

Article

Addressing the Algorithm Selection Problem through an Attention-Based Meta-Learner Approach

Enrique Díaz de León-Hicks , Santiago Enrique Conant-Pablos * , José Carlos Ortiz-Bayliss 
and Hugo Terashima-Marín 

Tecnologico de Monterrey, School of Engineering and Sciences, Monterrey 64849, Mexico;
enriquedlh97@hotmail.com (E.D.d.L.-H.); jcobayliss@tec.mx (J.C.O.-B.); terashima@tec.mx (H.T.-M.)

* Correspondence: sconant@tec.mx

Abstract: In the algorithm selection problem, where the task is to identify the most suitable solving technique for a particular situation, most methods used as performance mapping mechanisms have been relatively simple models such as logistic regression or neural networks. In the latter case, most implementations tend to have a shallow and straightforward architecture and, thus, exhibit a limited ability to extract relevant patterns. This research explores the use of attention-based neural networks as meta-learners to improve the performance mapping mechanism in the algorithm selection problem and fully take advantage of the model's capabilities for pattern extraction. We compare the proposed use of an attention-based meta-learner method as a performance mapping mechanism against five models from the literature: multi-layer perceptron, k -nearest neighbors, softmax regression, support vector machines, and decision trees. We used a meta-data dataset obtained by solving the vehicle routing problem with time window (VRPTW) instances contained in the Solomon benchmark with three different configurations of the simulated annealing meta-heuristic for testing purposes. Overall, the attention-based meta-learner model yields better results when compared to the other benchmark methods in consistently selecting the algorithm that best solves a given VRPTW instance. Moreover, by significantly outperforming the multi-layer perceptron, our findings suggest promising potential in exploring more recent and novel advancements in neural network architectures.



Citation: Díaz de León-Hicks, E.; Conant-Pablos, S.E.; Ortiz-Bayliss, J.C.; Terashima-Marín, H. Addressing the Algorithm Selection Problem through an Attention-Based Meta-Learner Approach. *Appl. Sci.* **2023**, *13*, 4601. <https://doi.org/10.3390/app13074601>

Academic Editor: DaeEun Kim

Received: 8 March 2023

Revised: 31 March 2023

Accepted: 3 April 2023

Published: 5 April 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: offline algorithm selection; meta-learning; attention mechanism; neural network; combinatorial optimization; vehicle routing problem with time windows; simulated annealing; machine learning

1. Introduction

Given the nature of NP-hard combinatorial optimization problems, the search space grows exponentially with the number of inputs. Thus, it is unfeasible to employ exact methods, except for in small cases [1]. Instead, approximation techniques, such as heuristics, produce solutions for larger instances (although they cannot guarantee the optimality of such solutions) [2,3]. Moreover, we now understand that no single algorithm will yield the best performance in all instances of a problem [4]. However, an “intelligent” technique may benefit from considering the structure of the problem at hand [5]. Consequently, a new problem arises: selecting the most appropriate algorithm for a particular problem instance, a task formally known as the algorithm selection problem (ASP) [6,7].

A recent approach to this problem involves machine learning to identify the patterns between problem instances and algorithms by leveraging the meta-features that characterize such problem instances [6,8]. Thus, we must face the challenge of deciding the performance model and the type of machine learning technique to use.

This work investigates attention-based neural networks as an alternative performance mapping mechanism within the algorithm selection framework. The architecture used is

similar to the encoder from the transformer by Vaswani et al. [9], a neural network architecture that leverages the attention mechanism for modeling long-term dependencies in data. This work aims to prove that attention-based neural networks can select the best algorithm for a given problem instance offline. Then, we can state that these neural networks constitute an effective alternative to learning performance mappings from problem instances to algorithm space. The rationale behind our proposal is to leverage the attention mechanism, specifically using multiple attention heads to differentially weigh the importance of the features that characterize the problem instances [9]. We decided to use multi-head attention instead of a single attention head so that multiple positions can be jointly attended [10] and leverage the specialization among attention heads that the approach encourages [11]. Moreover, this investigation is conducted in the context of the vehicle routing problem with time windows (VRPTW) [1], which aims to find a collection of routes covering a set of customers, all of them starting and ending at a depot and within a specific time interval referred to as the time window [1].

Additionally, it is relevant to note that this work focuses not on the application problem, i.e., the VRPTW, but on showing how attention-based neural networks are competent algorithm selectors. Consequently, we do not aim to beat the state-of-the-art solvers for the VRPTW. Instead, our objective is to show that the proposed attention-based model, when compared against other common approaches found in the literature, consistently chooses the best algorithm to solve a particular problem instance which, in this case, happens to be a VRPTW instance. Nevertheless, we assume these results should be generalizable to other problem domains.

Although recent advances in neural network architectures and techniques have led to significant performance improvements in many domains [12–16], the literature related to using neural networks to select meta-heuristics for solving combinatorial optimization problems has mainly relied on simple multi-layer perceptrons (MLPs) [17–20], with many studies using the one hidden layer architecture implemented in the WEKA data mining software [21]. With this in mind, we define our main contribution as the proposal of an attention-based neural network as a meta-learner for selecting meta-heuristics (in this particular case, tested on three different SA configurations) for solving the VRPTW. This contribution exhibits a twofold impact, as described below:

- We show that leveraging novel architectures, such as the attention-based neural network proposed in this study, can lead to good performance compared to other commonly used techniques, including MLPs.
- We introduce the attention-based meta-learner as a new baseline model to complement the traditional meta-learners in the literature.

Our first impact stems from our experimental evaluation, comparing our proposed meta-learner against the most commonly used meta-learners in the recent literature. In this sense, our results suggest that employing more sophisticated neural network architectures can be a promising alternative to traditional meta-learners and result in a good performance. The second impact of our contribution arises from proposing our attention-based meta-learner as a new baseline for future studies, thus providing the research community with a more comprehensive evaluation framework that better reflects the current advances in neural networks.

The remainder of this work is organized as follows. Section 2 provides the necessary background and fundamentals of this research and a general presentation of relevant related work. In Section 3, we describe the attention-based meta-learner and the dataset employed in this work. Section 4 deals with the experimentation conducted, the results obtained, and their discussion. Finally, Section 5 presents general conclusions and suggested directions for future work.

2. Background and Related Work

The ASP was first introduced as a framework for choosing the best algorithm from a set of candidates that would yield the best performance for a particular problem in-

stance [22]. It was originally formulated as learning the selection mapping $S(f(x))$ from problem instances $x \in \mathcal{P}$, characterized by features $f(x) \in \mathcal{F}$, into algorithm space $\mathcal{A}$ such that, when solving the problem instance $x \in \mathcal{P}$ with the selected algorithm $a \in \mathcal{A}$, the performance metric $y(a, f(x)) \in \mathcal{Y}$ is maximized [6,7]. A more modern outlook approaches the problem from a meta-learning perspective and focuses on learning the mapping $S(f(x))$ through machine learning. In this case, the data that describe the four elements $\{\mathcal{P}, \mathcal{A}, \mathcal{Y}, \mathcal{F}\}$ are regarded as the meta-data [6,7]. In such an approach, characterizing problem instances using relevant features plays an essential role in identifying the algorithm or configuration that yields the best performance for a specific problem instance. The set of features $\mathcal{F}$ can include various types of information, such as the size of the problem instance, the type of data, or any other relevant characteristic. A critical aspect of the algorithm selection problem is determining the best features that accurately capture the relevant characteristics of the problem instance and lead to the selection of the most suitable algorithm. Identifying the optimal set of features that adequately characterizes a given problem instance is challenging. Moreover, it requires domain knowledge and experimentation to determine which features can be used to predict the performance of a particular algorithm or class of algorithms [6].

Multiple methods, both online and offline [23], have been proposed for computing the best problem-algorithm mapping, with the direction of research favoring the latter. Among offline methods, the ones following the meta-learning approach have gained attention and shown that using meta-learners, in combination with common meta-heuristics such as genetic algorithms (GAs) [17] or simulated annealing (SA) [24], can yield good results.

Additionally, the literature that focuses explicitly on using supervised learning to select meta-heuristics for solving combinatorial optimization problems shows that the most common approaches employ mainly straightforward implementations of machine learning techniques. Among these implementations, we can mention neural networks [17,18], linear and logistic regression [24], k -nearest neighbor (k -NN) [17], support vector machines (SVM) [18,25], and decision trees (DT) [18,26,27]. Karimi-Mamaghan et al. [23] and Cohen-Shapira et al. [28] show that this applies to the most relevant and recent papers focusing on researching the ASP for selecting meta-heuristics to solve combinatorial optimization problems. Thus, works that fail to include one of these three main components (machine learning, meta-heuristics, or combinatorial optimization problems), as identified by Karimi-Mamaghan et al. [23], are considered to not be within the scope of this work and, for that reason, excluded.

Furthermore, meta-learners based on neural networks have proven to be adequate performance mapping mechanisms [17], even when most of the research has been mainly limited to simple architectures. For example, Smith-Miles and Lopes proposed a meta-learning framework for analyzing the relationships between quadratic assignment problem characteristics and meta-heuristic performance, in which a multi-layer perceptron (MLP) was used as meta-learner [19]. Similarly, Kanda et al. used an MLP as a meta-learner for selecting the best meta-heuristic for the traveling salesman problem (TSP) [20]. This raises the question of whether exploring more recent neural network architectures can improve the pattern-extraction process to further leverage the meta-features that characterize the problem instances and thus lead to better learning of the mappings of problem instances to algorithms.

Moreover, Huisman et al. [29] classified neural network meta-learners into three categories: metric-, model-, and optimization-based techniques. Metric-based techniques focus on learning a feature space based on the network's weights, which is then used for new tasks by comparing new inputs with example inputs in the learned space. This learning approach, in which the learned feature space stored in the network's parameters is not updated when new tasks are seen, is the focus of our proposed attention-based meta-learner. Alternatively, model-based meta-learning relies on an adaptive internal state that sequentially processes meta-data to capture task-specific information [29]. Besides, Huisman et al. [29] defined optimization-based meta-learning as an approach that optimize

for fast learning by treating meta-learning as a bi-level optimization problem, with a base-learner making task-specific updates at the inner level and performance across tasks being optimized at the outer level.

In this work, we focus on the VRPTW as an application problem. It is an extension of the VRP in which servicing a particular customer i , out of C customers, can only be carried out during a time interval $[t_{max}, t_{min}]$ corresponding to the customer's time window and defined by a starting time t_{max} and ending time t_{min} [1,30]. The problem, essentially, consists of identifying k routes for the vehicles. Such routes together and subject to additional constraints must cover the set of customers where, for each customer i , there is a given demand m_i . Furthermore, no vehicle can supply more customers beyond its capacity q_k , and each customer must be visited exactly once by a single vehicle [1,30]. We do not provide further details on the VRPTW since it is only the application problem and not the focus of this work. However, we invite the interested reader to consult the works of Kallehauge et al. [30] and Tan et al. [1], which provide a more detailed description of this problem.

Moreover, the modern approach to the ASP from the meta-learning perspective has already shown that simple meta-learners, such as MLPs, can yield good results when used to select the best meta-heuristic for solving the VRPTW [31]. This work, therefore, focuses on the meta-heuristic approach, given that using these types of methods (e.g., SA [32,33] and GAs [34,35]) has proven to be an effective strategy to consistently obtain adequate solutions when solving routing problems. As an introductory study, we built a portfolio of algorithms consisting of three different configurations of the simulated annealing (SA) meta-heuristic to test attention-based neural networks as algorithm selectors. From this point on, and for clarity, we will treat each of these configurations as a different algorithm. These algorithms generate meta-data by solving various VRPTW instances. Subsequently, we use the generated meta-data to evaluate the effectiveness of an attention-based meta-learner as an alternative performance mapping mechanism by selecting the algorithm out of the three that best solves a specific VRPTW instance.

However, as mentioned before, this research's focus differs from the VRPTW itself; thus, we do not aim to improve on the current state-of-the-art solutions. Alternatively, this work aims to study the use of an attention-based meta-learner in the context of the ASP for selecting the best algorithm for a specific problem instance. We chose the VRPTW to evaluate the proposed meta-learner and compare it against other common meta-learners found in the most relevant and recent literature. All the meta-learners considered for comparison share the three main components considered for this work: machine learning, meta-heuristics, and combinatorial optimization problems (which in this work happens to be the VRPTW). Moreover, we opted for such meta-learners because the literature consistently presents them as baseline methods for this problem.

As attention mechanisms have become more relevant, they have been used in tasks such as text classification [36] and image-based analysis [37]. However, although these mechanisms have been implemented differently, they all share the same principles, based on which Niu et al. [10] identified four main categories. The first category concerns the softness of attention, which includes soft and hard attention [10]. Soft attention is deterministic and differentiable and computes a weighted average of all keys to build the context vector, while hard attention samples keys to do this [10]. This category also includes global attention, similar to soft attention, and local attention, a blend of hard and soft attention that considers only a subset of the input at a time. The second category is related to the form of the input feature. It is divided into item-wise, which processes explicit items or a sequence of items, or location-wise, which processes a feature map or an image at each decoder step [10]. The third category concerns the input representation, which includes distinctive, multi-input, self-, and hierarchical attention [10]. The fourth and final category refers to how the output is represented, such as a probability distribution, sequence of tokens, or fixed-length vector [10]. Moreover, Niu et al. [10] also identified seven types of score functions used

within the attention mechanisms to determine how to compute the relevance or similarity between the keys and queries to generate the attention weights.

The proposed attention-based meta-learner has a similar architecture to the encoder block used in transformers [9], which employs a scaled multiplicative score function within the attention mechanism [10]. Transformers represent a network architecture for processing sequential data without necessarily processing it in order. Instead, transformers leverage self-attention to look at the whole sequence and give more importance to the most relevant input data segments. This network architecture was initially developed for natural language processing (NLP) [9]. Nevertheless, it has also been proven helpful in other domains, including combinatorial optimization and, specifically, for solving VRPs using modified transformer architectures to learn heuristics and construct solutions incrementally [38,39].

Nonetheless, as opposed to Kool et al. [38] and Peng et al. [39], this work will not use the proposed attention-based meta-learner to solve specific VRPTW instances directly. Instead, it will select a specific algorithm that will, on average, generate the best solution for a given problem instance. Therefore, to the best of the authors' knowledge, no previous work has proposed using an attention-based neural network as a meta-learner for selecting, from a given set of meta-heuristics, the one that best solves a particular VRPTW instance, as proposed in this work.

3. Solution Model

This section describes the attention-based meta-learner architecture employed in this work. We also describe the dataset considered and briefly explain how we generated it.

3.1. Multi-Head Attention for Problem-Algorithm Mapping

This work relies on the multi-head attention mechanism in the transformer [9], explicitly using it within an attention-based neural network as a meta-learner. The rationale behind using multi-head attention lies, in part, in the type of features that can be computed to represent the structure of a VRPTW instance and how this representation can be transformed to encode it better. We do this by following an approach similar to that of Misir et al. [40] and perform an intermediate feature-to-feature mapping step, with the main difference being that we use multi-head attention in an attention block to do this. This step aims to generate a new and improved representation for a given VRPTW instance. We express this as a three-step process where an input vector x_i , containing d_x features that characterize a given VRPTW instance, is incrementally transformed into a new representation $h_i^{(\ell)}$, where $\ell \in 1, \dots, n$, and n corresponds to the number of attention blocks used. The number of attention blocks determines the number of times the intermediate feature-to-feature mapping step is performed so that mapping from a particular problem instance to a point in the algorithm space can be learned.

As shown by Rasku et al. [41], multiple features can be computed to represent different aspects of a particular problem instance. For example, we can mention generic ones, such as those related to the graph topology, as well as specific ones related to the location of the depot, clients and the demand, among others. With the proposed three-step process, we aim to provide an approach to better extract the patterns encoded by these features. Additionally, we strive to avoid saturating a single attention mechanism due to using multiple types of features (geometrical or topological, for example) by employing multiple heads and thus splitting the cognitive load. Moreover, the decision to use multiple attention heads instead of a single one was mainly taken due to the added capability enabling joint attention to information from different representation subspaces at different positions from the input vector $x_i \in \mathbb{R}^{d_x}$ [10]. The motivation for this was the variety of features used to characterize the VRPTW instances. The goal was to leverage the approach to facilitate specialization behavior among the attention heads, wherein certain heads develop specialized functions within the model [11].

The three-step process is described as follows:

- Step I.** Initially, for the first step in the input layer, a new representation of d_h dimensions of the input vector x_i is learned via a linear transformation $h_i^{(0)} = Wx_i + b$, where the weight matrix $W \in \mathbb{R}^{d_x \times d_h}$ and the bias $b \in \mathbb{R}^{d_h}$ correspond to learnable parameters of the network [38]. In this work, d_x and d_h are set to 26 and 196, respectively. The value of d_x was defined empirically and corresponds to the 26 features described in Section 3.3 used to characterize each VRPTW instance. In contrast, d_h was defined via Bayesian optimization as the optimal value for this hyperparameter, as described in Section 4. The idea of using a linear transformation to generate a new initial representation of the input vector x_i was adopted since it has been previously used in attention-based networks with promising results [9,38,39]. By improving the representation of the input data, this approach can enhance the effectiveness of the updates made by the attention block in the next step.
- Step II.** In the second step, which corresponds to the intermediate feature-to-feature mapping process [40], this new and improved representation $h_i^{(0)}$ —which characterizes the given VRPTW instance—is then fed to the attention block. The objective here is to generate, once again, an improved representation of the $h_i^{(0)}$ vector such that, by leveraging the attention mechanism, it is weighted to better encode the features characterizing the given problem instance in regards to the algorithm space mapping.

As mentioned, the attention block is similar to the encoder block used in the transformer [9]. However, it differs from the original model in that, since there is no use in capturing information from the input order, no positional encodings are added [9]. In addition to that, the attention block follows the original encoder block implementation [9], consisting of a multi-head attention layer (MLA) and a dense layer (FF), each with a corresponding skip-connection and batch normalization (BN) [38,39]. This work uses $M = 8$ attention heads as the optimal value found when optimizing the model hyperparameters via Bayesian optimization, as described in Section 4.

This work uses only one block; however, as for the original transformer architecture, n attention blocks can be defined sequentially so that $h_i^{(\ell)}$ represents the output obtained after a particular block ℓ (where $\ell \in 1, \dots, n$). Nevertheless, since we only use one block, the initial linear projection $h_i^{(0)}$ is equivalent to the general input of an attention block $h_i^{(\ell-1)}$, as defined by Kool et al. [38]. Hence, following their notation, where $\hat{h}_i$ and $h_i^{(1)}$ are the outputs from the MHA and FF sub-layers, respectively, both sub-layers can be simplified and expressed as in Equations (1) and (2).

$$\hat{h}_i = \text{BN}(h_i^{(0)} + \text{MHA}_i(h_i^{(0)})) \quad (1)$$

$$h_i^{(1)} = \text{BN}(\hat{h}_i + \text{FF}(\hat{h}_i)) \quad (2)$$

- Step III.** Finally, in the third step, the output layer takes the resulting $h_i^{(1)}$ as input representation and generates the final probability vector via log softmax after applying global average pooling.

3.2. Portfolio of Algorithms

The portfolio of algorithms used in this work consists of three different configurations of the simulated annealing (SA) meta-heuristic, as stated before. Simulated annealing (SA) is a well-known meta-heuristic with many practical applications. It was initially proposed by Kirkpatrick et al. [42] in the 1980s. The main idea behind SA is to allow the search to explore low-quality solutions at the initial stages of the search as a means to escape from local optima and, with time, keep only the solutions that improve the current solution. We

achieve this behavior by starting the search with an initial (and high) temperature T_{max} that decreases throughout the search until it reaches the final (and low) temperature T_{min} . When the temperature is high (at the beginning of the search process), the probability of accepting low-quality solutions is also high. However, the temperature and the probability of accepting low-quality solutions decrease progressively following the cooling schedule, converging into a simple iterative improvement algorithm.

Moreover, this work uses three different configurations of SA—each differentiated from the other by their neighborhood operator—to conform the algorithm space from which the proposed attention-based meta-learner will choose the specific configuration that best solves each of the VRPTW instances. The pseudocode of the SA algorithm implemented for this work is depicted in Algorithm 1.

Algorithm 1 Simulated Annealing Algorithm.

```

1:  $\alpha \leftarrow$  a constant cooling rate
2:  $k \leftarrow$  iterations at a given temperature
3:  $S \leftarrow$  an initial feasible solution
4:  $E_S \leftarrow$  the cost of the initial solution  $S$ 
5:  $T_{max} \leftarrow$  initial temperature
6:  $T_{min} \leftarrow$  final temperature
7:  $T \leftarrow T_{max}$ 
8: while  $T \geq T_{min}$  do
9:   for  $iteration = 1, \dots, k$  do
10:     $S' \leftarrow Neighbor(S)$ 
11:     $E_{S'} \leftarrow$  the cost of the neighbor solution  $S'$ 
12:     $\Delta E \leftarrow E_{S'} - E_S$ 
13:    if  $\Delta E < 0$  or  $e^{-\Delta E/T} > rand(0,1)$  then
14:       $S \leftarrow S'$ 
15:    end if
16:  end for
17:   $T \leftarrow \alpha \times T$ 
18: end while
19: return  $S$ 

```

3.3. Dataset

We used the VRPTW instances from the Solomon benchmark [43] (composed of problems with 25, 50, and 100 customers) to generate a dataset to test the proposed mechanism for learning the performance mapping between problem instances and the algorithm space. The generated dataset comprises 26 features that characterize each VRPTW instance and a target corresponding to one of the three algorithms that, on average, solved a particular instance at the lowest cost.

Subsequently, we used the three algorithms to solve each VRPTW instance to define the specific configuration that best solved each one. Figure 1 presents a kernel density plot that compares the distributions of the resulting best cost solutions for the three problem sizes in the Solomon benchmark. The plot shows that although there is an overlap in the cost of the solutions across the problem sizes, the cost of the solutions generally increases with the problem size. The distribution of solutions for larger problem sizes is shifted to the right, indicating that the solutions are generally more expensive. This trend is consistent with the intuition that larger problems are more complex than smaller ones and require more resources to solve, which leads to higher costs. It is worth noting that the kernel density plot provides a probabilistic representation of the distribution of the solutions rather than a point estimate of the mean or median solution cost. Therefore, while the trend of increasing cost with problem size is evident in the plot, the exact magnitude of the increase and the shape of the distribution could be better understood by calculating summary statistics such as the mean and standard deviation. However, since the VRPTW is only the application problem and not the focus of this research, we do not show these values.

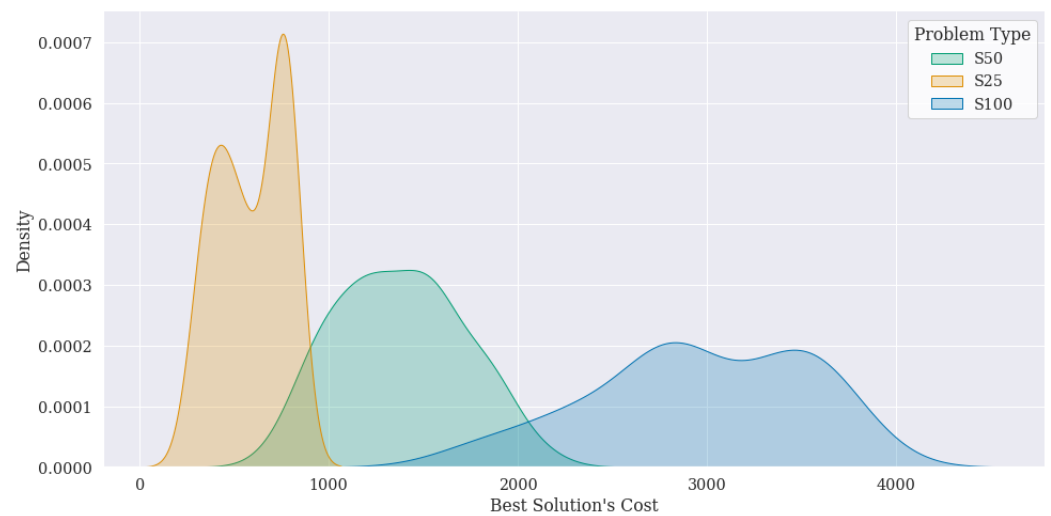


Figure 1. Distribution of best cost solutions for each VRPTW problem type.

As mentioned, each of the three algorithms employed in this work corresponds to an instance of SA and they differ only by their neighborhood operators, for which three different heuristics were used: intraroute 2-opt, intraroute Or-opt, and GENI exchange. Such heuristics are briefly described below and exemplified in Figure 2. Other than that, the three algorithms share the same values for the remaining hyperparameters, such as initial temperature ($T_{max} = 35$), final temperature ($T_{min} = 0.1$), cooling rate ($\alpha = 0.95$), and the number of iterations at a given temperature ($k = 50$).

Intraroute 2-opt reverses the order of the customers inside a segment of a randomly selected route. Figure 2a illustrates the behavior of intraroute 2-opt. In the example, a randomly selected segment from the original route $D - 1 - 3 - 2 - 4 - D$ was reversed from $3 - 2$ to $2 - 3$, which resulted in the updated route $D - 1 - 2 - 3 - 4 - D$.

Intraroute Or-opt moves a random segment of customers from a random route to another random position within the same route. This heuristic is exemplified in Figure 2b; in this case, the randomly selected segment was $2 - 3$ and was moved from the last position before returning to the depot at the second position right after visiting customer 1, effectively modifying the original route from $D - 1 - 4 - 5 - 2 - 3 - D$ to $D - 1 - 2 - 3 - 4 - 5 - D$. **GENI exchange** randomly selects two routes and inserts a randomly selected customer from the first route into a position between two random customers from a different random segment within the second route. GENI exchange is shown in Figure 2c where customer 8 was randomly selected from route 1 $D - 1 - 2 - 3 - 8 - D$, and was reduced to $D - 1 - 2 - 3 - D$ after removing it from the route. In the case of route 2 $D - 9 - 6 - 5 - 7 - 4 - D$, it was updated by selecting a random position in which customer 8 from route 1 was inserted. In this case, customer 8 was inserted between segment $9 - 7$, resulting in a new segment $9 - 8 - 7$ and yielding the final extended route 2 as $D - 9 - 8 - 7 - 6 - 5 - 4 - D$.

Furthermore, each problem was solved thirty times with each of the three algorithms for selecting the specific algorithm that produced the best result for a particular VRPTW instance. Then, we computed the mean cost of the best solution across all iterations. We selected the meta-heuristic that generated the solution with the lowest average cost as the best meta-heuristic for that particular problem instance.

Once again, it is important to stress that the VRPTW is not the actual focus of this research but a field for testing our solution approach. Although we care about the cost of the solutions, the importance relies on clearly defining a single algorithm out of the three that, on average, produces the best solution for a given VRPTW instance. For this reason, the hyperparameters used for the meta-heuristic and especially the three neighborhood operators that differentiate them from each other were specifically selected so that each

of the three configurations generates, on average, the best solution for only a third of the VRPTW instances.

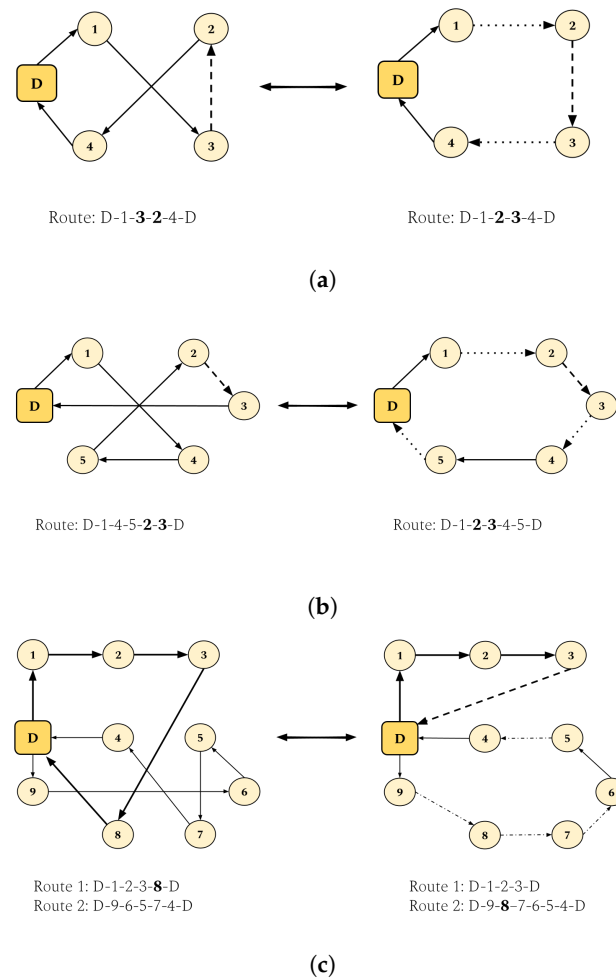


Figure 2. Heuristics used as neighborhood operators by SA. (a) Intraroute 2-opt; (b) Intraroute Or-opt; (c) Geni exchange.

Finally, we complemented the dataset by characterizing each of the VRPTW instances with 26 different features to provide an adequate description of the structure of each problem instance. Such features are summarized as follows:

- F1. Total number of vehicles [41,44].
- F2. Maximum capacity of a vehicle [41,44].
- F3. Total number of customers [45].
- F4-F5. X and Y coordinates of depot [44].
- F6-F10. Average, total, standard deviation, skewness, and kurtosis of demand [41,44].
- F11. Standard deviation of demand as a proportion of vehicle capacity. The standard deviation is computed from the values obtained by having the demand for each customer node divided by the total vehicle capacity [44].
- F12. Minimum vehicles required computed as total demand divided by total vehicle capacity [44].
- F13. The largest demand of a single customer node divided by the total vehicle capacity [44].
- F14. Average customers per route computed as number of customers divided by number of vehicles [46].
- F15. Average service time [31].
- F16. Sum of ready time [31].

- F17–F20. Average, standard deviation, skewness and kurtosis of time windows [41].
- F21. The total demand of a problem instance divided by the total vehicle capacity [44].
- F22–F26. Average, total, standard deviation, skewness and kurtosis of the overlap of two time windows [31].

In the case of features 22 to 26, the overlap O_{ij} of the time windows of customers i and j is computed as the ratio between the intersection and union of such time windows [31], as depicted in Equation (3).

$$O_{ij} = \frac{\text{intersection}(i, j)}{\text{union}(i, j)} \quad (3)$$

4. Experiments and Results

This section provides details of the experiments conducted, their main results, and a brief discussion of the most relevant findings.

We compared the proposed approach for learning the problem–algorithm mapping against five algorithm selectors frequently described in the recent literature related to this work. These algorithms contain the three main components (machine learning, meta-heuristics, and combinatorial optimization problems) identified by Karimi-Mamaghan et al. [23].

Moreover, we used Bayesian optimization [47] across all models for tuning their corresponding hyperparameters. This approach for hyperparameter tuning has proven to be an effective strategy for this task [48–50]. The five baseline methods against which the proposed attention-based meta-learner (AM)—with a single attention block—is compared are a multi-layer perceptron (MLP) [51], a k -nearest neighbor selector (KNN) [52], a softmax regression (SR) [53], a support vector machine (SVM) [54] and a decision tree (DT) [55].

The optimization scheme used for setting the hyperparameters of all models consisted of 300 iterations of Bayesian optimization coupled with 33 steps of random exploration. Regarding the search strategy, we used a Gaussian process along with Tikhonov regularization [56] to inform the overall search strategy for optimal hyperparameters with the predicted uncertainty via the acquisition function. We can define this Gaussian process as:

$$y \sim \mathcal{GP}(m(x), k(x_i, x_j)) \quad (4)$$

for which $m(x)$, the mean function, corresponds to the mean of the data and $k(x_i, x_j)$, the kernel function, is a Matérn covariance function [56] as defined by:

$$k(x_i, x_j) = \frac{1}{\Gamma(\nu)2^{\nu-1}} \left(\frac{\sqrt{2\nu}}{l} d_{ij} \right)^\nu K_\nu \left(\frac{\sqrt{2\nu}}{l} d_{ij} \right) \quad (5)$$

where Γ stands for the gamma function, K_ν stands for a modified Bessel function, and d_{ij} represents the Euclidean distance between vectors x_i and x_j [56]. We set parameters ν and l to $\frac{5}{2}$ and 1.0, respectively. Furthermore, the parameters of the kernel were optimized by maximizing the log-marginal-likelihood via the limited-memory Broyden–Fletcher–Goldfarb–Shannon algorithm extension for handling bound constraints (L-BFGS-B) [57] with 15,000 maximum iterations and function evaluations, 20 line search steps per iteration and an absolute step size of 1.00×10^{-8} for the numerical approximation of the Jacobian using forward differences.

We conducted the training process and hyperparameter optimization of all selectors using 85.8% of the previously described dataset following a 10-fold cross-validation scheme. We reserved the remaining 14.2% of the dataset for reporting the final results of the evaluation of the selectors.

This work's main point of interest is studying how often a selector chooses the correct algorithm for a particular instance that, on average, generates the best solution for a given VRPTW instance. For this reason, we decided to use accuracy as the metric to be optimized during the Bayesian optimization for the testing set across the ten folds. Table 1 shows

the hyperparameters and bounds used for each selector and the corresponding optimal values obtained.

Table 1. Hyperparameter ranges of values and optimal values obtained from Bayesian optimization.

Method	Hyperparameter	Ranges of Values	Optimal Value
AM	Epochs	(1, 1000)	1000
	Learning rate	(0.0, 0.01)	0.00043197
	Dropout	(0.1, 0.9)	0.12383
	Embedding size	d_h (16, 256)	196
	Attention heads	(2, 12)	8
MLP	Epochs	(1, 1000)	108
	Learning rate	(0.0, 0.1)	0.0048487
	L2 regularizartion	(0.0001, 1000)	0.30201379
	Hidden layer size	(1, 1000)	222
KNN	Neighbors	(1, 15)	1
	Leaf size	(1, 100)	42
SR	L2 regularization	(0.0001, 1000)	876.51276
	Maximum iterations	(1, 1000)	895
SVM	Kernel	Linear, Radial Basis Function, Sigmoid, Polynomial	Radial Basis Function
	L2 regularization	(0.0001, 1000)	957.93164
DT	Maximum depth	(1, 100)	95

Additionally, in the case of the DT model, we used entropy as a criterion to measure the quality of splits via information gain [58,59]. Moreover, we employed the Newton–CG algorithm for the optimization problem in the case of the SR selector [60,61]. For the MLP method, we used the ReLU [62] activation function along with the adaptive moment estimation (Adam) [63] algorithm and the negative log-likelihood loss. Furthermore, in the case of the KNN model, a ball-tree (metric-tree) algorithm was used for computing the nearest neighbors [64,65]. Finally, for the AM, we used the root mean squared propagation (RMSProp) [63] optimization algorithm along with the negative log-likelihood loss. Table 2 shows the results corresponding to the values for the accuracy, precision, recall, and F1-score, all computed on the 14.2% of the dataset previously described that was reserved exclusively for reporting the final results of the selectors’ evaluation.

Table 2. Selector results. Final values for accuracy, precision, recall and F1-score, with the best value for each metric marked in bold.

Metric	AM	SVM	KNN	DT	MLP	SR
Accuracy	87.38	86.72	84.32	81.98	72.79	70.72
Precision	89.03	87.56	85.20	83.23	73.93	71.80
Recall	87.33	86.67	83.87	82.00	72.67	70.77
F1	86.90	86.63	83.76	81.46	72.43	70.63

The results displayed in Table 2 show that the AM is consistently selecting, more often than the other five selectors, the specific algorithm that generates, on average, the lowest cost solution given a particular VRPTW instance. Moreover, compared to the other algorithm selectors, the AM shows a remarkable improvement in reducing the proportion of false positives when performing a selection.

Furthermore, a remarkable result is the significant difference in how often the AM selects the best algorithm for a given instance concerning the MLP. Overall, the AM selector shows an improvement of roughly 14% compared to the MLP, which confirms the idea that exploring more novel neural network architectures improves the learning of the mappings from problem instances to the algorithm space. This improvement is possible due to the

enhanced pattern-extraction process and further leveraging of the meta-features used to characterize the problem instances.

Overall, based on the results shown in Table 2, the AM selector performs better for all the analyzed metrics in selecting the most suitable algorithm to solve a particular problem instance. The latter supports the idea that leveraging the multi-head attention mechanism is a good approach for generating an improved representation of a problem instance based on its underlying structure and, thus, better learning the problem-algorithm mapping.

This improvement in performance for selecting the best algorithm for a given VRPTW instance was further validated via an F-test [66] employing the implementation provided by Raschka [67] used for similar studies, focusing on the accuracy as the main point of comparison. We conducted the test considering a significance level of 0.05 and under the null hypothesis that there is no difference between the accuracy of the algorithm selectors in terms of how often they choose the algorithm that, on average, produces the best solution for a given problem instance:

$$H_0 = p_{AM} = p_{MLP} = p_{KNN} = p_{SR} = p_{SVM} = p_{DT} \quad (6)$$

The test reported an F statistic of 25.707, which corresponds to a p -value of 4.591383×10^{-93} . This result suggests that the null hypothesis should be rejected, thus supporting the improvement of the accuracy of the AM selector which, when compared to that of the other five approaches shown in Table 2, is statistically significant.

We validated the perceived advantage in performance of the AM selector by conducting pair-wise McNemar tests (within-subjects chi-squared tests) [68] to compare its selection accuracy with that of the other five selectors. For the McNemar tests, we used the implementation provided by Raschka [67]. Moreover, a Bonferroni correction [69] was applied, reducing the assumed significance level for the test from 0.05 to 0.01 to avoid increasing the chances of Type I error when performing the five pair-wise McNemar tests. Table 3 summarizes the results from the pair-wise McNemar tests.

Table 3. p -values for pair-wise McNemar tests between AM and the other five selectors.

Method	AM
MLP	5.661648×10^{-15}
KNN	4.185436×10^{-10}
SR	3.547531×10^{-17}
SVM	9.443043×10^{-7}
DT	9.148237×10^{-13}

The resulting p -values for all five tests were smaller than the assumed significance threshold ($\alpha = 0.01$), therefore rejecting the null hypothesis for each test and indicating the difference in the higher selection accuracy between the AM and each of the other five selectors is statistically significant, with a significance level of 0.01.

5. Conclusions and Future Work

This study proposes using an attention-based neural network as a meta-learner to select a suitable algorithm that best solves a particular VRPTW instance. In our experiments, the proposed AM selector showed a statistically significant advantage over the other five baseline selectors from the literature for choosing the best algorithm for a specific VRPTW instance. Overall, the proposed model proved superior for all the analyzed metrics when selecting a suitable algorithm, i.e., accuracy, precision, recall, and F1-score. Thus, we can conclude that the attention-based meta-learner represents an acceptable alternative for learning problem-algorithm mapping.

Moreover, the evidence supports employing more novel neural network architectures as performance mapping mechanisms. We showed that the proposed AM selector performs better when comparing it against the simpler MLP architecture. The improvement

concerning the simpler MLP occurs not only in how often it chooses the best algorithm configuration for a specific problem instance but also in its clear advantage in reducing the proportion of false positives when performing a selection. This particular advantage can be observed explicitly in the precision scores for both selectors since, in the case of the MLP, all metrics have roughly the same value, whereas for the AM, the precision score is around two percentage points higher than the remaining metrics and 15.1 percentage points higher than that of the MLP selector, which corresponds to a 20% overall improvement in precision. Such evidence suggests that extracting patterns from the meta-features that characterize problem instances can still be significantly improved. It may be worth researching other architectures to improve the overall process of learning the mappings from problem instances to suitable algorithms.

In other words, our model selects the best algorithm out of three options for a given VRPTW instance. Moreover, by significantly outperforming the benchmark models, this method shows that exploring more complex and advanced performance mapping mechanisms appears to have good results that could significantly improve solving a problem instance via the algorithm selection framework. It also shows that neural networks, especially those with more complex architectures, yield better results and are worth studying.

An exciting path for future research would be to explore other novel neural network architectures and approaches for the selection process and improve the final problem instance representation. Some possible approaches that would be worth studying could include an adapted MMINet [70], local explanations [71], as well as other non-linear dimension reduction architectures such as EVNET [72]. We could also explore methodologies based on alternative machine learning techniques, such as self-organizing maps, and even explore the use of unsupervised approaches, such as the unsupervised modified graph embedding technique proposed by Lu et al. [73] or mixed methodologies such as the one introduced by Rovira et al. [74] combining unsupervised clustering and feature correlation. Furthermore, it would be interesting to explore using a different score function for the attention mechanism, such as employing a parameterized additive score function, which has outperformed multiplicative functions such as the one used in this work [10].

Additionally, employing the attention-based meta-learner for selecting heuristics within an online selection hyper-heuristic seems a potentially good research direction. Since the AM selector incrementally improves the representation of a problem instance, another interesting direction for future work would be to explore using additional features that better capture the structure of the problem. For example, we could integrate geometric, topological, local search probing, branch-and-cut probing, or nearest neighborhood features [41]. Moreover, we would like to validate the results obtained in this research by studying applications on larger datasets and a more extensive set of heuristics or meta-heuristics. Of course, exporting this model to other problem domains, such as routing problems, scheduling, and scheduling timetabling, would also represent an exciting possibility.

Author Contributions: Conceptualization, E.D.d.L.-H. and S.E.C.-P.; Methodology, S.E.C.-P.; Software, E.D.d.L.-H. and S.E.C.-P.; Validation, E.D.d.L.-H., S.E.C.-P. and H.T.-M.; Formal analysis, E.D.d.L.-H.; Investigation, E.D.d.L.-H. and S.E.C.-P.; Data curation, E.D.d.L.-H.; Writing—original draft, E.D.d.L.-H.; Writing—review & editing, S.E.C.-P., J.C.O.-B. and H.T.-M.; Supervision, S.E.C.-P. and J.C.O.-B.; Project administration, S.E.C.-P. and J.C.O.-B.; Funding acquisition, H.T.-M. All authors have read and agreed to the published version of the manuscript.

Funding: The APC was funded by Tecnológico de Monterrey.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing not applicable. No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Tan, K.; Lee, L.; Zhu, Q.; Ou, K. Heuristic methods for vehicle routing problem with time windows. *Artif. Intell. Eng.* **2001**, *15*, 281–295. [\[CrossRef\]](#)
2. Lara-Cárdenas, E.; Sánchez-Díaz, X.; Amaya, I.; Cruz-Duarte, J.M.; Ortiz-Bayliss, J.C. A Genetic Programming Framework for Heuristic Generation for the Job-Shop Scheduling Problem. In Proceedings of the Advances in Soft Computing, Mexico City, Mexico, 12–17 October 2020; Martínez-Villaseñor, L., Herrera-Alcántara, O., Ponce, H., Castro-Espinoza, F.A., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 284–295.
3. Sanchez, M.; Cruz-Duarte, J.M.; Ortiz-Bayliss, J.C.; Ceballos, H.; Terashima-Marin, H.; Amaya, I. A Systematic Review of Hyper-Heuristics on Combinatorial Optimization Problems. *IEEE Access* **2020**, *8*, 128068–128095. [\[CrossRef\]](#)
4. Wolpert, D.; Macready, W. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [\[CrossRef\]](#)
5. Ho, Y.; Pepyne, D. Simple Explanation of the No-Free-Lunch Theorem and Its Implications. *J. Optim. Theory Appl.* **2002**, *115*, 549–570. [\[CrossRef\]](#)
6. Kotthoff, L. Algorithm Selection for Combinatorial Search Problems: A Survey. *AI Mag.* **2014**, *35*, 48–60. [\[CrossRef\]](#)
7. Smith-Miles, K.; van Hemert, J. Discovering the suitability of optimisation algorithms by learning from evolved instances. *Ann. Math. Artif. Intell.* **2011**, *61*, 87–104. [\[CrossRef\]](#)
8. Smith-Miles, K.A. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Comput. Surv.* **2009**, *41*, 1–25. [\[CrossRef\]](#)
9. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.
10. Niu, Z.; Zhong, G.; Yu, H. A review on the attention mechanism of deep learning. *Neurocomputing* **2021**, *452*, 48–62. [\[CrossRef\]](#)
11. Voita, E.; Talbot, D.; Moiseev, F.; Sennrich, R.; Titov, I. Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. *arXiv* **2019**, arXiv:1905.09418.
12. Dargan, S.; Kumar, M.; Ayyagari, M.R.; Kumar, G. A survey of deep learning and its applications: A new paradigm to machine learning. *Arch. Comput. Methods Eng.* **2020**, *27*, 1071–1092. [\[CrossRef\]](#)
13. Lakshmana, K.; Kaluri, R.; Gundluru, N.; Alzamil, Z.S.; Rajput, D.S.; Khan, A.A.; Haq, M.A.; Alhussen, A. A review on deep learning techniques for IoT data. *Electronics* **2022**, *11*, 1604. [\[CrossRef\]](#)
14. Abdou, M.A. Literature review: Efficient deep neural networks techniques for medical image analysis. *Neural Comput. Appl.* **2022**, *34*, 5791–5812. [\[CrossRef\]](#)
15. Peng, S.; Cao, L.; Zhou, Y.; Ouyang, Z.; Yang, A.; Li, X.; Jia, W.; Yu, S. A survey on deep learning for textual emotion analysis in social networks. *Digit. Commun. Netw.* **2022**, *8*, 745–762. [\[CrossRef\]](#)
16. Aslani, S.; Jacob, J. Utilisation of deep learning for COVID-19 diagnosis. *Clin. Radiol.* **2023**, *78*, 150–157. [\[CrossRef\]](#)
17. Sadeg, S.; Hamdad, L.; Kada, O.; Benatchba, K.; Habbas, Z. Meta-learning to Select the Best Metaheuristic for the MaxSAT Problem. In Proceedings of the International Symposium on Modelling and Implementation of Complex Systems, Batna, Algeria, 24–26 October 2020; pp. 122–135.
18. Miranda, E.S.; Fabris, F.; Nascimento, C.G.; Freitas, A.A.; Oliveira, A.C. Meta-learning for recommending metaheuristics for the maxsat problem. In Proceedings of the 2018 7th Brazilian Conference on Intelligent Systems (BRACIS), Sao Paulo, Brazil, 22–25 October 2018; pp. 169–174.
19. Smith-Miles, K.; Lopes, L. Measuring instance difficulty for combinatorial optimization problems. *Comput. Oper. Res.* **2012**, *39*, 875–889. [\[CrossRef\]](#)
20. Kanda, J.; de Carvalho, A.; Hruschka, E.; Soares, C.; Brazdil, P. Meta-learning to select the best meta-heuristic for the Traveling Salesman Problem: A comparison of meta-features. *Neurocomputing* **2016**, *205*, 393–406. [\[CrossRef\]](#)
21. Ibrahim, F.A.; Shiba, O.A. Data Mining: WEKA Software (An Overview). *J. Pure Appl. Sci.* **2019**, *18*, 3.
22. Rice, J.R. The Algorithm Selection Problem. In *Advances in Computers*; Elsevier: Amsterdam, The Netherlands, 1976; Volume 15, pp. 65–118.
23. Karimi-Mamaghan, M.; Mohammadi, M.; Meyer, P.; Karimi-Mamaghan, A.M.; Talbi, E.G. Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. *Eur. J. Oper. Res.* **2022**, *296*, 393–422. [\[CrossRef\]](#)
24. de la Rosa-Rivera, F.; Nunez-Varela, J.I.; Ortiz-Bayliss, J.C.; Terashima-Marín, H. Algorithm Selection for Solving Educational Timetabling Problems. *Expert Syst. Appl.* **2021**, *174*, 114694. [\[CrossRef\]](#)
25. Cao, W.; Wu, Y.; Wang, Q.; Zhang, J.; Zhang, X.; Qiu, M. A Novel RVFL-Based Algorithm Selection Approach for Software Model Checking. In Proceedings of the International Conference on Knowledge Science, Engineering and Management, Singapore, 6–8 August 2022; pp. 414–425.
26. Pavelski, L.; Delgado, M.; Kessaci, M.E. Meta-learning for optimization: A case study on the flowshop problem using decision trees. In Proceedings of the 2018 IEEE Congress on Evolutionary Computation (CEC), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
27. Müller, D.; Müller, M.G.; Kress, D.; Pesch, E. An algorithm selection approach for the flexible job shop scheduling problem: Choosing constraint programming solvers through machine learning. *Eur. J. Oper. Res.* **2022**, *302*, 874–891. [\[CrossRef\]](#)
28. Cohen-Shapira, N.; Rokach, L. Learning dataset representation for automatic machine learning algorithm selection. *Knowl. Inf. Syst.* **2022**, *64*, 2599–2635. [\[CrossRef\]](#)
29. Huisman, M.; Van Rijn, J.N.; Plaat, A. A survey of deep meta-learning. *Artif. Intell. Rev.* **2021**, *54*, 4483–4541. [\[CrossRef\]](#)

30. Kallehauge, B.; Larsen, J.; Madsen, O.B.; Solomon, M.M., Vehicle Routing Problem with Time Windows. In *Column Generation*; Desaulniers, G., Desrosiers, J., Solomon, M.M., Eds.; Springer: Boston, MA, USA, 2005; pp. 67–98.
31. Gutierrez-Rodríguez, A.E.; Conant-Pablos, S.E.; Ortiz-Bayliss, J.C.; Terashima-Marín, H. Selecting meta-heuristics for solving vehicle routing problems with time windows via meta-learning. *Expert Syst. Appl.* **2019**, *118*, 470–481. [[CrossRef](#)]
32. Konstantakopoulos, G.D.; Gayialis, S.P.; Kechagias, E.P. Vehicle routing problem and related algorithms for logistics distribution: A literature review and classification. *Oper. Res.* **2022**, *22*, 2033–2062. [[CrossRef](#)]
33. Lin, S.W.; Ying, K.C.; Lee, Z.J.; Chen, H.S. Vehicle Routing Problems with Time Windows Using Simulated Annealing. In Proceedings of the 2006 IEEE International Conference on Systems, Man and Cybernetics, Taipei, Taiwan, 8–11 October 2006.
34. Ochelska-Mierzejewska, J.; Poniszewska-Marañda, A.; Marañda, W. Selected Genetic Algorithms for Vehicle Routing Problem Solving. *Electronics* **2021**, *10*, 3147. [[CrossRef](#)]
35. Ngo, T.S.; Jaafar, J.; Aziz, I.A.; Aftab, M.U.; Nguyen, H.G.; Bui, N.A. Metaheuristic Algorithms Based on Compromise Programming for the Multi-Objective Urban Shipment Problem. *Entropy* **2022**, *24*, 388. [[CrossRef](#)]
36. Liu, G.; Guo, J. Bidirectional LSTM with attention mechanism and convolutional layer for text classification. *Neurocomputing* **2019**, *337*, 325–338. [[CrossRef](#)]
37. Yan, X.; Hu, S.; Mao, Y.; Ye, Y.; Yu, H. Deep multi-view learning methods: A review. *Neurocomputing* **2021**, *448*, 106–129. [[CrossRef](#)]
38. Kool, W.; van Hoof, H.; Welling, M. Attention, Learn to Solve Routing Problems! In Proceedings of the International Conference on Learning Representations, New Orleans, LO, USA, 6–9 May 2019.
39. Peng, B.; Wang, J.; Zhang, Z. A deep reinforcement learning algorithm using dynamic attention model for vehicle routing problems. In Proceedings of the International Symposium on Intelligence Computation and Applications, Ghaziabad, India, 6–8 December 2019; pp. 636–650.
40. Mısı, M.; Gunawan, A.; Vansteenwegen, P. Algorithm selection for the team orienteering problem. In Proceedings of the European Conference on Evolutionary Computation in Combinatorial Optimization (Part of EvoStar), Madrid, Spain, 20–22 April 2022; pp. 33–45.
41. Rasku, J.; Kärkkäinen, T.; Musliu, N. Feature extractors for describing vehicle routing problem instances. In *OASICS*; Dagstuhl Publishing: Saarbrücken, Germany, 2016.
42. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by simulated annealing. *Science* **1983**, *220*, 671–680. [[CrossRef](#)]
43. Solomon, M.M. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Oper. Res.* **1987**, *35*, 254–265. [[CrossRef](#)]
44. Steinhaus, M. *The Application of the Self Organizing Map to the Vehicle Routing Problem*; University of Rhode Island: Kingston, RI, USA, 2015.
45. Kanda, J.; Carvalho, A.; Hruschka, E.; Soares, C. Selection of algorithms to solve traveling salesman problems using meta-learning. *Int. J. Hybrid Intell. Syst.* **2011**, *8*, 117–128.
46. Daduna, J.R.; Voß, S. Practical Experiences in Schedule Synchronization. In Proceedings of the Computer-Aided Transit Scheduling, Lisbon, Portugal, 6–9 July 1993; Daduna, J.R., Branco, I., Paixao, J.M.P., Eds.; Springer: Berlin/Heidelberg, Germany, 1995; pp. 39–55.
47. Wu, J.; Chen, X.Y.; Zhang, H.; Xiong, L.D.; Lei, H.; Deng, S.H. Hyperparameter optimization for machine learning models based on Bayesian optimization. *J. Electron. Sci. Technol.* **2019**, *17*, 26–40.
48. Turner, R.; Eriksson, D.; McCourt, M.; Kiili, J.; Laaksonen, E.; Xu, Z.; Guyon, I. Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. In Proceedings of the NeurIPS 2020 Competition and Demonstration Track, Virtual, 6–12 December 2020; pp. 3–26.
49. Lindauer, M.; Eggensperger, K.; Feurer, M.; Biedenkapp, A.; Deng, D.; Benjamins, C.; Ruhkopf, T.; Sass, R.; Hutter, F. SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization. *J. Mach. Learn. Res.* **2022**, *23*, 54–61.
50. Riboni, A.; Ghioldi, N.; Candelieri, A.; Borrotti, M. Bayesian optimization and deep learning for steering wheel angle prediction. *Sci. Rep.* **2022**, *12*, 8739. [[CrossRef](#)]
51. Ramchoun, H.; Ghanou, Y.; Ettaouil, M.; Janati Idrissi, M.A. Multilayer perceptron: Architecture optimization and training. *Int. J. Interact. Multimed. Artif. Intell.* **2016**. [[CrossRef](#)]
52. Cunningham, P.; Delany, S.J. k-Nearest neighbour classifiers—A Tutorial. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–25. [[CrossRef](#)]
53. Movellan, J.R. Tutorial on Multinomial Logistic Regression. MPLab Tutorials. 2006. Available online: <http://mplab.ucsd.edu> (accessed on 8 March 2023).
54. Ghosh, S.; Dasgupta, A.; Swetapadma, A. A study on support vector machine based linear and non-linear pattern classification. In Proceedings of the 2019 International Conference on Intelligent Sustainable Systems (ICISS), Palladam, India, 21–22 February 2019; pp. 24–28.
55. Hernández, V.A.S.; Monroy, R.; Medina-Pérez, M.A.; Loyola-González, O.; Herrera, F. A practical tutorial for decision tree induction: Evaluation measures for candidate splits and opportunities. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–38. [[CrossRef](#)]
56. Rasmussen, C.E.; Williams, C. *Gaussian Processes for Machine Learning* the Mit Press; MIT Press: Cambridge, MA, USA, 2006; Volume 32, pp. 68.
57. Morales, J.L.; Nocedal, J. Remark on “Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization”. *ACM Trans. Math. Softw. (TOMS)* **2011**, *38*, 1–4. [[CrossRef](#)]

58. Kaul, S.; Fayaz, S.A.; Zaman, M.; Butt, M.A. Is decision tree obsolete in its original form? A Burning debate. *Rev. D'Intelligence Artif.* **2022**, *36*, 105–113. [[CrossRef](#)]
59. Nanfack, G.; Temple, P.; Frénay, B. Constraint Enforcement on Decision Trees: A Survey. *ACM Comput. Surv. (CSUR)* **2022**, *54*, 1–36. [[CrossRef](#)]
60. Zeng, X.; Liu, Y.; Liu, W.; Yuan, C.; Luo, X.; Xie, F.; Chen, X.; de la Chapelle, M.L.; Tian, H.; Yang, X.; et al. Evaluation of classification ability of Logistic Regression model on SERS data of miRNAs. *J. Biophotonics* **2022**, *15*, e202200108. [[CrossRef](#)] [[PubMed](#)]
61. Sharma, M.; Agarwal, S.K.; Bundele, M. Decisive Analysis of multiple logistic regression apropos of hyper-parameters. *Indian J. Comput. Sci. Eng.* **2022**, *13*, 188–196. [[CrossRef](#)]
62. Dubey, S.R.; Singh, S.K.; Chaudhuri, B.B. Activation Functions in Deep Learning: A comprehensive Survey and Benchmark. *Neurocomputing* **2022**, *503*, 92–108. [[CrossRef](#)]
63. Zou, F.; Shen, L.; Jie, Z.; Zhang, W.; Liu, W. A sufficient condition for convergences of adam and rmsprop. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 11127–11135.
64. Zuo, C.; Pan, Z.; Yin, Z.; Guo, C. A nearest neighbor multiple-point statistics method for fast geological modeling. *Comput. Geosci.* **2022**, *167*, 105208. [[CrossRef](#)]
65. Zhu, J.; Yu, E.; Li, Q.; Chen, H.; Shitz, S.S. Ball-Tree-Based Signal Detection for LMA MIMO Systems. *IEEE Commun. Lett.* **2022**, *26*, 602–606. [[CrossRef](#)]
66. Snedecor, G.W.; Cochran, W.G. *Statistical Methods*, 8th ed.; Iowa State University Press: Ames, IA, USA, 1989; Volume 1191.
67. Raschka, S. MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack. *J. Open Source Softw.* **2018**, *3*, 638. [[CrossRef](#)]
68. McNemar, Q. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika* **1947**, *12*, 153–157. [[CrossRef](#)]
69. Groenwold, R.H.; Goeman, J.J.; Le Cessie, S.; Dekkers, O.M. Multiple testing: When is many too much? *Eur. J. Endocrinol.* **2021**, *184*, E11–E14. [[CrossRef](#)]
70. Özdenizci, O.; Erdoğan, D. Stochastic mutual information gradient estimation for dimensionality reduction networks. *Inf. Sci.* **2021**, *570*, 298–305. [[CrossRef](#)]
71. Björklund, A.; Mäkelä, J.; Puolamäki, K. SLISEMAP: Supervised dimensionality reduction through local explanations. *Mach. Learn.* **2022**, *112*, 1–43. [[CrossRef](#)]
72. Zang, Z.; Cheng, S.; Lu, L.; Xia, H.; Li, L.; Sun, Y.; Xu, Y.; Shang, L.; Sun, B.; Li, S.Z. EVNet: An Explainable Deep Network for Dimension Reduction. *IEEE Trans. Vis. Comput. Graph.* **2022**. [[CrossRef](#)]
73. Lu, X.; Long, J.; Wen, J.; Fei, L.; Zhang, B.; Xu, Y. Locality preserving projection with symmetric graph embedding for unsupervised dimensionality reduction. *Pattern Recognit.* **2022**, *131*, 108844. [[CrossRef](#)]
74. Rovira, M.; Engvall, K.; Duwig, C. Identifying key features in reactive flows: A tutorial on combining dimensionality reduction, unsupervised clustering, and feature correlation. *Chem. Eng. J.* **2022**, *438*, 135250. [[CrossRef](#)]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.