algorithms

MDPI

Article

A New Approach to Efficiently Solving the Traveling Salesman Problem (TSP) by Combining Artificial Intelligence Techniques and Ant Colony Metaheuristics

Baudoin Nguimeya Tsofack [1], Garrik Brel Jagho Mdemaya [2,*], Milliam Maxime Zekeng Ndadji [3], Maxwell Ndognkom Manga [1] and Mthulisi Velempini [2]

[1] Department of Computer Science, University of Yaounde I, Yaounde P.O. Box 337, Cameroon; baudoinguimeya@gmail.com (B.N.T.); manga.maxwell@univ-yaounde1.cm (M.N.M.)
[2] Department of Computer Science, University of Limpopo, Polokwane P.O. Box 0700, South Africa; mthulisi.velempini@ul.ac.za
[3] Department of Mathematics and Computer Science, University of Dschang, Dschang P.O. Box 96, Cameroon; ndadji.maxime@univ-dschang.org
* Correspondence: jaghobrel@gmail.com

Abstract

The efficient resolution of complete NP problems, such as the Traveling Salesman Problem (TSP), particularly for large instances, remains a major challenge in operations research and combinatorial optimization, especially for many businesses, particularly in sectors such as logistics, urban planning, and networks, where efforts are made daily to optimize routes and delivery times. Optimization methods inspired by collective behavior, such as Ant Colony Optimization (ACO), offer competitive results for solving these types of problems. The main problem is the size of the instances because, when it becomes large, many existing algorithms fail to converge to a good solution within a reasonable timeframe: the execution time is generally very long, and the solution obtained is generally far from being the optimal solution to the problem. In this article, we propose a new way of approaching the resolution of the TSP through new metaheuristics inspired by artificial intelligence techniques and ant colony theory. To evaluate the effectiveness of our methodology, particularly the Multi-colony Ant Colony Optimization version 2-SK (MACOV2SK) method, simulations were performed on several instances of the TSP, focusing on large-scale instances. The experimental results clearly demonstrate that the proposed approach significantly improves upon several other approaches in the literature in terms of execution time and solution quality, especially for large-scale problems.

Keywords: operational research; multi-colony ant colony optimization; K-means; traveling salesman problem

Academic Editors: Frank Werner and Alicia Cordero

Received: 23 January 2026
Revised: 4 May 2026
Accepted: 6 May 2026
Published: 6 July 2026

1. Introduction

Artificial intelligence has become indispensable to many scientific and engineering disciplines, notably combinatorial optimization, where its ability to handle complex NP-hard problems offers considerable advantages [1]. Among these problems, the TSP is a well-known and widely studied problem, due to its theoretical importance and numerous practical applications, such as logistics, network routing, planning [2,3], and wireless networks, which also constitute a major application area for the TSP. The TSP is to determine a minimal-cost route that visits each city exactly once before returning to the starting

point, given a set of cities and the distances between each pair of cities. This problem is fundamental both from a theoretical point of view, as it is an NP-hard problem representing an archetype of challenges associated with computational complexity, and from a practical point of view, as it models many real-world situations where a route needs to be optimized, whether in logistics, mobile robotics, wireless network planning, or resource management in distributed systems [3]. Effectively solving the TSP is therefore crucial to reducing costs, improving system performance, and ensuring scalability in environments with a very large number of points to visit. It comes in several variants, including symmetric, asymmetric, time window, and stochastic [4]. In this work, we focus specifically on the symmetric version, where the distance between two points is identical in both directions. This choice is justified by the fact that many real-world systems, particularly in wireless networks, are based on Euclidean or radio-electric distances that are inherently symmetric, making this variant particularly relevant and representative while offering a more stable mathematical structure for the design of large-scale algorithms [5,6]. Wireless networks are indeed a major area of application for the TSP. For example, in a large-scale wireless sensor, a data mule or collector drone must visit all nodes in the network to retrieve locally stored data. The collector's movements follow an optimized route: poor planning results in excessive energy consumption, significant latency in data collection, and even loss of information. In a typical scenario, a drone operating in a smart agricultural area must visit several hundred or even thousands of sensors scattered across a large area. The goal is to minimize the total distance covered while ensuring regular passage times to meet data freshness constraints. This type of application illustrates the value of TSP on very large instances where exact methods are not feasible and only heuristic or metaheuristic approaches remain feasible.

In the past, multiple heuristic approaches have been proposed to address the TSP, including ant colony methods [7], advanced local searches [8,9], evolutionary algorithms [10,11], and hybrid techniques [12,13]. However, with the rapid growth in the number of instances to be processed, these approaches have limitations. On the one hand, several methods aim to reduce computation time by simplifying the search space, but the quality of the solutions obtained is often suboptimal. On the other hand, techniques capable of achieving near-optimal solutions, such as advanced versions of ACO or Lin–Kernighan (LK)-type heuristics [14], require significant computational resources and struggle to converge when the number of cities becomes very high. This duality between speed and quality poses a major problem for applications that require both scalability and accuracy. To overcome these limitations in this paper, we propose a hybrid approach that combines k-means clustering and a new multi-colony version (named Multi-Ant Colony Optimization version 2—MACOV2) of the ACO algorithm based on Multi-Ant Colony Optimization (MACO) [2]. The general idea is to use k-means to efficiently divide the space into coherent sub-regions, before using several ant colonies working in parallel to explore these sub-regions and locally optimize the routes. Combining the two paradigms significantly accelerates convergence while maintaining a high-quality solution thanks to a richer exploration of the search space. A global merging and improvement strategy then ensures the consistency and optimality of the final route. The experimental evaluations that we conducted on large instances show that our approach offers an effective compromise between computation time and solution quality, surpassing traditional ACO methods in both accuracy and scalability. These results demonstrate the relevance of a hybrid approach structured by spatial clustering for large-scale combinatorial optimization. The remainder of this paper is structured as follows: Section 2 presents a literature review on solving the TSP. Section 3 details the proposed hybrid architecture. Section 4 analyzes the experimental results and discusses their significance. Finally, Section 5 concludes this work.

2. Literature Review

The TSP has already been the subject of intense research. In recent years, several approaches have been proposed to solve the TSP. Among these approaches, there are exact methods that calculate all possible permutations to find the optimal path. Unfortunately, because of the exponential complexity, exact methods are only used for very small problems (less than 15 instances) because, for large problems, these methods can take billions of years to provide the optimal solution, which is computationally infeasible [15,16]. In addition to exact methods, there are heuristic and metaheuristic methods such as the ones by Habib et al. [17], Yousefikhoshbakht et al. [18] and Almufti et al. [19] that are commonly used to solve the TSP. These methods provide solutions more quickly but do not guarantee anything about the optimality of their output. Linear programming techniques have also been used to solve the TSP [20]. This approach has been very effective and is still used in many solutions, for example, the Concorde solver for the TSP [21], which helped to find optimal solutions for all reference instances of the TSPLIB library. Thanks to the multiple research studies carried out on heuristic methods, several have shown their effectiveness in finding solutions to the TSP, including genetic algorithms, ant colony algorithms, and hybrid heuristics [5]. Here we have focused particularly on the ant colony heuristic. It is a distributed approach that belongs to a family of bio-inspired metaheuristics [6].

Ant colony algorithms use agents called artificial ants that mimic the behavior of real ants [5]. Although practical and relatively fast for most optimization problems, including the TSP, these algorithms depend on the parameters and the size of the instance [5]. Among its different versions, Ant System (AS), which is the very first ant colony algorithm to solve the TSP, was developed by Dorigo et al. [22]. Unfortunately, its purely probabilistic choice makes it inefficient in terms of diversification, thus favoring the risks of premature convergence. To improve the performance of AS, Dorigo and al. [2] modified the AS transition rule to provide a better balance between exploring new areas of the resolution space and exploiting accumulated knowledge about the problem to solve it. To this end, they introduced a new rule that depends on a parameter q_0 ($0 < q_0 < 1$). It is a randomly generated number, uniformly distributed in (0,1). With this version, the choice of the next cities is no longer only probabilistic but also based on the parameter q_0. The resulting algorithm is called ACO [2] and has been effective for small and some medium-sized instances.

In [2], the authors developed a new variant of the ACO heuristic called MACO that uses several colonies. This variant has given very satisfactory results in execution time and solution quality for most medium-sized instances (less than 100) and some large instances. Ref. [23] proposed an approach based on k-means clustering and the Shuffled Frog Leaping Algorithm (SFLA), which it uses to solve the TSP. The proposed approach consists of three parts: separating cities into k clusters, finding the shortest path for each cluster, and merging the clusters. Experimental results have shown that the algorithm performs better as the number of clusters increases for problems with a large number of cities.

In [24], Gozde et al. designed a highly efficient optimized algorithm for the TSP, using GPU parallelization, machine learning, and artificial intelligence. The proposed algorithm consists of three steps: grouping points in the data set using the k-means algorithm, finding the shortest path within each group using the ant colony algorithm, and connecting the groups to each other at the nearest point. In addition to this approach, hybrid and parallel approaches have also been developed to strengthen the effectiveness of the ant colony heuristic in terms of exploiting solutions in solving the TSP, such as Multi-Ant Colony Optimization–Lin–Kernighan–Helsgaun (MACO-LKH) and the ant colony system-genetic algorithm (ACS-AG) [2].

These approaches provide good solutions, but the execution time is still high. Today, with artificial intelligence, route planning and optimization can be carried out much more efficiently, as shown by several works in the literature [6]. Indeed, the incorporation of machine learning techniques for the analysis or resolution of combinatorial problems is observed through its results. Bengio et al. [5] argue that the integration between machine learning and combinatorial optimization should be further pushed and outline a methodology to achieve this. A central idea of their work is to treat generic optimization problems as data instances and to investigate which underlying distribution of problems is the most relevant for training a model tailored to a specific task. In [6], the authors combined machine learning techniques with classical optimization methods. In their paper, they proposed improvements to reduce the computational complexity of their initial deep learning model. Furthermore, they explored the possibility of adding a local search phase to further improve performance and reduce execution time.

The authors in [25] designed an algorithm that takes advantage of ACO to solve, improve overall performance, and shorten the solution time in the TSP. They used techniques such as clustering on ACO parameters, dynamic pheromone evaporation, and diversity of solutions in the population. To observe the working performance, a study was carried out on the problems with the number of nodes varying from 51 to 2392, and it was observed that the proposed method was efficient, but the time complexity was too high.

The literature for resolving the TSP in particular proposes several approaches, each with advantages and limitations, which we can summarize as follows: (1) The weakness of exact methods, limited by their exponential complexity, has led researchers in the field to explore heuristics and metaheuristics. (2) Among heuristics, ACO is the most widely used for solving the TSP, followed by genetic algorithms and then hybrid heuristics. ACO has evolved from the initial ant system to a multi-colony ant system and hybrid approaches, increasingly integrating clustering, parallelization, and machine learning with heuristics to balance exploration and exploitation of the search domain [26]. Regarding the integration of clustering with ACO-based heuristics, its main objective is to transform an NP-hard problem of size N into several subproblems of size N/k in order to reduce the complexity of the exploration process. This is the case in the work of [26], which demonstrated that the use of K-means reduces the combinatorial complexity of ACO, making the search for local solutions extremely fast. Despite the advantages of clustering integration techniques with heuristics, these methods often suffer from a loss of optimality during merging if it is not performed correctly. Indeed, reconnecting the different solutions to obtain the global solution is often the weak point of these clustering-based techniques.

As for the integration of machine learning with heuristics, it has opened new perspectives for ant guidance. Ref. [27] used reinforcement learning to dynamically adjust ACO parameters during execution, thus improving convergence. The main drawback of these approaches is that they require significant hardware resources for model training. Regarding hybridization between ACO and the genetic algorithm (GA), the addition of genetic operators within ACO has stabilized the search. Middendorf [28] was among the first to integrate crossover operators to maintain the diversity of the ant population. Unfortunately, the use of an inappropriate crossover operator (order crossover, for example, in the case of the TSP) broke the optimal edges found by ACO, thus weakening the optimal solution and highlighting the importance of choosing the right crossover operators for this type of hybridization. Although all these strategies demonstrate remarkable performance on large instances, these modern methods still face a trade-off between solution quality and time complexity.

Building on this literature, the MACOV2SK method addresses the following shortcomings: It solves large instances more easily thanks to K-means clustering, thus enabling the more efficient handling of TSP instances where classical MACO would stagnate. The use of the ERX operator as the crossover operator in MACO preserves edge connections, and the cluster-merging strategy, based on the K-OPT improvement heuristic, significantly improves the process of obtaining the overall optimal solution. Finally, compared with ML-based approaches, MACOV2SK offers a lighter, training-free alternative, eliminating the need for large resources on large instances while ensuring good performance regardless of instance size.

3. Contribution

In this section, we present a new hybrid approach combining the automatic classification algorithm k-means and an ant colony heuristic, MACOV2, which is an improvement of the well-known MACO heuristic [2], for solving the TSP and other complex problems in computer science, especially for large instances. Our approach is broken down into three steps: first, we use the K-means method, which divides large problem instances into smaller subproblems or clusters of K; second, we use the highly efficient MACOV2 heuristic for solving medium-sized instances to effectively resolve each cluster or subproblem resulting from the clustering; and finally, we combine the solutions obtained from the different clusters to find the solution to the original problem.

3.1. Presentation of the Various Used Algorithms

3.1.1. Presentation of MACO

MACO [2] is one of the most recent and effective variants of the ant colony heuristic, using multiple colonies. The idea is based on the fact that as ants move within a colony, they leave pheromone trails that mark their passage along a given path. Pheromones are very short-lived hormones and are easily recognizable among ants of the same colony. When a food source is found, the path leading to it is more heavily impregnated with pheromones than others. Gradually, all the ants become aware of this path and attempt to shorten it. Modifying the parameters exposes the colony to the risk of premature convergence. MACO significantly improves on previous versions in the literature by modifying strategies for pheromone updating and ant movement. The MACO steps are as follows.

Step 1: Initializing the Algorithm

The modeling of ant behavior in MACO [2] is as follows: Initially, the ants from the different colonies are placed on the vertices of the graph (i.e., on each city). They move from one vertex to another using the edges of the graph. There are no pheromones on the edges. The global pheromone matrix is initialized with zero values for each pair of nodes. The ants move randomly per colony and construct their first tour. At the end of this first experiment, the global communication matrix (pheromone) is initialized with the best solution from the different colonies.

Step 2: Route Construction

During the different iterations, each ant agent within the different colonies possesses the following characteristics:

- The ant deposits a trace of pheromones on edge (i, j) when it moves from city i to city j;
- It chooses the destination city according to a probability that depends on the distance between this city and its position and the quantity of pheromones present on the edge (transition rule);

- In order to pass through each city only once, the ant cannot go to a city it has already passed through; therefore, the ant must have a local memory. In the case of MACO, each colony is equipped with a local memory, and the local memory specific to each ant agent disappears in favor of a global memory specific to each sub-colony.
- To prevent an ant from retracing its steps, it keeps a list of the cities it has already visited. This list, called a taboo list, is reset each time the ant completes a tour. This taboo list constitutes the ant's memory.

Step 3: Updating the Pheromone Matrix

Another feature of MACO is its pheromone trail update strategy, which is slightly different from previous versions of the ACO algorithm. Pheromone trails are modeled by the variables $\tau_{ij}(t)$, which give the intensity of the trail along (i, j) at time t. For the TSP, the information contained in the pheromone trail is based on the frequency with which ants choose to visit city i followed by city j. The transition probability $p_{ij}^{k,l}$ of ant k from vertex i to vertex j is given by Equation (1):

$$p_{ij}^{k,l}(t) = \begin{cases} \frac{[\eta_{ij}]^{\beta}[\tau_{ij}(t)]^{\alpha}}{\Sigma u \in v_{k_l}[\eta_{iu}]^{\beta}[\tau_{iu}(t)]^{\alpha}} & \text{if } j \in v_{k_l} \\ 0 & \text{else} \end{cases} \quad (1)$$

where v_{k_l} represents the list of cities to be visited by the ants of colony l, η_{ij} represents a measure of visibility that corresponds to the inverse of the distance between cities i and j, and t_{ij} represents the quantity of pheromone present at stop i, j at time t. α and β are two parameters used to modulate the relative importance of pheromones and visibility.

The pheromone update is performed once all the ants have visited all the cities and is described by Equation (2):

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \triangle\tau_{ij} \quad (2)$$

The MACO approach is therefore presented by Algorithm 1.

3.1.2. Presentation of Modified MACO (MACOV2)

MACOV2 is an improvement of the MACO heuristic [2] that aims to improve the diversification of the solution space, reduce the risks of premature convergence, and improve solution quality. This improvement in the diversification process will significantly reduce the risks of the local optimum trap of the MACO method, as well as premature convergence, and to a lesser extent, improve the exploitation process and the quality of the optimal solution compared with the MACO method. Recall that the effectiveness of a metaheuristic lies in its ability to strike an ideal balance between good diversification of the search domain and effective exploitation of the solutions within that domain. In the MACO method, when an ant k wants to move from one city (or node) i to another city j, it uses the probabilistic formula of Equation (1) to optimize the search domain and accelerate convergence. However, there can be a risk of premature convergence for large instances. To reduce the risk of premature convergence and improve the exploration process a modification to the original algorithm is proposed as follows.

The innovation of MACOV2 lies in the following: it replaces the stochastic and blind component that the classical MACO algorithm uses to augment the search with a crossover operator derived from genetic algorithms. This crossover operator is the Edge Recombination Crossover (ERX), which is efficient for problems like the TSP. This substitution significantly transforms the search mechanism, as it is now able to guide the search by synthesizing the characteristics of the best existing solutions to discover new avenues or promising descendants of the search space. These key transformations

will address the following limitations: thanks to K-means clustering, large instances are more easily solved, thus enabling more efficient management of TSP instances, where the classical MACO method had weaknesses; the integration of the ERX operator as a crossover operator in MACO preserves edge connections; and the cluster-merging strategy, based on the K-OPT tower improvement heuristic, significantly improves the process of obtaining the global optimal solution. Finally, compared with machine learning-based approaches, MACOV2SK will offer a lighter, untrained alternative, eliminating the need for significant resources for large instances while guaranteeing good performance regardless of instance size.

Algorithm 1: MACO algorithm.

Input: m: number of ants per colony;
L: Number of colonies;
n: number
of cities;
Begin
    Nc: = 0 ; // cycle number initialization
    D_{ij}: = 0 ; // Global matrix initialization
    d_{ij}: = t0 ; // local matrix of pheromones initialization
    Initialize t0 ; // with each ant's home city
    Place each colony randomly in a starting point (city);
    for $k = 1$ *to* m **do**
        Construction of a tour by each ant at random;
        Gradual deposition of pheromones in the d_{ij} matrix of each colony;
        Evaluation and selection of the best colony;
    Updated the global D_{ij} pheromone matrix with the best colony ;
    while $N_C < N_{max}$ **do**
        for $i = 1$ *to* n **do**
            for $j = 1$ *to* l **do**
                for $k = 1$ *to* m **do**
                    Select the city Vi to be added to the current tour with Formula (1);
                    Evaluate the solution of the ant K on route (i,j);
                    Update globally the trace according to city pair (i,j) if it's better than the previous ants according to formula of evaporation;
                    Progressively insert the path (i, j) into the taboo list to construct the solution of the ant K;
        N_C ++;
    S= $Best_{solution}$; // Each colony provides a partial solution

Based on algorithm parameters at Table 1, the results in Table 2 present a comparative study of the MACOV2 and MACO methods in terms of execution time and solution quality on several TSP instances. A more in-depth analysis of these results will be presented in the Results Section. The MACOV2 approach is described in Algorithm 2.

Visibility η (from Equation (1) depends on the distance between i and j. The choice of the next city is not purely stochastic or probabilistic but rather a hybrid of the two. The communication matrix is updated dynamically. This version has been tested and compared with MACO and is more efficient. The disadvantage of this version is that like MACO, it is very sensitive to variations in its parameters [20,29].

Table 1. Algorithm parameters.

Parameter	Role	Value
Initial quantity of pheromone	Information on the quality of a lead	0
Number of colonies	Represents the number of colonies in the heuristic	K
q_0	Rate of intensification and diversification	0.9
$1-\rho$	Pheromone evaporation rate	0.85
N_{max}	Maximum number of iterations	100
α	Pheromone intensity control	0.95
β	Visibility control	1
ϵ	MACOV2 pheromone update parameter	0.1

Table 2. Results obtained by MACOV2 on small, medium, and large TSP instances.

Instance	MACOV2		
	Best Sol.	**Runtime (s)**	**Average Sol.**
eil51 (426)	426	0.63	426.02
eil71 (538)	538	2.36	538.6
kroA150 (26,524)	26,524	10.35	26,694.82
kroB150 (26,130)	26,130	14.02	26,232.05
d198 (15,780)	15,780	37.9	15,802.58
tsp225 (3916)	3916	20.55	3916.04
A280 (2579)	2579	14.80	2579.2
Lin318 (42,029)	42,029.48	108.55	42,045.4
pr439 (107,217)	107,217	88.5	108,129.6
pcb442 (50,778)	50,778.01	27.04	50,781.22
rat575 (6773)	6773.9	77.04	6786.22
d657 (48,913)	48,913.9	62.84	49,702.09
d1655 (62,128)	63,616	996.2	64,133.5
pr2392 (378,032)	378,032.5	1809.2	387,756.32

Algorithm 2: MACOV2 algorithm.

Input: m: number of ants per colony;
L: Number of colonies;
N_{max}: maximum number of cycles;
j: number of parents to cross;
n: number of cities;

Begin
- Nc: = 0 ; `// cycle number initialization`
- D_{ij}: = 0 ; `// Global matrix initialization`
- d_{ij}: = t0 ; `// local matrix of pheromones initialization`
- Initialize t0 ; `// with each ant's home city`
- Place each colony randomly in a starting point (city);
- **for** $k = 1$ *to* m **do**
  - Construction of a tour by each ant at random;
  - Gradual deposition of pheromones in the d_{ij} matrix of each colony;
  - Evaluation and selection of the best colony;
- Updated the global D_{ij} pheromone matrix with the best colony;
- **while** $N_C < N_{max}$ **do**
  - **for** $i = 1$ *to* n **do**
    - **for** $j = 1$ *to* l **do**
      - **for** $k = 1$ *to* m **do**
        - Select the city Vi to be added to the current tour with Formula (1);
        - Evaluate the solution of the ant K on route (i,j);
        - Update globally the trace according to city pair (i,j) if it's better than the previous ants;
        - Progressively insert the path (i, j) into the taboo list to construct the solution of the ant K;
        - randomly choose j pairs of solutions and cross them to define a new ant;
  - N_C ++;
- S= $Best_{solution}$; `// Each colony provides a partial solution`

3.1.3. Presentation of the K-Means Algorithm for the TSP

The clustering of K-means is an unsupervised learning algorithm that groups unlabeled data into different groups. It is one of the most common statistical data analysis methods, used to establish a relationship regarding the structure of available data [24]. It is an iterative algorithm that divides unlabeled data into K distinct groups, such that each data set belongs to only one group with similar properties. The letter K defines the predefined number of clusters to be created. Automatic classification consists of grouping, in an unsupervised manner (without prior intervention by an expert), a set of objects or, more broadly, data, in such a way that the objects of the same group (called a cluster) are closer (according to a chosen similarity/dissimilarity criterion) to each other than those of other groups (clusters). The k-means algorithm identifies a number of centroids in a data set. It calculates the arithmetic mean of all data objects belonging to a cluster. Each data point is assigned to its nearest cluster. The algorithm aims to minimize cluster sizes. Simultaneously, other clusters are kept as diverse as possible. In the case of the TSP, the k-means algorithm randomly initializes several cluster centers on the graph of cities. Each point is assigned its nearest cluster center with each execution of the algorithm.

For our study, the choice of the K-means algorithm is primarily justified by two reasons: (1) It is less expensive in terms of average complexity, which is O(n), than other algorithms. This is very important and must be taken into account for problems like the TSP, unlike the hierarchical clustering method, where the complexity is $O(n^2)$. (2) The K-means technique tends to create clusters of more compact or balanced sizes, which facilitates local problem solving. Regarding the choice of K values, we used the elbow method. We justify our choice with the following reasons: (a) The elbow method is faster than the Silhouette scoring method or the Davies–Bouldin Index method. (b) The elbow method is better suited for large data sets such as the TSP. (c) In terms of interpretation, the method produces a very intuitive visual graph and is also simple to implement.

3.1.4. Critical Analysis of the Elbow Method

- The main drawback of the elbow method lies in its visual ambiguity. It is sometimes difficult to pinpoint the exact point of curvature, unlike the Silhouette scoring method, which provides a clear numerical value between -1 and 1 to designate the best K (the maximum). The elbow method, however, relies on the analyst's subjective visual assessment.
- The elbow method is quick but subjective; it only measures compactness.

3.1.5. Impact of K on the Solution in MACOV2SK

- Execution time: The total time increases rapidly with the number of clusters; increasing K would reduce computation time.
- (Optimal) solution quality: If K is too small, sub-segmentation occurs, resulting in the loss of specific information. If K it is too large, overfitting occurs, with each point becoming almost its own cluster. Consequently, there is a fragmentation threshold beyond which the overall solution degrades significantly.

An illustrative example is given in Figure 1, and the K-means method is described in Algorithm 3.

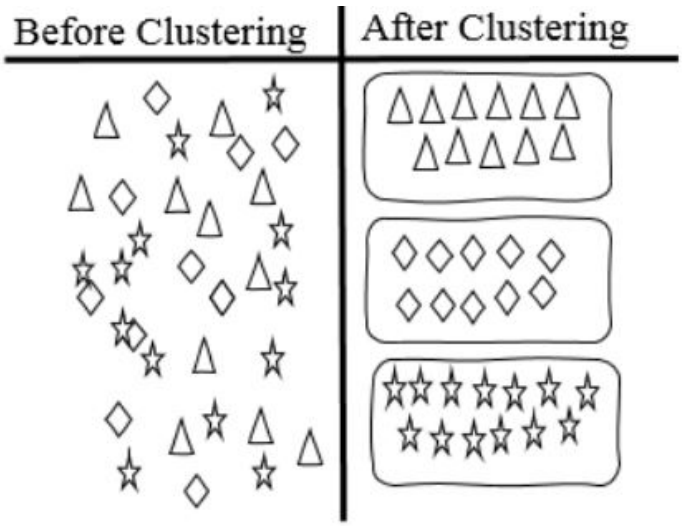

Figure 1. Clustering [24].

Algorithm 3: Classical K-means algorithm.

Input: m: number of ants per colony;
n: number of cities;
Output: S: the optimal solution;
Begin
    Select K initial elements as the centers of the K groups;
    Place the objects in the group with the closest center;
    Recalculate the center of gravity of each group;
    Iterate the algorithm until the objects no longer change groups;

3.2. Implementation of the Hybridization Approach (MACOV2SK)

The MACOV2SK heuristic is a hybrid approach that combines a metaheuristic ACO and an automatic classification algorithm (K-means) to address complex computer science problems such as the TSP. The basic idea of this strategy is to use the MACOV2 method which is efficient in solving small and medium-sized instances to solve large instances by subdividing them into smaller subproblems using the K-means method and then combining all the solutions. This is a low-level hybridization because the result of one method is used as input for another. The solution we implement is divided into three detailed steps as follows.

3.2.1. Phase 1: Division of the Initial TSP Instance Using the K-Means Algorithm

For a given TSP instance of size N, we transform the N-sized problem into K subproblems by applying the K-means method. This step aims to subdivide the initial N-sized problem into K subproblems of varying sizes. The implementation of this algorithm will be done in Python 3.12.12 using the Sklean library. Since there a k groups, then the selection of the center of each group is done in O(k); thereafter, the assignment of data (n cities) to the nearest cluster is done in O(n); finally, recalculating the center of gravity for each group is done in O(T). Hence, the average complexity of this function is O(k.n.T), where: K is the number of clusters, n is the number of cities, and T is the number of iterations. The cluster merging method is displayed on Figure 2 and the graph for the elbow method is shown in Figure 3, and the steps of the elbow method are as follows:

- It performs K-mean clustering on a data set for different values of K (K ranges from 1 to 10);
- For each value of K, it calculates the WCSS distance;
- It plots a curve that shows how the calculated WCSS values correspond to the number of clusters;
- The sharp point on the curve resembles the elbow, and this point is considered the best value of K.

WCSS is the sum of the distances between each data point and its squared centroid in each cluster. att48 is the title of the graph and represents the instance from which we obtained this graph. It should be noted that for all the tested instances, the elbow method was applied to obtain the optimal number of clusters K.

3.2.2. Phase 2: Resolving the TSP on Each Cluster Using MACOV2

Since the MACO or MACOV2 heuristics are effective in solving problems like the TSP for small and medium-sized instances, we will leverage this strength to find the optimal solution for each cluster by simulating pseudo-parallelism. Specifically, this involves considering each cluster obtained by the K-means algorithm as a separate instance of the TSP and then solving each sub-instance using the MACOV2 heuristic. The parameters used

by MACOV2 are presented in Table 2. For this function, its average complexity is O(k.n.m), where: K is the number of colonies, m the number of cities, and n the number of ants per colony.

3.2.3. Phase 3: Grouping of Solutions

To obtain the overall solution after dividing the TSP into several groups using K-means, the reconnection is performed according to the following method (Figure 2):

- First, we determine the connection order of the clusters: indeed, the different clusters of MACOV2SK are not connected randomly. We treat each cluster center as a unique city and solve the whole as a TSP. This defines the overall order of cluster visits (for example, Cluster 1–Cluster 3–Cluster 2...).
- Next, we proceed with the actual cluster reconnection: At this stage, we use the edge breaking and reconnection technique and then re-optimize the final solution using the K-opt improvement heuristic. The reconnection principle is as follows:
- We identify the two closest points between Cluster A and Cluster B according to the connection order established in the previous step.
- Then, we break the cycles by removing one edge from the cycle of Group A and one edge from the cycle of Group B.
- The endpoints of these broken edges are reconnected to merge the two loops into one, creating two new edges (bridges) that link the point in Cluster A to the point in Cluster B.
- Once all the groups are connected, the final solution may exhibit inefficiencies at the junction points; that is, the bridges are not necessarily optimal for all the nodes, especially for large instances. Therefore, the K-opt (K = 3) heuristic is applied to the final solution to reorganize the connections between the cluster boundaries to further reduce the total distance, if possible.

An example is shown in Figure 2.

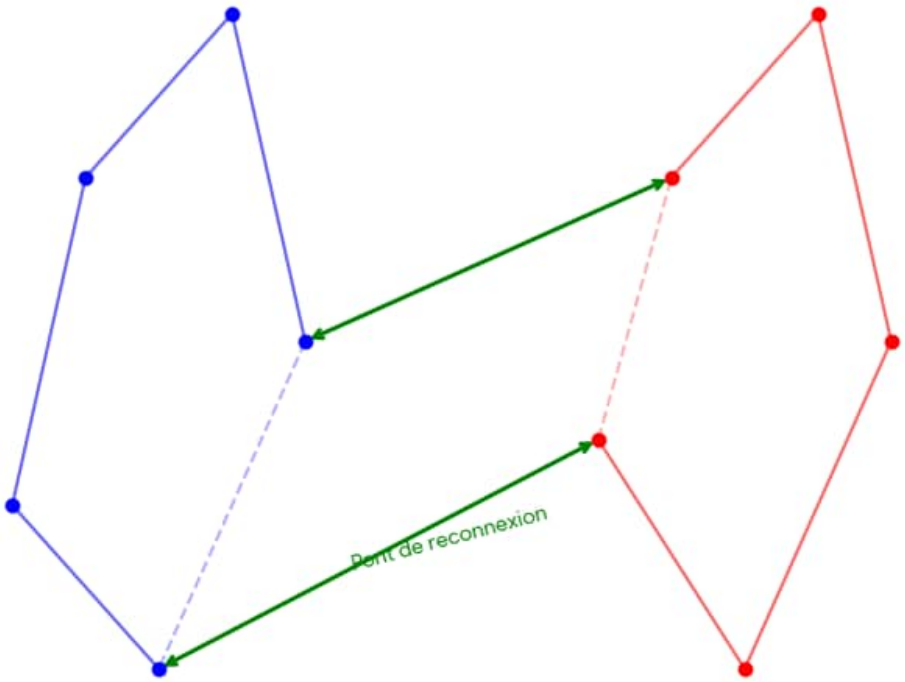

Figure 2. Cluster merging method.

The implementation of this algorithm will be done in python using the Sklearn library, and its average complexity is O(k.n.T) where: K is the number of clusters, N the number of cities and T the number of iterations.

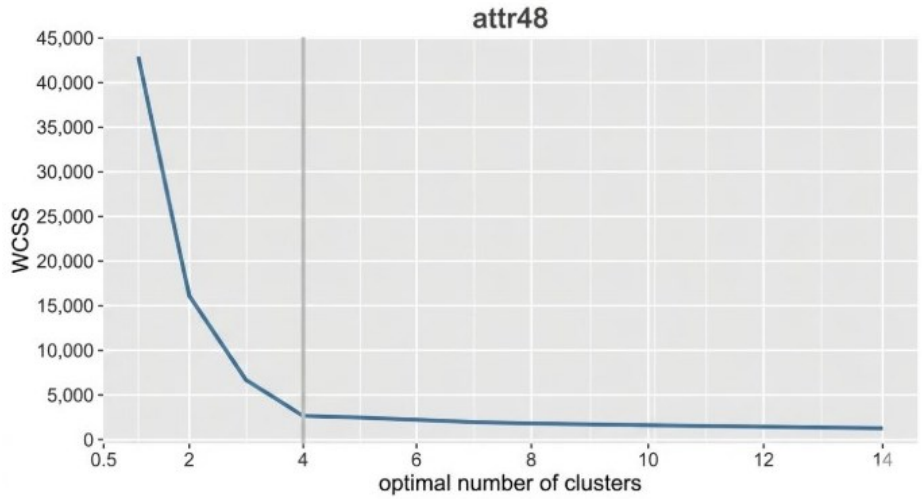

Figure 3. Obtaining the optimal number of clusters K by using the elbow method.

3.2.4. Presentation of the K-Opt Heuristic

The k-opt heuristic is a local search algorithm primarily used to improve solutions. Its goal is to reduce the total distance of a tour by locally modifying its edges. The algorithm starts with a known initial solution and attempts to improve it through successive steps by removing K stops from the graph and reconnecting the graph in a different way. Its operating principle is as follows:

- K edges are chosen from the current tour.
- These edges are removed, dividing the tour into K disjoint segments.
- These segments are reconnected by adding K new edges to form a new complete cycle (Hamiltonian cycle).
- If the new tour is shorter than the previous one, it is kept, and the process is repeated. Otherwise, another combination is tried.
- The algorithm stops when no further modification of K edges can shorten the tour. The solution is then said to be optimal.

Algorithm 4 shows how our hybridization process is implemented.

Algorithm 4: MACOV2SK algorithm.

Input: *m*: number of ants per colony;
n: number of cities;
Output: *S*: the optimal solution;
1 **Begin**
2 Division of the initial TSP instance using the K-means algorithm;
3 Resolving the TSP on each cluster using MACOV2;
4 Grouping of solutions;
5 Return the optimal solution S;

4. Experimental Results

In this section, we compare our algorithms (MACOV2 and MACOV2SK) with the MACO [2], SFLA [23], AACO-NC [25], K-EACO[24] and ACO-ML [1] algorithms on data or instances from the TSP online instance library named TSPLIB95 [30] in order to evaluate the

performance of our algorithms in terms of execution speed and solution quality compared with the best algorithms in the literature. We focus particularly on large instances, although some small and medium-sized instances will also be tested to effectively evaluate our approach to solving the TSP regardless of path length.

Two factors motivated our choice of parameter values used in this paper: (1) We relied on the technique used by [2] in MACO, who manually performed tests with the values of different parameters to choose the correct parameters, and the result was satisfactory. (2) We also conducted tests by manually varying these values to assess the robustness of these parameters; the values retained in this article are those that proved effective. By varying these parameters for our case study, we observed that for very small α (<0.5), the algorithm converges more quickly, but the optimal solution is not very good; the same is true for very large α (>1.5). By varying β, we note that for very small values of β (<0.5), the algorithm converges very slowly to the optimal solution, and when β becomes too large, the algorithm converges quickly; but in most cases, the optimal solution is not good, and sometimes even very bad. Finally, the algorithm becomes very unstable when ρ takes on large values; however, this does not have a major impact on execution time. The parameters used in the MACOV2 and MACOV2SK algorithms are shown in Table 1.

In this section, we compare the algorithms using the Wilcoxon test and here are the test results in terms of solution quality:

- MACOV2SK and AACO-NC : MACOV2SK is consistently better than AACO-NC across all 12 instances.
- Number of pairs (n): 12.
- Sum of positive ranks (W+): 78.
- Sum of negative ranks (W−): 0.
- Value p (p-value): 0.00034.
- Conclusion: The difference is highly significant. MACOV2SK consistently outperforms AACO-NC.
- MACOV2SK and MACO: MACOV2SK achieves better (or similar) results than MACO across all tested instances.
- Number of pairs (n): 12.
- Sum of positive ranks (W+): 78.
- Sum of negative ranks (W−): 0.
- Valeur p (p-value): 0.00048.
- Conclusion: The result is highly significant. Statistically, MACOV2SK outperforms MACO.

In short, the Wilcoxon test confirms that the MACOV2SK algorithm is the most efficient of the three. Not only does it provide the solutions closest to the optimum (lowest Average Error), but it is also the fastest in the majority of complex cases.

4.1. Analysis of Differences

By calculating the difference ($d = KEACO - MACOV2SK$) for each line, we find that in 100% of cases (11/11), the execution times of K-EACO are greater than those of MACOV2SK.

4.2. Analysis of the Results

In Tables 2–6 , we can see the following:

- The first column (Instance) represents the problem instances, and the numbers in parentheses represent the known optimal solution.
- The “Best Sol” column represents the best solution obtained after 100 iterations by the algorithms during the experiment.

- The "Average Sol" column represents the average of the solutions obtained after 100 iterations by the algorithms during the experiment.
- The "Average Error" column represents the relative error between the best solution of the experiment and the average solution obtained after 100 iterations.
- The "Runtime" column represents the average times in seconds taken by each metaheuristic used to produce the best solution during the experiment.

From the experimental results presented in Tables 2 and 3, we observe the following:

- In Table 2 and Figure 4, the MACOV2 algorithm significantly improves the execution time of the MACO algorithm for all instances evaluated with solutions that are practically identical in terms of quality. However, we can note very low degradation in terms of solution quality in MACOV2 for certain instances, benefiting from a very good difference in terms of execution time. This result allows us to validate the MACOV2 algorithm as the first major contribution of this paper.
- In Tables 3–5, we observe that MACOV2SK significantly improves the overall execution times of large instances, as well as the quality of the solution, compared with AACO-NC (Figure 5), the SFLA (Figure 6), K-EACO (Figure 7) and MACO, considered among the best recent algorithms in the literature for solving the TSP.
- In Tables 4 and 5 , we also observe that from the point of view of solution quality and execution time, our MACOV2SK approach is clearly superior to several other approaches in the literature based on clustering and ant colonies (Figure 6), such as that proposed by [23,24].

Table 3. Comparative results of MACOV2SK in terms of execution time and solution quality.

Instance	Algorithm	Best Sol.	Average Sol.	Average Error	Runtime (s)
pr439 (107,217)	MACOV2SK	107,217	107,234	0.01%	39.80
	AACO-NC	107,969	110,532.7	2.31%	45.89
	MACO	107,980.33	110,730.4	3.27%	98.65
Pcb442 (50,778)	MACOV2SK	50,983	51,186.13	0.80%	34.01
	AACO-NC	51,131	51,421.4	1.26%	35.48
	MACO	51,102.6	51,230.9	0.89%	58.41
rat575 (6773)	MACOV2SK	6773	6783.01	0.14%	68.15
	AACO-NC	6789	6846.36	1.08%	72.98
	MACO	6860	6901.3	1.89%	98.00
d493 (35,004)	MACOV2SK	35,104	35,281.0	0.79%	48.5
	AACO-NC	35,281	35,694.04	1.97%	62.70
	MACO	35,735	35,841.4	2.39%	79.80
u574 (36,905)	MACOV2SK	36,905.48	37,105.94	0.54%	78.95
	AACO-NC	37,329	37,933.1	2.78%	83.82
	MACO	37,981.96	38,024.82	3.03%	101.3
d657 (48,913)	MACOV2SK	48,913.9	49,676.00	1.55%	52.20
	AACO-NC	49,696	50,227.57	2.68%	68.46
	MACO	50,204	50,227.06	2.68%	81.04
u724 (41,910)	MACOV2SK	41,910.76	42,196.94	0.68%	68.75
	AACO-NC	42,210	42,498.2	1.40%	80.43
	MACO	42,268.96	42,378.9	1.11%	97.00
rat783 (8806)	MACOV2SK	8860	8862.5	0.64%	78.60
	AACO-NC	8865	8936.6	1.48%	87.22
	MACO	8940.96	8976.9	1.94%	104.10

Table 3. *Cont.*

Instance	Algorithm	Best Sol.	Average Sol.	Average Error	Runtime (s)
d1655 (62,128)	MACOV2SK	62,923	63,127	1.60%	873.84
	AACO-NC	62,983	63,823.24	2.72%	981.09
	MACO	63,616	64,133.7	3.22%	1390.00
rl1323 (270,199)	MACOV2SK	270,210.48	272,310.11	0.78%	285.06
	AACO-NC	273,289	276,945.11	2.49%	335.31
	MACO	281,785	280,195	3.69%	940.60
fl1400 (20,127)	MACOV2SK	20,227.48	20,310.11	0.90%	685.06
	AACO-NC	20,356	20,657.99	2.63%	789.77
	MACO	21,125	21,265	5.65%	940.60
pr2392 (378,032)	MACOV2SK	379,034.48	384,863	1.80%	985.06
	AACO-NC	384,881	387,758.59	2.57%	1730.30
	MACO	387,750	387,990.06	2.63%	1840.60

Table 4. Comparison of the MACOV2SK approach with another k-means approach [23] in terms of solution quality for solving small, medium, and large TSP instances from the TSPlib library.

Instance	MACOV2SK (Opt. Solution)	SFLA (Opt. Solution) [23]
Berlin52	7542	8000
Ch130	6110	6748
Tsp255	3919.6	4364
Pcb442	5019.5	60,882
Pr1002	279,045	578,593
u724	42,196	43,805
rat783	8862.5	9004
D1655	63,127	64,560
fl1400	20,310.11	21,267
Pr2392	384,863	395,397

Table 5. Comparison of the MACOV2SK approach with another k-means-based approach (K-EACO [24]) in terms of execution time for solving the TSP on small, medium, and large instances from the TSPlib library.

Instance	Runtime (s) MACOV2SK	Runtime (s) K-EACO [24]
Berlin52	1.4	16.09
eli76	1.6	30.88
bier127	2.6	88.27
kroA200	9.5	197.74
pr439	29.04	170
pr1002	202.04	307.01
u724	68.75	148.05
rat783	78.60	190.04
D1655	873.84	1435.60
fl1400	685.06	1123.8
Pr2392	985.06	1895.37

Table 6. MACOV2SK comparison with ACO-ML [1] in terms of solution quality for solving small, medium, and large TSP instances from the TSPlib library.

Instance	MACOV2SK (Average Sol)	ACO-ML: Heuristic with ML (Average Sol) [1]
eil51	426.03	522.85
kroA100	21,282.34	26,611.4
Tsp255	3919.6	5016.11
Pcb442	5019.5	64,697.14
rat575	6783.01	8602.61
u724	42,196.00	43,503.2
rat783	8862.5	11,212.15
D1655	63,127.00	77,146.27
fl1400	20,310.11	26,845.26
Pr2392	384,863.00	482,353.93

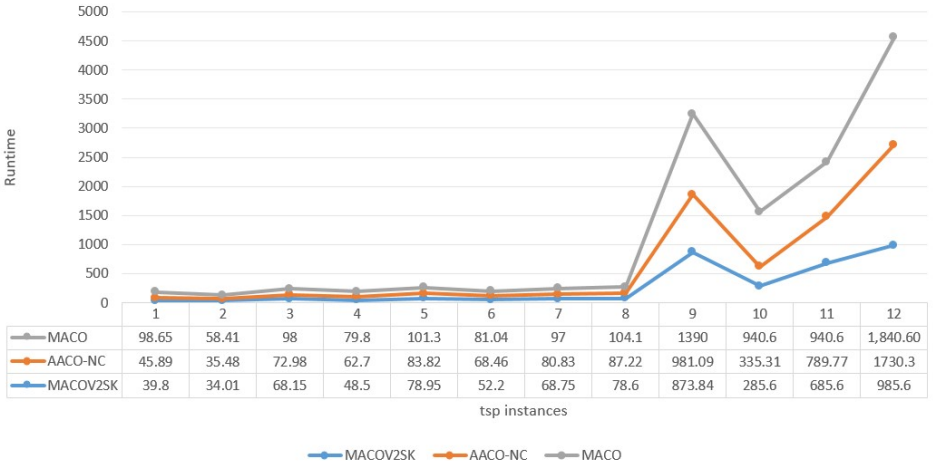

Figure 4. Comparison of MACOV2SK, MACOV2, MACO and AACO-NC methods in terms of execution time on large TSP instances.

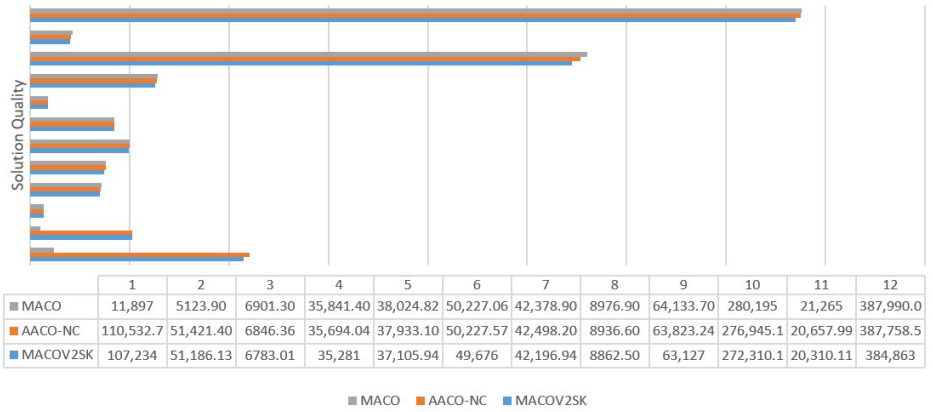

Figure 5. Comparison of MACOV2SK, MACOV2, MACO and AACO-NC methods in terms of solution quality on large TSP instances.

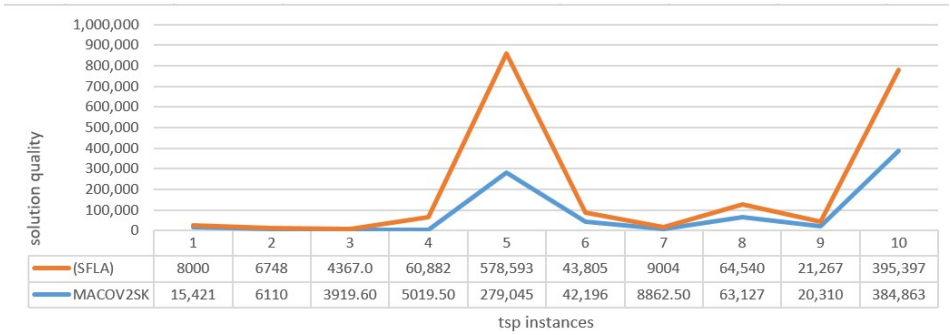

Figure 6. Comparison graph of our k-means hybrid approach with another k-means hybrid approach, the SFLA [23], on TSP resolution in terms of solution quality.

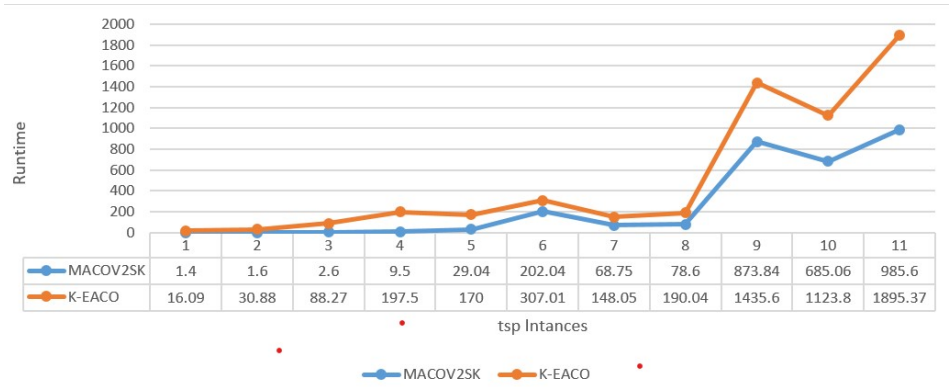

Figure 7. Comparison of MACOV2SK and K-EACO [24] heuristics in terms of execution time.

4.3. Demonstration of MACOV2SK Superiority over Other Methods

The superiority of MACOV2SK over other methods is based on the following key points: (1) Decomposing large instances of the TSP using the K-means method significantly reduces the search space exploration process, thus allowing ant colonies to converge much more quickly towards high-quality local solutions, where methods like MACO, AACO, or the SFLA failed by spreading themselves too thin across an overly broad search area. Figure 4 illustrates MACOV2SK superiority over the others in terms of execution time. (2) The decomposition of large TSP instances using the K-means method significantly reduces the search space exploration time. Figure 5 illustrates the superiority of MACOV2SK over MACO and AACO-NC in terms of solution quality. (3) The ERX crossover operator introduced in the MACOV2 method, by combining the best solutions found by the ants to create hybrid offspring, accelerates the discovery of the global optimum for the problem. This is not the case for MACO, ACO-ML, and K-EACO, which optimize exploration and the search for the global optimum through unguided and sometimes counterproductive stochastic movements. (4) The merging strategy used in our paper has the advantage of assembling the optimized subtowers by cluster by intelligently managing the junction points between clusters.

5. Conclusions and Perspectives

This article proposes new methodologies for addressing large-scale TSP resolution. It combines both artificial intelligence methods (K-means) and ant colony metaheuristics. The various contributions of this work were tested on standard TSPLIB instances and compared with recent approaches from the literature (MACO, SFLA, AACO-NC, and K-EACO). The results clearly demonstrate the effectiveness of our approaches, particularly the MACOV2SK approach, which stands out for its ability to solve large instances in concurrent time. It also offers good performance in terms of solution time and quality compared with

the MACO, SFLA, AACO, and K-EACO approaches with a better average error (Figure 8). Future work will focus on applying the MACOV2SK method to other classes of problems, such as the Vehicle Routing Problem (VRP) or the Multi-Traffic Problem (MTSP).

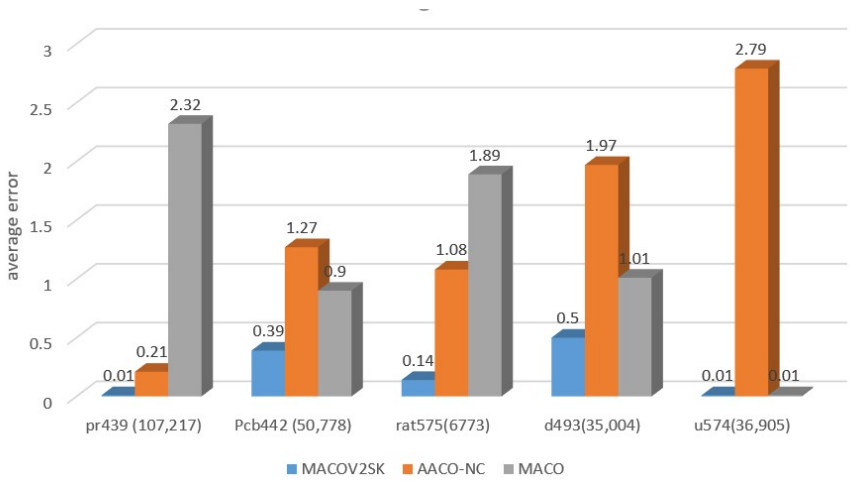

Figure 8. Comparison of the different algorithms in terms of Average Error.

Author Contributions: Conceptualization, B.N.T. and G.B.J.M.; methodology, B.N.T., G.B.J.M., M.M.Z.N. and M.V.; software, B.N.T., G.B.J.M., M.M.Z.N., M.N.M. and M.V.; validation, B.N.T., G.B.J.M., M.M.Z.N., M.N.M. and M.V.; formal analysis, B.N.T. and G.B.J.M.; investigation, G.B.J.M. and M.M.Z.N.; resources, B.N.T., G.B.J.M., M.M.Z.N. and M.V.; data curation, B.N.T., G.B.J.M., M.M.Z.N., M.N.M. and M.V.; writing—original draft preparation, B.N.T., G.B.J.M. and M.M.Z.N.; writing—review and editing, B.N.T., G.B.J.M., M.M.Z.N., M.N.M. and M.V.; visualization, B.N.T., G.B.J.M., M.M.Z.N., M.N.M. and M.V.; supervision, B.N.T., G.B.J.M., M.M.Z.N. and M.V.; project administration, B.N.T. and G.B.J.M.; funding acquisition, M.V. All authors have read and agreed to the published version of the manuscript.

Funding: This study was funded by the National Research Foundation of South Africa (Grant Number: 141918).

Data Availability Statement: The data used in this paper are available online at this address: https://github.com/Garrik10/maco_kmeans_data, accessed on 28 April 2026.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Petr Stodola, R.S. Using machine learning in combinatorial optimization: Extraction of graph features for travelling salesman problem. *Knowl.-Based Syst.* **2025**, *314*, 113216. [CrossRef]
2. Soh, M.; Nguimeya, B.; Djamegni, C.T. A Multi Ant Colony Heuristic Approach For Solving The Traveling Salesman Problem. *Rev. Afr. Rech. Inform. Math. Appl.* **2021**, *34*, *1–15* . [CrossRef]
3. Khachai, D.; Sadykov, R.; Battaia, O.; Khachay, M. Precedence constrained generalized traveling salesman problem: Polyhedral study, formulations, and branch-and-cut algorithm. *Eur. J. Oper. Res.* **2023**, *309*, 488–505.
4. Bock, S.; Bomsdorf, S.; Boysen, N.; Schneider, M. A survey on the Traveling Salesman Problem and its variants in a warehousing context. *Eur. J. Oper. Res.* **2025**, *322*, 1–14.
5. Bengio, Y.; Lodi, A.; Prouvost, A. Machine learning for combinatorial optimization: A methodological tour d'horizon. *Eur. J. Oper. Res.* **2021**, *290*, 405–421. [CrossRef]
6. Vitali, T.; Mele, U.J.; Gambardella, L.M.; Montemanni, R. Learning the travelling salesperson problem requires rethinking generalization. Constraints. *AIEEE Trans. Evol. Comput.* **2022**, 50–64. . [CrossRef]
7. Murugananthan, V.; Rehan, M.Y.E.S.; Srinivasan, R.; Kavitha, M.; Kavitha, R. Traveling salesman problem with ant colony optimization. In *Proceedings of the 2023 2nd International Conference on Edge Computing and Applications (ICECAA)*; IEEE: Piscataway, NJ, USA, 2023; pp. 481–485.

8. Hamza, A.; Darwish, A.H.; Rihawi, O. A new local search for the bees algorithm to optimize multiple traveling salesman problem. *Intell. Syst. Appl.* **2023**, *18*, 200242. [CrossRef]
9. Pan, X.; Jin, Y.; Ding, Y.; Feng, M.; Zhao, L.; Song, L.; Bian, J. H-tsp: Hierarchically solving the large-scale traveling salesman problem. In Proceedings of the AAAI Conference on Artificial Intelligence, Washington, DC, USA, 7–14 February 2023; Volume 37, pp. 9345–9353.
10. Cariou, C.; Moiroux-Arvis, L.; Pinet, F.; Chanet, J.P. Evolutionary algorithm with geometrical heuristics for solving the Close Enough Traveling Salesman Problem: Application to the trajectory planning of an Unmanned Aerial Vehicle. *Algorithms* **2023**, *16*, 44. [CrossRef]
11. Liu, Y.; Xu, L.; Han, Y.; Zeng, X.; Yen, G.G.; Ishibuchi, H. Evolutionary multimodal multiobjective optimization for traveling salesman problems. *IEEE Trans. Evol. Comput.* **2023**, *28*, 516–530. [CrossRef]
12. Chen, P.; Wang, Q. Learning for multiple purposes: A Q-learning enhanced hybrid metaheuristic for parallel drone scheduling traveling salesman problem. *Comput. Ind. Eng.* **2024**, *187*, 109851. [CrossRef]
13. Gunay-Sezer, N.S.; Cakmak, E.; Bulkan, S. A hybrid metaheuristic solution method to traveling salesman problem with drone. *Systems* **2023**, *11*, 259. [CrossRef]
14. Zheng, J.; He, K.; Zhou, J.; Jin, Y.; Li, C.M. Combining reinforcement learning with Lin-Kernighan-Helsgaun algorithm for the traveling salesman problem. In Proceedings of the AAAI conference on artificial intelligence, Online, 2–9 February 2021; Volume 35, pp. 12445–12452.
15. François, A.; Cappart, Q.; Rousseau, L. How to Evaluate Machine Learning Approaches for Combinatorial Optimization: Application to the Travelling Salesman Problem. The Concorde TSP Solver. 2019. Available online: https://www.math.uwaterloo.ca/tsp/index.html (accessed on 15 December 2025).
16. Ali AI, K.J. The asymmetric m-traveling salesmen problem: A duality based branch-and-bound algorithm. *Discret. Appl. Math.* **1989**, *13*, 259–276.
17. Habib, A.; Akram, M. Optimizing traveling salesman problem using tabu search metaheuristic algorithm with Pythagorean fuzzy uncertainty. *Granul. Comput.* **2024**, *9*, 16. [CrossRef]
18. Yousefikhoshbakht, M. Solving the traveling salesman problem: A modified metaheuristic algorithm. *Complexity* **2021**, *2021*, 6668345. [CrossRef]
19. Almufti, S.M.; Shaban, A.A. Comparative analysis of metaheuristic algorithms for solving the travelling salesman problems. *Int. J. Sci. World* **2025**, *11*, 26–30. [CrossRef]
20. Zhang, T.; Gruver, W.S.M. Team scheduling by genetic search. Proceedings of the second international conference on intelligent processing and manufacturing of materials. *Int. J. Comput. Math.* **1999**, 839–844. . . [CrossRef]
21. Taleb, M.A. Parallélisation d'un algorithme génétique pour le problème d'ordonnancement sur machine unique avec temps de réglages dépendants de la séquence. *Int. J. Comput. Math.* **2008** .
22. Dorigo, M.; Stuetzle, T. Ant Colony Optimization: Overview and Recent Advances. In *Handbook of Metaheuristics*; IRIDIA—Technical Report Series Technical Report, No. TR/IRIDIA/2009-013; Springer: Cham, Switzerland, 2009.
23. Karakoyun, M. A New Approach Based on K-means Clustering and Shuffled Frog Leaping Algorithm to Solve Travelling Salesman Problem. In Proceedings of the 7th International Symposium on Innovative Technologies in Engineering and Science, Şanlıurfa, Turkey, 22–24 November 2019; pp. 22–24.
24. Baydogmus, G.K. Solution for TSP/mTSP with an Improved Parallel Clustering and Elitist ACO. *Comput. Sci. Inf. Syst.* **2023**, *20*, 195–214. [CrossRef]
25. Stodola, P.; Otrısal, P.; Hasilova, K. Adaptive ant colony optimization with node clustering applied to the travelling salesman problem. *Swarm Evol. Comput.* **2022**, *70*, 101056. [CrossRef]
26. Hou, Z.; Yan, R.; Wang, S. On the K-Means Clustering Model for Performance Enhancement of Port State Control. *J. Mar. Sci. Eng.* **2022**, *10*, 1608. [CrossRef]
27. Umberto Junior Mele, M.G. A New Constructive Heuristic Driven by Machine Learning for the Traveling Salesman Problem. *Algorithms* **2021**, *14*, 267. [CrossRef]
28. Guntsch, M.; Middendorf, M. A population based approach for ACO. In *Evolutionary Computation*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 72–81.
29. Toth, P.; Vigo, E.D. *The Vehicle Routing Problem*; Digital Library Google Scholar; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2001.
30. Universität Heidelberg. TSPLIB95, Traveling Salesman Problem Library. 2019. Available online: https://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/index.html (accessed on 28 April 2026).