot). Then, the set of customers neighboring c_i can be referred to as the neighborhood of c_i, denoted as $\mathcal{N}(c_i)$ (Prodhon and Prins, 2016). The CVRP is an extended version of the Vehicle Routing Problem (VRP). In CVRP, a set of valid routes R is defined as a collection of routes, where a route can be defined as a sequence of nodes, with the first and last nodes being the depot D and the remaining nodes representing customers c_k, in which the total demand of customer nodes does not exceed the capacity Q of identical vehicles in the fleet. Therefore, a CVRP solution is considered feasible when it consists of valid routes, ensuring each customer is visited exactly once. The CVRP aims to determine a feasible solution that minimizes the sum of route costs for all routes in the solution (Laporte, 2009).

Metaheuristics for solving the CVRP Heuristics and local search mechanisms are the main components of metaheuristics (Prodhon and Prins, 2016). In Glover, 1997, a tabu search heuristic was proposed, allowing the algorithm to escape local optima by preventing cyclic evaluation. Following this, Granular Neighborhoods mechanism (GNs), proposed by Toth and Vigo, 2003, defines a heuristic filtering of less promising neighbors. This mechanism, when combined with the tabu search, has empirically demonstrated an excellent trade-off between computation time and solution quality (Toth and Vigo, 2003), even with very large scale problems, up to 30,000 customers (Accorsi and Vigo, 2021). Another way for enhancing tabu search involves bolstering the intensification mechanism while preserving solution diversity through the integration of the path relinking (Glover, 1997; Glover, Laguna, and Marti, 2000) procedure. In parallel, Prins, 2004 introduced a new representation of the solution, called the giant tour, and a splitting mechanism to transform it into a VRP solution. In Prins, 2004, this was implemented alongside population-based metaheuristic, and further refined and applied to solve a broad range of VRP variants (Vidal et al., 2014), and has particularly proven to effectively solve the CVRP (Vidal, 2022) up to 1,000 customers. Path relinking, as proposed by Glover, 1997, has been proven to enhance the intensification strategy of tabu search (Laguna, Martí, and Campos, 1999; Ho and Gendreau, 2006). In Sörensen and Schittekat, 2013, a metaheuristic algorithm is proposed to hybridize path relinking with GRASP (Greedy Randomized Adaptive Search Procedure) and VND (Variable Neighborhood Descent). However this study highlights a limited contribution of path relinking compared with the simple hybridization of GRASP and VND.

Hybridizing machine learning with metaheuristic for solving CVRP The integration of machine learning (ML) and optimization algorithms can be classified into three strategies, as outlined by Bengio, Lodi, and Prouvost, 2021: (1) end-to-end learning, (2) learning based on problem properties, and (3) learning repeated decisions. The concept of learning based on problem properties entails utilizing ML to formulate a configuration (in a broad sense) for the optimization algorithm. Meanwhile, the mechanism of learning repeated decisions is applied by constructing an in-loop ML-assisted optimization algorithm to allow the algorithm to learn from its own decisions and adapt its behavior. Recent studies have aimed to enhance the performance of metaheuristics by hybridizing them with ML when solving the CVRP.

The hybridizing mechanism may initially involves by analyzing the structure of the parent problem of the CVRP, that is the VRP, through statistic analysis and classification model (Arnold and Sörensen, 2019b). Arnold and Sörensen, 2019b, and then further analyzed in more depth by Lucas, Billot, and Sevaux, 2019, propose a classification model to classify between near-optimal and non-optimal solutions by extracting useful features, which could be used to reduce search space to potentially good solutions. The method the employ not only uses the structure of the instances (i.e., instance feature) but also respects the structure of the resulting solution (i.e., solution features). Later on, Arnold and Sörensen, 2019a leveraged the extracted knowledge to improve their proposed algorithm for solving the CVRP and even extend it to solve large-scale CVRP (Arnold, Gendreau, and Sörensen, 2019). However, Arnold and Sörensen, 2019a did not provide a statistical analysis of the contribution of application guidance to the proposed algorithm. Apart from that, Li, Yan, and Wu, 2021 and Xin et al., 2021 attempted to accelerate and improve the Lin-Kernighan heuristic (LKH-3) (Helsgaun, 2017) by hybridizing with ML model. The proposed hybrid model is able to accelerate and improve the baseline algorithm. However, they did not provide a statistical confirmation of the significant improvement between algorithms, most of the proposed learning models use instance-based features. Nonetheless, according to Arnold and Sörensen, 2019b and Lucas, Billot, and Sevaux, 2019, the solution-based features tend to be stronger than the instance feature for making the prediction, indicating a future opportunity for enhancing the proposed hybrid ML model.

1.2 Research Questions and Contributions

In Arnold and Sörensen, 2019a, the authors have shown that clearly defining the preferred structural properties of an optimal VRP solution is highly valuable for designing an efficient heuristic. Furthermore, with more interpretation by using an explainable learning model, we can change our way of solving VRP (Lucas, Billot, and Sevaux, 2019). An ML model, through the learning phase, builds predictive models that can map data features into a class (Guidotti et al., 2018). Then, the explanation of these models gives insights into how the models utilize features to make decisions. As Guidotti et al., 2018 describes, the ranking of the relative importance of the problem attributes can be incorporated into guidance rules. Inspired by those advances, in this research, our main questions are:

1. How can we extract the most important features of a good solution and use them to guide a heuristic?
2. To what extent does the developed heuristic bring a significant improvement for solving the CVRP?

To address these questions, we propose a simple mechanism of hybridization between ML and metaheuristics by adopting the concept of "learning to configure algorithms". Here, we introduce an explainable learning framework for classifying the quality of the VRP solution. This is done by generating a dataset of VRP features and developing a classification model that can identify the features that have the most significant influence, then explaining the developed model to study the feature that influences the quality of the solution. Furthermore, we introduce a metaheuristic for solving the CVRP that consists of neighborhood search and path relinking. In our proposed metaheuristic, we also present a novel mechanism of path relinking for solving CVRP that hybridizes with the split algorithm. Ultimately, we present a feature-based guidance applied to the proposed metaheuristic. In summary, the main design steps and contributions of this research are the following:

1. We generate a dataset of features related to the instances and solutions of VRP. This dataset contains optimal solutions and near-optimal solutions. This dataset is generated by solving all 10,000 𝕏𝕄𝕃100 instances, introduced in Queiroga et al., 2021.
2. We develop and explain a machine learning model that classifies a solution as good or not based on its features. By explaining this model, the most important features are identified.
3. Based on this knowledge, we formulate a guidance for managing the diversity of the solution during the optimization processes whenever the metaheuristic algorithm solves a problem.
4. For implementing our proposed guidance, we develop a new metaheuristic algorithm for solving CVRP. Our proposed algorithm is composed of neighborhood improvement and path relinking.

5. In our proposed metaheuristic, we propose a novel mechanism of path relinking for solving the CVRP. Our proposed path relinking mechanism compromises with the giant tour concatenation and hybridizes with the split algorithm. We show that our proposed path relinking mechanism can contribute statistically significantly to the proposed algorithm's overall mechanism but also compete with state-of-the-art algorithms.
6. Based on a computational experiment, we also show that our proposed guidance can significantly improve our proposed algorithm.

Lastly, the rest of this paper is structured as follows: In Section 2, we provide a detailed learning framework and explain how the model can classify between optimal and near-optimal solutions. In Section 3, we describe our proposed metaheuristic algorithm for solving the CVRP. Moving forward, Section 4 focuses on how we hybridize our proposed metaheuristic algorithm with our guidance, resulting from our explained learning model. Lastly, section 5 shows our experimentation to evaluate our hybridization mechanism. We also measure the performance of the proposed hybridized algorithm with the state-of-the-art metaheuristic algorithm for solving the CVRP.

2 Learning From Solutions

Here, let assume $\mathcal{X}$ as the set training samples, with size N, that consist of a set of problem features $x^{(i)}$ and its label $y^{(i)}$, such that: $x^{(i)}, y^{(i)} \in \mathcal{X}^{(i)}$, where $i \in N$. For every $x \in \mathcal{X}$, we have a set of $p$features, such that $x = [x_1, \ldots, x_p]$. Here, the label $y \in \mathcal{X}$ represents the quality of the solution as a binary variable, such as:

$$y = \begin{cases} 1 & \text{if it corresponds to an optimal solution} \\ 0 & \text{if it correspond to a near-optimal solution} \end{cases} \tag{1}$$

Then, the aim of the learning model $f(x)$ is to classify between optimal and near-optimal solutions, such as:

$$f(x) = \begin{cases} 1 & \text{if it corresponds to an optimal solution} \\ 0 & \text{if it corresponds to a near-optimal solution} \end{cases} \tag{2}$$

In this paper, our proposed methodology to develop a learning model is composed of three main steps: (1) generating a dataset, (2) performing classification using several learning models, and (3) explaining the developed learning model by using SHAP values (as shown in Figure 1). The dataset alongside the source code and the documentation for performing the analysis can be downloaded at https://github.com/bachtiarherdianto/MS-Feature.

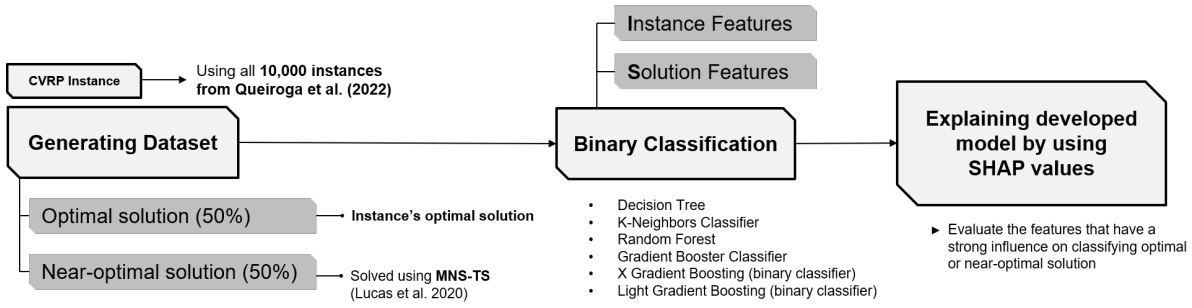

Figure 1: Outline of the learning framework

2.1 Data Generations and Feature Extractions

The CVRP has a great variety of instances according to the following attributes (Uchoa et al., 2017): (1) the positioning and number of customers, (2) the positioning of the depot, (3) the distribution of demand, and (4) the average route size or the number of routes. In this research, we use the 10,000 $\mathbb{XML}$100 instances[1] from Queiroga et al., 2021 to generate a dataset that is used to develop a learning model $f(x)$.

[1]Detailed information related to the problem instances is available at http://vrp.galgos.inf.puc-rio.br/index.php/en/

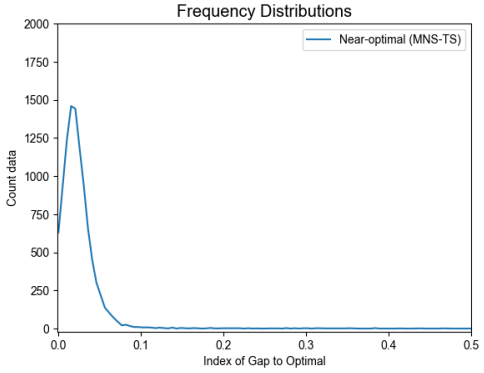

Figure 2: The summary of solutions obtained from the MNS-TS algorithm with a maximum time limit of $T_{max} = 10$ seconds. It reveals that the majority of instances were solved with a gap to the optimal solution ranging from 0% to 5%. Furthermore, three instances were solved with a gap to the optimal solution exceeding 45%.

In parallel, for near-optimal solution, we use MNS-TS algorithm (Soto et al., 2017; Lucas et al., 2020) for solving all $\mathbb{XML}100$ instances. The MNS-TS, which is a short term for *Multiple Neighborhood Search with Tabu Search*, has already demonstrated good capabilities to solve the Open Vehicle Routing Problem (OVRP) Soto et al., 2017 and large size instances of CVRP Lucas et al., 2020 up to 30,000 customer nodes. For obtaining a near-optimal solution, we set $T_{max} = 10$ seconds, meaning that the MNS-TS algorithm solved every $\mathbb{XML}100$ instance with a time limit of 10 seconds in five runs. The relative difference between the solution obtained by the MNS-TS with $T_{max} = 10$ seconds and the objective value from the optimal solution is represented as the gap-to-optimal solution. The gap-to-optimal value is calculated as follows:

$$\text{Gap-to-Optimal} = \frac{\text{Obtained Solution} - \text{Optimal Solution}}{\text{Optimal Solution}} \times 100\% \tag{3}$$

As shown in Figure 2, the dataset used for developing the learning model consists of 20,000 data points, where 50% of them are optimal solutions, and the remaining are near-optimal solutions.

Feature Extractions We organize the features into two groups for developing the proposed learning model: (1) instances features and solution features. The instance features are utilized from Arnold and Sörensen, 2019b and Lucas et al., 2020. The solution features are mainly inspired by Arnold and Sörensen, 2019b and Lucas et al., 2020 (from Equation (20) to Equation (37)). Here, we propose two new solutions feature for the analysis, that is, Equation (38) and Equation (39). A detailed explanation of features is described in Appendix A.1. Then, a detailed preliminary statistical analysis of these features is described in Appendix A.2.

2.2 Learning Model: Binary Classification

Several supervised classification algorithms have been developed, such as K-Nearest Neighbors (Fix, 1985; Cover and Hart, 1967), Decision Tree Classifier (Breiman et al., 2017), and Random Forest (Breiman, 2001). Some boosting algorithms for classification, such as the Gradient Boosting classifier (Friedman, 2001), Extreme-Gradient Boosting (X-Gradient Boosting) (Chen and Guestrin, 2016), and Light Gradient Boosting (Ke et al., 2017) have also been tested. Those algorithms were fed using our dataset of VRP features, implemented in Python, and performed on a 64-bit mini-computer with an AMD Ryzen 7 PRO 5850U processor and 16 GB RAM running on Ubuntu 22.04.1 operating system. Based on a classical train/test procedure, the performances are compared to the F_1-score. As described by Sokolova, Japkowicz, and Szpakowicz, 2006, the F_1-score can be calculated as:

$$F_1\text{-score} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{4}$$

The full results of the comparison supervised classification algorithm are shown in Table 1, where the precision and recall are complementary metrics used to measure the performance of the classification model, in particular the binary classification model (Sokolova, Japkowicz, and Szpakowicz, 2006).

Table 1: F_1-scores from various classification algorithms used in this proposed model

Algorithm	**Precision**	**Recall**	F_1**-score**
K-Nearest Neighbors Classifier	0.476	0.351	0.404
Decision Tree Classifier	0.537	0.538	0.538
Random Forest Classifier	0.523	0.500	0.511
Gradient Boosting Classifier	**0.672**	**0.661**	**0.666**
X-Gradient Boosting Classifier	0.632	0.621	0.627
Light Gradient Boosting Classifier	0.652	0.652	0.652

From Table 1, we can see that the Gradient Boosting classifier resulted in the highest F_1-score, 0.666. This value is quite low but the prediction performance is not the objectify of our approach that aims at identifying some strategic features. Thus, in the following section, we will explain how the Gradient Boosting classifier makes predictions, particularly focusing on identifying the behavior of the features.

2.3 Explaining The Learning Model

Explainable AI offers necessary insights into how a developed AI system learns, makes decisions, and represents information (Arrieta et al., 2020). Various approaches exist to achieve model explainability, as described in Arrieta et al., 2020. One notable approach is the SHAP model (SHapley Additive exPlanations) (Lundberg and Lee, 2017; Lundberg et al., 2020; Baptista, Goebel, and Henriques, 2022). The SHAP calculates the impact of each feature on the predictions made by the trained model Lundberg and Lee, 2017. As shown in Table 1, here we will calculate the SHAP value of the Gradient Boosting classifier. The distribution of the SHAP values for each feature from the Gradient Boosting classifier is illustrated in Figure 3.

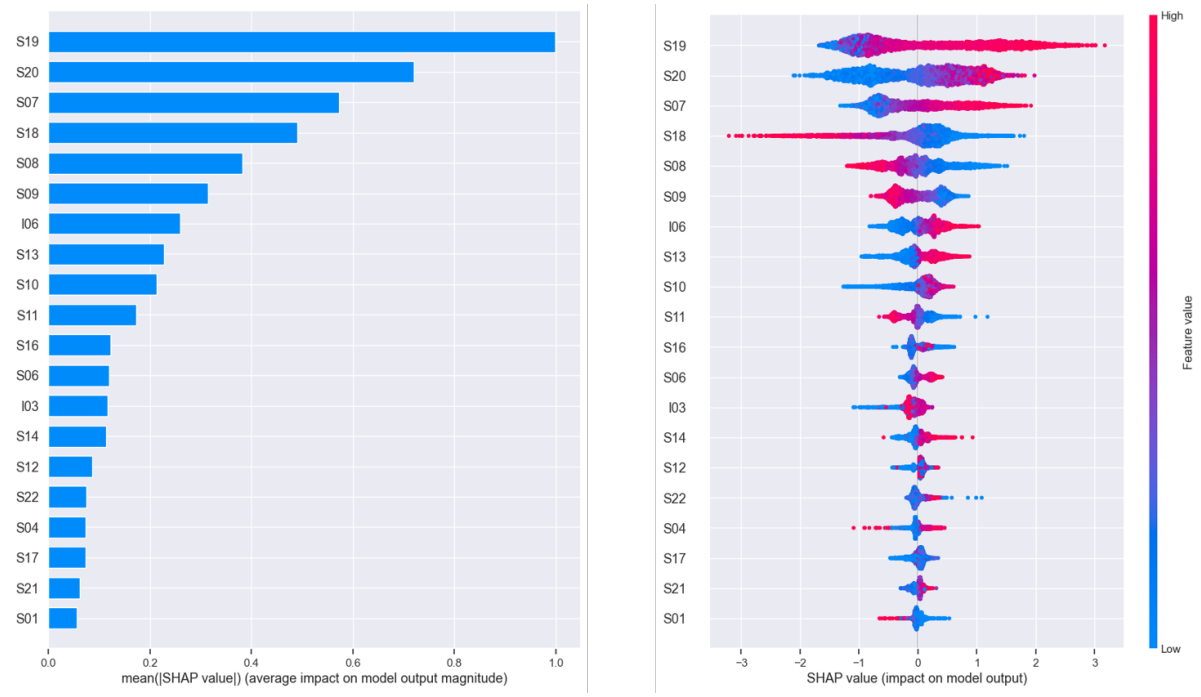

Figure 3: The global feature importance plot (left) and the local explanation summary plot (right) from the learning model

As the order of the displayed features is the order of the highest SHAP value, from Figure 3, we can conclude that S19 (the average capacity utilization of obtained solution) and S20 (the standard deviation of capacity utilization of obtained solution) have stronger influence among all other features. Here, the horizontal axis represents the SHAP value. The larger the SHAP value in the positive direction (resp. negative), the larger the contribution of the feature in the positive direction (resp. negative), and vice versa. The color represents the largeness of a feature value. The color becomes red as the value of the feature increases and turns blue as the value decreases. Concerning feature S19, the higher the value of the feature, the larger the SHAP value in the positive direction. This can be interpreted as the higher the average capacity utilization of a solution, the higher the quality of the solution. Meanwhile, the lower values of S20 have negative SHAP values, and higher values of S20 have positive SHAP. However, it also shows that, in particular, lower SHAP values of S20 have positive SHAP values. It also shows that lower S20 extend further towards the negative side of SHAP values, suggesting high S20 has a stronger negative impact on the quality of solution than the positive impact of high S20 on the quality of solution. From this information, we can define that the larger capacity utilization of every route in the obtained solution is a sign of better quality of the resulting solution.

2.4 Integration into the Metaheuristic

Guidotti et al., 2018 describes that the ranking of the relative importance of the problem attributes can be incorporated into rules. As shown in Figure 3, S19 and S20 have a higher magnitude in their corresponding SHAP value than other features. Thus, based on our explanation related to the features, particularly about features S19 and S20 in Section 2.3, we define a hypothesis about how to control the diversity of solution in the pool of elite solutions when solving the CVRP, based on the most important features.

Hypothesis: *features related to capacity utilization can be used to control the diversity of a pool of elite set of solutions*

To test our hypothesis, we first develop a metaheuristic algorithm to solve the CVRP, and then we will enhance the proposed metaheuristic algorithm using guidance from our previous explanation of the learning model. In Section 5.2, we will perform a computational experiment to investigate our hypothesis.

3 Development of the Metaheuristic Algorithm

This section proposes a new metaheuristic algorithm for solving the CVRP named the Multiple Search (MS) algorithm. The general overview of the MS algorithm is shown in Algorithm 1. The MS algorithm consists of a construction phase (line 2) and a further improvement phase that comprises neighborhood search improvement and path relinking. Moreover, to enhance diversity and intensification mechanism, we introduce a new hybrid split and path relinking mechanism into our proposed metaheuristic. Subsequent paragraphs provide a detailed description of the proposed algorithm, including the novel hybrid split and path relinking.

Algorithm 1 MS for solving CVRP

input: CVRP instance $\mathbf{I}$
1: **procedure** MS-CVRP($\mathbf{I}$)
2: $\mathbb{E} \leftarrow$ GeneratingInitialSolutions($\mathbf{I}$) ▷ pool of elite solutions
3: $S_{best} \leftarrow \arg\min_{s \in \mathbb{E}}$ Cost(s)
4: $\vartheta \leftarrow 0$ ▷ non-improving iteration counter
5: **repeat**
6: $(\mathbb{E}, S_{best}) \leftarrow$ NeighborhoodSearch($\mathbb{E}, S_{best}$)
7: $(\mathbb{E}, S_{best}) \leftarrow$ Split-PathRelinking($\mathbb{E}, S_{best}$)
8: $(\mathbb{E}, S_{best}) \leftarrow$ EliteSetManagement($\mathbb{E}, S_{best}, S_0, \vartheta, \mathbf{I}$) ▷ control member of the elite Set
9: **until** T_{max}
10: **return** S_{best}
11: **end procedure**

3.1 Generating Initial Solutions

The initial solution is constructed using a savings algorithm proposed by Clarke and Wright, 1964. As demonstrated by several authors (Arnold and Sörensen, 2019a; Accorsi and Vigo, 2021), the algorithm can be accelerated by only considering a pruned number n_{cw} of neighbors j in $\mathcal{N}_{cw}(C)$ for each customer i in C when computing the saving value. The number of neighborhoods for the Clarke and Wright algorithm, $\mathcal{N}_{cw}$, is set to 100, summarized in Table 3. As in the proposed algorithm, the minimum number of generated initial solutions is $\mathbb{E}_{min}$. Thus, after the algorithm generates an initial solution by using the Clarke and Wright algorithm, another solution is generated by performing tour perturbations. The main idea of tour perturbation is to destroy two random routes from the solution, resulting from the Clarke and Wright algorithm, and insert all customers into their best position in the remaining routes. If we do not find any sufficient position for insertion, we construct a new route for that customer node with 50% chance. The solution will be accepted whenever it is a feasible solution with a total cost smaller than the solution generated by the Clarke and Wright algorithm or has the same/smaller number of routes as the estimated number of routes $R_{estimated}$. The $R_{estimated}$ is a lower bound of the number of routes needed for the current instance and is calculated as follows.

$$R_{estimated} = \frac{\sum_{j \in N} q(c_j)}{Q} \tag{5}$$

3.2 Pool of the Elite Set Solutions

The pool of the elite set of solutions is a collection of a small number of solutions found during the search process. The size of the pool is described in Table 3, consisting of its minimum size $\mathbb{E}_{min}$, and its maximum size $\mathbb{E}_{max}$. The pool of elite set solutions starts empty. If the pool is not yet full, the candidate is simply added to the pool, provided it differs from all existing elite set members. Once the pool is full, if the new candidate is better than the current members, it replaces the worst member. The detailed process is shown in Algorithm 2.

Algorithm 2 Accepting a new solution for the elite set

input: new solution S, current elite set $\mathbb{E}$

1: **procedure** UpdateEliteSet(S, $\mathbb{E}$)
2: **if** $S \notin \mathbb{E}$ **then** ▷ should contain unique solutions
3: **if** Size($\mathbb{E}$) $< \mathbb{E}_{max}$ **then**
4: $\mathbb{E} \leftarrow \mathbb{E} \cup S$
5: **else**
6: $S_{worst} \leftarrow \arg\max_{s \in \mathbb{E}}$ Cost(s) ▷ worst solution in elite set
7: **if** Cost(S) $<$ Cost(S_{worst}) **then**
8: $\mathbb{E} \leftarrow \mathbb{E} \setminus S_{worst}$
9: $\mathbb{E} \leftarrow \mathbb{E} \cup S$
10: **else if** $(\text{Cost}(S) - \text{Cost}(S_{worst}))/\text{Cost}(S_{worst}) < 0.2$ **then** ▷ proximity allowance
11: $S_{best} \leftarrow \arg\min_{s \in \mathbb{E}}$ Cost(s) ▷ best solution in elite set
12: $\delta_{new} \leftarrow$ ProximityMeasurement(S_{best}, S)
13: $\delta_{old} \leftarrow$ ProximityMeasurement(S_{best}, S_{worst})
14: **if** $\delta_{new} > \delta_{old}$ **then**
15: $\mathbb{E} \leftarrow \mathbb{E} \setminus S_{worst}$
16: $\mathbb{E} \leftarrow \mathbb{E} \cup S$
17: **end if**
18: **end if**
19: **end if**
20: **end if**
21: **return** $\mathbb{E}$
22: **end procedure**

In Algorithm 2, we introduce the proximity allowance (line 10). This mechanism is used whenever the new solution is worse than the worst solution member of the pool, but the solution interval between them is less than 20%. Then, the proximity allowance, measured by using Algorithm 3, means that the pool will accept the solution to enhance the diversity by measuring its distance toward the best solution in the pool.

Algorithm 3 Proximity distance between two solutions

input: solution S_i, solution S_j

procedure ProximityMeasurement(S_i, S_j)
  $\delta \leftarrow 0$
  for $c \in C$ **do** ▷ iterate for all customer
    if $\text{Next}(S_i, c) \neq \text{Next}(S_j, c) \wedge \text{Next}(S_i, c) \neq \text{Prev}(S_j, c)$ **then**
      $\delta \leftarrow \delta + 1$
    end if
    if $\text{Prev}(S_i, c) = D \wedge \text{Prev}(S_j, c) \neq D \wedge \text{Next}(S_j, c) \neq D$ **then**
      $\delta \leftarrow \delta + 1$
    end if
  end for
  return δ
end procedure

3.2.1 Diversity Control Mechanism

The diversity control mechanism controls the level of diversity of the small number of solutions in the pool while maintaining the quality of solutions (Sörensen and Sevaux, 2006). Martí, Resende, and Ribeiro, 2013 proposed a multi-start mechanism where the algorithm will be restarted and construct a new initial solution whenever it finds a local optimum that is unlikely changed.

Build upon by these studies, we try to manage the solutions in the pool by measuring its non-improving iterations. The detailed mechanism used in the proposed algorithm is shown in Algorithm 4. The algorithm will re-generate solutions whenever ϑ exceeds the maximum limit of non-improving iterations, $\Theta_{\mathbb{E}}$.

Algorithm 4 Diversity control mechanism

input: elite set $\mathbb{E}$, current best solution S_{best}, best solution before S_0 improvement iteration counter ϑ, CVRP instance $\mathbf{I}$

procedure EliteSetManagement($\mathbb{E}$, S_{best}, S_0, ϑ, $\mathbf{I}$)
  $\Theta_{\mathbb{E}} \leftarrow 4000$ ▷ maximum non-improving iteration
  if $S_{best} < S_0$ **then**
    $\vartheta \leftarrow 0$
    return $\mathbb{E}$
  else
    $\vartheta \leftarrow \vartheta + 1$
    if $\vartheta > \Theta_{\mathbb{E}}$ **then**
      $\mathbb{E} \leftarrow \varnothing$ ▷ re-start elite set
      $\mathbb{E} \leftarrow$ GeneratingInitialSolutions($\mathbf{I}$) ▷ re-generate initial solutions
      $\vartheta \leftarrow 0$
      return $\mathbb{E}$
    end if
  end if
end procedure

3.3 Neighborhood Improvement

In the proposed algorithm, the neighborhood improvement processes are composed of two major steps: perturbation and local search improvement. The full mechanism of neighborhood search is shown in Algorithm 5. The perturbation mechanism is done by destroying and reconstructing a set of tours in the solution. The local search improvement is implemented through various local search operators, which are categorized into two groups. Within each group, the local search operators are randomly ordered.

3.3.1 Perturbation Mechanism

The perturbation mechanism prevents the algorithm from becoming trapped in local optima during optimization processes. The proposed perturbation mechanism used for this metaheuristic can be seen in Algorithm 6. In summary, the algorithm attempts to change the structure of solutions by randomly destroying a set of routes and inserting the customer nodes (from the destroyed route) around their neighbor nodes. If there is no suitable location for relocating the node, then a new route is created for that customer node.

Algorithm 5 Neighborhood Search mechanism

input: elite set $\mathbb{E}$, current best solution S_{best}
procedure NeighborhoodSearch($\mathbb{E}, S_{best}$)
  for $n \leftarrow 1$ **to** $\mathbb{E}_{min}$ **do**
    $S \leftarrow \arg\min_{s \in \mathbb{E}} \text{Cost}(s)$ ▷ best solution in current elite set
    $\mathbb{E} \leftarrow \mathbb{E} \setminus S$
    $S' \leftarrow$ DestroyRepairTour(S, S_{best})
    $S'' \leftarrow$ LocalSearchImprovement(S', S_{best})
    $\mathbb{E} \leftarrow$ UpdateEliteSet($S'', \mathbb{E}$)
  end for
  return $\mathbb{E}$, S_{best}
end procedure

Perturbation mechanism according to Estimated Route Sizes In the proposed metaheuristic, we apply a perturbation strategy based on the estimation of the number of customers in a route. This estimation is calculated using Equation (6). We categorize the perturbation strategy as destroy strategy and rebuilding strategy. For example, if an instance is categorized as a short route, we will destroy a random pair of routes. However, if an instance is categorized as a long route, we will only destroy one random route. This mechanism based on $k_{estimated}$ value is also applied whenever we perform parameter setting in the beginning of the algorithm, summarized in Table 3.

$$k_{estimated} = \frac{Q}{\left(\frac{\sum_{j \in N} q(c_j)}{N+1}\right)} \qquad \begin{cases} \text{long routes:} & \text{if } k_{estimated} > 20 \\ \text{short routes:} & \text{otherwise} \end{cases} \tag{6}$$

Apart from that, there are three strategies for rebuilding a set of routes in the solution that have been destroyed, *i.e.,* FullInsertion, FullGranularInsertion, and GranularInsertion. The FullInsertion means that the algorithm searches for the best insertion position in all the remaining routes of the solution. Meanwhile, the FullGranularInsertion means the algorithm searches the best insertion position around its maximum granular neighborhood Γ_{max}. Then, the GranularInsertion means that the algorithm searches the best insertion position around its currently granular neighborhood. We also referred to its value of $k_{estimated}$ for applying these strategies. For example, if the instance is categorized as a short route, we will apply the FullInsertion for the base perturbation mechanism. However, if there is no improvement of the best solution when the pool $\mathbb{E}$ is already generated three times, then the FullGranularInsertion is applied. If there is still no improvement, then the GranularInsertion is applied until the new best solution found / algorithm is terminated. The FullInsertion is applied again if a new best solution is found. Meanwhile, if instances are categorized as long routes, the FullInsertion is applied until no best solution is found even though the pool $\mathbb{E}$ is already generated twice. Then, if it happens, the GranularInsertion is applied until the new best solution found / algorithm is terminated. Similarly, if a new best solution is found, the FullInsertion is applied again.

3.3.2 Local Search Improvement

In metaheuristics, local search is known as a strong tool for improving a solution (Arnold and Sörensen, 2019a). The basic idea underlying local search is that high-quality solutions to an optimization problem can be found by iteratively improving a solution using small (local) modifications called moves.

Algorithm 6 Perturbation mechanism by destroying and rebuilding a set of routes

input: solution S, current best solution S_{best}

procedure DestroyRepairTour(S, S_{best})
  $s_{destroy} \leftarrow$ DefineStrategyForDestroyingRoutes(S)
  $s_{build} \leftarrow$ DefineStrategyForBuildingRoutes(S)
  $L_{free} \leftarrow$ EraseRoutes($s_{destroy}, S$) ▷ destroy customers according to $s_{destroy}$
  $L'_{free} \leftarrow$ ReOrderingCustomerList(L_{free})
  for $c \in L'_{free}$ **do**
    $n_c \leftarrow$ GetListNeighborsOf(c)
    $p_c \leftarrow$ GetBestPositionInsertion(s_{build}, c, n_c) ▷ insertion according to s_{build}
    if $\exists p_c \in S$ **then** ▷ has the insertion position been found?
      $S \leftarrow$ InsertCustomerAt(c, p_c, S)
    else
      $S \leftarrow$ BuildOneCustomerRoute(c, S)
    end if
  end for
  if $S < S_{best}$ **then** ▷ in case found an improved solution
    $S_{best} \leftarrow S$
  end if
  return S
end procedure

In this proposed algorithm, we utilized several intra-route local and inter-route local searches. Intra-route local search means that local search is applied to improve nodes within a route. Inter-route local search means that local search is applied between, at least, two different routes. These local searches are described below:

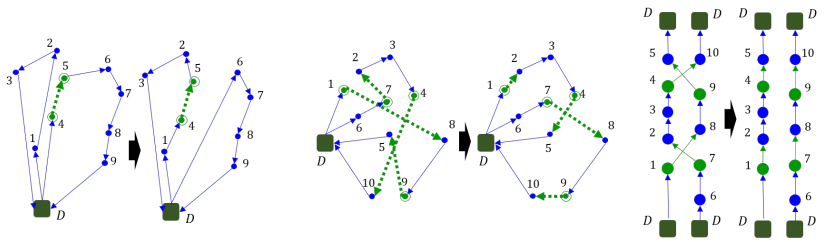

Figure 4: The mechanism of path-move (left) and CROSS-Exchange (right). On far right we detailed the mechanism of CROSS-Exchange

- **Relocate and Swap:** Relocate operation involves inserting an existing customer node into a new position, while the Swap operation involves exchanging two different customer nodes. Both operations are employed for improvement, within a route and between two different routes.
- **2-Opt and 2-Opt*:** The main concept behind the 2-Opt move is to eliminate two edges from the current route and substitute them with two new edges, thereby creating a new route. Originally designed for the Traveling Salesman Problem by Jünger, Reinelt, and Rinaldi, 1995. A variant of this move involves by exchanging sub-sequences at the terminations of two tours, which we denoted as 2-Opt*.
- **CROSS-Exchange:** CROSS-Exchange, introduced by Taillard et al., 1997, is a local search operator designed to remove four edges from two different routes and replace them with four new edges. The mechanism of CROSS-Exchange is shown in Figure 4 (right), where on the far right is shown its detailed mechanism.
- **Path-Move:** Path-move is a mechanism involving the relocation of a path comprising two consecutive customers. This movement happens either within a route or between two routes, as described by Soto et al., 2017. Additionally, we introduce another variant, named Double Path-move. Here, instead of only moving a path, we also tried to move 2 consecutive paths, consisting 3 consecutive customers. The basic mechanism of the single-path move is shown in Figure 4 (left).

- **Chain-Move:** During a local search, we sometimes encounter movements that, although interesting, violate certain constraints. Because of the potential solution improvement, there may be a temptation to execute these moves. One approach to deal with this case is to employ ejection chains, introduced by Glover, 1996 and first formalized for solving VRP by Rego, 2001. These chains represent a sequence of movements, where the initial one is not feasible. Subsequent movements are intended to repair the violated constraint. The detailed mechanism of chain-move, performed by relocate chain move, is shown in Figure 5.

Algorithm 7 Local search improvement phase

input: solution S, current best solution S_{best}

procedure LocalSearchImprovement(S, S_{best})
  $p \leftarrow 0$
  repeat
    $S_{eval} \leftarrow$ RandomNeighborhoodSearch(p, S, S_{best})
    if Cost(S_{eval}) < Cost(S) **then**
      $S \leftarrow S_{eval}$
      $p \leftarrow 0$ ▷ repeat the improvement
      if Cost(S_{eval}) < Cost(S_{best}) **then**
        $S_{best} \leftarrow S_{eval}$
      end if
    else
      $p \leftarrow p + 1$ ▷ move to next level of local search group
    end if
  until $p < 2$
  return S
end procedure

The list of local searches used in the proposed algorithm is summarized in Table 2. Algorithm 7 and Algorithm 8 show how we applied these local search operators to improve the solution. As shown in Algorithm 8, we categorized the local search operators into two levels: the first level contains intra-route and inter-route local searches. The second level contains inter-route chain-move operators. For executing the local search operator, as shown Algorithm 8, we define **R** as the currently used pointer and **L** as the last used pointer, and the search will stop whenever a better solution S_{eval} is found or there is no improvement S_{eval} after all operators are evaluated.

Table 2: Various local search operators

Intra-Route	Inter-Route	Inter-Route Chain-Move
Relocate	Relocate	Ejection-Chain: Relocate
Swap	Swap	Ejection Chain: Single-Path move
2-Opt move	2-Opt* move	
	Single-Path move	
	Double-Path move	
	Cross-Exchange	

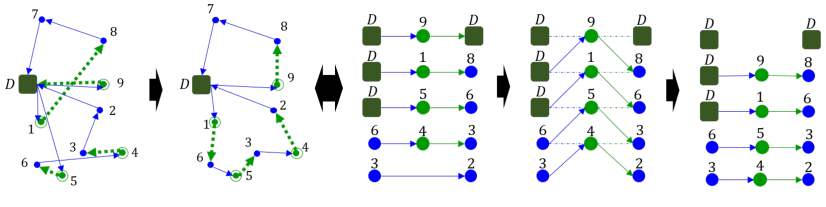

Figure 5: The mechanism of inter-route ejection-chain move. As shown in the right, this mechanism is composed of several relocate moves. The chain moves are stimulated when encountering a move that, although interesting, violates certain constraints.

Algorithm 8 Randomized-order local search operators mechanism in every level

input: pointer level p, solution S, current best solution S_{best}

procedure RandomNeighborhoodSearch(p, S, S_{best})
  $\mathbb{L}_{LS} \leftarrow$ GetListLocalSearchOperatorForLevel(p)
  $\mathbb{L}'_{LS} \leftarrow$ RandomReOrdering($\mathbb{L}_{LS}$)
  $\mathbf{R} \leftarrow 0$ ▷ define the current pointer
  $\mathbf{L} \leftarrow 0$ ▷ define the last pointer
  repeat
    $\mathbb{LS} \leftarrow$ GetOperator($\mathbb{L}'_{LS}, \mathbf{L}$) ▷ calling the local search operator
    $S_{eval} \leftarrow$ LocalSearchExecution($\mathbb{LS}, S$)
    if $S_{eval} < S$ **then**
      $\mathbf{R} \leftarrow \mathbf{L}$ ▷ update the current pointer
      $S \leftarrow S_{eval}$
      if $S < S_{best}$ **then**
        $S_{best} \leftarrow S$
      end if
    end if
    $\mathbf{L} \leftarrow (\mathbf{L} + 1) \cdot \mod (\text{Size}(\mathbb{L}_{LS}))$ ▷ update the last pointer
  until $\mathbf{R} = \mathbf{L}$
  return S
end procedure

Tabu Search Heuristic and Growing Neighborhood Size Tabu search is utilized by tracking recent moves in a tabu list L_{tabu} Soto et al., 2017. The move is banned whenever the algorithm attempts to perform a move recorded in the tabu list. This rule prevents cycling and forces other solutions to be explored (shown in Algorithm 9). In the proposed algorithm, for compressing computational resources while maintaining the performance of the algorithm, we utilized a growing granular concept. Hence, in this research, we initialize granular size at the beginning of iterations as Γ_0. If any better solution is not found during the search process, the Γ_0 will increase by γ, until Γ_{max} is reached. If the algorithm finds a new best solution, the granular size is reset again as Γ_0. The description of all granular parameters used in the proposed algorithm is shown in Table 3.

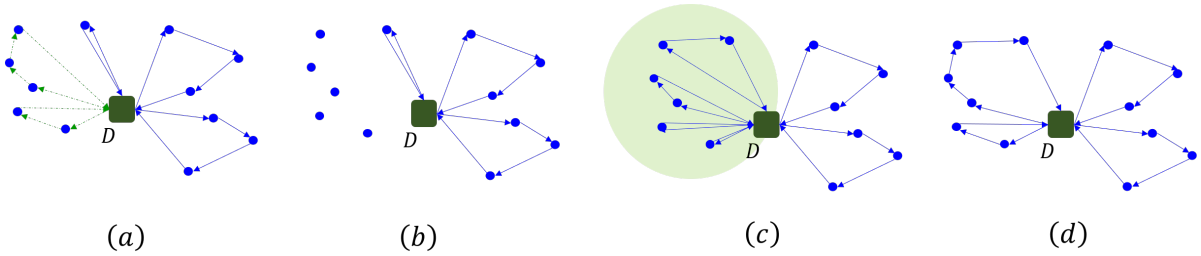

Figure 6: Mechanism of pruned neighborhood improvement, compromise of (a) perturbation by destroying a set of routes in the solution, (b) the destroyed customer nodes are inserted to the remaining routes or build new routes, (c) local search improvement that only focused to the last moved customer nodes, and (d) resulting new solution

Mechanism of Pruned Neighborhood Improvement To efficiently perform the search process, the local search improvement mechanism only focused in the area already perturbed (*i.e.*, around customer nodes that were just inserted into routes). As shown in Algorithm 9, we utilized the concept of selective vertex caching and vertex-wise granular management (Accorsi and Vigo, 2021) to limit the search area. We denoted the coverage area (the green area in Figure 6 (c)) as N_c, and the maximum coverage area is summarized in Table 3.

Algorithm 9 Move evaluation processes

input: local search operator $\mathbb{LS}$, solution S

procedure LocalSearchExecution($\mathbb{LS}$, S)
  $L_{tabu} \leftarrow \varnothing$ ▷ initialize tabu list
  for $c_i \in N_c$ **do**
    $L_n \leftarrow$ GetListNeighborhoodOf(c_i)
    $c_j \leftarrow$ GetBestPossiblePositionMovingFrom(L_n)
    if $\exists c_j \in S \wedge (c_i, c_j) \notin L_{tabu} \wedge$ CostMoveOf($\mathbb{LS}, c_i, c_j$) $<$ Cost(S) **then**
      $S \leftarrow$ PerformMove($\mathbb{LS}, c_i, c_j$)
      $L_{tabu} \leftarrow L_{tabu} \cup (c_i, c_j)$ ▷ update tabu list
    end if
  end for
  return S
end procedure

3.4 Path Relinking

In path relinking, the initial solution should be improved toward guiding solution (Ho and Gendreau, 2006). However, in the VRP solution, the number of routes (strings of solution) is mostly more than one, so it is a challenge to ensure that the initial solution perfectly transforms toward the guiding solution. Apart from that, Prins, 2004 introduces an approach in VRP that represents the solution as a permutation of visits, a string, called as *Giant Tour* (GT), and transforming the GT into VRP solution using the split algorithm (described in Algorithm 14).

Algorithm 10 A Novel Hybrid Split and Path Relinking mechanism

input: elite set $\mathbb{E}$, current best solution S_{best}

1: **procedure** Split-PathRelinking($\mathbb{E}, S_{best}$)
2: $(S_i, S_g) \leftarrow$ GetRandomParents($\mathbb{E}, S_{best}$)
3: $T_i \leftarrow$ RandomConcatenation(S_i) ▷ transform into giant tour
4: $T_g \leftarrow$ RandomConcatenation(S_g) ▷ transform into giant tour
5: $(\Delta_{pr}, L_{pr}) \leftarrow$ GetRestrictedNeighborhood(T_i, T_g)
6: $L'_{pr} \leftarrow$ ReOrderingCustomerList(L_{pr})
7: $N_{pr} \leftarrow {}^{\Delta_{pr}}/_{2} \cdot \eta_{pr}$
8: $(\mathbb{E}, S_{best}) \leftarrow$ EvaluateNeighborhood($T_i, T_g, N_{pr}, L'_{pr}, \mathbb{E}, S_{best}$)
9: **return** $\mathbb{E}$, S_{best}
10: **end procedure**

In this research, we proposed a hybrid split and path relinking. This mechanism aims to transform initial and guiding solutions into GT solutions. Then, we perform path relinking between them and transform the intermediate solution, resulting during path relinking processes, with the split algorithm. After the intermediate solution is transformed with the split algorithm, it is evaluated to become or not a new member of the pool $\mathbb{E}$. Using this approach, we can transform the initial solution into guiding solutions and generate several intermediate (new) solutions during the process. By using these intermediate solutions, we can improve the quality of the solution and increase the diversity of the pool $\mathbb{E}$. The proposed hybrid split and path relinking steps are shown in Algorithm 10. The first step of the algorithm for performing path relinking is to identify the restricted neighborhood. The L_{pr} represents the list of restricted neighborhoods, and Δ_{pr} represents the proximity distance between initial solution T_i and guiding solution T_g. Then, the process is continued with neighborhood search processes among the restricted neighborhood L_{pr}.

For more details about neighborhood search processes during path relinking, in Algorithm 10, line 7 is described in Appendix A.4. Furthermore, the truncated mechanism for path relinking (Glover, Laguna, and Marti, 2000) has been combined in the proposed mechanism to accelerate the search processes while maintaining the diversity of solutions explored.

Truncated Path Relinking Resende and Ribeiro, 2005 demonstrated that there tends to be a higher concentration of better solutions close to the initial solutions explored by path relinking. Additionally, we may reduce the computational time while still possible to obtain good solutions by adapting this mechanism. As described by Glover, Laguna, and Marti, 2000, adapting the truncated path relinking mechanism when exploring the restricted neighborhood can be done by introducing η_{pr} as the index defining the portion of the path to be explored, where $0 < \eta_{pr} \leq 1$, which the best value is shown in Table 3. As we utilized η_{pr}, instead of evaluating all Δ_{pr} restricted neighborhoods, we will use N_{pr} as the main loop for evaluating the restricted neighborhood.

4 Hybridizing Metaheuristics with Feature-Based Guidance

As outlined in our hypothesis in Section 2.4, we aim to use the most important feature to formulate rules for enhancing the performance of our proposed metaheuristic algorithm. In this section, we detail how we construct a rule based on our explainability learning model to boost the performance of our proposed metaheuristic algorithm, which has been described before in Section 3.

4.1 Guidance for Diversity Control

In Section 3.2.1, we simply rely on the number of non-improving iterations ϑ to determine whether the pool $\mathbb{E}$ is still worth to improve or should the algorithm re-generate the pool $\mathbb{E}$. Hence, through Section 2.3, we understand that the quality of the solution can be measured by its capacity utilization.

Algorithm 11 Guided diversity control mechanism

input: elite set $\mathbb{E}$, current best solution S_{best}, best solution before S_0
feature threshold $\mathbf{W}$, improvement iteration counter ϑ, CVRP instance $\mathbf{I}$

procedure GuidedEliteSetManagement($\mathbb{E}$, S_{best}, S_0, $\mathbf{W}$, ϑ, $\mathbf{I}$)
  $\mathcal{C} \leftarrow \lceil \mathbf{W} \times \mathbf{M} \rceil$ ▷ threshold non-improving iteration
  if $S_{best} < S_0$ **then**
    $\vartheta \leftarrow 0$
    return $\mathbb{E}$
  else
    $\vartheta \leftarrow \vartheta + 1$
    if $\vartheta > \mathcal{C}$ **then**
      $\mathbb{E} \leftarrow \varnothing$ ▷ re-start elite set
      $(\mathbb{E}, \alpha, \beta) \leftarrow$ GeneratingInitialSolutions($\mathbf{I}$) ▷ re-generate initial solutions
      $\mathbf{W} \leftarrow \dfrac{(\mathbf{W} + \alpha + \beta)}{2}$ ▷ updating threshold
      $\vartheta \leftarrow 0$
      return $\mathbb{E}$
    end if
  end if
end procedure

Then, besides only using non-improving solution φ, it attempts to incorporate the value of its capacity utilization to decide whether the pool $\mathbb{E}$ is still worth improving or not. The detailed calculation for $S19(s)$ and $S20(s)$ for every solution S is described in Equation (38) and Equation (39), as:

$$\hat{c_1} = \sum_{s=1}^{\mathbb{E}} S19(s) \qquad \hat{c_2} = \sum_{s=1}^{\mathbb{E}} S20(s) \tag{7}$$

Then, assume in pool $\mathbb{E}$ is contains ε member of solutions, where $\mathbb{E}_{min} \leq \varepsilon \leq \mathbb{E}_{max}$. Then, for measuring the average value of $S19(s)$ and $S20(s)$ in pool $\mathbb{E}$, denoted as α and β respectively, we calculated as:

$$\alpha = \sum_{\kappa=1}^{\varepsilon} \hat{c_1} \qquad \beta = \sum_{\kappa=1}^{\varepsilon} \hat{c_2} \tag{8}$$

As shown in Figure 3, we conclude that higher average capacity utilization and lower standard deviation of capacity utilization mean a sign that better the quality of the obtained solution.

Thus, by assuming that α is a representation of the average capacity utilization of obtained solutions in the pool $\mathbb{E}$ (S19), and β is a representation of the standard deviation of capacity utilization of obtained solutions in the pool $\mathbb{E}$ (S20), we formulated a threshold for determining whether the pool $\mathbb{E}$ still worth to improve as:

$$\mathcal{C} := \lceil \mathbf{W} \cdot \mathbf{M} \rceil = \lceil (\alpha - \beta) \cdot \mathbf{M} \rceil := \left\lceil \left(\sum_{\kappa=1}^{K} \hat{c_1} - \sum_{\kappa=1}^{K} \hat{c_2} \right) \cdot \mathbf{M} \right\rceil \tag{9}$$

In Equation (9), the $\mathbf{M}$ is dented as a big constant value, where $\mathbf{M} = 4000$. Hence, the guidance version of how the algorithm controls the diversity of the pool $\mathbb{E}$ is shown in Algorithm 11.

4.2 Metaheuristic with Guidance for Diversity Control

As shown in Equation (9), we denoted the guidance as $\mathbf{W}$, where it can be applied as shown in Algorithm 11. Then, the full mechanism of the guided version of Algorithm 1 can be written as follows: In the following section,

Algorithm 12 Guided MS for solving CVRP

input: CVRP instance $\mathbf{I}$
procedure GUIDED-MS-CVRP($\mathbf{I}$)
  $(\mathbf{W}, \mathbb{E}) \leftarrow$ GENERATINGINITIALSOLUTIONS($\mathbf{I}$)
  $S_{best} \leftarrow \arg\min_{s \in \mathbb{E}} \text{COST}(s)$
  $\vartheta \leftarrow 0$ ▷ non-improving iteration counter
  repeat
    $(\mathbb{E}, S_{best}) \leftarrow$ NEIGHBORHOODSEARCH($\mathbb{E}, S_{best}$)
    $(\mathbb{E}, S_{best}) \leftarrow$ SPLIT-PATHRELINKING($\mathbb{E}, S_{best}$)
    $(\mathbf{W}, \mathbb{E}, S_{best}) \leftarrow$ GUIDEDELITESETMANAGEMENT($\mathbb{E}, S_{best}, S_0, \mathbf{W}, \vartheta, \mathbf{I}$) ▷ control member of the elite Set
  until T_{max}
  return S_{best}
end procedure

we will perform a computational experiment to measure the performance of the guided version of the proposed algorithm, shown in Algorithm 12, compared with Algorithm 1. This experiment investigates whether we can improve the performance of our proposed algorithm by utilizing Equation (9) as well as to test our hypothesis in Section 2.4.

5 Experiment and Analysis

The algorithm was implemented in C++ and compiled using g++ 8.3.0. For all algorithms tested in this paper, the experiment was performed on a 64-bit mini-computer with an AMD Ryzen 7 PRO 5850U processor and 16 GB RAM, running on Ubuntu 22.04.1 operating system. As the randomized processes of the algorithm, we follow an experiment procedure from Accorsi and Vigo, 2021 where each experiment involved a set number of five runs for every instance, with the seed of the pseudo-random engine defined as the run counter minus one. Throughout the experimentation, we refer to the following:

- BKS: the total cost value of the best-known solutions. All the information related to the instances and best-known solutions are available at http://vrp.galgos.inf.puc-rio.br/index.php/en/.
- Gap: the difference between the obtained solution and the best-known solution of the problem. The gap value is calculated as follows:

$$\text{Gap} = \frac{\text{Obtained Solution} - \text{BKS}}{\text{BKS}} \times 100\% \tag{10}$$

The source code of the proposed algorithm and all the detailed computational results both for the proposed algorithm and the baseline algorithm for every run can be downloaded from https://github.com/bachtiarherdianto/MS-CVRP alongside the instruction for replicating the experiment of the proposed algorithm.

Table 3: Parameters of the proposed algorithm

Parameter		Value		Described in
$\mathcal{N}_{cw}$	Size of saving table	100		Section 3.1
$\mathbb{E}_{max}$	Maximum size of pool elite solution	3 2	(long routes) (short routes)	Section 3.2
$\mathbb{E}_{min}$	Minimum size of pool elite solution	2		Section 3.2
$\textsc{Size}(N_c)$	Maximum coverage area	50		Section 3.3.2
Γ_0	Initial granular size	10 5	(long routes) (short routes)	Section 3.3.2
γ	Granular's growth	5		Section 3.3.2
Γ_{max}	Maximum granular size	25		Section 3.3.2
$\mathbb{T}$	Size of the tabu list	50		Section 3.3.2
η_{pr}	Truncated index	0.4		Section 3.4

5.1 Parameter Tuning

The parameters used consist of parameters for pruning the Clarke and Wright when constructing a solution, parameters to control the size of the pool of elite set solution, and the search intensification. Furthermore, we also vary several parameters based on the estimated number of customers for every route. The variation mechanism is similar to that described in Section 3.3.1 and defined using Equation (6). The details for these values are summarized in Table 3.

5.2 Computational Experiment with $\mathbb{XML}$100 Instances

To investigate whether our proposed guided metaheuristic in Algorithm 12 can outperform our proposed metaheuristic in Algorithm 1, we will perform a computation experiment using $\mathbb{XML}$100 instances, from Queiroga et al., 2021. The computational experiment was performed by using 100 randomly sampled $\mathbb{XML}$100 instances. In this experiment, we use the MNS-TS algorithm as a baseline for comparing our proposed algorithms, as it was also already used for generating near-optimal data, described in Section 2.1. In this experiment, the baseline algorithms and all proposed algorithms are given a 60-second time budget to solve each sampled $\mathbb{XML}$100 instances. Computational results of the baseline algorithm, all proposed proposed algorithms on sampled $\mathbb{XML}$100 instances, are summarized in Table 4.

Table 4: Summary of the experiment to investigate the contribution of the proposed guidance to the proposed algorithm on $\mathbb{XML}$100 instances (Queiroga et al., 2021) with $T_{max} = 60$ seconds in 5 runs. The experiment aimed to assess the performance of all our proposed algorithms compared to the baseline algorithm. The MNS-TS algorithm (Soto et al., 2017; Lucas et al., 2020) is used as the baseline algorithm.

Measurement	**MNS-TS**		**MS**		**Guided-MS**	
	Avg	**Best**	**Avg**	**Best**	**Avg**	**Best**
Minimum Gap	0.08	0.02	0.00	0.00	0.00	0.00
Average Gap	2.06	1.87	0.83	0.59	0.78	0.53
Median Gap	2.05	1.84	0.74	0.35	0.66	0.27
Maximum Gap	5.89	5.80	3.64	3.61	3.28	2.68

The comparison between the baseline algorithm and all proposed algorithms is shown Figure 7 (left). We can see that all proposed algorithms can perform better than the baseline algorithm. Then, the comparison of proposed algorithms, between without guidance and with guidance, is shown on the left side of Figure 7. Further statistical analysis is performed to measure the effectiveness of the proposed guidance.

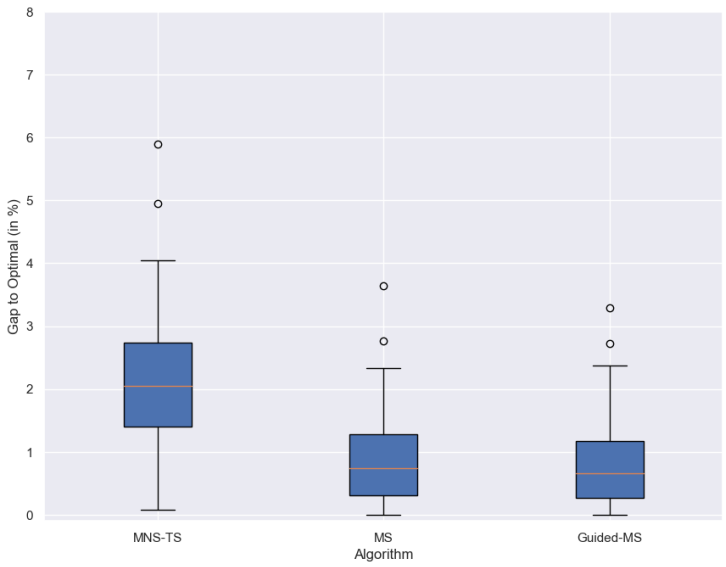

Figure 7: Summary of the computational results between baseline algorithm and all the proposed algorithms from Table 4.

Statistical Analysis of Feature-Based Guidance Further statistical analysis is performed to assess whether a significant difference exists between the MS algorithm and the Guided-MS algorithm, summarized in Table 4. This analysis measures the effect of the proposed feature-based guidance. Following the recommendation of Demšar, 2006; Accorsi, Lodi, and Vigo, 2022, non-parametric tests are preferred over parametric tests. Therefore, the one-tailed Wilcoxon signed-rank test is performed. In this test, we formulate Hypothesis H_0 and Hypothesis H_1, as follows:

Hypothesis H_0 *AvgCost(Guided-MS)* $\equiv$ *AvgCost(MS)*

Hypothesis H_1 *AvgCost(Guided-MS)* $<$ *AvgCost(MS)*

As recommended by Accorsi, Lodi, and Vigo, 2022, for comparing two algorithms through the one-tailed Wilcoxon signed-rank test, we set $\alpha = {}^{0.025}/_{2} = 0.0125$, meaning that if p-value $> \alpha$, then we failed to reject Hypothesis H_0 meaning that the average results of the two methods are not statistically different. However, if p-value $\leq \alpha$, then Hypothesis H_0 is rejected. Here, we can conclude that the average results are statistically different, and the alternative Hypothesis H_1 can be accepted. It means that the Guided-MS performs better than the MS. Here, the one-tailed Wilcoxon signed-rank test results in p-value $= 2.26 \cdot 10^{-10}$ (< 0.0125), meaning that the guidance gives a significant improvement of the proposed metaheuristic algorithm.

Path Relinking Contributions An additional statistical analysis is performed to measure the contribution of the proposed path relinking to all the proposed algorithms. Following the previous analysis through the one-tailed Wilcoxon signed-rank test, we set $\alpha = {}^{0.025}/_{2} = 0.0125$, so that if p-value $\leq \alpha$, then Hypothesis H_0 is rejected, which refers to a statistically significant contribution of the proposed mechanism of path relinking to the guided metaheuristics. The summary of the computational results on 50 sampled $\mathbb{XML}100$ instances is described in Table 5, while the detailed results are described in Appendix A.6. Here, for MS algorithm, the one-tailed Wilcoxon signed-rank test results into p-value $= 1.64 \cdot 10^{-8}$ (< 0.0125), while for Guided-MS algorithm, the one-tailed Wilcoxon signed-rank test results into p-value $= 4.98 \cdot 10^{-8}$ (< 0.0125). Then, we can conclude that our proposed path relinking mechanism can give a statistically significant contribution to the overall mechanism of the proposed algorithm

Table 5: Summary of the experiment to measure the contribution of the proposed mechanism of the path relinking with $T_{max} = 60$ seconds in 5 runs. The experiment was performed for both proposed metaheuristic algorithms, MS and Guided-MS.

Instance	MS				Guided-MS			
	with PR		without PR		with PR		without PR	
	Avg (Gap)	Best (Gap)	Avg (Gap)	Best (Gap)	Avg (Gap)	Best (Gap)	Avg (Gap)	Best (Gap)
Minimum Gap	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Average Gap	0.74	0.49	0.85	0.58	0.70	0.44	0.82	0.56
Median Gap	0.75	0.25	0.86	0.49	0.66	0.23	0.82	0.46
Maximum Gap	2.33	2.29	2.45	2.36	2.37	2.33	2.43	2.29

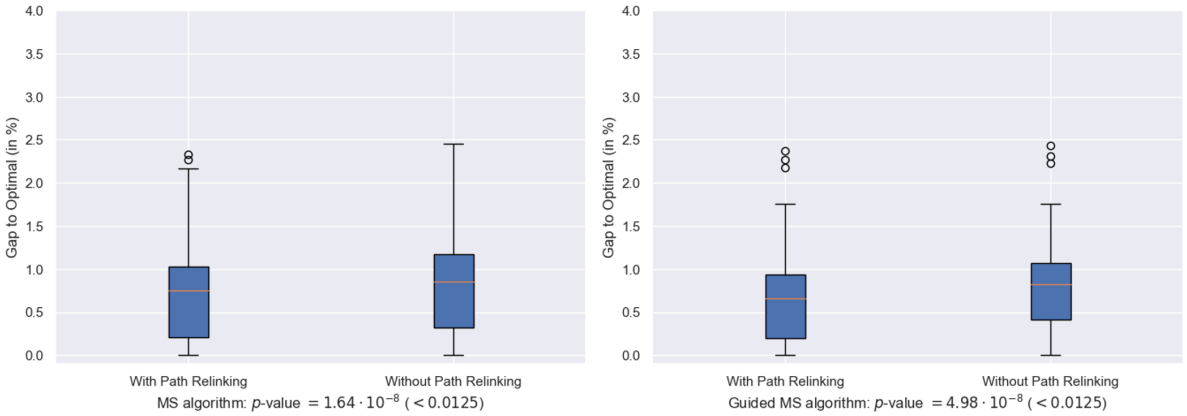

Figure 8: Summary the computational results 50 sampled $\mathbb{XML}100$ instances to show the contribution of the proposed mechanism path relinking

5.3 Testing the Guided Metaheuristic on $\mathbb{X}$ Instances

The detailed result on the $\mathbb{X}$ instances (Uchoa et al., 2017) for the Guided-MS algorithm is presented in Table 6. We compared the performance of the Guided-MS with several state-of-the-art metaheuristic algorithms for solving the CVRP, such as: LKH-3[2] (Helsgaun, 2017), HGS[3] (Vidal, 2022), and FILO[4] Accorsi and Vigo, 2021. As all the source codes are available, we execute our proposed algorithm and all the state-of-the-art algorithms under similar conditions using a similar mini-computer.

We follow experiments from Vidal, 2022, where we set the termination criteria for all algorithms is the maximum computation time, $T_{max} = N \times {}^{240}/_{100}$ seconds, in which N represents the number of customer nodes. As also described in Section 5, here we execute all the baseline algorithms five times with five different random seeds, where each seed was executing one algorithm run. From table Table 6, although the proposed guided metaheuristic cannot outperform all the state-of-the-art metaheuristic algorithms, it can still outperform the LKH-3. Also, for some small instances, the proposed guided metaheuristic can perform similarly with the HGS and FILO. Hence, we can conclude that the proposed guided metaheuristic is competitive with the state-of-the-art metaheuristics tested in the Table 6.

[2]We refer to the LKH-3.0.9 solver at: http://webhotel4.ruc.dk/ keld/research/LKH-3/
[3]We refer to the HGS solver at: https://github.com/vidalt/HGS-CVRP
[4]We refer to the FILO solver at: https://github.com/acco93/filo

Table 6: Comparison of solution quality with $T_{max} = N \times {}^{240}/_{100}$ seconds in 5 runs. The table shows the computational results on the 𝕏 instances Uchoa et al., 2017 of the Guided-MS with state-of-the-art methods, such as LKH-3 (Helsgaun, 2017), HGS (Vidal, 2022), and FILO (Accorsi and Vigo, 2021)

Instance	HGS		FILO		LKH-3		Guided-MS		BKS
	Avg (Gap)	Best (Gap)	Avg (Gap)	Best (Gap)	Avg (Gap)	Best (Gap)	Avg (Gap)	Best (Gap)	
X-n101-k25	27591 (0)	27591 (0)	27592.2 (0)	27591 (0)	27618.2 (0.1)	27591 (0)	27630.4 (0.14)	27591 (0)	27591
X-n106-k14	26412.2 (0.19)	26399 (0.14)	26379.2 (0.07)	26362 (0)	26384 (0.08)	26381 (0.07)	26477.8 (0.44)	26389 (0.1)	26362
X-n110-k13	14971 (0)	14971 (0)	14971 (0)	14971 (0)	14986.8 (0.11)	14971 (0)	15053 (0.55)	14971 (0)	14971
X-n115-k10	12747 (0)	12747 (0)	12747 (0)	12747 (0)	12754.6 (0.06)	12747 (0)	12747 (0)	12747 (0)	12747
X-n120-k6	13332 (0)	13332 (0)	13332 (0)	13332 (0)	13339.8 (0.06)	13332 (0)	13364.2 (0.24)	13332 (0)	13332
X-n125-k30	55539 (0)	55539 (0)	55899.2 (0.65)	55692 (0.28)	55965.6 (0.77)	55679 (0.25)	56044.8 (0.91)	55716 (0.32)	55539
X-n129-k18	28940 (0)	28940 (0)	28962 (0.08)	28948 (0.03)	28996.2 (0.19)	28948 (0.03)	29065.8 (0.43)	28972 (0.11)	28940
X-n134-k13	10916 (0)	10916 (0)	10935.4 (0.18)	10933 (0.16)	10973.8 (0.53)	10937 (0.19)	10954 (0.35)	10916 (0)	10916
X-n139-k10	13590 (0)	13590 (0)	13590 (0)	13590 (0)	13639.4 (0.36)	13602 (0.09)	13628.8 (0.29)	13590 (0)	13590
X-n143-k7	15700 (0)	15700 (0)	15726.6 (0.17)	15726 (0.17)	15789.8 (0.57)	15726 (0.17)	15829 (0.82)	15726 (0.17)	15700
X-n148-k46	43448 (0)	43448 (0)	43580.4 (0.3)	43474 (0.06)	43562.2 (0.26)	43448 (0)	43531 (0.19)	43448 (0)	43448
X-n153-k22	21225 (0.02)	21225 (0.02)	21273.2 (0.25)	21225 (0.02)	21247.8 (0.13)	21228 (0.04)	21276.4 (0.27)	21231 (0.05)	21220
X-n157-k13	16876 (0)	16876 (0)	16876 (0)	16876 (0)	16881 (0.03)	16876 (0)	16884.6 (0.05)	16876 (0)	16876
X-n162-k11	14138 (0)	14138 (0)	14164.4 (0.19)	14153 (0.11)	14194.2 (0.4)	14171 (0.23)	14185.6 (0.34)	14171 (0.23)	14138
X-n167-k10	20557 (0)	20557 (0)	20567.4 (0.05)	20557 (0)	20713.2 (0.76)	20601 (0.21)	20743 (0.9)	20617 (0.29)	20557
X-n172-k51	45607 (0)	45607 (0)	45624 (0.04)	45607 (0)	45941.4 (0.73)	45715 (0.24)	45737.6 (0.29)	45730 (0.27)	45607
X-n176-k26	47812 (0)	47812 (0)	47990.4 (0.37)	47825 (0.03)	48113.4 (0.63)	47951 (0.29)	48545.8 (1.53)	48105 (0.61)	47812
X-n181-k23	25569 (0)	25569 (0)	25574.4 (0.02)	25569 (0)	25635.8 (0.26)	25601 (0.13)	25602.6 (0.13)	25577 (0.03)	25569
X-n186-k15	24145 (0)	24145 (0)	24161.2 (0.07)	24149 (0.02)	24256.6 (0.46)	24189 (0.18)	24310.4 (0.69)	24182 (0.15)	24145
X-n190-k8	16999 (0.11)	16987 (0.04)	16993.6 (0.08)	16987 (0.04)	17097.4 (0.69)	17008 (0.16)	17178.6 (1.17)	17055 (0.44)	16980
X-n195-k51	44225 (0)	44225 (0)	44361 (0.31)	44270 (0.1)	44518.6 (0.66)	44295 (0.16)	44458.4 (0.53)	44386 (0.36)	44225
X-n200-k36	58605.6 (0.05)	58578 (0)	59072.4 (0.84)	58659 (0.14)	58864.8 (0.49)	58754 (0.3)	59350 (1.32)	59166 (1)	58578
X-n204-k19	19565 (0)	19565 (0)	19584.6 (0.1)	19570 (0.03)	19759 (0.99)	19700 (0.69)	19759 (0.99)	19682 (0.6)	19565
X-n209-k16	30662.2 (0.02)	30656 (0)	30699.4 (0.14)	30685 (0.09)	31023.8 (1.2)	30993 (1.1)	30976.4 (1.05)	30809 (0.5)	30656
X-n214-k11	10870.4 (0.13)	10865 (0.08)	10908.4 (0.48)	10870 (0.13)	11131.2 (2.54)	11036 (1.66)	11075.6 (2.02)	10968 (1.03)	10856
X-n219-k73	117597.2 (0)	117595 (0)	117604 (0.01)	117595 (0)	117688 (0.08)	117622 (0.02)	117612.2 (0.01)	117606 (0.01)	117595
X-n223-k34	40465 (0.07)	40437 (0)	40530.2 (0.23)	40505 (0.17)	40757.6 (0.79)	40576 (0.34)	40710.8 (0.68)	40645 (0.51)	40437
X-n228-k23	25743 (0)	25743 (0)	25786.8 (0.17)	25743 (0)	25936.8 (0.76)	25816 (0.29)	25831 (0.35)	25810 (0.26)	25742
X-n233-k16	19247 (0.09)	19230 (0)	19334 (0.54)	19324 (0.49)	19405.4 (0.91)	19321 (0.47)	19393 (0.85)	19337 (0.56)	19230
X-n237-k14	27042 (0)	27042 (0)	27043.8 (0.01)	27042 (0)	27164.2 (0.45)	27056 (0.05)	27294.4 (0.93)	27187 (0.54)	27042
X-n242-k48	82893.4 (0.17)	82780 (0.04)	82956.8 (0.25)	82825 (0.09)	83318.4 (0.69)	83217 (0.56)	83247.4 (0.6)	83078 (0.4)	82751
X-n247-k50	37374.4 (0.27)	37286 (0.03)	37552.2 (0.75)	37481 (0.56)	37386.8 (0.3)	37320 (0.12)	37632.4 (0.96)	37568 (0.79)	37274
X-n251-k28	38767 (0.21)	38684 (0)	38811.2 (0.33)	38699 (0.04)	39033.6 (0.9)	38909 (0.58)	38994.8 (0.8)	38953 (0.7)	38684
X-n256-k16	18874.2 (0.19)	18851 (0.06)	18880 (0.22)	18880 (0.22)	19098.2 (1.38)	18939 (0.53)	18927.4 (0.47)	18911 (0.38)	18839
X-n261-k13	26586.4 (0.11)	26577 (0.07)	26685.4 (0.48)	26633 (0.28)	26983 (1.6)	26826 (1.01)	27241 (2.57)	27011 (1.71)	26558
X-n266-k58	75734.8 (0.34)	75692 (0.28)	75856 (0.5)	75642 (0.22)	76081.6 (0.8)	75916 (0.58)	76467.2 (1.31)	76231 (1)	75478
X-n270-k35	35304.4 (0.04)	35303 (0.03)	35372.4 (0.23)	35324 (0.09)	35486.6 (0.55)	35436 (0.41)	35522.2 (0.66)	35445 (0.44)	35291
X-n275-k28	21245 (0)	21245 (0)	21267.8 (0.11)	21245 (0)	21375.4 (0.61)	21273 (0.13)	21378.6 (0.63)	21245 (0)	21245
X-n280-k17	33592.4 (0.27)	33584 (0.24)	33642 (0.41)	33601 (0.29)	33849.6 (1.03)	33717 (0.64)	34044.4 (1.62)	33913 (1.22)	33503
X-n284-k15	20267 (0.26)	20244 (0.14)	20323.6 (0.54)	20290 (0.37)	20492.6 (1.37)	20291 (0.38)	20514.6 (1.48)	20412 (0.97)	20215
X-n289-k60	95394.2 (0.26)	95332 (0.19)	95718 (0.6)	95586 (0.46)	96259.4 (1.16)	96012 (0.9)	96261.6 (1.17)	96101 (1)	95151
X-n294-k50	47212.8 (0.11)	47191 (0.06)	47304.2 (0.3)	47266 (0.22)	47530.4 (0.78)	47402 (0.51)	47456.8 (0.63)	47384 (0.47)	47161
X-n298-k31	34246.2 (0.04)	34231 (0)	34317.6 (0.25)	34276 (0.13)	34497.4 (0.78)	34368 (0.4)	34539 (0.9)	34458 (0.66)	34231
X-n303-k21	21784.8 (0.22)	21760 (0.11)	21858.4 (0.56)	21819 (0.38)	21992.8 (1.18)	21910 (0.8)	21972.4 (1.09)	21909 (0.8)	21736
X-n308-k13	25897.8 (0.15)	25876 (0.07)	25902.2 (0.17)	25868 (0.03)	26181.2 (1.25)	26050 (0.74)	26237.4 (1.46)	26113 (0.98)	25859
X-n313-k71	94193.2 (0.16)	94157 (0.12)	94649 (0.64)	94365 (0.34)	95097.2 (1.12)	95017 (1.04)	95147 (1.17)	94933 (0.95)	94043
X-n317-k53	78362 (0.01)	78355 (0)	78395.8 (0.05)	78379 (0.03)	78556.2 (0.26)	78405 (0.06)	78731.6 (0.48)	78420 (0.08)	78355
X-n322-k28	29888 (0.18)	29869 (0.12)	29970.6 (0.46)	29909 (0.25)	30310.2 (1.6)	30083 (0.83)	30128.4 (0.99)	30079 (0.82)	29834
X-n327-k20	27584 (0.19)	27571 (0.14)	27638.2 (0.39)	27588 (0.2)	27984 (1.64)	27894 (1.31)	27958.6 (1.55)	27830 (1.08)	27532
X-n331-k15	31133.2 (0.1)	31105 (0.01)	31131.2 (0.09)	31103 (0)	31377.4 (0.89)	31230 (0.41)	31368.8 (0.86)	31242 (0.45)	31102
X-n336-k84	139731.4 (0.45)	139647 (0.39)	139989.6 (0.63)	139848 (0.53)	140453.6 (0.97)	140248 (0.82)	140724.4 (1.16)	140648 (1.1)	139111
X-n344-k43	42145.8 (0.23)	42092 (0.1)	42271 (0.53)	42194 (0.34)	42528.4 (1.14)	42418 (0.88)	42530.4 (1.14)	42475 (1.01)	42050
X-n351-k40	25971.6 (0.29)	25940 (0.17)	26036.6 (0.54)	25992 (0.37)	26295.4 (1.54)	26167 (1.05)	26262 (1.41)	26193 (1.15)	25896
X-n359-k29	51735.8 (0.45)	51623 (0.23)	51763.8 (0.5)	51659 (0.3)	52171.4 (1.29)	52088 (1.13)	52244.8 (1.44)	52103 (1.16)	51505
X-n367-k17	22814 (0)	22814 (0)	22844.4 (0.13)	22814 (0)	23084 (1.18)	22899 (0.37)	23220.8 (1.78)	23098 (1.24)	22814
X-n376-k94	147736 (0.02)	147718 (0)	147748 (0.02)	147726 (0.01)	147860.4 (0.1)	147758 (0.03)	147891.4 (0.12)	147837 (0.08)	147713
X-n384-k52	66158.4 (0.35)	66088 (0.24)	66208.8 (0.43)	66163 (0.36)	66634.4 (1.07)	66370 (0.67)	66646.8 (1.09)	66435 (0.77)	65940
X-n393-k38	38276.8 (0.04)	38260 (0)	38384 (0.32)	38323 (0.16)	38703.4 (1.16)	38567 (0.8)	38598.4 (0.88)	38462 (0.53)	38260
X-n401-k29	66330 (0.27)	66255 (0.15)	66321.6 (0.25)	66306 (0.23)	66879.6 (1.1)	66693 (0.81)	66685.2 (0.8)	66509 (0.54)	66163

Table 7: Comparison of solution quality with $T_{max} = N \times {}^{240}/_{100}$ seconds in 5 runs. The table shows the computational results on the $\mathbb{X}$ instances Uchoa et al., 2017 of the Guided-MS with state-of-the-art methods, such as LKH-3 (Helsgaun, 2017), HGS (Vidal, 2022), and FILO (Accorsi and Vigo, 2021) (continued)

Instance	HGS		FILO		LKH-3		Guided-MS		BKS
	Avg (Gap)	Best (Gap)	Avg (Gap)	Best (Gap)	Avg (Gap)	Best (Gap)	Avg (Gap)	Best (Gap)	
X-n411-k19	19744.4 (0.16)	19731 (0.1)	19807.6 (0.48)	19765 (0.27)	20016.4 (1.54)	19959 (1.25)	20153.4 (2.24)	20027 (1.6)	19712
X-n420-k130	107900.2 (0.09)	107831 (0.03)	108068 (0.25)	107948 (0.14)	108462.2 (0.62)	108383 (0.54)	108427.6 (0.58)	108333 (0.5)	107798
X-n429-k61	65514.2 (0.1)	65483 (0.05)	65722.8 (0.42)	65630 (0.28)	66263 (1.24)	66105 (1)	66265.2 (1.25)	66151 (1.07)	65449
X-n439-k37	36404.4 (0.04)	36395 (0.01)	36421.6 (0.08)	36395 (0.01)	36670 (0.77)	36593 (0.56)	36543.6 (0.42)	36467 (0.21)	36391
X-n449-k29	55582.4 (0.63)	55435 (0.37)	55582.8 (0.63)	55531 (0.54)	56613.4 (2.5)	56506 (2.3)	56230.8 (1.81)	56087 (1.55)	55233
X-n459-k26	24182.8 (0.18)	24165 (0.11)	24222.8 (0.35)	24219 (0.33)	24595.2 (1.89)	24522 (1.59)	24555.4 (1.73)	24498 (1.49)	24139
X-n469-k138	222928.4 (0.5)	222702 (0.4)	224538.8 (1.22)	224360 (1.14)	223803 (0.89)	223072 (0.56)	226318.8 (2.03)	225764 (1.78)	221824
X-n480-k70	89600 (0.17)	89479 (0.03)	89838 (0.43)	89782 (0.37)	90181.2 (0.82)	89999 (0.61)	90075.4 (0.7)	89958 (0.57)	89449
X-n491-k59	66775 (0.44)	66690 (0.31)	66828.4 (0.52)	66670 (0.28)	67713.6 (1.85)	67516 (1.55)	67392.4 (1.37)	67245 (1.15)	66487
X-n502-k39	69274.2 (0.07)	69263 (0.05)	69263.2 (0.05)	69243 (0.02)	69400 (0.25)	69320 (0.14)	69355.2 (0.19)	69328 (0.15)	69226
X-n513-k21	24248.6 (0.2)	24236 (0.14)	24273.8 (0.3)	24247 (0.19)	24546 (1.43)	24464 (1.09)	24589.2 (1.6)	24512 (1.29)	24201
X-n524-k153	154904.8 (0.2)	154796 (0.13)	155332.4 (0.48)	155259 (0.43)	154820.2 (0.15)	154723 (0.08)	156370.4 (1.15)	156146 (1)	154593
X-n536-k96	95203.4 (0.38)	95061 (0.23)	95696.2 (0.9)	95656 (0.85)	96938 (2.21)	96316 (1.55)	96296.4 (1.53)	95982 (1.2)	94868
X-n548-k50	86905.2 (0.24)	86856 (0.18)	86807.4 (0.12)	86748 (0.06)	87070.2 (0.43)	86997 (0.34)	87168.6 (0.54)	87123 (0.49)	86700
X-n561-k42	42792.8 (0.18)	42777 (0.14)	42904 (0.44)	42828 (0.26)	43157.8 (1.03)	43085 (0.86)	43255.4 (1.26)	43071 (0.83)	42717
X-n573-k30	50911.8 (0.47)	50873 (0.39)	50846.8 (0.34)	50796 (0.24)	51222 (1.08)	51060 (0.76)	51265 (1.17)	51164 (0.97)	50673
X-n586-k159	191005.4 (0.36)	190889 (0.3)	191604.6 (0.68)	191369 (0.55)	191772 (0.77)	191445 (0.59)	193338.4 (1.59)	192882 (1.35)	190316
X-n599-k92	108794.4 (0.32)	108738 (0.26)	109081.2 (0.58)	109014 (0.52)	110087.8 (1.51)	109971 (1.4)	110018.8 (1.45)	109903 (1.34)	108451
X-n613-k62	59821.2 (0.48)	59789 (0.43)	59881 (0.58)	59773 (0.4)	60529.4 (1.67)	60406 (1.46)	60439.6 (1.52)	60352 (1.37)	59535
X-n627-k43	62613.2 (0.72)	62534 (0.6)	62384.8 (0.36)	62299 (0.22)	63284 (1.8)	63010 (1.36)	63176.2 (1.63)	62944 (1.25)	62164
X-n641-k35	64085 (0.63)	64029 (0.54)	63929.6 (0.39)	63845 (0.26)	64734.8 (1.65)	64632 (1.49)	64724.2 (1.64)	64526 (1.33)	63694
X-n655-k131	106842.4 (0.06)	106805 (0.02)	106819.6 (0.04)	106807 (0.03)	107050.4 (0.25)	107016 (0.22)	107035 (0.24)	107001 (0.21)	106780
X-n670-k130	147198.2 (0.59)	146991 (0.45)	147847.8 (1.04)	147286 (0.65)	147606.6 (0.87)	147240 (0.62)	149282 (2.02)	148293 (1.34)	146332
X-n685-k75	68513 (0.45)	68467 (0.38)	68616.6 (0.6)	68543 (0.5)	69374.8 (1.72)	69137 (1.37)	69417.6 (1.78)	69296 (1.6)	68205
X-n701-k44	82636.8 (0.87)	82501 (0.71)	82348.8 (0.52)	82326 (0.49)	83173.6 (1.53)	82536 (0.75)	83254.2 (1.62)	83139 (1.48)	81923
X-n716-k35	43617.8 (0.56)	43559 (0.43)	43673.8 (0.69)	43635 (0.6)	44308.8 (2.16)	44068 (1.6)	44593.2 (2.81)	44320 (2.18)	43387
X-n733-k159	136564 (0.28)	136529 (0.25)	136770.4 (0.43)	136537 (0.26)	137210.2 (0.75)	137125 (0.69)	137483.8 (0.95)	137363 (0.86)	136190
X-n749-k98	78010.6 (0.96)	77922 (0.85)	77883.6 (0.8)	77832 (0.73)	78815.6 (2)	78635 (1.77)	78285.2 (1.32)	78167 (1.16)	77314
X-n766-k71	114842.2 (0.37)	114783 (0.32)	115138.2 (0.63)	114913 (0.43)	116322.8 (1.67)	116063 (1.44)	116109.4 (1.48)	115961 (1.35)	114454
X-n783-k48	73207.2 (1.13)	73093 (0.98)	72878.8 (0.68)	72674 (0.4)	73933.4 (2.14)	73826 (1.99)	73754.6 (1.89)	73558 (1.62)	72394
X-n801-k40	73738.4 (0.59)	73575 (0.37)	73446.8 (0.19)	73378 (0.1)	73988 (0.93)	73748 (0.6)	73950.4 (0.88)	73807 (0.68)	73305
X-n819-k171	158703.2 (0.37)	158651 (0.34)	159434 (0.83)	159235 (0.7)	160358.8 (1.42)	159978 (1.17)	160809.2 (1.7)	160521 (1.52)	158121
X-n837-k142	195311 (0.81)	194955 (0.63)	194759.6 (0.53)	194652 (0.47)	195682.6 (1)	195484 (0.9)	196300.6 (1.32)	195957 (1.15)	193737
X-n856-k95	89088.6 (0.14)	89055 (0.1)	89133.2 (0.19)	89082 (0.13)	89551 (0.66)	89413 (0.5)	89535.6 (0.64)	89359 (0.44)	88965
X-n876-k59	100160.8 (0.87)	100059 (0.77)	99736 (0.44)	99713 (0.42)	100799.8 (1.51)	100544 (1.25)	100706.8 (1.42)	100502 (1.21)	99299
X-n895-k37	54344.8 (0.9)	54249 (0.72)	54303.2 (0.82)	54208 (0.65)	56757.4 (5.38)	55974 (3.92)	55578.2 (3.19)	55426 (2.91)	53860
X-n916-k207	331787.4 (0.79)	331606 (0.74)	331461.8 (0.69)	331011 (0.56)	331701.2 (0.77)	331121 (0.59)	334212.8 (1.53)	333520 (1.32)	329179
X-n936-k151	133656.6 (0.71)	133288 (0.43)	133854.8 (0.86)	133627 (0.69)	134221.6 (1.14)	134036 (1)	135267.4 (1.92)	134847 (1.61)	132725
X-n957-k87	85707.6 (0.28)	85697 (0.27)	85595.6 (0.15)	85570 (0.12)	86145.4 (0.8)	85953 (0.57)	86022.6 (0.65)	85818 (0.41)	85465
X-n979-k58	120073.2 (0.92)	119838 (0.72)	119607.4 (0.53)	119348 (0.31)	121858.6 (2.42)	121395 (2.03)	120384.6 (1.18)	119950 (0.82)	118987
X-n1001-k43	73419.4 (1.47)	73322 (1.34)	72887.4 (0.74)	72808 (0.63)	74141.4 (2.47)	73867 (2.09)	74688.6 (3.23)	74462 (2.91)	72359
Minimum Gap	0	0	0	0	0.03	0	0	0	
Average Gap	0.26	0.19	0.37	0.25	1.03	0.72	1.09	0.8	
Median Gap	0.18	0.09	0.36	0.22	0.9	0.59	1.07	0.78	
Maximum Gap	1.47	1.34	1.22	1.14	5.38	3.92	3.23	2.91	

6 Conclusion

In this paper, we have presented a method to enhance a metaheuristic algorithm for solving CVRP by formulating a feature-based guidance. To formulate the proposed feature-based guidance, we have developed and explained the solution learning model that can classify whether the resulting solution is good or not based on both instances and solutions features inputs.

Furthermore, to implement our proposed guidance, we have developed a new metaheuristic algorithm composed of neighborhood improvements and path relinking. As our proposed algorithm utilizes path relinking, we have also proposed a novel mechanism of path relinking that compromises with the giant tour concatenation and hybridizes with the split algorithm. Here, we also show that our proposed path relinking mechanism can make a statistically significant contribution to the overall mechanism of the proposed algorithm. Finally, the resulting proposed guidance can statistically improve the performance of our proposed metaheuristic algorithm. Furthermore, we have also shown that our proposed guided metaheuristic competes with other state-of-the-art algorithms for solving the CVRP.

From a more general perspective, we understand that by only utilizing simple feature-based guidance, we are still not unlocking the full performance of ML to improve the metaheuristic algorithm. For future work, we will try to enhance the influence of the ML on the proposed hybrid algorithm. Thus, we plan to adapt the mechanism of learning repeated decisions. In the next proposed hybrid algorithm, our objective is to develop a hybrid algorithm that will adjust its decision and its behavior automatically.

A Appendix

A.1 Feature of VRP

Mathematical notations for features of VRP In the context of VRP features, let R represents the set of all routes in a solution. A route, denoted as $r_k = c_1, c_2, ..., c_k \in R$, comprises a sequence of nodes. The rank of the neighborhood between nodes c_i and c_j is denoted as $\mathcal{R}_{ij}$. We define $\mathcal{R}_k$ as the average rank of the neighborhood for each member of route k, calculated as $\mathcal{R}_k = {}^{(\sum i,j \in k \mathcal{R}ij)}/_{k}$. Each route $k \in \{r_1, r_2, \ldots, r_k\} \in R$ consists of edges $(D, c_1), (c_1, c_2), \ldots, (c_k, D) \in E$, where D represents the depot. The Euclidean distance between nodes c_j and c_k is denoted as $d(c_j, c_k)$. Additionally, $x(c_k)$ and $y(c_k)$ represent the x and y-coordinates of node c_k respectively, with $|x|$ indicating the absolute value of x. The demand of node c_j is denoted as $q(c_j)$. The x-coordinate of the center of gravity G_k of route k is given by $x(G_k) = {}^{(\sum x(c_k)+x(D))}/_{k+1}$, and the y-coordinate is computed as $y(G_k) = {}^{(\sum y(c_k)+y(D))}/_{k+1}$ (Arnold and Sörensen, 2019b). Additionally, we define L_{G_k} as the line passing through D and G_k, and $d(L_{G_k}, c_i)$ as the distance between line L_{G_k} and customer c_i. The distance is positive when the customer is on the right side of the line and negative otherwise. We also introduce $\text{rad}(c_j, c_k)$ to denote the difference in radians between nodes c_j and c_k concerning the depot D, representing the angle spanned between the lines connecting each node with the depot. Finally, $I(r_1, r_2)$ denotes the number of intersections between routes r_1 and r_2

Instance Features Here are the specific details about features that depend on the respective instance. These features are drawn from the research of Arnold and Sörensen, 2019b and Lucas et al., 2020.

I01: Number of customers

$$\text{I01}(S) = N = V - 1 \tag{11}$$

I02: Number of vehicles

$$\text{I02}(S) = R \tag{12}$$

I03: Degree of capacity utilization

$$\text{I03}(S) = \frac{\sum_{j \in N} q(c_j)}{Q \cdot R} \tag{13}$$

I04: Average distance between each pair of customers

$$\text{I04}(S) = \frac{\sum_{i,j \in N \setminus i=j} d(c_i, c_j)}{N} \tag{14}$$

I05: Standard deviation of the pairwise distance between customers

$$\text{I05}(S) = \sqrt{\frac{\sum_{i,j \in N \setminus i=j} (d(c_i, c_j) - \text{I04})^2}{N}} \tag{15}$$

I06: Average distance from customers to the depot

$$\text{I06}(S) = \frac{\sum_{i \in N} d(c_i, D)}{N} \tag{16}$$

I07: Standard deviation of the distance from customers to the depot

$$\text{I07}(S) = \sqrt{\frac{\sum_{i \in N} (d(c_i, D) - \text{I06})^2}{N}} \tag{17}$$

I08: Average radians of customers towards the depot

$$\text{I08}(S) = \frac{\sum_{i \in N} \text{rad}(c_i, D)}{N} \tag{18}$$

I09: Standard deviation of the radians of customers towards the depot

$$\text{I09}(S) = \sqrt{\frac{\sum_{i \in N} (\text{rad}(c_i, D) - \text{I08})^2}{N}} \tag{19}$$

Solution Features This provides detailed information about features that rely on the outcomes of solutions. Most of these features are drawn from Arnold and Sörensen, 2019b and Lucas et al., 2020. Additionally, we propose several additional features related to the capacity utilization in the resulting solutions. Moreover, we also introduce additional features associated with its longest distance, depicted in Figure 9. In total, there are 22 solution features that capture the structure and attributes of the resulting solution.

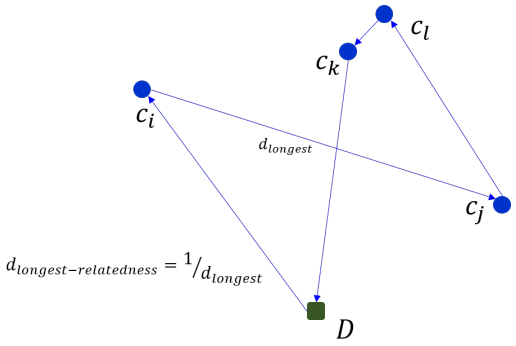

Figure 9: Illustration of longest-relatedness distance in a route of a solution

S01: Average width of each route

$$\mathrm{S01}(S) = \frac{1}{R} \cdot \sum_{k \in R} \left(\max_{j \in k} d(L_{G_k}, c_j) - \min_{j \in k} d(L_{G_k}, c_j) \right) \tag{20}$$

S02: Standard deviation of the width of each route

$$\mathrm{S02}(S) = \sqrt{\frac{\sum_{k \in R} ((\max_{j \in k} d(L_{G_k}, c_j) - \min_{j \in k} d(L_{G_k}, c_j)) - \mathrm{S01})^2}{R}} \tag{21}$$

S03: Average span of each route

$$\mathrm{S03}(S) = \frac{1}{R} \cdot \sum_{k \in R} \max_{i,j \in k} \mathrm{rad}(c_i, c_j) \tag{22}$$

S04: Standard deviation of the span of each route

$$\mathrm{S04}(S) = \sqrt{\frac{\sum_{k \in R} (\max_{i,j \in k} \mathrm{rad}(c_i, c_j) - \mathrm{S03})^2}{R}} \tag{23}$$

S05: Average depth of each route

$$\mathrm{S05}(S) = \frac{1}{R} \cdot \sum_{k \in R} \max_{j \in k} d(c_j, D) \tag{24}$$

S06: Standard deviation of the depth of each route

$$\mathrm{S06}(S) = \sqrt{\frac{\sum_{k \in R} (\max_{j \in j} d(c_j, D) - \mathrm{S05})^2}{R}} \tag{25}$$

S07: Average of the distance of the first and last edge of each route divided by the total length of the route

$$\mathrm{S07}(S) = \frac{1}{2 \cdot R} \cdot \sum_{k \in R} \frac{d(D, c_1) + d(c_k, D)}{\sum_{i,j \in k} d(c_i, c_j)} \tag{26}$$

S08: Mean length of the longest edge of each route

$$\mathrm{S08}(S) = \frac{1}{R} \cdot \sum_{k \in R} \max_{i,j \in k} d(c_i, c_j) \tag{27}$$

S09: Length of the longest edge of all each route, divided by the average of the length of each route

$$\mathrm{S09}(S) = \frac{\max_{k \in R} \max_{i,j \in k} d(c_j, c_k)}{(\sum_{k \in R} \sum_{i,j \in k} d(c_i, c_j))/R} \tag{28}$$

S10: Length of the longest interior edge of each route divided by mean of the length of each route

$$\mathrm{S10}(S) = \frac{\max_{k \in R} \max_{i,j \in k-1} d(c_{i+1}, c_k)}{\sum_{k \in R} \sum_{i,j \in k} d(c_i, c_j)/R} \tag{29}$$

S11: Mean length of the first and last edges of each route

$$\mathrm{S11}(S) = \frac{1}{2 \cdot R} \cdot \sum_{k \in R} (d(D, c_1) + d(c_k, D)) \tag{30}$$

S12: Average of demand of the first and last customer of each route

$$\mathrm{S12}(S) = \frac{1}{2 \cdot R} \cdot \sum_{k \in R} (q(c_1) + q(c_k)) \tag{31}$$

S13: Average of demand of the farthest customer

$$\mathrm{S13}(S) = \frac{1}{R} \cdot \sum_{k \in R} \max_{j \in k} q(c_j) \tag{32}$$

S14: Standard deviation of the demand of the farthest customer

$$\mathrm{S14}(S) = \sqrt{\frac{\sum_{k \in R} \max_{j \in k} q(c_j) - \mathrm{S13})^2}{R}} \tag{33}$$

S15: Standard deviation of the length of each route

$$\mathrm{S15}(S) = \sqrt{\frac{\sum_{k \in R} (\sum_{i,j \in k} d(c_i, c_j) - (\sum_{k \in R} \sum_{i,j \in k} {}^{d(c_i, c_j)}/_{R}))^2}{R}} \tag{34}$$

S16: Mean distance between each route from their centre of gravity

$$\mathrm{S16}(S) = \frac{1}{R \cdot (R-1)} \cdot \sum_{r_1 \in R} \sum_{r_2 \in R \setminus r_1} d(G_{r_1}, G_{r_2}) \tag{35}$$

S17: Standard deviation of the number of customers of each route

$$\mathrm{S17}(S) = \sqrt{\frac{\sum_{k \in R} (k - \frac{V}{R})^2}{R}} \tag{36}$$

S18: Average of the degree of the neighborhood for every route

$$\mathrm{S18}(S) = \mathrm{S18} = \frac{1}{R} \cdot \sum_{k \in R} \mathcal{R}_k \tag{37}$$

S19: The average of the capacity utilization for every route

$$\mathrm{S19}(S) = \frac{1}{R} \cdot \sum_{k \in R} \sum_{j \in k} q(c_j)/Q \tag{38}$$

S20: Standard deviation of the capacity utilization for every route

$$\mathrm{S20}(S) = \sqrt{\frac{\sum_{k \in R} \left(\sum_{j \in k} q(c_j)/Q - \left(\sum_{k \in R} \sum_{j \in k} q(c_j)/Q \right) \right)^2}{R}} \tag{39}$$

S21: Average length of the longest distance-relatedness

$$\mathrm{S21}(S) = \frac{1}{R} \cdot \sum_{k \in R} \frac{1}{\max_{i,j \in k} d(c_i, c_j)} \tag{40}$$

S22: Standard deviation of the length of the longest distance-relatedness

$$\mathrm{S22}(S) = \sqrt{\frac{\sum_{k \in R} \left({}^{1}/_{\max_{i,j \in k} d(c_i, c_j)} - \left(\sum_{k \in R} {}^{1}/_{\max_{i,j \in k} d(c_i, c_j)/R} \right) \right)^2}{R}} \tag{41}$$

A.2 Feature Engineering

Feature engineering, also known as feature discovery, involves extracting characteristics, properties, or attributes from raw data to facilitate the training of downstream statistical models (Hastie et al., 2009). As outlined in Section 2.1, the dataset generated for this research is divided into two groups: optimal and near-optimal solutions. Both optimal and near-optimal solutions are obtained by solving all 10,000 𝕏𝕄𝕃100 instances (Queiroga et al., 2021). These instances consist of a similar number of customer nodes, totaling 100 nodes each. However, they vary across different categories, such as the position of their depots, the distribution of customers, the distribution of demands, and the average size of the routes in their optimal solutions. In total, there are 378 groups of instances. The first 172 groups contain 27 instances each, while the remaining 206 groups contain 26 instances respectively. Furthermore, the dataset comprises 20,000 data points, with an equal distribution of optimal and near-optimal solutions. The features utilized for classification in the proposed learning model are categorized into two groups: instance features and solution features, which are comprehensively described in Appendix A.1.

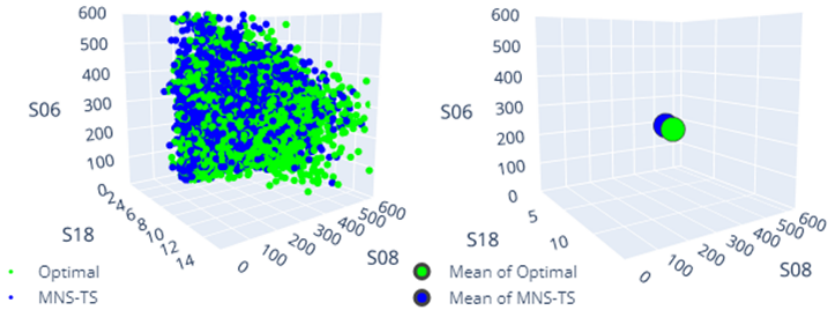

Figure 10: Illustration between the optimal and near-optimal solution in the simplified 3D features space

A scatter chart illustrating the distribution between optimal and near-optimal solutions concerning features S06, S08, and S18 is presented in Figure 10. From the scatter chart, it is obvious that the optimal and near-optimal solutions exhibit proximity, making it challenging to distinguish between the two groups.

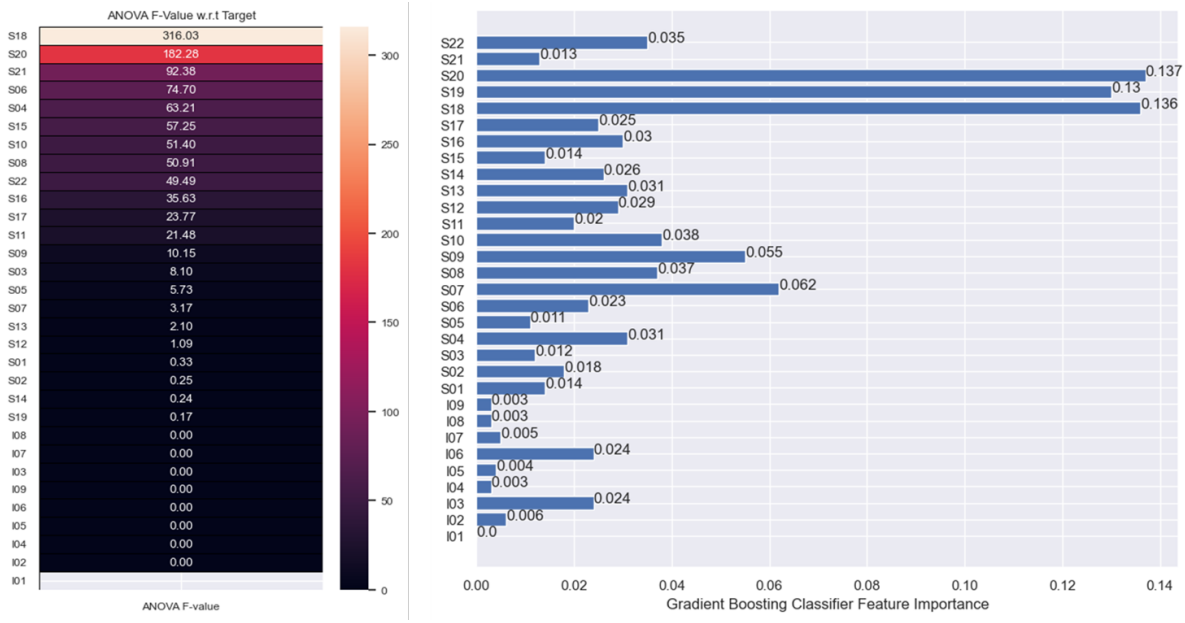

Figure 11: Order of ANOVA F-value concerning its label (left) and the order of impurity feature importance (right)

By quickly analyzing how these features correlated with the quality of solutions using ANOVA F-values, shown in Figure 11 (left), we can see that the higher F-value suggests that the feature is statistically significant concerning the target variable. According to ANOVA F-value, S18 is the most statistically significant to the target variable.

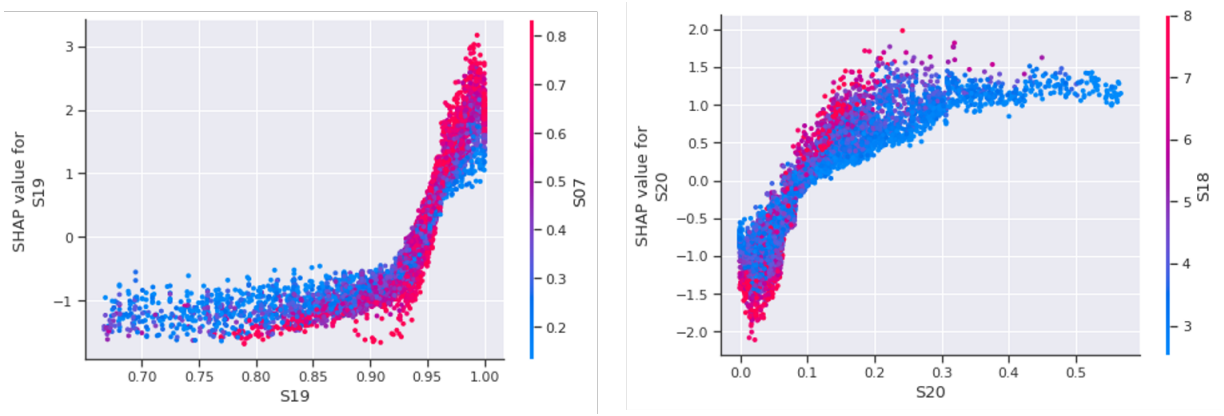

Figure 12: Dependency plot between S07 and S19 (left) and S20 and S18 (right). In dependency plot between S07 and S19 (left), it is shown that S19 has a strong interaction with S07, meaning the average distance of the first and last edge of each route has a strong interaction with capacity utilization for every route. Meanwhile, the dependency plot between S20 and S18 (right) shows that S19 has a strong interaction with S07, meaning the average distance of the first and last edge of each route has a strong interaction with capacity utilization for every route.

However, as it does not capture the full range of relationships between features and the target variable, we cannot conclude that S18 is the most important feature of the target variable. Thus, by testing several classification algorithms, shown in Table 1, according to the resulted F_1-scores, we can conclude that the Gradient Boosting algorithm resulted from the best prediction. We can find its feature importance value by performing further based on the impurity reduction criteria used during tree constructions, shown in Figure 11 (right). This approach assumes that features that lead to the largest decrease in impurity are considered more important. However, the disadvantage of these feature importance, compared with the order of features in the global feature importance plot in Figure 3 (left), is that these values ignore the interaction between features. Meanwhile, the order of features in the global feature importance plot in Figure 3 captures interactions between features, as the SHAP value provides a more comprehensive understanding of feature importance by showing how these features behave for making a decision (Lundberg and Lee, 2017; Lundberg et al., 2020). The interaction between features can be depicted by using a dependency plot. Here Figure 12 shows the dependency plot between strongly correlated features.

A.3 Concatenation Method and Split Algorithm

Prins, 2004 shows an innovative approach for solving CVRP, called *"route-first cluster-second"* paradigm. The approach is started by forming a giant tour (refer to Figure 13).

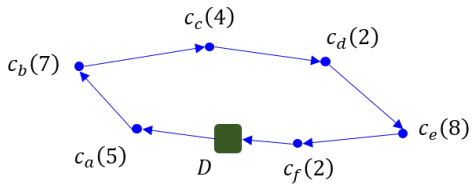

Figure 13: Example of giant tour of VRP

In this research, the giant tour T of solution S is formed by using a simple randomized concatenation. The concatenation process starts by identifying the head customer of every route $r \in R$. Thus, the process is continued by randomized concate the route, as shown in Algorithm 13.

Algorithm 13 Randomized concatenation mechanism

input: solution S

procedure CONCATENATION(S)
  $\mathbb{L}_{head} \leftarrow \varnothing$ ▷ list of first customer
  for $r \in R$ **do**
    $\mathbb{L}_{head}[r] \leftarrow$ GETFIRSTCUSTOMEROFROUTE(r)
  end for
  $\mathbb{L}_{head} \leftarrow$ RANDOMREORDERING($\mathbb{L}_{head}$)
  $T \leftarrow \varnothing$ ▷ initialize a giant tour
  for $c \in \mathbb{L}_{head}$ **do**
    $r \leftarrow$ GETCUSTOMERLISTFROM(c) ▷ get the route
    $T \leftarrow T \cup r$ ▷ append route in to the giant tour
  end for
  return T
end procedure

In forming a giant tour T, we then process the solution T to perform path relinking (see Section 3.4). Subsequently, during the path relinking, we will produce several intermediate solutions T_{eval} (described in Appendix A.4). If the intermediate solution is sufficient (see Algorithm 15), it will transform into a VRP solution (see Figure 15), through Figure 14, using a split algorithm, described in Algorithm 14.

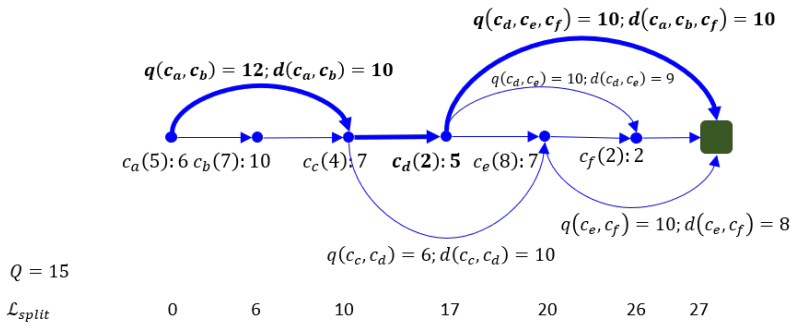

Figure 14: Example of an acyclic graph of a giant tour Figure 13

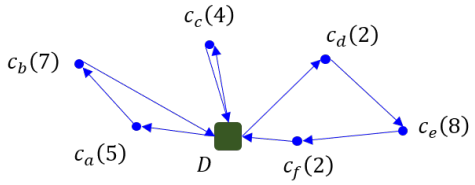

Figure 15: CVRP solution after performing split algorithm to the giant tour Figure 13 through Figure 14

The shortest path, shown in Figure 14, can be calculated using Bellman algorithm for solving a directed acyclic graph (Bellman, 1958). In detail, let's construct an auxiliary graph $H = (X, U)$. Here, X consists of $V + 1$ nodes labeled from $D = 0$ to V, and U includes edges $(i - 1, j)$ for each customer subsequence $(c_i, c_{i+1}, \ldots, c_j)$ that a vehicle can visit. The optimal splitting is determined by finding the shortest path from node D to node V in H, as shown in Algorithm 14.

Algorithm 14 Split and mechanism of constructing a CVRP solution

input: giant tour solution T

procedure SPLIT(T)
  $\mathcal{L}_{split}[0] \leftarrow 0$ ▷ initialization split label
  $\mathcal{P}_{split}[0] \leftarrow 0$ ▷ predecessor nodes
  for $i \leftarrow 1$ **to** V **do**
    $\mathcal{L}_{split}[i] \leftarrow \infty$
  end for
  for $i \leftarrow 1$ **to** V **do** ▷ start splitting processes
    $j \leftarrow i, \sum q \leftarrow 0$
    repeat
      if $i = j$ **then**
        $\sum d \leftarrow d(D, c_i) + c(c_i, D)$ ▷ initialize a route that only served one customer c_i
      else
        $\sum d \leftarrow \sum d + d(c_{j-1}, D) + d(c_{j-1}, c_j) + d(c_j, D)$ ▷ adding customers in the route
      end if
      if $\sum q \leq Q \wedge \mathcal{L}_{split}[i-1] + \sum d < \mathcal{L}_{split}[j]$ **then**
        $\mathcal{L}_{split}[j] \leftarrow \mathcal{L}_{split}[i-1] + \sum d$
        $\mathcal{P}_{split}[j] \leftarrow i - 1$
      end if
      $j \leftarrow j + 1$
    until $j > V \vee \sum q > Q$
  end for
  $S \leftarrow \varnothing$ ▷ initialization CVRP solution
  repeat
    $r \leftarrow \varnothing$ ▷ initialize an empty route
    for $k \leftarrow \mathcal{P}_{split}[j] + 1$ **to** j **do** ▷ forming a route
      $r \leftarrow$ PUSHBACK(c_k) ▷ add customer c_k at the end of the route
    end for
    $S \leftarrow S \cup r$ ▷ append a route to the VRP solution
    $j \leftarrow \mathcal{P}_{split}[j]$
  until j = 0
  return S
end procedure

Since the split algorithm assesses each subsequence in a constant time, given the incremental updates to total demand, $\sum q$, and cost the total cost, $\sum d$, of the corresponding route as j advances, instead of re-calculating them through a loop from i to j. As a result, the complexity is directly proportional to the number of edges in H, which is at worst $O(n^2)$.

A.4 Neighborhood Search for Path Relinking

To generate intermediate solutions for investigating potential improvements, we need to explore the solution space between the initial and guiding solutions. To explore this solution space, we perform a neighborhood search between the initial solution and the guiding solution, as shown in Algorithm 10, line 7. This neighborhood search process involves systematically examining neighboring solutions by making small modifications to the current solutions. An overview of the neighborhood search in the proposed path relinking is shown in Algorithm 15.

Algorithm 15 Iteratively neighborhood evaluation processes

input: initial solution T_i, guiding solution T_g, number of loop N_{pr}
list of restricted neighborhood Lpr, elite set $\mathbb{E}$, current best solution S_{best}

1: **procedure** EvaluateNeighborhood(T_i, T_g, N_{pr}, L_{pr}, $\mathbb{E}$, S_{best})
2: $T \leftarrow T_i$
3: $L_{tabu} \leftarrow \varnothing$ ▷ initialize tabu list
4: **for** $move \leftarrow 1$ **to** N_{pr} **do**
5: $(c_{swap}, p_i, p_j) \leftarrow$ GetPositionSwap(T, T_g, L_{pr}, L_{tabu})
6: $T_{eval} \leftarrow$ Swap(T, p_i, p_j) ▷ intermediate solution
7: **if** Cost(T_{eval}) < Cost(T) **then**
8: $S_{eval} \leftarrow$ Split(T_{eval}) ▷ transform into VRP solution
9: $\mathbb{E} \leftarrow$ UpdateEliteSet($S_{eval}, \mathbb{E}$)
10: **if** Cost(S_{eval}) < Cost(S_{best}) **then**
11: $S_{best} \leftarrow S_{eval}$
12: **end if**
13: **end if**
14: $L_{tabu} \leftarrow L_{tabu} \cup c_{swap}$ ▷ update tabu list
15: **end for**
16: **return** $\mathbb{E}$, S_{best}
17: **end procedure**

As depicted in Algorithm 15, line 6, these neighborhood searches involve swapping the customer nodes towards the guiding solution. The aim is to identify solutions that closely resemble the initial solutions but may potentially yield better objective function values. Moreover, the search in path relinking will automatically terminate whenever the intermediate solution is better than the guiding solution, $\text{Cost}(T_{eval}) \leq \text{Cost}(T_g)$

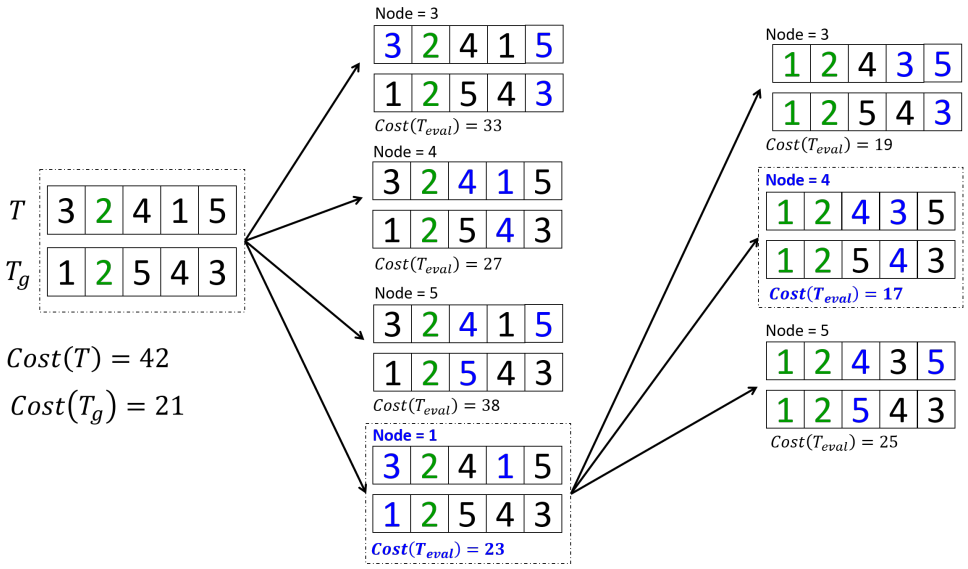

Figure 16: Illustrated of neighborhood evaluation in proposed path relinking. The process to get the best swap position is also described in Algorithm 16. As also shown in Algorithm 15 line 7, the intermediate solution T_{eval} will continue to split processes only when $\text{Cost}(T_{eval}) \leq \text{Cost}(T)$.

In our proposed path relinking, we implement the concept of truncated path relinking. Therefore, as illustrated in Algorithm 15, the number of search moves, denoted as N_{pr}, can be assumed to be less than or equal to the size of L_{pr}. Additionally, when instances are categorized as having long routes (determined using Equation (6)), after constructing S_{eval} (Algorithm 15, line 8), we conduct an additional local search improvement using only level 1 local search operators (as described in Section 3.3.2), *i.e.,* only considering intra- and inter-route local search operators, and without utilizing inter-route chain-move operators.

Algorithm 16 Get best swap position for transforming T toward T_g

input: current initial solution T, guiding solution T_g
list of restricted neighborhood L_{pr}, tabu list L_{tabu}

1: **procedure** GetPositionSwap(T, T_g, L_{pr}, L_{tabu})
2: $c_{swap} \leftarrow -1, p_i \leftarrow -1, p_j \leftarrow -1, f_{best} \leftarrow \infty$ ▷ initialization
3: **for** node $\in L_{pr}$ **do**
4: **if** node $\notin L_{tabu}$ **then**
5: $pos_i \leftarrow$ GetPositionNode(T, node) ▷ position of node in T
6: $pos_g \leftarrow$ GetPositionNode(T_g, node) ▷ position of node in T_g
7: $\varepsilon \leftarrow$ PossibilitySwap(p_i, p_g)
8: **if** $\top\varepsilon$ **then**
9: **if** CostSwapOn(pos_i, pos_g) $< f_{best}$ **then**
10: $f_{best} \leftarrow$ CostSwapOn(pos_i, pos_g)
11: $c_{swap} \leftarrow$ node
12: $p_i \leftarrow pos_i$
13: $p_j \leftarrow pos_g$
14: **end if**
15: **end if**
16: **end if**
17: **end for**
18: **return** c_{swap}, p_i, p_j
19: **end procedure**

A.5 Result on Randomly 100 Sampled XML100 Instances

Table 8: Comparison of solution quality with $T_{max} = 60$ seconds in 5 runs

Instance	MNS-TS		MS		Guided-MS		Optimal
	Avg (Gap)	Best (Gap)	Avg (Gap)	Best (Gap)	Avg (Gap)	Best (Gap)	
XML100_1111_2	38434.6 (0.16)	38410 (0.09)	38459 (0.22)	38459 (0.22)	38459 (0.22)	38459 (0.22)	38374
XML100_1113_20	18462.4 (1.44)	18426 (1.24)	18200 (0)	18200 (0)	18200 (0)	18200 (0)	18200
XML100_1121_3	31812.4 (1.3)	31798 (1.25)	31512 (0.34)	31471 (0.21)	31509.8 (0.33)	31471 (0.21)	31405
XML100_1124_23	11277.8 (1.69)	11220 (1.17)	11093.6 (0.03)	11090 (0)	11090 (0)	11090 (0)	11090
XML100_1145_22	11747 (2.61)	11747 (2.61)	11457.4 (0.08)	11457 (0.08)	11457 (0.08)	11457 (0.08)	11448
XML100_1151_3	33600.4 (1.87)	33592 (1.85)	33123 (0.42)	33058 (0.23)	33070.8 (0.27)	32983 (0)	32983
XML100_1151_7	52154.8 (2.26)	51847 (1.65)	51437.8 (0.85)	51261 (0.5)	51401.4 (0.78)	51283 (0.55)	51004
XML100_1151_19	28491.8 (1.67)	28443 (1.5)	28218.8 (0.7)	28184 (0.57)	28131.6 (0.39)	28074 (0.18)	28023
XML100_1155_20	10311.2 (0.46)	10276 (0.12)	10274 (0.1)	10264 (0)	10270.6 (0.06)	10264 (0)	10264
XML100_1215_5	8771.2 (1.91)	8730 (1.43)	8607 (0)	8607 (0)	8607 (0)	8607 (0)	8607
XML100_1231_5	34083 (4.02)	34039 (3.88)	33018.2 (0.77)	32925 (0.48)	33004.2 (0.72)	32825 (0.18)	32767
XML100_1231_24	42513.8 (2.86)	42475 (2.77)	42293.8 (2.33)	42275 (2.29)	42308.6 (2.37)	42295 (2.33)	41330
XML100_1251_4	40355.8 (2.56)	40264 (2.33)	39738.8 (0.99)	39521 (0.44)	39728 (0.96)	39498 (0.38)	39349
XML100_1251_7	35173.8 (2.05)	35102 (1.85)	34677 (0.61)	34553 (0.25)	34656.8 (0.55)	34553 (0.25)	34466
XML100_1251_8	33242.8 (2.94)	33166 (2.7)	32843.8 (1.7)	32715 (1.3)	32753.6 (1.42)	32647 (1.09)	32294
XML100_1251_9	33978.4 (3.7)	33886 (3.42)	33031.8 (0.81)	32897 (0.4)	33033.6 (0.81)	32836 (0.21)	32767
XML100_1251_10	34009 (1.84)	33968 (1.72)	33678 (0.85)	33561 (0.5)	33670 (0.83)	33561 (0.5)	33394
XML100_1251_12	49466.6 (2.74)	49449 (2.71)	48991.8 (1.76)	48229 (0.17)	48762.8 (1.28)	48372 (0.47)	48146
XML100_1251_13	43110.2 (1.54)	43007 (1.29)	42500.8 (0.1)	42488 (0.07)	42497 (0.09)	42458 (0)	42458
XML100_1251_14	24982.2 (1.67)	24920 (1.41)	24691 (0.48)	24581 (0.03)	24646 (0.3)	24581 (0.03)	24573
XML100_1251_15	49226 (2.68)	49226 (2.68)	48549.6 (1.27)	48548 (1.27)	48503.2 (1.17)	48290 (0.73)	47940
XML100_1251_16	37127 (2.81)	37115 (2.78)	36573.2 (1.28)	36275 (0.45)	36526 (1.15)	36202 (0.25)	36111
XML100_1251_17	39646.2 (1.86)	39632 (1.82)	39153.2 (0.59)	39147 (0.58)	39147.8 (0.58)	39145 (0.57)	38923
XML100_1251_19	33962.2 (2.64)	33854 (2.32)	33378.8 (0.88)	33206 (0.36)	33335.2 (0.75)	33206 (0.36)	33088
XML100_1252_15	28145.6 (2.04)	28103 (1.89)	27939.2 (1.29)	27627 (0.16)	27939.2 (1.29)	27627 (0.16)	27583
XML100_1252_16	24703.8 (4.04)	24694 (4)	24608.2 (3.64)	24602 (3.61)	24523.8 (3.28)	24381 (2.68)	23744
XML100_1261_7	22279 (2.82)	22258 (2.73)	22137.8 (2.17)	22071 (1.86)	22139.2 (2.18)	22092 (1.96)	21667
XML100_1313_1	17023.8 (0.66)	16978 (0.39)	16943 (0.18)	16912 (0)	16943 (0.18)	16912 (0)	16912
XML100_1315_6	10799.4 (0.37)	10770 (0.09)	10803.6 (0.41)	10760 (0)	10803.6 (0.41)	10760 (0)	10760
XML100_1322_7	20873 (0.94)	20872 (0.93)	20684.4 (0.03)	20679 (0)	20679 (0)	20679 (0)	20679
XML100_1334_10	13220.6 (0.64)	13170 (0.25)	13184.2 (0.36)	13170 (0.25)	13184.2 (0.36)	13170 (0.25)	13137
XML100_1351_2	35501.8 (2.59)	35470 (2.5)	34640 (0.1)	34605 (0)	34640 (0.1)	34605 (0)	34605
XML100_1351_16	32497 (2.12)	32487 (2.09)	31969 (0.46)	31858 (0.11)	31969 (0.46)	31858 (0.11)	31822
XML100_1353_22	17729 (1.51)	17705 (1.37)	17568.2 (0.59)	17534 (0.4)	17557.2 (0.53)	17555 (0.52)	17465
XML100_2112_1	17853.8 (0.7)	17845 (0.65)	17730 (0)	17730 (0)	17730 (0)	17730 (0)	17730
XML100_2143_25	12681.4 (0.73)	12654 (0.51)	12590 (0)	12590 (0)	12590 (0)	12590 (0)	12590
XML100_2151_13	26520 (0.08)	26520 (0.08)	26569.4 (0.27)	26552 (0.2)	26565.2 (0.25)	26543 (0.17)	26499
XML100_2151_19	28718.6 (1.82)	28716 (1.81)	28259 (0.19)	28206 (0)	28244.6 (0.14)	28206 (0)	28206
XML100_2231_1	27441.8 (2.56)	27423 (2.49)	27053.8 (1.11)	27046 (1.08)	27054.2 (1.11)	27046 (1.08)	26756
XML100_2231_20	26693.2 (1.61)	26680 (1.56)	26486.6 (0.82)	26476 (0.78)	26475.2 (0.78)	26435 (0.62)	26271
XML100_2231_22	23820.2 (2.65)	23793 (2.53)	23246.4 (0.18)	23215 (0.04)	23247 (0.18)	23215 (0.04)	23205
XML100_2233_14	9270.2 (1.74)	9130 (0.2)	9112 (0)	9112 (0)	9112 (0)	9112 (0)	9112
XML100_2234_9	8335.6 (0.37)	8309 (0.05)	8305.8 (0.01)	8305 (0)	8305.8 (0.01)	8305 (0)	8305
XML100_2235_22	6760.2 (1.03)	6742 (0.76)	6691 (0)	6691 (0)	6691 (0)	6691 (0)	6691
XML100_2251_13	33608.2 (3.28)	33507 (2.97)	33236.8 (2.14)	32725 (0.57)	33236.8 (2.14)	32725 (0.57)	32540
XML100_2251_15	26146.8 (2.46)	26096 (2.26)	25728.8 (0.82)	25568 (0.19)	25727.6 (0.82)	25568 (0.19)	25519
XML100_2251_21	23007.6 (2.74)	22861 (2.08)	22411.8 (0.08)	22395 (0)	22411.4 (0.07)	22395 (0)	22395
XML100_2252_5	15190.2 (0.59)	15171 (0.46)	15161 (0.4)	15161 (0.4)	15161 (0.4)	15161 (0.4)	15101
XML100_2316_3	8095 (0.65)	8045 (0.02)	8043 (0)	8043 (0)	8043 (0)	8043 (0)	8043
XML100_2322_16	16544 (1.16)	16529 (1.07)	16419.4 (0.4)	16354 (0)	16418.8 (0.4)	16354 (0)	16354
XML100_2331_25	29033.2 (0.85)	28958 (0.59)	28911 (0.42)	28873 (0.29)	28909.6 (0.42)	28848 (0.2)	28789
XML100_2351_26	33957.6 (2.97)	33913 (2.84)	33074.8 (0.29)	32978 (0)	33033.6 (0.17)	32978 (0)	32978
XML100_3112_2	25942.8 (1.01)	25936 (0.99)	25777.2 (0.37)	25701 (0.07)	25770.8 (0.34)	25701 (0.07)	25683

Table 9: Comparison of solution quality with $T_{max} = 60$ seconds in 5 runs (continued)

Instance	MNS-TS		MS		Guided-MS		Optimal
	Avg (Gap)	Best (Gap)	Avg (Gap)	Best (Gap)	Avg (Gap)	Best (Gap)	
XML100_3113_9	21602.6 (1.03)	21482 (0.46)	21392.4 (0.04)	21383 (0)	21392.4 (0.04)	21383 (0)	21383
XML100_3113_18	22206 (0.91)	22180 (0.79)	22023.4 (0.08)	22006 (0)	22023.4 (0.08)	22006 (0)	22006
XML100_3151_5	39806.4 (1.54)	39762 (1.43)	39397 (0.5)	39331 (0.33)	39383.4 (0.46)	39331 (0.33)	39202
XML100_3151_7	49299.4 (1.6)	49221 (1.44)	48875.2 (0.73)	48784 (0.54)	48886.6 (0.75)	48653 (0.27)	48523
ML100_3151_15	44679.8 (1.07)	44616 (0.93)	44539.6 (0.75)	44487 (0.63)	44545.4 (0.77)	44516 (0.7)	44207
XML100_3151_17	46290.6 (1.49)	46250 (1.41)	45963 (0.78)	45725 (0.25)	45936.8 (0.72)	45725 (0.25)	45609
XML100_3151_21	38630 (2.08)	38630 (2.08)	38334.6 (1.3)	38315 (1.25)	38328.6 (1.28)	38315 (1.25)	37843
XML100_3151_24	58362.8 (2.17)	58115 (1.74)	57507.6 (0.68)	57482 (0.63)	57514 (0.69)	57514 (0.69)	57121
XML100_3154_6	15639.2 (1.4)	15627 (1.32)	15472.2 (0.32)	15453 (0.19)	15472.2 (0.32)	15453 (0.19)	15423
XML100_3154_25	16125.4 (1.23)	16107 (1.12)	16038 (0.68)	15939 (0.06)	16038 (0.68)	15939 (0.06)	15929
XML100_3163_10	21404 (0.69)	21398 (0.66)	21333.4 (0.36)	21257 (0)	21328.4 (0.34)	21271 (0.07)	21257
XML100_3164_20	15148 (2.23)	15148 (2.23)	14918.8 (0.69)	14817 (0)	14908.6 (0.62)	14817 (0)	14817
XML100_3165_17	14145.4 (0.21)	14144 (0.2)	14199 (0.59)	14124 (0.06)	14199 (0.59)	14124 (0.06)	14116
XML100_3231_6	49053.8 (2.84)	49042 (2.82)	47982.6 (0.59)	47917 (0.46)	47944 (0.51)	47917 (0.46)	47699
XML100_3231_9	58118.4 (2.64)	58066 (2.54)	57443 (1.44)	57303 (1.2)	57430.6 (1.42)	57273 (1.14)	56625
XML100_3231_11	64613 (2.4)	64549 (2.29)	64217.6 (1.77)	64135 (1.64)	64214 (1.76)	64092 (1.57)	63101
XML100_3231_12	50992.4 (1.9)	50944 (1.81)	50880.4 (1.68)	50862 (1.64)	50880.4 (1.68)	50862 (1.64)	50040
XML100_3231_17	42865.4 (2.2)	42482 (1.29)	42401 (1.09)	42356 (0.99)	42367.2 (1.01)	42319 (0.9)	41942
XML100_3231_25	57689 (2.05)	57666 (2.01)	57514.6 (1.74)	57463 (1.65)	57518.2 (1.75)	57483 (1.68)	56531
XML100_3251_1	45411.2 (1.6)	45396 (1.56)	45349.8 (1.46)	45287 (1.32)	45338.6 (1.44)	45290 (1.33)	44697
XML100_3251_3	38713.4 (1.39)	38690 (1.33)	38492.8 (0.81)	38472 (0.76)	38498 (0.82)	38498 (0.82)	38183
XML100_3251_4	34705.6 (1.59)	34648 (1.42)	34603 (1.29)	34588 (1.24)	34531.4 (1.08)	34297 (0.39)	34163
XML100_3251_5	49007.4 (3.58)	48950 (3.46)	48164.4 (1.8)	48162 (1.8)	48167.6 (1.81)	48162 (1.8)	47312
XML100_3251_6	46754.2 (2.87)	46700 (2.75)	46029.4 (1.27)	45879 (0.94)	45983.6 (1.17)	45869 (0.92)	45450
XML100_3251_9	42020.2 (2.07)	41996 (2.01)	41725.4 (1.36)	41643 (1.16)	41676.8 (1.24)	41550 (0.93)	41167
XML100_3251_15	63555.6 (4.95)	63495 (4.85)	61839.6 (2.11)	61830 (2.1)	61861.2 (2.15)	61786 (2.03)	60559
XML100_3251_16	58940.4 (3.12)	58856 (2.97)	57658.2 (0.88)	57208 (0.09)	57324.8 (0.29)	57221 (0.11)	57158
XML100_3251_18	37141.6 (3.38)	37080 (3.21)	36430.4 (1.4)	36235 (0.86)	36423.6 (1.38)	36229 (0.84)	35927
XML100_3251_19	57357 (3.45)	57275 (3.3)	56276 (1.5)	56040 (1.08)	56211 (1.39)	56040 (1.08)	55443
XML100_3251_21	46288.6 (3.49)	46233 (3.37)	45960.6 (2.76)	45935 (2.7)	45945.2 (2.73)	45921 (2.67)	44726
XML100_3251_24	55898.6 (2.97)	55725 (2.65)	54438.2 (0.28)	54397 (0.2)	54429.4 (0.26)	54379 (0.17)	54286
XML100_3251_25	40614.8 (2.84)	40542 (2.65)	40044.4 (1.39)	39888 (1)	40028.4 (1.35)	39945 (1.14)	39495
XML100_3252_7	31995.6 (3.54)	31976 (3.48)	31047.8 (0.48)	30996 (0.31)	31047.8 (0.48)	30996 (0.31)	30901
XML100_3252_10	27628.8 (2.03)	27546 (1.73)	27505 (1.58)	27485 (1.5)	27495.6 (1.54)	27485 (1.5)	27078
XML100_3252_26	27748 (2.67)	27720 (2.57)	27639.8 (2.27)	27633 (2.25)	27639.4 (2.27)	27631 (2.24)	27026
XML100_3271_2	39713.4 (5.89)	39680 (5.8)	37792.8 (0.77)	37503 (0)	37741.2 (0.64)	37503 (0)	37503
XML100_3331_5	53502.2 (3.84)	53143 (3.14)	51677.8 (0.29)	51599 (0.14)	51666 (0.27)	51527 (0)	51526
XML100_3351_3	41516.8 (1.71)	41502 (1.68)	41395.2 (1.42)	41214 (0.97)	41379.4 (1.38)	41214 (0.97)	40817
XML100_3351_6	33191.6 (1.91)	33182 (1.88)	32860.6 (0.9)	32805 (0.72)	32851.2 (0.87)	32805 (0.72)	32569
XML100_3351_8	58244.8 (2.88)	58042 (2.52)	57058.4 (0.79)	56973 (0.64)	57031.4 (0.74)	56973 (0.64)	56613
XML100_3351_12	65912.8 (1.38)	65735 (1.1)	65509.8 (0.76)	65157 (0.21)	65425.4 (0.63)	65191 (0.27)	65018
XML100_3351_14	57390.6 (2.39)	57351 (2.32)	56634.2 (1.04)	56495 (0.79)	56520 (0.83)	56393 (0.61)	56053
XML100_3351_16	56980.2 (1.45)	56889 (1.29)	56351.4 (0.33)	56318 (0.27)	56334 (0.3)	56318 (0.27)	56165
XML100_3351_18	45080.6 (2.82)	44941 (2.5)	44500 (1.5)	44428 (1.33)	44464.4 (1.42)	44292 (1.02)	43843
XML100_3351_19	48983 (2.34)	48910 (2.19)	48379.6 (1.08)	48201 (0.71)	48271.2 (0.86)	48069 (0.43)	47861
XML100_3351_23	58812.6 (2.16)	58749 (2.05)	58119.2 (0.95)	57955 (0.67)	58027.8 (0.79)	57838 (0.46)	57571
XML100_3352_11	30698 (2.21)	30698 (2.21)	30339.2 (1.01)	30307 (0.91)	30346.2 (1.04)	30307 (0.91)	30035
Minimum Gap	0.08	0.02	0.00	0.00	0.00	0.00	
Average Gap	2.06	1.87	0.83	0.59	0.78	0.53	
Median Gap	2.05	1.84	0.74	0.35	0.66	0.27	
Maximum Gap	5.89	5.80	3.64	3.61	3.28	2.68	

A.6 Path Relinking Contributions on 50 Sampled $\mathbb{XML}$100 Instances

Table 10: Comparison of solution quality with $T_{max} = 60$ seconds in 5 runs

Instance	MS				Guided-MS				Optimal
	with PR		without PR		with PR		without PR		
	Avg (Gap)	Best (Gap)	Avg (Gap)	Best (Gap)	Avg (Gap)	Best (Gap)	Avg (Gap)	Best (Gap)	
XML100_1113_20	18200 (0)	18200 (0)	18200 (0)	18200 (0)	18200 (0)	18200 (0)	18200 (0)	18200 (0)	18200
XML100_1145_22	11457.4 (0.08)	11457 (0.08)	11459 (0.1)	11457 (0.08)	11457 (0.08)	11457 (0.08)	11457 (0.08)	11457 (0.08)	11448
XML100_1151_3	33123 (0.42)	33058 (0.23)	33209 (0.69)	33072 (0.27)	33070.8 (0.27)	32983 (0)	33194.6 (0.64)	33075 (0.28)	32983
XML100_1215_5	8607 (0)	8607 (0)	8607 (0)	8607 (0)	8607 (0)	8607 (0)	8607 (0)	8607 (0)	8607
XML100_1231_24	42293.8 (2.33)	42275 (2.29)	42342.6 (2.45)	42306 (2.36)	42308.6 (2.37)	42295 (2.33)	42332.4 (2.43)	42275 (2.29)	41330
XML100_1251_4	39738.8 (0.99)	39521 (0.44)	39765.4 (1.06)	39662 (0.8)	39728 (0.96)	39498 (0.38)	39807.4 (1.16)	39796 (1.14)	39349
XML100_1251_7	34677 (0.61)	34553 (0.25)	34708.8 (0.7)	34568 (0.3)	34656.8 (0.55)	34553 (0.25)	34648 (0.53)	34556 (0.26)	34466
XML100_1251_9	33031.8 (0.81)	32897 (0.4)	33067.8 (0.92)	32897 (0.4)	33033.6 (0.81)	32836 (0.21)	33069.6 (0.92)	32922 (0.47)	32767
XML100_1251_10	33678 (0.85)	33561 (0.5)	33724 (0.99)	33656 (0.78)	33670 (0.83)	33561 (0.5)	33713.4 (0.96)	33442 (0.14)	33394
XML100_1251_15	48549.6 (1.27)	48548 (1.27)	48556.2 (1.29)	48509 (1.19)	48503.2 (1.17)	48290 (0.73)	48563.8 (1.3)	48505 (1.18)	47940
XML100_1251_16	36573.2 (1.28)	36275 (0.45)	36573.6 (1.28)	36177 (0.18)	36526 (1.15)	36202 (0.25)	36643.2 (1.47)	36195 (0.23)	36111
XML100_1251_17	39153.2 (0.59)	39147 (0.58)	39158.6 (0.61)	39131 (0.53)	39147.8 (0.58)	39145 (0.57)	39145.4 (0.57)	39131 (0.53)	38923
XML100_1252_15	27939.2 (1.29)	27627 (0.16)	27958.8 (1.36)	27653 (0.25)	27939.2 (1.29)	27627 (0.16)	27966.6 (1.39)	27693 (0.4)	27583
XML100_1261_7	22137.8 (2.17)	22071 (1.86)	22164 (2.29)	22021 (1.63)	22139.2 (2.18)	22092 (1.96)	22150.4 (2.23)	21998 (1.53)	21667
XML100_1313_1	16943 (0.18)	16912 (0)	16949.8 (0.22)	16912 (0)	16943 (0.18)	16912 (0)	16948.4 (0.22)	16912 (0)	16912
XML100_1315_6	10803.6 (0.41)	10760 (0)	10822 (0.58)	10774 (0.13)	10803.6 (0.41)	10760 (0)	10822 (0.58)	10774 (0.13)	10760
XML100_1322_7	20684.4 (0.03)	20679 (0)	20807 (0.62)	20686 (0.03)	20679 (0)	20679 (0)	20788 (0.53)	20687 (0.04)	20679
XML100_1351_2	34640 (0.1)	34605 (0)	34666.8 (0.18)	34605 (0)	34640 (0.1)	34605 (0)	34659.2 (0.16)	34605 (0)	34605
XML100_2112_1	17730 (0)	17730 (0)	17730 (0)	17730 (0)	17730 (0)	17730 (0)	17730 (0)	17730 (0)	17730
XML100_2143_25	12590 (0)	12590 (0)	12590 (0)	12590 (0)	12590 (0)	12590 (0)	12590 (0)	12590 (0)	12590
XML100_2231_20	26486.6 (0.82)	26476 (0.78)	26531 (0.99)	26486 (0.82)	26475.2 (0.78)	26435 (0.62)	26525.2 (0.97)	26486 (0.82)	26271
XML100_2231_22	23246.4 (0.18)	23215 (0.04)	23277 (0.31)	23205 (0)	23247 (0.18)	23215 (0.04)	23259.8 (0.24)	23215 (0.04)	23205
XML100_2233_14	9112 (0)	9112 (0)	9112 (0)	9112 (0)	9112 (0)	9112 (0)	9112 (0)	9112 (0)	9112
XML100_2235_22	6691 (0)	6691 (0)	6691 (0)	6691 (0)	6691 (0)	6691 (0)	6691 (0)	6691 (0)	6691
XML100_2251_15	25728.8 (0.82)	25568 (0.19)	25836 (1.24)	25777 (1.01)	25727.6 (0.82)	25568 (0.19)	25780.2 (1.02)	25658 (0.54)	25519
XML100_2251_21	22411.8 (0.08)	22395 (0)	22441.2 (0.21)	22415 (0.09)	22411.4 (0.07)	22395 (0)	22415.2 (0.09)	22415 (0.09)	22395
XML100_2316_3	8043 (0)	8043 (0)	8055.6 (0.16)	8043 (0)	8043 (0)	8043 (0)	8055.6 (0.16)	8043 (0)	8043
XML100_3151_5	39397 (0.5)	39331 (0.33)	39442.4 (0.61)	39392 (0.48)	39383.4 (0.46)	39331 (0.33)	39453.8 (0.64)	39379 (0.45)	39202
XML100_3151_7	48875.2 (0.73)	48784 (0.54)	48914.2 (0.81)	48778 (0.53)	48886.6 (0.75)	48653 (0.27)	48864.6 (0.7)	48777 (0.52)	48523
XML100_3151_17	45963 (0.78)	45725 (0.25)	45968.8 (0.79)	45889 (0.61)	45936.8 (0.72)	45725 (0.25)	45978.4 (0.81)	45926 (0.7)	45609
XML100_3151_21	38334.6 (1.3)	38315 (1.25)	38374.6 (1.4)	38327 (1.28)	38328.6 (1.28)	38315 (1.25)	38347.4 (1.33)	38304 (1.22)	37843
XML100_3154_25	16038 (0.68)	15939 (0.06)	16089.2 (1.01)	16047 (0.74)	16038 (0.68)	15939 (0.06)	16089.2 (1.01)	16047 (0.74)	15929
XML100_3163_10	21333.4 (0.36)	21257 (0)	21335 (0.37)	21273 (0.08)	21328.4 (0.34)	21271 (0.07)	21335 (0.37)	21273 (0.08)	21257
XML100_3164_20	14918.8 (0.69)	14817 (0)	14950.2 (0.9)	14907 (0.61)	14908.6 (0.62)	14817 (0)	14938.2 (0.82)	14907 (0.61)	14817
XML100_3165_17	14199 (0.59)	14124 (0.06)	14227.4 (0.79)	14145 (0.21)	14199 (0.59)	14124 (0.06)	14226 (0.78)	14138 (0.16)	14116
XML100_3231_6	47982.6 (0.59)	47917 (0.46)	48080.4 (0.8)	47934 (0.49)	47944 (0.51)	47917 (0.46)	48002.6 (0.64)	47924 (0.47)	47699
XML100_3231_11	64217.6 (1.77)	64135 (1.64)	64259.2 (1.84)	64141 (1.65)	64214 (1.76)	64092 (1.57)	64209.4 (1.76)	64135 (1.64)	63101
XML100_3231_17	42401 (1.09)	42356 (0.99)	42443.8 (1.2)	42356 (0.99)	42367.2 (1.01)	42319 (0.9)	42395.4 (1.08)	42356 (0.99)	41942
XML100_3251_3	38492.8 (0.81)	38472 (0.76)	38497.2 (0.82)	38472 (0.76)	38498 (0.82)	38498 (0.82)	38507 (0.85)	38498 (0.82)	38183
XML100_3251_4	34603 (1.29)	34588 (1.24)	34607.2 (1.3)	34582 (1.23)	34531.4 (1.08)	34297 (0.39)	34588.8 (1.25)	34574 (1.2)	34163
XML100_3251_16	57658.2 (0.88)	57208 (0.09)	57717.4 (0.98)	57328 (0.3)	57324.8 (0.29)	57221 (0.11)	57754.8 (1.04)	57315 (0.27)	57158
XML100_3251_24	54438.2 (0.28)	54397 (0.2)	54450.4 (0.3)	54411 (0.23)	54429.4 (0.26)	54379 (0.17)	54738.4 (0.83)	54382 (0.18)	54286
XML100_3251_25	40044.4 (1.39)	39888 (1)	40064.2 (1.44)	39991 (1.26)	40028.4 (1.35)	39945 (1.14)	40044.6 (1.39)	39942 (1.13)	39495
XML100_3252_26	27639.8 (2.27)	27633 (2.25)	27646.4 (2.3)	27633 (2.25)	27639.4 (2.27)	27631 (2.24)	27650.4 (2.31)	27633 (2.25)	27026
XML100_3271_2	37792.8 (0.77)	37503 (0)	37921 (1.11)	37503 (0)	37741.2 (0.64)	37503 (0)	37762.4 (0.69)	37503 (0)	37503
XML100_3351_3	41395.2 (1.42)	41214 (0.97)	41440.4 (1.53)	41381 (1.38)	41379.4 (1.38)	41214 (0.97)	41415 (1.47)	41381 (1.38)	40817
XML100_3351_6	32860.6 (0.9)	32805 (0.72)	32878.4 (0.95)	32803 (0.72)	32851.2 (0.87)	32805 (0.72)	32887.2 (0.98)	32847 (0.85)	32569
XML100_3351_8	57058.4 (0.79)	56973 (0.64)	57115.4 (0.89)	56973 (0.64)	57031.4 (0.74)	56973 (0.64)	57076.8 (0.82)	56973 (0.64)	56613
XML100_3351_14	56634.2 (1.04)	56495 (0.79)	56637.4 (1.04)	56532 (0.85)	56520 (0.83)	56393 (0.61)	56569.6 (0.92)	56512 (0.82)	56053
XML100_3351_23	58119.2 (0.95)	57955 (0.67)	58156.8 (1.02)	58063 (0.85)	58027.8 (0.79)	57838 (0.46)	58112.8 (0.94)	57945 (0.65)	57571
Minimum Gap	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
Average Gap	0.74	0.49	0.85	0.58	0.70	0.44	0.82	0.56	
Median Gap	0.75	0.25	0.86	0.49	0.66	0.23	0.82	0.46	
Maximum Gap	2.33	2.29	2.45	2.36	2.37	2.33	2.43	2.29	

References

[1] Luca Accorsi, Andrea Lodi, and Daniele Vigo. "Guidelines for the computational testing of machine learning approaches to vehicle routing problems". In: *Operations Research Letters* 50.2 (2022), pp. 229–234.

[2] Luca Accorsi and Daniele Vigo. "A fast and scalable heuristic for the solution of large-scale capacitated vehicle routing problems". In: *Transportation Science* 55.4 (2021), pp. 832–856.

[3] Florian Arnold, Michel Gendreau, and Kenneth Sörensen. "Efficiently solving very large-scale routing problems". In: *Computers & Operations Research* 107 (2019), pp. 32–42. ISSN: 0305-0548.

[4] Florian Arnold and Kenneth Sörensen. "Knowledge-guided local search for the vehicle routing problem". In: *Computers & Operations Research* 105 (2019), pp. 32–46.

[5] Florian Arnold and Kenneth Sörensen. "What makes a VRP solution good? The generation of problem-specific knowledge for heuristics". In: *Computers & Operations Research* 106 (2019), pp. 280–288.

[6] Alejandro Barredo Arrieta et al. "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI". In: *Information fusion* 58 (2020), pp. 82–115.

[7] Marcia L Baptista, Kai Goebel, and Elsa MP Henriques. "Relation between prognostics predictor evaluation metrics and local interpretability SHAP values". In: *Artificial Intelligence* 306 (2022), p. 103667.

[8] Richard Bellman. "On a routing problem". In: *Quarterly of applied mathematics* 16.1 (1958), pp. 87–90.

[9] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. "Machine learning for combinatorial optimization: a methodological tour d'horizon". In: *European Journal of Operational Research* 290.2 (2021), pp. 405–421.

[10] Leo Breiman. "Random forests". In: *Machine Learning* 45.1 (2001), 5–32. ISSN: 0885-6125.

[11] Leo Breiman et al. *Classification And Regression Trees*. Routledge, 2017. ISBN: 9781315139470.

[12] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794.

[13] Geoff Clarke and John W Wright. "Scheduling of vehicles from a central depot to a number of delivery points". In: *Operations research* 12.4 (1964), pp. 568–581.

[14] Thomas Cover and Peter Hart. "Nearest neighbor pattern classification". In: *IEEE transactions on information theory* 13.1 (1967), pp. 21–27.

[15] George Dantzig, Ray Fulkerson, and Selmer Johnson. "Solution of a large-scale traveling-salesman problem". In: *Journal of the operations research society of America* 2.4 (1954), pp. 393–410.

[16] Janez Demšar. "Statistical Comparisons of Classifiers over Multiple Data Sets". In: *Journal of Machine Learning Research* 7.1 (2006), pp. 1–30.

[17] Evelyn Fix. *Discriminatory analysis: nonparametric discrimination, consistency properties*. Vol. 1. USAF school of Aviation Medicine, 1985.

[18] Jerome H Friedman. "Greedy function approximation: a gradient boosting machine". In: *Annals of statistics* (2001), pp. 1189–1232.

[19] F. Glover, M. Laguna, and R. Marti. "Fundamentals of scatter search and path relinking". English. In: *Control and Cybernetics* Vol. 29, no 3 (2000), pp. 653–684.

[20] Fred Glover. "Ejection chains, reference structures and alternating path methods for traveling salesman problems". In: *Discrete Applied Mathematics* 65.1-3 (1996), pp. 223–253.

[21] Fred Glover. "Tabu search and adaptive memory programming—advances, applications and challenges". In: *Interfaces in Computer Science and Operations Research: Advances in Metaheuristics, Optimization, and Stochastic Modeling Technologies* (1997), pp. 1–75.

[22] Riccardo Guidotti et al. "A survey of methods for explaining black box models". In: *ACM computing surveys (CSUR)* 51.5 (2018), pp. 1–42.

[23] Trevor Hastie et al. *The elements of statistical learning: data mining, inference, and prediction.* Vol. 2. Springer, 2009.

[24] Keld Helsgaun. "An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems". In: *Roskilde: Roskilde University* 12 (2017).

[25] Sin C Ho and Michel Gendreau. "Path relinking for the vehicle routing problem". In: *Journal of heuristics* 12 (2006), pp. 55–72.

[26] Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi. "Chapter 4 The traveling salesman problem". In: *Network Models*. Vol. 7. Handbooks in Operations Research and Management Science. Elsevier, 1995, pp. 225–330.

[27] Guolin Ke et al. "Lightgbm: A highly efficient gradient boosting decision tree". In: *Advances in neural information processing systems* 30 (2017).

[28] Manuel Laguna, Rafael Martí, and Vicente Campos. "Intensification and diversification with elite tabu search solutions for the linear ordering problem". In: *Computers & Operations Research* 26.12 (1999), pp. 1217–1230.

[29] Gilbert Laporte. "Fifty years of vehicle routing". In: *Transportation science* 43.4 (2009), pp. 408–416.

[30] Sirui Li, Zhongxia Yan, and Cathy Wu. "Learning to delegate for large-scale vehicle routing". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 26198–26211.

[31] Flavien Lucas, Romain Billot, and Marc Sevaux. "A comment on "what makes a VRP solution good? The generation of problem-specific knowledge for heuristics"". In: *Computers & Operations Research* 110 (2019), pp. 130–134.

[32] Flavien Lucas et al. "Reducing space search in combinatorial optimization using machine learning tools". In: *Learning and Intelligent Optimization: 14th International Conference, LION 14, Athens, Greece, May 24–28, 2020, Revised Selected Papers 14*. Springer. 2020, pp. 143–150.

[33] Scott M Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *Advances in neural information processing systems* 30 (2017).

[34] Scott M Lundberg et al. "From local explanations to global understanding with explainable AI for trees". In: *Nature machine intelligence* 2.1 (2020), pp. 56–67.

[35] Rafael Martí, Mauricio GC Resende, and Celso C Ribeiro. "Multi-start methods for combinatorial optimization". In: *European Journal of Operational Research* 226.1 (2013), pp. 1–8.

[36] Christian Prins. "A simple and effective evolutionary algorithm for the vehicle routing problem". In: *Computers & operations research* 31.12 (2004), pp. 1985–2002.

[37] Caroline Prodhon and Christian Prins. "Metaheuristics for Vehicle Routing Problems". In: *Metaheuristics*. Cham: Springer International Publishing, 2016, pp. 407–437.

[38] Eduardo Queiroga et al. "10,000 optimal CVRP solutions for testing machine learning based heuristics". In: *AAAI-22 Workshop on Machine Learning for Operations Research (ML4OR)* (2021).

[39] César Rego. "Node-ejection chains for the vehicle routing problem: Sequential and parallel algorithms". In: *Parallel Computing* 27.3 (2001), pp. 201–222.

[40] Mauricio G.C. Resende and Celso C. Ribeiro. "GRASP with Path-Relinking: Recent Advances and Applications". In: *Metaheuristics: Progress as Real Problem Solvers*. Boston, MA: Springer US, 2005, pp. 29–63.

[41] David Simchi-Levi, Philip Kaminsky, and Edith Simchi-Levi. *Designing and managing the supply chain: Concepts, strategies, and case studies*. Irwin/McGraw-Hill series in operations and decision sciences. McGraw-Hill/Irwin, 2002. ISBN: 9780072845532.

[42] Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. "Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation". In: *Australasian joint conference on artificial intelligence*. Springer. 2006, pp. 1015–1021.

[43] Kenneth Sörensen and Marc Sevaux. "MA| PM: memetic algorithms with population management". In: *Computers & Operations Research* 33.5 (2006), pp. 1214–1225.

[44] María Soto et al. "Multiple neighborhood search, tabu search and ejection chains for the multi-depot open vehicle routing problem". In: *Computers & Industrial Engineering* 107 (2017), pp. 211–222.

[45] Kenneth Sörensen and Patrick Schittekat. "Statistical analysis of distance-based path relinking for the capacitated vehicle routing problem". In: *Computers & Operations Research* 40.12 (2013), pp. 3197–3205. ISSN: 0305-0548.

[46] Éric Taillard et al. "A tabu search heuristic for the vehicle routing problem with soft time windows". In: *Transportation science* 31.2 (1997), pp. 170–186.

[47] Paolo Toth and Daniele Vigo. "The granular tabu search and its application to the vehicle-routing problem". In: *Informs Journal on computing* 15.4 (2003), pp. 333–346.

[48] Paolo Toth and Daniele Vigo. *Vehicle routing: problems, methods, and applications*. SIAM, 2014.

[49] Eduardo Uchoa et al. "New benchmark instances for the Capacitated Vehicle Routing Problem". In: *European Journal of Operational Research* 257.3 (2017), pp. 845–858. ISSN: 0377-2217.

[50] Thibaut Vidal. "Hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood". In: *Computers & Operations Research* 140 (2022), p. 105643.

[51] Thibaut Vidal et al. "A unified solution framework for multi-attribute vehicle routing problems". In: *European Journal of Operational Research* 234.3 (2014), pp. 658–673. ISSN: 0377-2217.

[52] Liang Xin et al. "Neurolkh: Combining deep learning model with lin-kernighan-helsgaun heuristic for solving the traveling salesman problem". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 7472–7483.