

Feature-Free Algorithm Selection for the Traveling Salesman Problem Using Distance-Matrix Convolutional Networks

Fatima Zahrae Mellouk*, Bouchra Karoum, Imad Hafidi

IDIA Laboratory, National School of Applied Sciences, Sultan Moulay Slimane University, Morocco

Abstract Algorithm selection has emerged as an important challenge in solving the Traveling Salesman Problem (TSP), as the performance of metaheuristics can vary significantly across problem instances, directly affecting the quality of the resulting TSP tours. Existing learning-based approaches typically rely on either handcrafted meta-features or image-based representations of TSP instances. The former depends on predefined feature engineering, whereas the latter involves additional preprocessing. This paper proposes a feature-free deep learning approach based on a two-dimensional Convolutional Neural Network (2D-CNN) that uses the raw TSP distance matrix directly as input, thereby avoiding handcrafted feature extraction and image-generation preprocessing. A dataset of 3,000 subinstances was created from ten TSPLIB source instances, and six metaheuristics were evaluated to determine the best-performing solver for each instance. The proposed model achieves an accuracy of **0.91**, outperforming the evaluated image-based deep-learning approaches while remaining competitive with feature-engineered machine-learning models, which achieve up to **0.93** accuracy. Under a stricter Leave-One-TSPLIB-Family-Out evaluation, the model achieves a mean accuracy of **0.8233** and an F1-score of **0.8146**, providing additional evidence of cross-family predictive capability. These results demonstrate the potential of direct distance-matrix learning as an alternative representation for TSP algorithm selection.

Keywords Algorithm Selection, Deep Learning, Traveling Salesman Problem (TSP), Convolutional Neural Network, Metaheuristics.

AMS 2010 subject classifications 90C27, 90C59, 68T07

DOI: 10.19139/soic-2310-5070-4131

1. Introduction

The Traveling Salesman Problem (TSP) is a critical problem for various modern decision-making systems, such as vehicle routing [1], urban logistics [2], delivery services [3], production scheduling [4], and micro-electronics manufacturing [5]. The performance of such systems is not limited to solution computation; it also involves operational costs, service quality, and resource utilization. The selection of an inappropriate metaheuristic for a TSP instance can result in inefficient routes and increased resource utilization.

Across these application areas, solution methods are evaluated not only in terms of computational efficiency, but also according to operational cost, service quality, and resource utilization. Moreover, the wide variety of TSP instances leads to substantial variations in algorithm performance, making it difficult to identify a single metaheuristic that consistently performs well across all instances.

The No-Free-Lunch Theorem states that no metaheuristic can consistently outperform all others across every possible scenario. A method that performs extremely well on one set of instances may perform poorly on another [6]. As a result, the main challenge is not only to design new metaheuristics, but also to identify the most suitable

*Correspondence to: Fatima zahrae Mellouk (Email: fatimazahrae.mellouk@usms.ac.ma). IDIA Laboratory, National School of Applied Sciences, Sultan Moulay Slimane University. Bd Béni Amir, BP 77, Khouribga, 25000, Morocco.

solver for a given instance without relying on expensive trial-and-error. Rice formalized this issue as the algorithm selection problem [7], and it has since been further developed in the meta-learning literature [8, 9].

For the TSP, the influential study by Kanda et al. [10] showed that meta-learning can effectively connect tabular meta-features to the rankings of several metaheuristics (TS, GRASP, SA, GA, and ACO) across different graph types, including symmetric versus asymmetric and strongly versus weakly connected graphs. Their approach combines graph descriptors, such as edge and vertex statistics and complex network metrics, with algorithm-related features and subsampling landmarks. These representations are then used with standard machine-learning models, including MLP, k-NN, and decision trees, to generate solver rankings.

Research has demonstrated the usefulness of selecting an appropriate algorithm for a given TSP instance. However, these approaches generally assume that each TSP instance must first be converted into a manually constructed tabular feature vector. Although convenient, this representation reduces the rich geometric structure contained in the distance matrix, including local neighborhoods, clustering patterns, and spatial regularities, to a limited set of aggregated descriptors. Consequently, potentially relevant structural information may not be captured by the selected meta-features.

In addition, advances in deep neural networks for combinatorial optimization problems with convolutional neural networks [11], sequence models such as LSTMs [12], and efficient neural networks such as ResNet [13] and EfficientNet [14] have shown the potential of learning representations directly from structured data. Recent studies have further explored the interaction between learning and optimization, including deep reinforcement learning for combinatorial optimization [15], metaheuristics for neural-network optimization [16], and hybrid CNN–metaheuristic frameworks [17]. Nevertheless, as far as we know, prior works on TSP algorithm selection have not used the raw distance matrix as the primary input representation. Instead, several deep-learning-based approaches rely on image-like representations constructed from TSP instances, such as scatter plots, MST graphs, and k -nearest-neighbor projections. Such transformations introduce additional preprocessing choices, including resolution, scaling, marker size, and rendering parameters.

The distance matrix, in contrast, directly represents all pairwise distances of a TSP instance without requiring image construction or handcrafted meta-feature extraction. This representation therefore provides a natural basis for investigating whether a deep learning model can learn solver-discriminative structural patterns directly from pairwise distance information. This observation motivates our use of distance-matrix learning for TSP algorithm selection.

In this article, we address this gap by proposing, to the best of our knowledge, a feature-free TSP algorithm-selection framework that operates directly on distance matrices. Each 100×100 distance matrix is provided directly to the network, without manually engineered meta-features or image-generation preprocessing. We evaluate four standalone deep architectures: a simplified 2D-CNN baseline, a lightweight ResNet-Lite with residual connections, an EfficientNet-Lite designed to balance predictive accuracy and parameter efficiency, and an LSTM model that processes each row as a sequence to capture possible sequential regularities.

To better clarify the role of deep learning, we also carry out a systematic comparison with four classical machine-learning baselines trained on tabular meta-features: XGBoost, Random Forest, Decision Tree, and SVM. All models are trained and evaluated on the same collection of TSP instances under a unified experimental protocol.

The remainder of the paper is organized as follows. Section 2 introduces the Traveling Salesman Problem, and Section 3 reviews existing approaches to TSP algorithm selection. Section 4 presents the proposed methodology, while Section 5 reports the experimental results and comparisons. Finally, Sections 6 and 7 discuss the findings and conclude with directions for future work.

2. The Traveling Salesman Problem (TSP)

The Traveling Salesman Problem (TSP) is a well-known and extensively studied problem in the field of combinatorial optimization. Given a set of n cities together with the pairwise distances that separate them, the objective is to determine the shortest Hamiltonian cycle that visits each city exactly once and returns to the starting

city [18, 19]. From a formal standpoint, the TSP can be modeled as a complete weighted graph $G = (V, E, d)$, where the total cost associated with a tour π is defined as:

$$C(\pi) = \sum_{k=1}^n d(\pi_k, \pi_{k+1}).$$

The TSP is known to be NP-hard [20], and its search space grows factorially with n , making exact methods impractical for medium and large instances. For this reason, numerous metaheuristics such as Tabu Search, Simulated Annealing, Genetic Algorithms, and Ant Colony Optimization are commonly employed to obtain high-quality near-optimal solutions.

In the symmetric version studied in this work, distances satisfy $d(i, j) = d(j, i)$, enabling the instance to be represented by its distance matrix $D \in \mathbb{R}^{n \times n}$. This matrix encodes the geometric structure of the instance and serves as the basis of the deep learning-based algorithm selection approach is proposed in this study.

3. Related Work

Meta-learning has been key in automating algorithm selection for the Traveling Salesman Problem (TSP). One of the first contributions in this area came from Kanda et al. [21]. They studied neural predictors trained on TSPLIB instances, which were described using various hand-crafted meta-features. These included measures of graph structure and edge-weight variability. The results indicated that simple neural networks, such as MLP, were capable of linking instance attributes with the best solver. As a subsequent step, the same authors [22] evaluated multiple types of learning models, among which were k-NN, Decision Trees, SVM, and Naïve Bayes. It was noted that Decision Trees exhibited a particularly good performance, reaching an F measure of 0.95 for all kinds of TSP problems. Finally, a recent paper by Kanda et al. [10] formulated the problem as a ranking task. The study demonstrated that vertex and edge descriptors proved highly valuable when predicting the relative performance of multiple metaheuristic algorithms.

Building on these foundations, Bossenbroek et al. [23] introduced a hybrid meta-learning pipeline. This pipeline combines heuristic construction methods with machine-learning-based selectors to address large-scale TSP instances. Their experimental results showed that neural network models usually achieve the highest prediction accuracy. However, simpler learning algorithms can still perform competitively while requiring less computation. More recently, other general meta-learning frameworks have been proposed [24, 25]. These frameworks demonstrate that methods capable of transferring knowledge across different problem distributions can further improve robustness and generalization in algorithm selection settings.

With the rise of deep learning techniques, a new group of models that does not rely on manually created features has emerged. This allows for the automatic choice of algorithms for combinatorial optimization problems. Seiler et al. [26] introduced one of the earliest convolutional architectures for this purpose. They transformed each TSP instance into visual representations such as scatter plots, minimum spanning trees, and k-nearest-neighbor graphs. Then they trained a CNN to differentiate between competing solvers. Zhao et al. [27] built on this work with CTAS, a ResNet-based classifier that works with density-map images. They reported state-of-the-art performance among image-based selectors. Huerta et al. [28] further demonstrated the power of CNNs by encoding instances into grid-structured maps designed for anytime performance prediction. While these methods have been successful, they depend on manually designed image transformations. It introduces more preprocessing steps and decreases its overall applicability. Another method was proposed by Alissa et al. [29]. It is based on introducing a fully feature-free LSTM-based solver that directly works with the sequence of instances. While designed specifically for 1D Bin Packing, the proposed model achieved outstanding results (82-96% accuracy), beating the best individual solver. It proves that using deep sequential models, one can train a network that learns how to match an instance with an appropriate solver without any additional features.

Recently, a great number of papers appeared, considering graph neural networks as a promising representation of the intrinsic structure of TSP instances. In particular, Song et al. [30] introduced GINES, which is currently the first-ever GNN-based selector for TSP. It uses a message-passing mechanism for building structural embeddings

that encode coordinates and weighted edges. The experimental results prove the effectiveness of the proposed method compared to all considered alternatives.

Unlike previous CNN- and GNN-based methods, our model utilizes learned visual encodings rather than relying on the distance matrix itself as an invariant lossless representation of the TSP instance from which the CNN learns structure-related features of the solver. Unlike previous CNN- and GNN-based methods for TSP, our method utilizes the raw distance matrix as an input for the CNN. The distance matrix contains all pairwise distances without any loss of information because it does not employ spatial embedding. In comparison to graph-based approaches to TSP, the distance matrix is a much denser input representation for CNNs, making it easier to learn using global structure information.

4. Methodology

Our methodology follows a structured pipeline that transforms raw TSP instances into a distance-matrix representation, evaluates the performance of multiple meta-heuristics, and trains a convolutional neural network (CNN) to predict the most suitable solver for unseen instances. The workflow is illustrated in Figure 1.

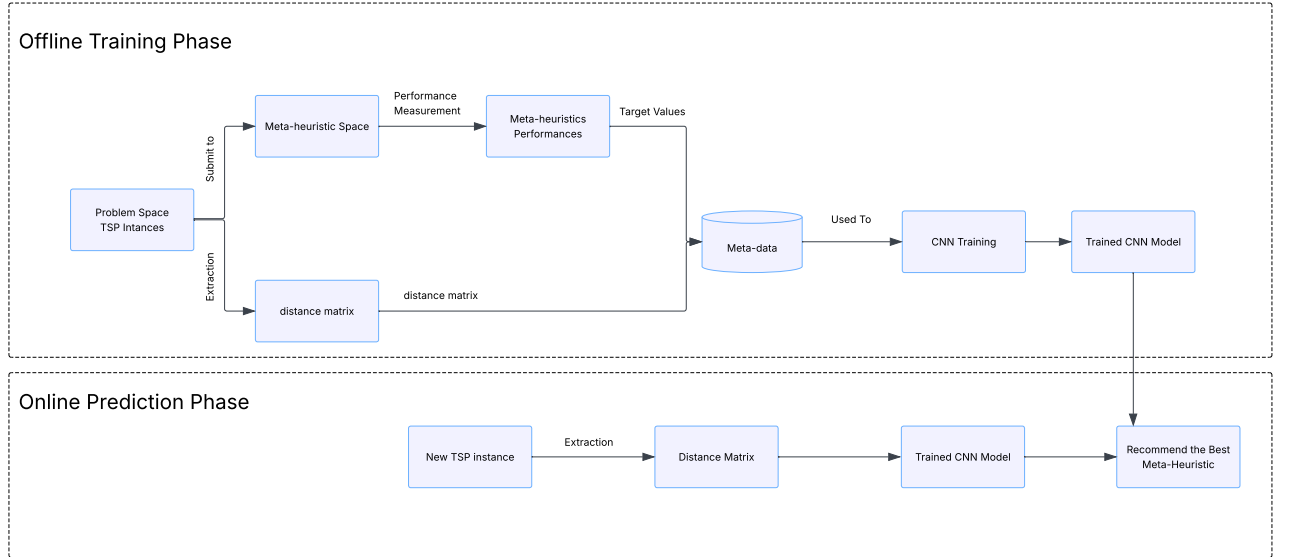


Figure 1. Overview of the proposed TSP algorithm-selection framework, distinguishing the offline training phase from the online prediction phase.

4.1. Problem Space

According to the extraction process defined in Kanda et al. [31], we created our problem space using a subset of coordinate-based instances from TSPLIB. The process was based on ten symmetric problem instances: d1655, f11400, nrw1379, pcb3038, pr2392, r11889, u1817, vm1748, f11577, and rat783. These instances were primarily selected because they are sufficiently large to support a consistent subinstance-generation protocol. For each of these ten problem instances, 300 problem subinstances of 100 cities were created. We selected this problem size as a tradeoff between making the evaluation of multiple metaheuristics computationally tractable and preserving sufficient geometric variety to allow meaningful discrimination between solvers. Each problem was created as a complete subgraph on 100 cities. All edge weights were copied from the original TSPLIB instance.

Hence, relevant properties of the original instances, such as Euclidean geometry, symmetry, and metricity, were preserved.

The resulting problem set comprises 3,000 TSP subinstances and was used as the experimental basis for evaluating the proposed solver-selection framework.

It should be emphasized that this selection was not originally designed to maximize structural coverage of the broader TSPLIB problem space. To assess this limitation quantitatively, we performed a separate post-hoc coverage analysis in which the ten source instances were compared with 88 valid coordinate-based TSPLIB instances. These additional instances were used exclusively as an external reference set for structural coverage analysis and were not used for CNN training, validation, testing, or algorithm-performance labeling.

The analysis considered scale-invariant descriptors related to pairwise-distance distributions, nearest-neighbor structure, radial dispersion, and geometric anisotropy. The ten selected sources covered 37.88% of the descriptor ranges on average (median: 34.35%). In addition, the first two principal components explained 63.98% of the structural variance, while the ten selected sources covered 7.07% of the corresponding two-dimensional PCA convex-hull area.

Table 1. Structural coverage analysis of the ten selected TSPLIB source instances relative to the coordinate-based TSPLIB reference set. The additional TSPLIB instances were used only for coverage analysis and not for CNN training or testing.

Coverage indicator	Result
Selected source instances	10
Reference TSPLIB instances	88
Mean descriptor-range coverage	37.88%
Median descriptor-range coverage	34.35%
Variance explained by PC1–PC2	63.98%
PCA 2D convex-hull coverage	7.07%
Mean nearest-source distance	2.9471

Table 1 summarizes the structural coverage of the ten selected source instances relative to the broader coordinate-based TSPLIB reference set. The results indicate that the ten sources represent only part of the structural diversity of this reference collection, with a mean descriptor-range coverage of 37.88% and a two-dimensional PCA convex-hull coverage of 7.07%. We therefore acknowledge the number and diversity of source instances as a limitation of the present benchmark. Future work will extend the benchmark with additional independent source instances, preferably selected according to structural diversity and coverage of underrepresented regions of the instance space.

4.2. Meta-heuristic Space

The selection of the metaheuristic algorithms was based on comparative studies of optimization methodologies [32] and reviews of recent advances in metaheuristics [33]. After reviewing the literature, six representative and complementary metaheuristics were selected: Simulated Annealing (SA) [34], Tabu Search (TS) [35], Ant Colony Optimization (ACO) [36], Genetic Algorithm (GA) [37], Artificial Bee Colony (ABC) [38], and Enhanced Fruit Fly Optimization Algorithm (EFOA) [39]. The objective was to construct a heterogeneous portfolio covering different search paradigms rather than to restrict the portfolio to the most frequently used metaheuristics.

EFOA was specifically included because of the improvements in global search capability and optimization precision reported over the conventional FOA by Huang et al. [40]. Overall, the selected algorithms provide complementary search behaviors: SA and TS represent trajectory-based search, whereas ACO, GA, and ABC employ population-based search mechanisms, and EFOA introduces an alternative population-based strategy aimed at balancing intensification and diversification. The inclusion of EFOA is not intended to imply superiority over other established swarm-intelligence methods such as PSO; rather, it was selected to broaden the diversity of search mechanisms represented in the portfolio. Extending the portfolio with additional algorithms such as PSO constitutes a direction for future work.

To make a fair comparison among the algorithms, they were executed under the same experimental conditions. Each metaheuristic was run 10 times for each problem instance. The algorithm parameters were derived from the literature and experimentally tuned. The best solution in terms of minimum tour length was used for evaluation. The configuration settings for the algorithms are as follows:

- **SA:** Initial temperature = 1000; cooling rate = 0.995; 100 iterations.
- **TS:** 100 iterations; tabu tenure = 10; perturbation every 30 iterations; tabu list size = 10.
- **EFOA:** Maximum number of iterations = 500.
- **GA:** 300 generations; population size = 50; tournament size = 20; mutation rate = 0.2; elitism rate = 0.2; crossover rate = 0.7.
- **ACO:** 50 ants; 200 iterations; evaporation rate = 0.5; $\alpha = 1$; $\beta = 3$.
- **ABC:** 50 bees; 1000 iterations; patience threshold = 50.

The parameter configurations adopted for the six metaheuristics were not selected arbitrarily. Prior to generating the algorithm-performance data, preliminary tuning experiments were conducted for each metaheuristic to identify parameter settings providing high-quality solutions, with tour lengths as close as possible to the known reference solutions. The configurations retained after this preliminary tuning stage were subsequently adopted as the nominal settings throughout the main experiments. Nevertheless, since the identity of the best-performing metaheuristic may remain sensitive to parameter choices, an additional configuration-sensitivity analysis was conducted around these nominal settings.

Each of the metaheuristics was applied to the entire set of generated TSP subinstances with the same stopping criteria. For each of the problems, the best tour length found by any of the metaheuristics was used to determine the best solver, thus facilitating the construction of a labeled meta-data set.

Table 2. Performance of metaheuristics on TSPLIB subinstances

Metaheuristic	Performance (%)
TS	34.10
SA	22.30
ABC	13.27
EFOA	11.53
GA	10.73
ACO	8.07

As can be seen in Table 2, there is a clear dominance of Tabu Search with (34.10%), followed by Simulated Annealing with (22.30%), and finally population-based methods like ABC and EFOA. This is in line with the complementary character of intensification-oriented trajectory-based heuristics and globally exploratory population-based heuristics. The resulting performance matrix is the basis for the labeled data set for training our proposed CNN-based algorithm selection approach.

4.3. Parameter-Configuration Sensitivity Analysis

To assess whether the algorithm-selection labels depend on the adopted metaheuristic configurations, an additional sensitivity analysis was conducted on a stratified subset of 300 TSP instances, with 30 instances selected from each of the ten TSPLIB source families. Each of the six metaheuristics was evaluated under three parameter profiles: a lower perturbation, the tuned nominal configuration used in the main experiments, and an upper perturbation. The purpose of this experiment was not to retune the metaheuristics, but to evaluate whether variations around the adopted nominal configurations modify the identity of the best-performing algorithm. For each instance and parameter profile, the metaheuristic obtaining the best solution was identified. Label stability was then computed as the percentage of instances for which the same metaheuristic remained the best-performing

Table 3. Sensitivity of the best-performing metaheuristic to parameter configurations.

Configuration comparison	Label stability (%)	Switch rate (%)
LOWER vs. NOMINAL	18.18	81.82
UPPER vs. NOMINAL	69.70	30.30
LOWER vs. UPPER	45.45	54.55

algorithm between two configurations, whereas the switch rate represents the percentage of instances for which the selected metaheuristic changed.

Table 3 shows that the identity of the best-performing metaheuristic is sensitive to parameter configuration. The LOWER configuration exhibits the greatest variation from the NOMINAL setting, with a label stability of 18.18%, whereas the UPPER configuration shows substantially greater agreement, with a stability of 69.70%. These results indicate that parameter configuration can substantially affect algorithm-selection labels. Consequently, the labels used in this study should be interpreted as the best-performing algorithms under the adopted nominal configurations rather than as parameter-independent ground truth.

4.4. Feature Extraction

For the purpose of ML-based comparison, 14 Edge and Vertex Metrics (EVM) features are computed for each instance, which summarize the degree-related and edge weight statistics, which are known for their strong prediction power [10]. In contrast, the deep learning-based approach works directly on the distance matrix, which is computed for each instance using the Euclidean distance between every pair of cities:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2},$$

thus providing a complete and lossless representation of the geometric structure of the instance.

It should be noted that, for the symmetric TSP considered in this study, $D_{ij} = D_{ji}$. The proposed CNN nevertheless receives the complete distance matrix and does not explicitly enforce this symmetry; thus, the matrix is processed as a general two-dimensional structured input. From an information perspective, either the upper or lower triangular part is sufficient to reconstruct the complete symmetric matrix and could provide a more compact representation. However, directly using a triangular representation would alter the regular two-dimensional structure exploited by the convolutional layers, and its effect on algorithm-selection performance therefore requires empirical evaluation. Future work could investigate symmetry-aware CNNs, triangular representations, or graph-based architectures that explicitly model pairwise city relationships while avoiding redundant matrix information.

4.5. Deep Learning Model Architecture

The proposed model is a two-dimensional Convolutional Neural Network (CNN) designed to operate directly on the distance matrix of each TSP instance. As illustrated in Figure 2, the network follows a hierarchical feature-extraction architecture composed of five convolutional blocks that progressively transform the input distance matrix into higher-level representations.

A normalized $N \times N$ Euclidean distance matrix is provided as input to the network. In the main experiments, $N = 100$, while the cross-size experiments additionally consider $N = 200$ and $N = 500$. The input is first processed by ConvBlock 1 and ConvBlock 2, which use 32 and 64 filters with 3×3 kernels, respectively. As illustrated in Figure 3a, each convolutional block consists of a two-dimensional convolutional layer, followed by batch normalization and a ReLU activation function. After ConvBlock 2, a max-pooling operation is applied to reduce the spatial dimensions of the feature maps.

The intermediate stage further refines the learned representation through ConvBlock 3 and ConvBlock 4, each using 128 filters with 3×3 kernels. A second max-pooling operation is then applied, followed by a dropout layer with a rate of 0.15 for regularization. ConvBlock 5 subsequently increases the number of filters to 256, providing a higher-dimensional representation before classification. Its output is regularized using a dropout layer with a rate of 0.20.

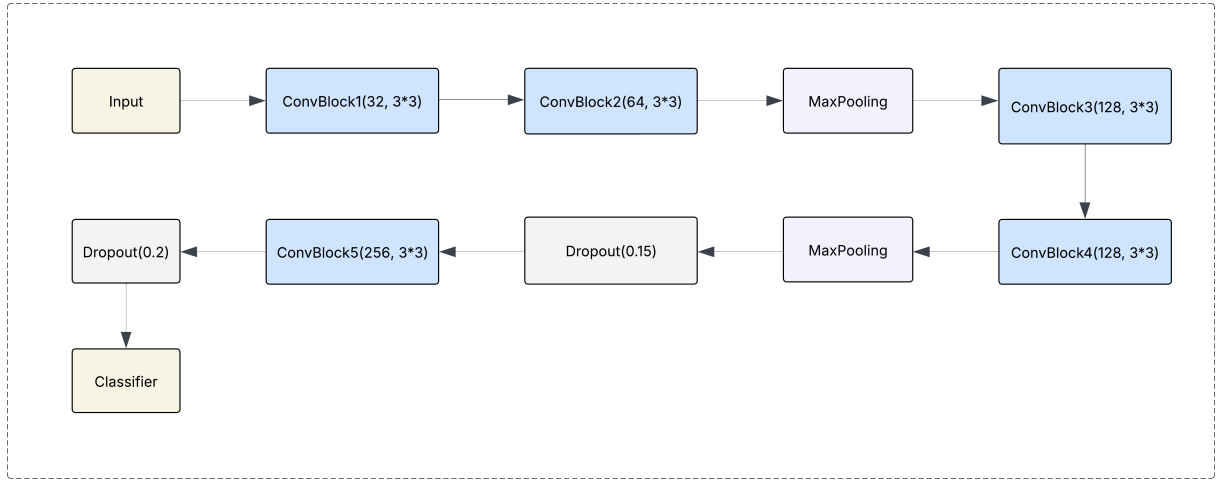


Figure 2. Architecture of the proposed convolutional neural network

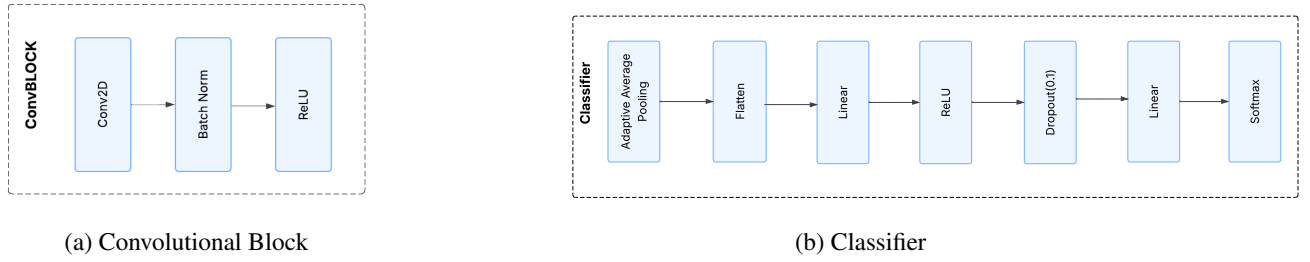


Figure 3. (a) The Internal structure of a convolutional block. (b) Classifier head of the proposed CNN model

The classifier head, illustrated in Figure 3b, first applies adaptive average pooling to obtain a fixed-dimensional representation independently of the spatial dimensions of the input distance matrix. The resulting feature map is then flattened and passed through a fully connected layer followed by a ReLU activation and dropout regularization. Finally, a fully connected output layer with a softmax activation produces the predicted class among the six candidate metaheuristics: TS, SA, ABC, GA, ACO, and EFOA. The adaptive pooling operation removes the direct dependence of the classifier dimensions on the input matrix size, allowing the same network architecture to process distance matrices of different sizes.

Overall, the architecture combines hierarchical convolutional feature extraction with a size-independent classifier head. Batch normalization is used to stabilize training, max-pooling progressively reduces the spatial dimensions of the feature maps, and dropout provides regularization. This design enables the model to learn solver-discriminative representations directly from TSP distance

5. Experimental Setup

All experiments were conducted on a Dell G16 7620 workstation equipped with a 12th-generation Intel Core i7-12700H processor, 16 GB of RAM, and an NVIDIA GeForce RTX 3060 GPU (6 GB). The system was running Windows 11 Pro (64-bit)

Regarding computational requirements, the construction of the labeled dataset represents a substantial offline computational cost. For the 3,000 TSP instances, six metaheuristics were evaluated with 10 independent runs per instance, resulting in 180,000 metaheuristic executions. Model training constitutes an additional offline cost, whereas prediction with the trained CNN is comparatively lightweight: the total prediction time for the 649 test instances was 0.214 s, corresponding to approximately 0.33 ms per instance. These offline costs should be considered when assessing the practical applicability of the framework, particularly when the trained selector is intended to be reused across multiple problem instances.

5.1. Distance Matrix Preprocessing

Before feeding the distance matrices to the CNN model, a Min-Max scaling was performed to normalize all data in the $[0,1]$ range. This was done by flattening each 100×100 matrix, normalizing using the statistics of the training set, and resizing it back to its original form. This provides all distance values with a consistent scale, thereby facilitating stable and efficient model training while preserving the structure of the corresponding TSP distance matrices.

Since the considered TSP instances are symmetric, the distance matrix satisfies $D_{ij} = D_{ji}$. In the present framework, the complete distance matrix is provided to the CNN, including both symmetric entries, and the network does not explicitly enforce or exploit this symmetry. Although one triangular part of the matrix would theoretically be sufficient to reconstruct the complete distance information, using such a representation would alter the two-dimensional organization processed by the current convolutional architecture. Therefore, the potential benefits of triangular or explicitly symmetry-aware representations require dedicated experimental evaluation and constitute a relevant direction for future work.

5.2. Performance Evaluation Metrics

The predictive performance of all models is assessed using standard multi-class classification metrics, namely Accuracy, Recall, and F1-score. Since computational efficiency plays a critical role in algorithm selection, we additionally report the total execution time on the test set as well as the average inference time per instance.

- **Accuracy:** Overall proportion of correctly classified instances:

$$\text{Accuracy} = \frac{\text{Correct predictions}}{\text{Total predictions}}. \quad (1)$$

- **Recall:** Ability to correctly identify instances of each class:

$$\text{Recall} = \frac{TP}{TP + FN}. \quad (2)$$

- **F1-score:** Harmonic mean of Precision and Recall, suitable for imbalanced settings:

$$P = \frac{TP}{TP + FP}, \quad F1 = \frac{2PR}{P + R}. \quad (3)$$

- **Execution time:** Total test-time cost and per-instance inference time, reflecting the practical suitability of the model.

These metrics jointly assess both the predictive accuracy and operational efficiency of the considered approaches.

6. Results and discussion

Table 4 summarizes the predictive performance of the proposed CNN. The model achieves an accuracy of **91%** and a F1-score of **0.91**, indicating strong and consistent selection capability across all six metaheuristics.

ABC, ACO, and EFOA exhibit the highest F1-scores (0.93), while TS obtains the highest recall (0.98). Performance on GA and SA remains competitive, with F1-scores of 0.87, confirming that the model captures solver-discriminative geometric patterns even for closely related search behaviors.

In terms of computational efficiency, inference is extremely fast: the total prediction time for all 649 test instances was **0.214 seconds**.

Table 4. Performance of the proposed CNN model

Metaheuristic	Precision	Recall	F1-score
ABC	0.95	0.91	0.93
ACO	1.00	0.87	0.93
GA	0.90	0.84	0.87
EFOA	0.95	0.91	0.93
SA	0.90	0.84	0.87
TS	0.85	0.98	0.91
Accuracy		0.91	
Precision		0.93	
Recall		0.89	
F1-score		0.91	
execution time		0.214 s	

Table 5. Confusion matrix of the proposed CNN for metaheuristic selection.

Actual \ Predicted	ABC	ACO	GA	EFOA	SA	TS
ABC	110	0	6	3	0	2
ACO	0	52	0	0	5	3
GA	3	0	54	0	1	6
EFOA	1	0	0	63	3	2
SA	0	0	0	0	110	21
TS	2	0	0	0	3	200

As shown in Table 5, the most pronounced misclassification occurs between SA and TS, with 21 SA instances classified as TS. Other notable errors include GA instances classified as TS and ABC instances classified as GA. The SA–TS confusion may partly reflect similarities in their trajectory-based search behavior, although the confusion matrix alone does not establish a causal relationship. From a practical perspective, not all misclassifications necessarily have the same optimization cost, since two competing solvers may achieve similar tour lengths. This motivates future investigation of probabilistic, cost-sensitive, or ranking-based algorithm-selection approaches rather than relying exclusively on hard classification.

6.1. Cross-Family Generalization Analysis

The LOFO results in Table 6 provide a more stringent assessment of generalization than the original random-split evaluation. The model achieves an average accuracy of 0.8233 ± 0.0225 and an F1-score of 0.8146 ± 0.0268 , with accuracy ranging from 0.8000 to 0.8667 across the held-out families. The decrease from 0.91 under the random split to 0.8233 under LOFO indicates that cross-family transfer is more challenging, likely due to structural differences among TSPLIB source families. Nevertheless, the model maintains at least 80% accuracy across all held-out families, showing useful predictive capability on unseen source structures. Overall, these results demonstrate that the model maintains consistent predictive performance across unseen TSPLIB source families.

Table 6. Cross-family algorithm-selection performance under the Leave-One-TSPLIB-Family-Out (LOFO) evaluation.

Held-out TSPLIB family	Accuracy	Precision	Recall	F1-score
d1655	0.8000	0.8111	0.8000	0.7952
fl1400	0.8333	0.8667	0.8333	0.8267
nrv1379	0.8000	0.8095	0.8000	0.8023
pcb3038	0.8333	0.8444	0.8333	0.8222
pr2392	0.8333	0.8343	0.8333	0.8322
rl1889	0.8333	0.8914	0.8333	0.8164
u1817	0.8667	0.8743	0.8667	0.8657
vm1748	0.8000	0.8325	0.8000	0.7979
fl1577	0.8333	0.8444	0.8333	0.8222
rat783	0.8000	0.8805	0.8000	0.7648
Mean $\pm$ Std	0.8233 $\pm$ 0.0225	0.8489 $\pm$ 0.0284	0.8233 $\pm$ 0.0225	0.8146 $\pm$ 0.0268

6.2. Cross-Size Generalization Analysis

The results in Table 7 provide additional insight into the scalability of the proposed framework beyond the 100-city setting. The model achieves an accuracy of 0.7533 and an F1-score of 0.7582 on 200-city instances. For 500-city instances, the accuracy decreases to 0.6275 and the F1-score to 0.6126, indicating that increasing the problem size introduces a more pronounced distribution shift. Nevertheless, the results show that the representation learned from smaller instances remains useful for larger problem sizes when limited size-specific adaptation is applied.

Table 7. Performance of the proposed framework on larger TSP instance sizes.

Instance size	Accuracy	Precision	Recall	F1-score
200 cities	0.7533	0.7655	0.7533	0.7582
500 cities	0.6275	0.6178	0.6275	0.6126

6.3. Effect of City-Index Permutation Augmentation

As shown in Table 8, city-index permutation augmentation preserves the overall predictive performance under the strict LOFO protocol. The mean accuracy remains unchanged at 0.8233, while precision slightly increases from 0.8489 to 0.8498 and F1-score from 0.8146 to 0.8184. These modest differences indicate that permutation augmentation does not substantially improve predictive performance, although comparable performance is maintained under the evaluated city re-indexings.

Table 8. Effect of city-index permutation augmentation on the learned distance-matrix representation under strict LOFO evaluation.

Setting	Accuracy	Precision	Recall	F1-score
Without permutation	0.8233 $\pm$ 0.0225	0.8489 $\pm$ 0.0284	0.8233 $\pm$ 0.0225	0.8146 $\pm$ 0.0268
With permutation	0.8233 $\pm$ 0.0135	0.8498 $\pm$ 0.0318	0.8233 $\pm$ 0.0105	0.8184 $\pm$ 0.0403

The results in (Table 9) show that the proposed CNN achieves the best overall performance among deep learning models trained on the raw distance matrix. With an accuracy and macro F1-score of **0.91**, it outperforms ResNet, EfficientNet, and LSTM, confirming that lightweight convolutional blocks are particularly well suited to capturing the geometric regularities encoded in distance matrices. ResNet remains competitive but slower, while EfficientNet

and especially LSTM show weaker generalization, indicating that deeper or sequential architectures are less adapted to this structured 2D representation.

Table 9. Comparison of deep-learning models trained on the raw distance matrix

Model	Input data	Accuracy	Precision	Recall	F1-score	Time (s)
Proposed CNN	distance matrix	0.91	0.93	0.89	0.91	0.214
ResNet	distance matrix	0.90	0.91	0.89	0.90	0.769
EfficientNet	distance matrix	0.87	0.89	0.86	0.87	0.125
LSTM	distance matrix	0.80	0.83	0.82	0.83	0.148

The results in (Table 10) indicate that deep learning models trained on image-based representations achieve significantly lower performance than those trained directly on the raw distance matrix. The best image-based model (CNN Images) reaches only **0.74** accuracy and **0.76** macro F1-score, compared to **0.91** accuracy for the proposed matrix-based CNN. ResNet, EfficientNet, and LSTM exhibit even weaker performance with images, reflecting the limitations of graphical representations that introduce information loss and unwanted variability (e.g., resolution, scaling, edge drawing).

Table 10. Comparison of deep learning models trained on image-based representations of TSP instances

Model	Input data	Accuracy	Precision	Recall	F1-score	Time (s)
CNN	Points + MST + kNNG images	0.74	0.78	0.74	0.76	0.176
ResNet	Points + MST + kNNG images	0.72	0.66	0.68	0.65	0.116
EfficientNet	Points + MST + kNNG images	0.67	0.65	0.64	0.64	0.141
LSTM	Points + MST + kNNG images	0.68	0.64	0.61	0.62	0.491

These results highlight the advantage of our contribution: using the distance matrix as input preserves the full metric structure of the TSP instance, enabling convolutional filters to extract fine-grained geometric patterns that are partially distorted or removed during image generation. The superior accuracy and robustness of the matrix-based CNN confirm the effectiveness of operating directly on raw distance information.

When comparing the proposed CNN with the feature-based machine-learning models, the results reveal a clear trade-off between predictive performance, computational efficiency, and representation requirements. As shown in Table 11, XGBoost, Random Forest, and SVM achieve a slightly higher test accuracy (0.93) than the proposed CNN (0.91). XGBoost also exhibits a substantially lower total prediction time on the test set (0.0075 s compared with 0.214 s for the CNN). The CNN prediction time of 0.214 s corresponds to the complete test set of 649 instances, equivalent to approximately 0.33 ms per instance. Therefore, the proposed CNN is not intended to outperform feature-based models in terms of raw inference speed or overall accuracy.

Table 11. Performance of machine-learning models trained on EVM meta-features. Prediction time corresponds to the complete test set.

Model	Input data	Accuracy	Precision	Recall	F1-score	Time (s)
XGBoost	EVM features	0.93	0.96	0.90	0.93	0.0075
Random Forest	EVM features	0.93	0.95	0.90	0.92	0.1875
Decision Tree	EVM features	0.84	0.83	0.85	0.83	0.00028
SVM	EVM features	0.93	0.95	0.90	0.92	0.0556

The main motivation for the proposed approach lies instead in its feature-free formulation. Conventional machine-learning models operate on a predefined vector of 14 EVM meta-features that must be computed before

prediction, whereas the proposed CNN learns directly from the raw TSP distance matrix. It therefore avoids dependence on handcrafted meta-features while maintaining competitive predictive performance.

It is also important to distinguish classifier inference time from the computational cost of the complete algorithm-selection pipeline. The prediction times reported in Table 11 correspond to classification over the test set once the EVM features are available and do not include their computation. Nevertheless, when these features are already available, XGBoost remains both slightly more accurate and faster. Thus, the main contribution of the proposed CNN lies in direct representation learning from distance matrices rather than superiority in inference speed.

6.4. Feature Importance Analysis

The feature importance results shown in (Figure 4) for Decision Tree, Random Forest, and XGBoost shows a consistent pattern across all models: prediction performance is driven mainly by a very small subset of EVM features. In particular, C_{min_E} and C_{min_V} exhibit the highest importance scores for all three algorithms, while most other features contribute only marginally. This confirms that machine learning baselines rely heavily on a few hand-engineered descriptors, resulting in a limited and biased representation of TSP instance structure.

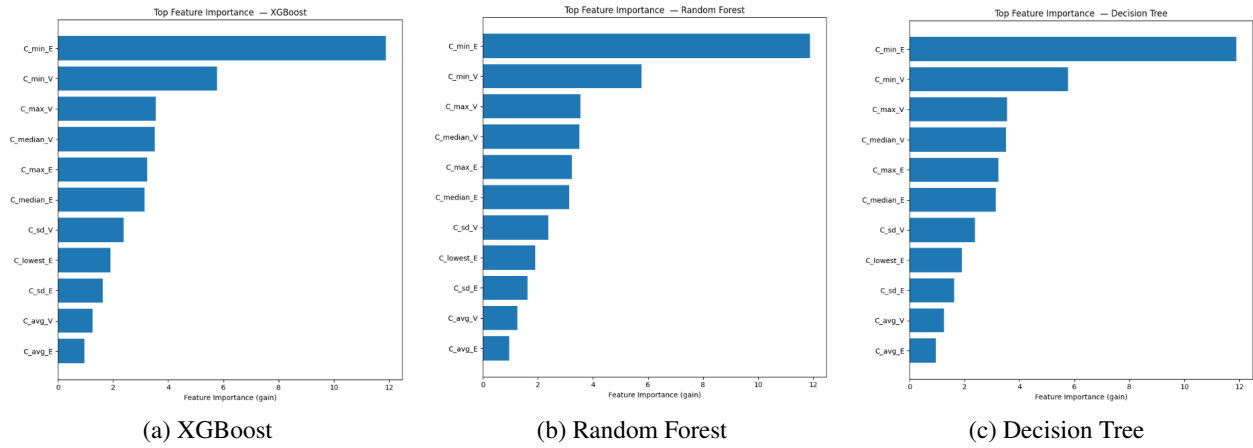


Figure 4. Feature importance analysis of the EVM meta-features for three machine learning models: XGBoost, Random Forest, and Decision Tree

The SVM with RBF kernel cannot provide meaningful feature-importance values because the RBF transformation maps the data into a high-dimensional nonlinear space, making the model’s decision function non-interpretable. This highlights the limited interpretability of kernel-based methods when relying on handcrafted features.

As can be seen, the results demonstrate some obvious limitations associated with the use of feature-based models. Although they are capable of providing competitive results on relatively small datasets, their expressiveness depends on the way in which the meta-features are created. In contrast, the CNN introduced in this paper does not rely on any manually engineered features and instead utilizes the full structure of the data provided in the distance matrix form.

Figure 5 demonstrates the results for all considered model classes and input representations globally. As can be seen, CNN using the distance matrix representation significantly outperforms all image-based deep learning methods and provides performance comparable to state-of-the-art machine learning models without relying on any hand-crafted features at all.

Table 12 provides a comparative overview of the Three families of approaches are considered in this study. Feature-based Methods rely heavily on handcrafted descriptors and thus require significant domain expertise, while image-based CNN approaches depend on complex visual encodings whose quality is sensitive to several problem-irrelevant parameters such as resolution or drawing conventions. In contrast, the proposed distance-matrix CNN offers a simpler and more direct representation of the TSP instance: no feature engineering or image construction

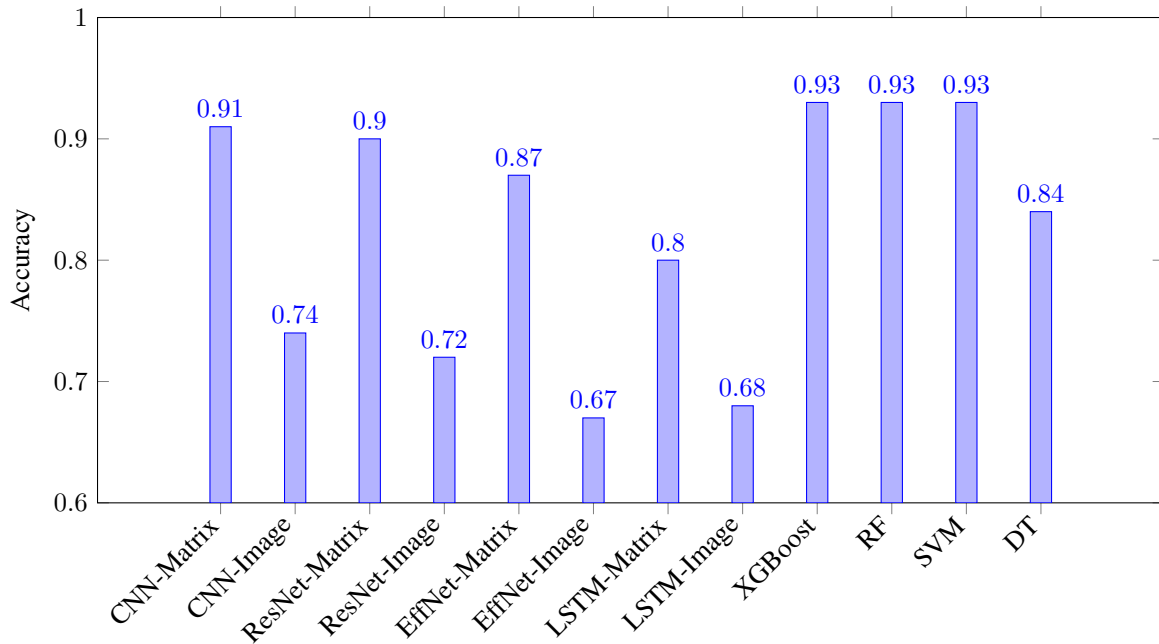


Figure 5. Global accuracy comparison across deep learning models (matrix-based and image-based) and machine-learning models (EVM meta-features)

Table 12. Comparative analysis of feature-based models, distance-matrix CNN, and image-based CNN approaches for algorithm selection

Aspect	Feature-based (EVM)	models	CNN (distance matrix)	CNN (image-based)
Input	Handcrafted meta-features		Raw $N \times N$ distance matrix	Images (scatter, MST, k-NNG)
Preprocessing	High (manual design)	feature	Very low (direct matrix input)	High (image construction pipeline)
Need for features	Yes		No	No
Sensitivity	Strong (scaling, choice)	feature	Moderate (matrix normalization)	High (resolution, color map, line width)
Information loss	High (summaries only)		Low (all distances preserved)	Moderate (projection/visualization choices)
Augmentation	Not applicable		Possible (permutations, symmetry)	Extensive (rotations, flips, resizing)
TSP variants	Hard (new features needed)		Moderate (matrix still valid)	Difficult (new images required)
Scalability	Good (fixed-size vector)		Moderate (matrix grows with N)	Limited (image size grows or loses detail)
Complexity	High (domain expertise)		Low	High (multi-step pipeline)

is needed, and the full pairwise metric information is preserved. This results in a substantially lower preprocessing

burden and reduces the risk of information loss, while maintaining good scalability and generalization across TSP instance types.

7. Conclusion and Future Work

This work proposed a feature-free CNN-based framework for TSP algorithm selection that operates directly on pairwise distance matrices, avoiding the need for handcrafted meta-features or image-generation preprocessing. Experiments conducted on 3,000 subinstances derived from ten TSPLIB source instances showed that the proposed distance-matrix CNN achieves an accuracy of **0.91**, outperforming the evaluated image-based deep-learning approaches while remaining competitive with feature-engineered machine-learning models, which achieve up to **0.93** accuracy. Under the stricter Leave-One-TSPLIB-Family-Out evaluation, the model achieved a mean accuracy of **0.8233** and an F1-score of **0.8146**, demonstrating predictive capability on source families excluded from training. These results support the use of pairwise distance information as a useful representation for learning-based TSP algorithm selection.

Several limitations also provide directions for future research. First, the current benchmark is derived from only ten TSPLIB source instances; therefore, future work should incorporate additional and structurally diverse TSP families to further assess generalization. Second, although the model can be applied to larger distance matrices, the cross-size experiments show decreasing performance as the instance size increases, motivating further investigation of size-robust representations and transfer-learning strategies. Future work should also investigate symmetry-aware or alternative matrix representations, larger and more diverse metaheuristic portfolios, and computational-budget normalization during solver labeling. Finally, hybrid architectures combining convolutional representations with attention-based or sequence-based mechanisms could be explored to further improve algorithm selection across heterogeneous TSP instances.

REFERENCES

1. Gilbert Laporte. The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2):231–247, June 1992.
2. Yusef Ahsini, Pablo Díaz-Masa, Belén Inglés, Ana Rubio, Alba Martínez, Aina Magraner, and J. Alberto Conejero. The electric vehicle traveling salesman problem on digital elevation models for traffic-aware urban logistics. *Algorithms*, 16(9):402, August 2023.
3. Marco Rinaldi, Stefano Primates, Martin Bugaj, Ján Rostáš, and Giorgio Guglieri. Development of heuristic approaches for last-mile delivery tsp with a truck and multiple drones. *Drones*, 7(7):407, June 2023.
4. H. Neil Geismar, Gilbert Laporte, Lei Lei, and Chelliah Sriskandarajah. The integrated production and transportation scheduling problem for a product with a short lifespan. *INFORMS Journal on Computing*, 20(1):21–33, February 2008.
5. Zulkifli Tahir, Nur Azman Abu, Shahrin Sahib, and Nanna Suryana Herman. Cnc pcb drilling machine using novel natural approach to euclidean tsp. In *2010 3rd International Conference on Computer Science and Information Technology*, volume 5, pages 481–485. IEEE, 2010.
6. David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 2002.
7. John R. Rice. *The Algorithm Selection Problem*, page 65–118. Elsevier, 1976.
8. Kate A. Smith-Miles. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys*, 41(1):1–25, January 2009.
9. Lars Kotthoff. *Algorithm Selection for Combinatorial Search Problems: A Survey*, page 149–190. Springer International Publishing, 2016.
10. Jorge Kanda, Andre de Carvalho, Eduardo Hruschka, Carlos Soares, and Pavel Brazdil. Meta-learning to select the best meta-heuristic for the traveling salesman problem: A comparison of meta-features. *Neurocomputing*, 205:393–406, September 2016.
11. Nikhil Ketkar and Jojo Moolayil. *Convolutional Neural Networks*, page 197–242. Apress, 2021.
12. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
13. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
14. Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.
15. Ifan Rizqa, Abdul Syukur, Aris Marjuni, and Nova Rijati. Deep reinforcement learning for generalizable multi-knapsack optimization in retail packaging. *Statistics, Optimization and Information Computing*, 16(4), August 2026.
16. Fakhita EDDAoudi, Halima LAKHBAB, and Mohamed NAIMI. Benchmarking metaheuristics for neural network optimization: A comprehensive study. *Statistics, Optimization and Information Computing*, 15(6):4922–4943, April 2026.

17. Sarah Hassan Awad Al-taei, Ban Hamed Al-miyahi, Melad Jameel Hamoud, Waleed Amin Mahmoud Al-Jawher, Hayder Najm, and Mohammed Salih Mahdi. A hybrid coa-cnn framework for credit card fraud detection using hyperparameter optimization and metaheuristic feature selection. *Statistics, Optimization amp; Information Computing*, 16(1):33–48, May 2026.
18. Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi. The traveling salesman problem. *Handbooks in operations research and management science*, 7:225–330, 1995.
19. Jérôme Monnot and Sophie Toulouse. The traveling salesman problem and its variations, July 2014.
20. Richard M. Karp. *Reducibility Among Combinatorial Problems*, page 219–241. Springer Berlin Heidelberg, November 2009.
21. Jorge Y. Kanda, Andre C.P.L.F. de Carvalho, Eduardo R. Hruschka, and Carlos Soares. Using meta-learning to recommend meta-heuristics for the traveling salesman problem. In *2011 10th International Conference on Machine Learning and Applications and Workshops*, page 346–351. IEEE, December 2011.
22. Jorge Y. Kanda, Andre C.P.L.F. de Carvalho, Eduardo R. Hruschka, and Carlos Soares. Using meta-learning to recommend meta-heuristics for the traveling salesman problem. In *2011 10th International Conference on Machine Learning and Applications and Workshops*, page 346–351. IEEE, December 2011.
23. M.B. Bossenbroek. Meta-learning using neural networks in large-scale traveling salesman problems. Master’s thesis, Erasmus University Rotterdam Erasmus School of Economics, 2023. Master’s thesis.
24. Pavel Brazdil, Jan N. van Rijn, Carlos Soares, and Joaquin Vanschoren. *Metalearning Approaches for Algorithm Selection I (Exploiting Rankings)*, page 19–37. Springer International Publishing, 2022.
25. Emma Hart, Quentin Renau, Kevin Sim, and Mohamad Alissa. *Evaluating the Robustness of Deep-Learning Algorithm-Selection Models by Evolving Adversarial Instances*, page 121–136. Springer Nature Switzerland, 2024.
26. Moritz Seiler, Janina Pohl, Jakob Bossek, Pascal Kerschke, and Heike Trautmann. *Deep Learning as a Competitive Feature-Free Approach for Automated Algorithm Selection on the Traveling Salesperson Problem*, page 48–64. Springer International Publishing, 2020.
27. Kangfei Zhao, Shengcai Liu, Jeffrey Xu Yu, and Yu Rong. Towards feature-free tsp solver selection: A deep learning approach. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.
28. Isaias I. Huerta, Daniel A. Neira, Daniel A. Ortega, Vicente Varas, Julio Godoy, and Roberto Asín-Achá. Improving the state-of-the-art in the traveling salesman problem: An anytime automatic algorithm selection. *Expert Systems with Applications*, 187:115948, January 2022.
29. Mohamad Alissa, Kevin Sim, and Emma Hart. Algorithm selection using deep learning without feature extraction. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO ’19*, page 198–206. ACM, July 2019.
30. Ya Song, Laurens Bliet, and Yingqian Zhang. Revisit the algorithm selection problem for tsp with spatial information enhanced graph neural networks. *arXiv preprint arXiv:2302.04035*, 2023.
31. Jorge Kanda, Andre Carvalho, Eduardo Hruschka, and Carlos Soares. Selection of algorithms to solve traveling salesman problems using meta-learning1. *International Journal of Hybrid Intelligent Systems*, 8(3):117–128, June 2011.
32. Agung Chandra and Aulia Naro. A comparative study of metaheuristics methods for solving traveling salesman problem. *International Journal of Information Science and Technology*, 6(2):1–7, 2022.
33. Bladimir Toaza and Domokos Esztergár-Kiss. A review of metaheuristic algorithms for solving tsp-based scheduling optimization problems. *Applied Soft Computing*, 148:110908, 2023.
34. Alexander G. Nikolaev and Sheldon H. Jacobson. *Simulated Annealing*, page 1–39. Springer US, 2010.
35. Manuel Laguna. *Tabu Search*, page 941–958. Springer Nature Switzerland, 2025.
36. Vittorio Maniezzo, Luca Maria Gambardella, and Fabio de Luigi. *Ant Colony Optimization*, page 101–121. Springer Berlin Heidelberg, 2004.
37. Seyedali Mirjalili. *Genetic Algorithm*, page 43–55. Springer International Publishing, June 2018.
38. Dervis Karaboga. Artificial bee colony algorithm. *scholarpedia*, 5(3):6915, 2010.
39. Ranjeet Kumar Ranjan and Vijay Kumar. A systematic review on fruit fly optimization algorithm and its applications. *Artificial Intelligence Review*, 56(11):13015–13069, March 2023.
40. Lan Huang, Gui-chao Wang, Tian Bai, and Zhe Wang. An improved fruit fly optimization algorithm for solving traveling salesman problem. *Frontiers of Information Technology & Electronic Engineering*, 18(10):1525–1533, 2017.