# Ant Colony Based Dynamic Voronoi Method for the Multi-Depot Multiple TSP

Sara Pérez-Carabaza*, Akemi Gálvez*, Andrés Iglesias*
*Dept. of Applied Mathemathics and Computational Sciences
*University of Cantabria*
Santander, Spain
perezcs@unican.es, galveza@unican.es, iglesias@unican.es

***Abstract*—This paper introduces a novel approach to solving the Multi-Depot Multiple Traveling Salesman Problem (MDMTSP), an extension of the classic Traveling Salesman Problem (TSP) characterized by multiple salesmen operating from various depots. The MDMTSP is particularly relevant in practical scenarios such as logistics and distribution, where efficient routing is crucial. Our approach integrates the Ant Colony System (ACS) with dynamically updated Voronoi regions, offering an innovative method to efficiently organize the assignment and routing of salesmen. This method not only optimizes the salesmen's routes but also ensures an efficient distribution of workload among them, leading to overall reduced travel distances. Experimental results demonstrate the effectiveness of our approach, highlighting significant improvements in route optimization compared to other existing methods.**

***Index Terms*—Multiple traveling salesman problem, Ant Colony Optimization, Voronoi Regions**

## I. Introduction

The multiple traveling salesman problem (mTSP) is a generalization of the well-known traveling salesman problem (TSP), allowing for more than one salesman within the solution. This characteristic renders the mTSP particularly suited to real-life scenarios such as robotics, transportation, and networking. Thus, the mTSP not only holds significant academic research value but also boasts extensive practical applications.

Depending on specific application requirements, the salesmen in mTSP scenarios can represent diverse entities, ranging from ground vehicles like trucks or robots to aerial vehicles such as drones [1]. Similarly, the destinations or 'cities' in these scenarios can represent a variety of entities, including customers, sensor nodes in wireless sensor networks, or targets in search and rescue operations [2].

This work delves into the Multi-Depot Multiple Traveling Salesmen Problem (MDMTSP), a more complex variant of the mTSP that introduces the concept of multiple depots [3]. While the single TSP has been extensively explored in literature, its multiple variant, particularly the MDMTSP, has not received equivalent attention [3]. This gap in research, particularly in the exploration of heuristic methods for solving the MDMTSP, signifies a interesting area for further investigation [4].

This research work has been supported by the project PID2021-127073OB-I00 of the MCIN/AEI/10.13039/501100011033/FEDER,EU, Spanish Ministry of Science and Innovation.

In this paper, we propose an innovative integration of the Ant Colony System (ACS) with dynamically updated Voronoi regions, specifically tailored to address the complexities of the MDMTSP. Our method, termed ACS-Voronoi, optimizes the assignment and routing of multiple salesmen who must visit a set of cities once and return to their respective depots. This methodology introduces a dynamic Voronoi region-based assignment of cities to each traveler, efficiently organizing work distribution. The experimental results highlight the benefits of our approach, not only in optimizing the routing of the salesmen but also in efficiently distributing the workload among them, leading to routes with reduced total lengths.

Voronoi regions have found applications in various complex optimization scenarios, including path planning [5] and Vehicle Routing Problems (VRP) [6]. Notably, the integration of Voronoi regions with ACS in dynamic path planning was shown in [5]. While this work also propose the use of Voronoi regions in conjunction with ACS, their application is distinctively different as they utilize these regions to divide an ocean environment map into a roadmap with edges. In the different context of VRP, Voronoi neighborhoods have been used to enhance efficiency, where a 'cluster-first, route-second' heuristic was proposed in [6]. This method involves clustering customers via Voronoi diagrams and refining routes through simulated annealing. Such instances demonstrate the versatility and capability of Voronoi methodologies in tackling a range of complex optimization challenges. In our ACS-Voronoi approach for the MDMTSP, we innovatively employ dynamic Voronoi regions to efficiently optimize the routes of multiple travelers, utilizing the characteristic distance matrix of TSP instances for efficient city assignment according to Voronoi regions.

The paper is organized as follows: Section II defines the combinatorial optimization problem. Section III reviews the existing literature, focusing on the MDMTSP and related issues. Section IV, describes the proposed ACS-based method, highlighting its main innovation: the inclusion of dynamic Voronoi regions. Section V analyzes its performance, along with the potential of the key contributions (the use of dynamic Voronoi regions and yield turn strategy), across several MDMTSP instances and compares it against other existing methods [7]. Finally, Section VI summarizes the main conclusions and outlines future research directions.

## II. Problem statement

The multiple TSP consists of determining the best routes for $m$ salesmen so that all the cities are visited once by each unique salesman. In fact, when there is only a single salesman, the multiple TSP reduces to the well-known TSP. Depending on whether all the salesmen start their tours from a unique depot or not, the multiple TSP can be classified as Single-Depot Multiple Traveling Salesman Problem (SDMTSP) or Multi-Depot Multiple Traveling Salesman Problem (MDMTSP). Moreover, MDMTSP can be further classified into fixed or non-fixed MDMTSP, depending on whether the $m$ travellers are required to return to their starting depot or whether they can finish in another depot. In this work, we consider the fixed destination MDMTSP, where all the travelers return to their original depots.

The fitness of each solution in MDMTSP is evaluated based on the total distance traveled by the $m$ travelers. Hence, the fitness function is formulated to minimize the sum of the distances for the $m$ closed tours contained in a solution $T^k$:

$$\textit{minimize} \sum_{\forall (i,j) \in T^k} dist(i,j) \tag{1}$$

This approach ensures that the optimal solution not only covers all cities but also minimizes the total distance traveled, aligning with the core objective of efficiency in routing problems.

## III. State-of-the-art

This section gives an overview of existing approaches to tackle the multiple TSP, focusing on those closely related to the proposed ACO-based method for the MDMTSP. For comprehensive reviews of the literature and a discussion on various applications of the mTSP, readers are referred to [3] and the more recent study [1].

Some exact solutions exist for the mTSP, but they are constrained to solving problems of limited size due to their high computational demands. For example, the authors in [8] have formulated and optimally solved a single-depot multiple TSP using Constraint Programming, taking into account minimum and maximum city limits per traveler. However, this approach is notably time-consuming, requiring approximately 2 hours to solve an instance involving 51 cities and 3 salesmen.

Another approach to solving the multiple TSP involves transforming it into a standard TSP, enabling the use of algorithms designed for the standard TSP. A notable instance of this transformation is presented in [9], where the MDMTSP is converted into a single asymmetric TSP by creating an extended graph with additional nodes representing the depots, and then solved using standard TSP exact methods. However, as pointed out in [3], methods that transform mTSP into standard TSP are often inefficient due to the degeneration of the resulting TSP problem.

Due to the high-computational complexity of the multiple TSP, heuristics and approximation methods are required to solve medium or large TSP instances. For instance, the authors in [10], combine the Invasive Weed Algorithm (IWO) with Partheno-Genetic algorithm to solve the fixed-destination MDMTSP. Venkatesh and A. Singh [11] propose an Artificial Bee Colony (ABC)-based method with local search for the SDMTSP, aiming to minimize both the total traveled distance and the maximum traveled distance per traveller.

Other works deal with extensions of MDMTSP, highlighting the diverse and evolving nature of multiple TSP research [12] [13]. On the one hand, in [12] approximation algorithms are applied to the many-visits variant of MDMTSP. On the other hand, an exact Branch and Bound method for non-fixed destination model with time windows is employed in [13], focusing on instances with only 6 to 10 cities.

Specially relevant to this work are those ACO-based approaches that deal with different versions of the mTSP problem [4], [7], [14], [15]. The ACO-based methods presented in [7] and [14] incorporate as an additional optimization objective the balanced work distribution among travelers. Namely, the authors propose and evaluate several multi-objective ACS-based methods for the SDMTSP, aiming to simultaneously optimize two objectives: total length and balanced subtours. While in [14], an ACO-based approach is proposed for mTSP, featuring a queen ant organizing teams of ants, each corresponding to a traveler, and utilizing dual pheromones to balance total travel distance and load among salesmen. On the other hand, S. Ghafurian and N. Javadian [15] propose an ACO-based solution tailored for a specific variant of the MDMTSP where the number of cities each traveler can visit is limited by minimum and maximum constraints. This method meticulously builds the tours for each traveler in an iterative manner, adjusting the feasible neighborhood within the ACO framework to ensure compliance with these city visit constraints. Following this work, [4] analyzes the impact of depot selection and constraints related to the number of travelers and cities per traveler, finding that these factors significantly affect the overall fitness of the solution and concluding that fewer travelers tend to yield shorter tour lengths. In contrast with these approaches, our ACO-based method offers greater flexibility by not imposing constraints on the number of cities per traveler, thereby accommodating a broader range of MDMTSP scenarios.

## IV. ACS-Voronoi based Method for MDMTSP

This section describes the proposed ACS-Voronoi method for MDMTSP. Initially, it outlines the fundamentals of the Ant Colony System (ACS). Subsequently, it delves into the novel Voronoi-based adaptations tailored for MDMTSP within the ACS framework. The section finishes with a detailed description of the algorithm, showcasing the integration of the proposed strategies.

### A. Ant Colony System

Ant Colony Optimization (ACO) is a framework comprising several algorithms inspired by the natural foraging behavior of ants. All ACO algorithms share a common structure, wherein every iteration a population of $M$ ants construct their tours based on heuristic knowledge specific to the problem and information learned through pheromone trails from the best solutions of previous iterations. During each iteration, the $M$ ant tours (candidate solutions) are constructed using transition rules that determine for the $k$-th ant located at node $i$ the next node $j$ from the nodes in the ant feasible neighborhood $N_i^k$. This selection is done through a probabilistic decision that considers the pheromone $\tau_{i,j}$ and heuristic $\eta_{i,j}$ values associated with traversing from node $i$ to node $j$. In TSP $\eta_{i,j}$ is set inversely proportional to the distance between both cities, i.e. $\eta_{i,j} = 1/dist(i,j)$. Among ACO various adaptations, the ACS stands out as a particularly effective variant for solving combinatorial optimization problems such as TSP [16]. The pseudorandom transition rule of ACS is given by Eq. (2), where $q$ is a uniform random variable, $q_0$ is a parameter of ACS (with $0 < q_0 < 1$), and $\beta$ is the heuristic influence parameter.

$$j = \begin{cases} argmax_{l \in N_i^k}\{\tau_{il} \cdot \eta_{il}^{\beta}\} & q \leq q_0 \\ \text{sample according to Eq. (3)} & \text{otherwise} \end{cases} \tag{2}$$

$$p_{i,j}^k = \frac{\tau_{i,j}\, \eta_{i,j}^{\beta}}{\sum_{l \in N_i^k} \tau_{i,j}\, \eta_{i,j}^{\beta}} \quad , \quad j \in N_i^k \tag{3}$$

This transition rule determines the next node $j$ according to Eq. (3), with a probability of (1-$q_0$). This equation, which is in fact the transition rule used by Ant System [16], states the probability $p_{i,j}^k$ for the $k$-th ant to travel from node $i$ to node $j$. On the other hand, Eq. (2) states that with a probability $q_0$, the experience accumulated by the ants is more strongly exploited, and the next node $j$ is set to $argmax_{l \in N_i^k}\{\tau_{il} \cdot \eta_{il}^{\beta}\}$, that is, the best possible move as indicated by probability distribution $p_{i,j}^k$ given by Eq. (3).

At the end of each iteration, once the $M$ ant tours are completed, the pheromone update process takes place. ACS applies pheromone reinforcement and evaporation only to the edges belonging to the best-found ant tour $T^{gb}$ according to Eq. (4).

$$\tau_{i,j} = (1-\rho)\tau_{i,j} + \frac{\rho}{f(T^{gb})} \quad \forall (i,j) \in T^{gb} \tag{4}$$

where $\rho$ is the pheromone evaporation parameter and $f(T^{gb})$ is the fitness of the best tour found so far. In addition to the global pheromone trail update rule, which is performed after all ants have completed their tours, ACS also considers a local pheromone update rule that is applied during the solution construction process. While building a solution of the TSP, ants visit edges and change their pheromone level by applying the local updating rule according to Eq. (5).

$$\tau_{i,j} = (1-\xi)\tau_{i,j} + \xi\tau_0 \tag{5}$$

where $\xi$ and $\tau_0$ are two algorithm parameters, with $0 < \xi < 1$. The value $\tau_0$ is set to be the same as the initial value of the pheromone trails.

### *B. Voronoi-Based Adaptations for MDMTSP*

This section describes the key proposed modifications of the ACS for MDMTSP. To begin with, in order to solve the MDMTSP using ACS, it is neccesary to define a proper codification of the solutions. In the context of MDMTSP, where multiple depots are involved, the codification must include not only the sequence of cities visited by each ant but also the assignment of each city to a specific depot/traveler. This ant tour $T^k$ can be represented as a set of $m$ tours, each associated with a particular traveler, and the sequence of cities visited by each traveller. Besides, the fitness of $T^k$ is evaluated based on the total distance traveled by the $m$ travelers, Eq. (1).

In order to assign a set of cities to each traveler, we propose a Voronoi-based strategy. This approach utilizes the concept of Voronoi diagrams to partition the set of cities into distinct regions, each corresponding to a specific traveler. We use the travelers' positions as generator points for the Voronoi regions, which are dynamically updated as the traveler positions change while constructing the ant tour. This Voronoi-based integration is incorporated within ACS by adapting the neighborhood $N_i^k$ of each traveler located at node $i$ to the set of unvisited cities that are within its Voronoi region. Additionally, beyond the neighborhood adaptation, the Voronoi regions play a crucial role in another key aspect of the multiple TSP: the distribution of work among the travelers. This is achieved through a 'yield turn' strategy, which efficiently balances the workload by allowing travelers to skip their turn when no unvisited cities are available in their Voronoi region. Below, the calculation of the Voronoi assignment is described, and a further description of the integration of Voronoi regions and the yield turn strategy within ACS is provided in Sec. IV-C.

As stated, the neighborhood of possible cities that a traveler located at node $i$ can move to is formed by the cities within its Voronoi region. To assign the $m$ Voronoi regions we do not compute the Voronoi geometric regions but instead follow a simple procedure that involves the distance matrix $dist$, whose elements $d(i,j)$ contain the distance to travel from city $i$ to city $j$. This distance matrix characterizes the TSP problem instances, and its use instead of the cities' coordinates saves on the computation of any distances. As an example, Fig. 1 shows the Voronoi assignment considering the distance matrix for the *berlin52* instance from the TSPLIB95 library [18] with $m$ equal to 3 travelers located at cities $s_1, s_2, s_3$. To obtain the Voronoi assignment, first we consider the submatrix formed by the $m$ rows corresponding to the nodes where the travelers are located (highlighted in bold in the image). This submatrix has a dimension $m$ by $n$, and contains the distance between each traveller (row) to the $n$ cities (columns). The Voronoi region associated with city $j$ is equal to the index of the row in the $j$-th column corresponding to the minimum distance. In the example of Fig. 1 Voronoi assignments are shown below the distance matrix.

### *C. Algorithm*

The ACS-Voronoi based algorithm for MDMTSP integrates the classic ACS framework with the Voronoi-based assignment

$$
\begin{array}{c}
\\ \\ s_1 \\ \\ s_2 \\ \\ s_3 \\ \\ \\ \\
\end{array}
\begin{bmatrix}
0 & 666 & 281 & \cdots & 1220 & 789 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\mathbf{291} & \mathbf{945} & \mathbf{509} & \cdots & \mathbf{984} & \mathbf{500} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\mathbf{1041} & \mathbf{1639} & \mathbf{1267} & \cdots & \mathbf{399} & \mathbf{285} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\mathbf{360} & \mathbf{390} & \mathbf{504} & \cdots & \mathbf{1330} & \mathbf{1044} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
1220 & 1716 & 1484 & \cdots & 0 & 625
\end{bmatrix}
$$
$$
\begin{array}{cccccc}
1 & 3 & 3 & \cdots & 2 & 2
\end{array}
$$

Fig. 1: Example of cities assignation to Voronoi regions generated by the vehicles positions $s_{1:m}$.

strategy to enhance the distribution of cities among multiple travelers. The algorithm proceeds as follows:

The algorithm requirements are the distance matrix $dist$, the number of travelers, and their depots $s^0_{1:m}$. Namely, each element $dist(i, j)$ contains the distance between city $i$ and $j$. Additionally, the algorithm requires typical parameters of ACS: the number of ants $M$ and the pheromone evaporation parameter $\rho$.

The algorithm starts by initializing the pheromone trails (line 4). Then, the main algorithm iteration loop (line 5 to line 25) runs until the stop condition is reached and the best found tour $T^{gb}$ is returned as a solution.

At the beginning of each iteration, during the solution construction loop (line 6 to 22), $M$ ant tours are generated. First, all tours are initialized by situating the travelers at their respective depots positions $s^0_{1:m}$. Next, the $k$-th ant tour is iteratively constructed until all cities are visited (line 8 to 21). Every turn, the Voronoi based assignation is updated considering the travelers' positions (line 10), following the method described in Section IV-B. Specifically, this method assign a Voronoi region to every city (stored in the vector $voronoi_{1:n}$) considering the travelers' positions as generator points. The solutions are iteratively constructed, where each traveler takes a turn and decides on their next city $j$ among its feasible neighbor $N^k_i$ (determined in line 12) following ACS transition rule (in line 14). Next, ACS local pheromone update rule given by Eq. (5) takes place. As previously described, the feasible neighbour $N^k_i$ for each traveler located at node $i$ is the set of unvisited cities that are within its Voronoi region. In the case where a traveller has no cities left to visit in its feasible neighbour, it yields its turn to other traveller (line 17). This yield turn strategy aims to lead to shorter and more efficient tours, and can result in a varied number of cities being assigned to each traveler. Once the constructed tour $T^k$ contains all cities the tour is finished by closing the tours, that is, each traveler returns to its depot (line 20).

Once the $M$ tours have been constructed, they are evaluated according to their total length as given by Eq. (1) and the best tour found so far $T^{gb}$ is updated. Additionally, at the end of each iteration the pheromones are updated according to ACS global update rule (line 24).

**Algorithm 1** ACS-Voronoi for MDMTSP

```
1 : Require: dist, m, s^0_{1:m}              // MDMTSP specifications
2 : Require: M, ρ                            // ACS parameters
3 : Initialize:
4 :       Initialize pheromone trails to τ_0
5 : Main Iteration Loop:
6 :    For each ant do
7 :       T^k ← initializeTour(s^0_{1:m})
8 :       While T^k is not complete do
9 :          s_{1:m} ← GetTravellerPositions(T^k)
10:          voronoi_{1:n} ← AssignVoronoi(dist,s_{1:m})
11:          For each traveller do
12:             N_i^k ← SetNeighbour(voronoi_{1:n}, T^k)
13:                If N_i^k not empty
14:                   T^k ← assign next node, Eq. (2)
15:                   ACS local update rule
16:                else
17:                   yield turn
18:                end if
19:          end for
20:          T^k ← closeTours(s^0_{1:m})
21:       end while
22:    end for
23:    Evaluate the M tours and save best ant tour T^{gb}
24:    ACS global pheromone update Eq. (4)
25: end for
26: Return: T^{gb}
```

When the stop condition is met, such as reaching a maximum number of algorithm iterations, the best tour found so far is returned as the solution to the MDMTSP. This tour represents the optimized route assignments for the multiple travelers, achieved through the synergistic combination of ACS and Voronoi-based strategies.

## V. EXPERIMENTAL RESULTS

This section presents the experimental results obtained from applying the ACS-Voronoi based method to the MDMTSP. It starts with an overview of our experimental setup, followed by the adaptation of the Nearest Neighbour heuristic for MDMTSP, serving both as a baseline for comparison and as a means to determine the initial pheromone value parameter for ACS. The section then progresses to a thorough analysis of different ACS-Voronoi variants, culminating in a comparative evaluation against other ACS-based methods for mTSP [7]. This comparison highlights the distinctive advantages and effectiveness of the proposed ACS-Voronoi based method.

### *A. Experimental Set-up*

In our experimental study, we employed instances from the TSPLIB95, a widely recognized library for the TSP [18]. For adapting these instances to the MDMTSP, $m$ depots per instance were randomly selected among the cities within each instance using a uniform distribution in MATLAB (seed value: 1), ensuring replicable and unbiased depot selection. The first and second columns of Tables I, II and III list the instance names and the corresponding depots selected for each MDMTSP instance.

For each MDMTSP instance, ACS methods were run 20 times to ensure robust statistical analysis, with a stopping criterion of 20,000 iterations.

Regarding the specific parameters of ACS, we opted for the values commonly recommended for the TSP [16], [17]. These values have a well-established performance in TSP scenarios, and an analysis of parameter tuning falls outside the scope of this work. The used parameters for all the ACS-based variants analysed in this work were set to the following values: $M = 10$, $q_0 = 0.9$, and $\rho = \xi = 0.1$. Additionally, in ACS, the initial pheromone value $\tau_0$ is set as $1/nC^{nn}$, where $C^{nn}$ is the length of a tour obtained by the nearest neighbor heuristic. To tailor this for MDMTSP, we developed two adaptations of the NN heuristic, which were used to calculate $\tau_0$ for our ACS-Voronoi method. Further details on these adaptations are provided in the following subsection.

### *B. Nearest Neighbour Adaptation for MDMTSP*

The Nearest Neighbour (NN) heuristic, traditionally used in the classic TSP, involves constructing a tour by sequentially moving to the closest unvisited city from the current location [16]. This approach, while simple, can yield efficient paths and serves as a foundational method in various optimization algorithms. In the case of ACO algorithms, this heuristic is used to set the initial pheromone value parameter ($\tau_0$).

We propose two distinct adaptations of the NN heuristic for MDMTSP. These adaptations are designed to determine the $\tau_0$ in our ACS-Voronoi method and also to act as tailored baseline methods for comparative analysis. Additionally, these adaptations have the potential to be employed as initial solutions in other metaheuristics, such as genetic algorithms or simulated annealing.

- **NN-Balanced**: This adaptation involves each salesman, starting from their respective depot, iteratively selecting the nearest unvisited city. All salesmen simultaneously make a move to their respective closest city in each iteration, ensuring a balanced workload distribution.
- **NN-Static**: This adaptation of the NN heuristic integrates Voronoi regions, using the salesmen's depots as the generating points. Each salesman, starting from their depot, is guided by the traditional NN heuristic but with a key modification: they are restricted to selecting the nearest unvisited city within their designated Voronoi region. This approach rationalizes the distribution of cities, ensuring that each salesman operates within an area proximate to their starting depot. As an example, Fig. 2 illustrates the solution generated by this heuristic for the *ch150* instance, with three travelers starting at cities 28, 52, and 60, highlighted in red.

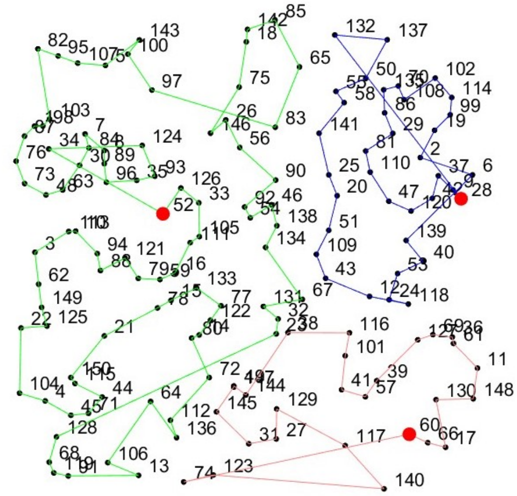


Fig. 2: The NN-Static Voronoi solution for *ch150* instance with depots 28-52-60 (l=8289).

| Name | Depots | Algorithm | Fitness |
|---|---|---|---|
| berlin52 | 22-38-1 | NN-Balanced | 12292 |
| | | NN-Static | 9413 |
| berlin52 | 16-8-5 | NN-Balanced | 14634 |
| | | NN-Static | 10965 |
| berlin52 | 10-18-21-29 | NN-Balanced | 14276 |
| | | NN-Static | 9778 |
| berlin52 | 22-36-11-46 | NN-Balanced | 12793 |
| | | NN-Static | 10456 |
| kroA100 | 42-73-1 | NN-Balanced | 32999 |
| | | NN-Static | 25975 |
| kroA100 | 31-15-10 | NN-Balanced | 34466 |
| | | NN-Static | 30423 |
| kroA100 | 54-42-69-21 | NN-Balanced | 33761 |
| | | NN-Static | 29734 |
| kroA100 | 88-3-68-42 | NN-Balanced | 29473 |
| | | NN-Static | 27851 |
| ch130 | 55-94-1 | NN-Balanced | 8991 |
| | | NN-Static | 8043 |
| ch130 | 40-20-13 | NN-Balanced | 10551 |
| | | NN-Static | 8637 |
| ch130 | 40-20-13 | NN-Balanced | 10551 |
| | | NN-Static | 8637 |
| ch130 | 71-55-90-27 | NN-Balanced | 10801 |
| | | NN-Static | 7949 |
| ch150 | 115-4-88-55 | NN-Balanced | 9268 |
| | | NN-Static | 7912 |
| ch150 | 28-52-60 | NN-Balanced | 10575 |
| | | NN-Static | 8289 |
| ch150 | 81-63-103-31 | NN-Balanced | 12579 |
| | | NN-Static | 8744 |
| ch150 | 132-5-101-63 | NN-Balanced | 10672 |
| | | NN-Static | 9041 |
| kroB200 | 84-145-1 | NN-Balanced | 44606 |
| | | NN-Static | 37899 |
| kroB200 | 61-30-19 | NN-Balanced | 43354 |
| | | NN-Static | 40526 |
| kroB200 | 108-84-138-41 | NN-Balanced | 51282 |
| | | NN-Static | 39923 |
| kroB200 | 176-6-135-84 | NN-Balanced | 53199 |
| | | NN-Static | 40106 |

TABLE I: Results of the two proposed NN heuristics for MDMTSP.

Table I shows the fitness (tour length) achieved by the two NN adaptations across 20 different MDMTSP instances. The results consistently demonstrate the superior performance of the NN-Static heuristic, particularly highlighting the efficacy of Voronoi regions in efficiently distributing the workload among the salesmen. Due to its better performance, this heuristic is selected for computing the initial pheromone values in the proposed ACS-Voronoi based method for MDMTSP. Nevertheless, we also want to acknowledge the potential merits of NN-Balanced. Despite resulting in longer tours, NN-Balanced ensures equitable distribution of cities among salesmen. This can be a significant consideration in practical applications where balanced workload is crucial.

### C. Analysis of ACS-Voronoi based method for MDMTSP

This subsection evaluates the performance of the ACS-Voronoi based method for MDMTSP, focusing on assessing the impact of dynamic Voronoi region updates and the yield strategy on the overall effectiveness of the solution. To this aim, we consider three variants of the proposed method for MDMTSP whose label and details are described below:

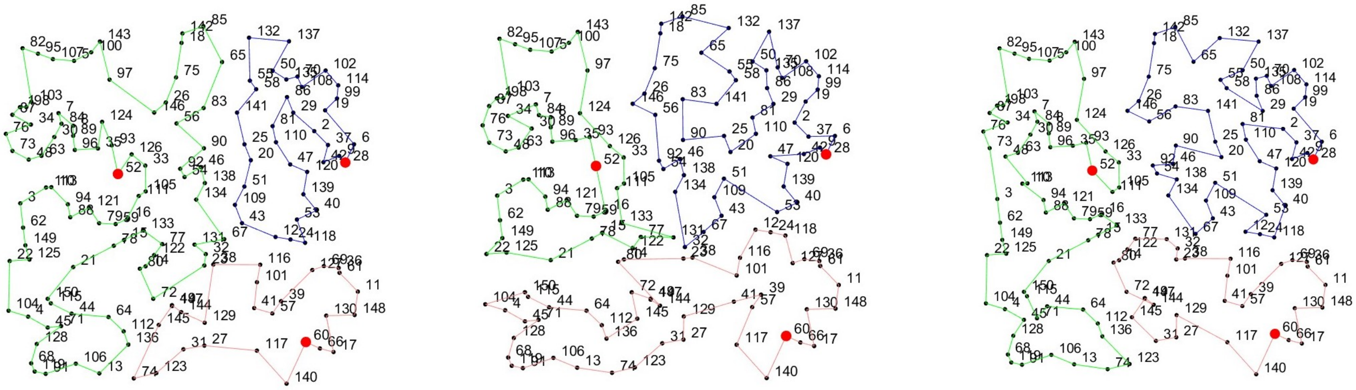
- **Dynamic-Yield**: This variant is the proposed ACS-Voronoi based method for MDMTSP, as described in Algorithm 1. It utilizes dynamic updates of Voronoi regions and incorporates the yield turn strategy, allowing salesmen to skip their turn if no unvisited cities are available in their Voronoi region. This enhances the efficiency of tour construction and could lead to shorter tours.
- **Static-Yield**: In this variant, Voronoi regions, determined based on the depots, remain static throughout the solution construction process. It serves to examine the effect of static versus dynamically updated Voronoi regions on the solution's fitness. Specifically, this variant omits the Voronoi update specified in line 10 of Algorithm 1.
- **Dynamic-Balanced**: This variant maintains the dynamic update of Voronoi regions but excludes the yield turn strategy. It assesses the impact on tour fitness by evenly distributing cities among salesmen without the flexibility provided by the yield strategy. Each salesman selects according to the transition rule, considering the unvisited nodes within their Voronoi-region. However, if a salesman has no nodes in his region, the set of all unvisited nodes is considered as the neighborhood. Specifically, this variant interchanges the yield turn strategy specified in line 17 of Algorithm 1 with the selection of the next city according to the transition rule defined by Eq. (2), considering all the unvisited nodes as $N_i^k$.

Table II presents the outcomes of applying these variants to various MDMTSP instances. The fourth column displays the average tour length obtained over 20 runs for each variant, the fifth column details the standard deviation of these solutions, and the final column contains the fitness of the best solution achieved in these 20 runs.

When comparing our approach (Dynamic-Yield) with the Static-Yield variant, we observe that Dynamic-Yield achieves better results in nearly all instances. This shows the clear benefit of dynamically update the Voronoi regions to the travelers' positions through the solutions construction process. However, it is notable that Static-Yield exhibits a lower standard deviation in the fitness of its solutions. This consistency can be attributed to its more limited search space, as each

| Name | Depots | Algorithm | Avg. | Std | Best |
|---|---|---|---|---|---|
| berlin52 | 22-38-1 | Static-Yield | 8463 | 20 | 8445 |
| | | Dynamic-Balanced | 8513 | 163 | 8258 |
| | | Dynamic-Yield | 8262 | 49 | 8188 |
| berlin52 | 16-8-5 | Static-Yield | 9062 | 0 | 9062 |
| | | Dynamic-Balanced | 8415 | 195 | 8070 |
| | | Dynamic-Yield | 8080 | 109 | 7798 |
| berlin52 | 10-18-21-29 | Static-Yield | 8933 | 31 | 8921 |
| | | Dynamic-Balanced | 9180 | 175 | 8929 |
| | | Dynamic-Yield | 8333 | 181 | 8032 |
| berlin52 | 22-36-11-46 | Static-Yield | 9086 | 24 | 9066 |
| | | Dynamic-Balanced | 8776 | 232 | 8461 |
| | | Dynamic-Yield | 7903 | 143 | 7788 |
| kroA100 | 42-73-1 | Static-Yield | 22042 | 43 | 22022 |
| | | Dynamic-Balanced | 23062 | 663 | 22388 |
| | | Dynamic-Yield | 21964 | 374 | 21502 |
| kroA100 | 31-15-10 | Static-Yield | 24955 | 59 | 24889 |
| | | Dynamic-Balanced | 25181 | 918 | 23996 |
| | | Dynamic-Yield | 24481 | 644 | 23342 |
| kroA100 | 54-42-69-21 | Static-Yield | 24052 | 6 | 24047 |
| | | Dynamic-Balanced | 25027 | 984 | 23661 |
| | | Dynamic-Yield | 22871 | 549 | 22140 |
| kroA100 | 88-3-68-42 | Static-Yield | 24614 | 69 | 24548 |
| | | Dynamic-Balanced | 23890 | 503 | 23284 |
| | | Dynamic-Yield | 23016 | 250 | 22713 |
| ch130 | 55-94-1 | Static-Yield | 6645 | 33 | 6592 |
| | | Dynamic-Balanced | 6701 | 256 | 6427 |
| | | Dynamic-Yield | 6362 | 84 | 6207 |
| ch130 | 40-20-13 | Static-Yield | 6999 | 52 | 6926 |
| | | Dynamic-Balanced | 7147 | 292 | 6655 |
| | | Dynamic-Yield | 6818 | 163 | 6553 |
| ch130 | 71-55-90-27 | Static-Yield | 6800 | 38 | 6734 |
| | | Dynamic-Balanced | 7341 | 276 | 6822 |
| | | Dynamic-Yield | 6726 | 203 | 6372 |
| ch130 | 115-4-88-55 | Static-Yield | 6733 | 47 | 6675 |
| | | Dynamic-Balanced | 7298 | 347 | 6778 |
| | | Dynamic-Yield | 6661 | 152 | 6364 |
| ch150 | 63-109-1 | Static-Yield | 7284 | 30 | 7233 |
| | | Dynamic-Balanced | 7255 | 263 | 6834 |
| | | Dynamic-Yield | 6948 | 109 | 6719 |
| ch150 | 28-52-60 | Static-Yield | 6957 | 33 | 6918 |
| | | Dynamic-Balanced | 7375 | 237 | 6817 |
| | | Dynamic-Yield | 6811 | 103 | 6632 |
| ch150 | 81-63-103-31 | Static-Yield | 7059 | 59 | 6956 |
| | | Dynamic-Balanced | 7819 | 371 | 7235 |
| | | Dynamic-Yield | 6922 | 161 | 6685 |
| ch150 | 132-5-101-63 | Static-Yield | 7227 | 60 | 7106 |
| | | Dynamic-Balanced | 7919 | 393 | 7290 |
| | | Dynamic-Yield | 6936 | 141 | 6680 |
| kroB200 | 84-145-1 | Static-Yield | 31386 | 306 | 30918 |
| | | Dynamic-Balanced | 33418 | 1464 | 30385 |
| | | Dynamic-Yield | 31799 | 1011 | 30540 |
| kroB200 | 61-30-19 | Static-Yield | 33056 | 391 | 32435 |
| | | Dynamic-Balanced | 32911 | 1497 | 30394 |
| | | Dynamic-Yield | 30948 | 560 | 30179 |
| kroB200 | 108-84-138-41 | Static-Yield | 32030 | 268 | 31671 |
| | | Dynamic-Balanced | 37279 | 1709 | 34999 |
| | | Dynamic-Yield | 31596 | 735 | 30762 |
| kroB200 | 176-6-135-84 | Static-Yield | 33537 | 210 | 33264 |
| | | Dynamic-Balanced | 38867 | 1721 | 35535 |
| | | Dynamic-Yield | 33285 | 1201 | 31358 |

TABLE II: Performance of ACS-Voronoi variants.



(a) Static-Yield (l=6965) (b) Dynamic-Balanced (l=7126) (c) Dynamic-Yield (l=6892)

Fig. 3: Illustration of solutions generated by the three ACS-Voronoi based variants for the *ch150* instance with depots 28-52-60.

traveler is restricted to the initial division of cities determined by their proximity to the depots. In terms of computational times, Dynamic-Yield showed an increase of approximately 3-4% compared to the Static-Yield variant, due to the additional processing required for dynamic Voronoi updates.

Comparing Dynamic-Yield with Dynamic-Balanced allows us to analyze the effectiveness of the yield turn strategy, where travelers can skip their turn when there are no unvisited cities available in their Voronoi region. Our approach consistently outperforms Dynamic-Balanced across all instances, leading to the conclusion that the yield turn strategy contributes significantly to reducing tour lengths.

Figure 3 illustrates example solutions obtained by the three variants for one of the instances. As can be observed when comparing Fig. 2 and Fig. 3(a), both the NN-Static and Static-Yield distribute cities among travellers according to the Voronoi regions from depots. However, Static-Yield leverages ACO optimization to enhance route efficiency within the same distribution framework, resulting in improved fitness. The solution of Dynamic-Balanced variant shown in Fig.3(b) displays an equal balance among the three travellers, with each visiting 50 cities. In contrast, the Dynamic-Yield variant further improves the total tour length by more effectively distributing the workload among travelers, as depicted in Fig.3(c).

In summary, our proposed Voronoi-ACS method (Dynamic-Yield) consistently outperforms the other two variants, demonstrating superior efficiency and adaptability in solving the MDMTSP. This success underscores the value of dynamically updating Voronoi regions and the strategic benefit of the yield turn strategy.

### D. Comparison with Other Methods

The final part of our experimental analysis involves a comparison of our proposed ACS-Voronoi based method with two other approaches: the Nearest Neighbour (NN) heuristic, which represents a baseline solution strategy for MDMTSP, and the ACS-based approach described in [7].

The NN heuristic is a well-known approach traditionally used for solving the classic TSP [16]. As adapted in Sec. V-B for the MDMTSP, this heuristic provides a robust baseline for comparison. Of the two adaptations—NN-Balanced and NN-Static—described previously, NN-Static was chosen for comparison due to its superior performance. Additionally, we compared our method with the ACS-based approach from [7]. In this approach, while each salesman uses the ACS transition rule for city selection, the turn of which salesman to move is chosen randomly. This ACS-based approach offers a well-suited contrast to our ACS-based method by illustrating the effectiveness of Voronoi regions and the yield turn strategy, in contrast to a process where the turn selection is randomized.

Table III shows the results over the same set of MDMTSP instances used in the previous experiments for the considered methods: the nearest neighbour heuristic which considers static Voronoi regions (labelled as NN-Static), the ACS-based approach proposed in [7], and the proposed Voronoi-ACS based method (labelled as Dynamic-Yield), which considers dynamic Voronoi regions and yield turn strategy. The results show that our Voronoi-ACS based method consistently outperforms the other two methods in all instances. On the one hand, the notable performance over the NN-Static heuristic underscores the superior efficiency of dynamically updated Voronoi regions in conjunction with the ACO metaheuristic. On the other hand, the remarkable results of our ACS-based method as compared to the ACS-based strategy method in [7], emphasize the significant advantages of combining dynamically updated Voronoi regions with the yield turn strategy.

## VI. Conclusions and Future Work

Our ACS-Voronoi based method has shown significant efficacy in addressing the MDMTSP, consistently achieving shorter tours compared to other methods. The innovative use of dynamic Voronoi regions and the strategic city allocation among travelers have been instrumental in enhancing tour efficiency. This adaptability underlines the potential of our approach. Furthermore, our approach's dynamic Voronoi region-based city assignment could be particularly relevant for dynamic versions of mTSP, where parts of the problem may change during runtime without prior knowledge. While

| Name | Depots | Algorithm | Avg. | Std | Best |
|---|---|---|---|---|---|
| berlin52 | 22-38-1 | NN-Static | 9413 | 0 | 9413 |
| | | ACS-based [7] | 8159 | 175 | 7879 |
| | | Dynamic-Yield | 8262 | 49 | 8188 |
| berlin52 | 16-8-5 | NN-Static | 10965 | 0 | 10965 |
| | | ACS-based [7] | 8166 | 141 | 7972 |
| | | Dynamic-Yield | 8080 | 109 | 7798 |
| berlin52 | 10-18-21-29 | NN-Static | 9778 | 0 | 9778 |
| | | ACS-based [7] | 8998 | 257 | 8465 |
| | | Dynamic-Yield | 8333 | 181 | 8032 |
| berlin52 | 22-36-11-46 | NN-Static | 10456 | 0 | 10456 |
| | | ACS-based [7] | 8207 | 176 | 7835 |
| | | Dynamic-Yield | 7903 | 143 | 7788 |
| kroA100 | 42-73-1 | NN-Static | 25975 | 0 | 25975 |
| | | ACS-based [7] | 23879 | 915 | 22305 |
| | | Dynamic-Yield | 21964 | 374 | 21502 |
| kroA100 | 31-15-10 | NN-Static | 30423 | 0 | 30423 |
| | | ACS-based [7] | 25736 | 856 | 24347 |
| | | Dynamic-Yield | 24481 | 644 | 23342 |
| kroA100 | 54-42-69-21 | NN-Static | 29734 | 0 | 29734 |
| | | ACS-based [7] | 25400 | 600 | 24101 |
| | | Dynamic-Yield | 22871 | 549 | 22140 |
| kroA100 | 88-3-68-42 | NN-Static | 27851 | 0 | 27851 |
| | | ACS-based [7] | 25539 | 772 | 24216 |
| | | Dynamic-Yield | 23016 | 250 | 22713 |
| ch130 | 55-94-1 | NN-Static | 8043 | 0 | 8043 |
| | | ACS-based [7] | 7069 | 235 | 6661 |
| | | Dynamic-Yield | 6362 | 84 | 6207 |
| ch130 | 40-20-13 | NN-Static | 8637 | 0 | 8637 |
| | | ACS-based [7] | 7209 | 272 | 6586 |
| | | Dynamic-Yield | 6818 | 163 | 6553 |
| ch130 | 71-55-90-27 | NN-Static | 7949 | 0 | 7949 |
| | | ACS-based [7] | 7624 | 246 | 7288 |
| | | Dynamic-Yield | 6726 | 203 | 6372 |
| ch130 | 115-4-88-55 | NN-Static | 7912 | 0 | 7912 |
| | | ACS-based [7] | 7570 | 204 | 7188 |
| | | Dynamic-Yield | 6661 | 152 | 6364 |
| ch150 | 63-109-1 | NN-Static | 8627 | 0 | 8627 |
| | | ACS-based [7] | 7733 | 358 | 7209 |
| | | Dynamic-Yield | 6948 | 109 | 6719 |
| ch150 | 28-52-60 | NN-Static | 8289 | 0 | 8289 |
| | | ACS-based [7] | 7796 | 240 | 7213 |
| | | Dynamic-Yield | 6811 | 103 | 6632 |
| ch150 | 81-63-103-31 | NN-Static | 8744 | 0 | 8744 |
| | | ACS-based [7] | 8173 | 347 | 7657 |
| | | Dynamic-Yield | 6922 | 161 | 6685 |
| ch150 | 132-5-101-63 | NN-Static | 9041 | 0 | 9041 |
| | | ACS-based [7] | 8467 | 266 | 7930 |
| | | Dynamic-Yield | 6936 | 141 | 6680 |
| kroB200 | 84-145-1 | NN-Static | 37899 | 0 | 37899 |
| | | ACS-based [7] | 36077 | 1275 | 33760 |
| | | Dynamic-Yield | 31799 | 1011 | 30540 |
| kroB200 | 61-30-19 | NN-Static | 40526 | 0 | 40526 |
| | | ACS-based [7] | 36134 | 1306 | 33926 |
| | | Dynamic-Yield | 30948 | 560 | 30179 |
| kroB200 | 108-84-138-41 | NN-Static | 39923 | 0 | 39923 |
| | | ACS-based [7] | 38376 | 877 | 36330 |
| | | Dynamic-Yield | 31596 | 735 | 30762 |
| kroB200 | 176-6-135-84 | NN-Static | 40106 | 0 | 40106 |
| | | ACS-based [7] | 39101 | 1511 | 36912 |
| | | Dynamic-Yield | 33285 | 1201 | 31358 |

TABLE III: Comparative results with other methods

there exists some research on the Dynamic Traveling Salesman problem (DTSP), literature on its multiple traveler variant, the Dynamic Multiple Traveling Salesman Problem (DMTSP), appears to be scarce. Addressing this gap, particularly in dynamic scenarios where problem parameters change in real-time, presents a significant opportunity for future research.

Furthermore, our ACS-Voronoi method for MDMTSP showcases significant flexibility, offering a robust foundation for future extensions that could incorporate various constraints, like a minimum or maximum number of nodes per traveler. This adaptability emphasizes our approach's versatility, uniquely positioning it to efficiently handle both constrained and unconstrained MDMTSP scenarios. This stands in contrast to other ACO-based approaches in the literature [4], [15], which are tailored for problems with mandatory minimum and maximum city limits per traveler, and thus may not be as adaptable to the unconstrained MDMTSP intances tackled in our study.

## References


[1] Cheikhrouhou, O., and Khoufi, I. (2021). A comprehensive survey on the Multiple Traveling Salesman Problem: Applications, approaches and taxonomy. Computer Science Review, 40, 100369.

[2] Cheikhrouhou, O., Koubâa, A., and Zarrad, A. (2020). A cloud based disaster management system. Journal of Sensor and Actuator Networks, 9(1), 6.

[3] Bektas, T. (2006). The multiple traveling salesman problem: an overview of formulations and solution procedures. Omega, 34(3), 209-219.

[4] Ramadhani, T., Hertono, G. F., and Handari, B. D. (2017, July). An Ant Colony Optimization algorithm for solving the fixed destination multi-depot multiple traveling salesman problem with non-random parameters. In AIP Conference Proceedings (Vol. 1862, No. 1). AIP Publishing.

[5] Xiong, C., Chen, D., Lu, D., Zeng, Z., and Lian, L. (2019). Path planning of multiple autonomous marine vehicles for adaptive sampling using Voronoi-based ant colony optimization. Robotics and Autonomous Systems, 115, 90-103.

[6] Fang, Z., Tu, W., Li, Q., Shaw, S. L., Chen, S., and Chen, B. Y. (2013). A Voronoi neighborhood-based search heuristic for distance/capacity constrained very large vehicle routing problems. International Journal of Geographical Information Science, 27(4), 741-764.

[7] Necula, R., Breaban, M., and Raschip, M. (2015, November). Tackling the bi-criteria facet of multiple traveling salesman problem with ant colony systems. In 2015 IEEE 27th international conference on tools with artificial intelligence (ICTAI) (pp. 873-880). IEEE.

[8] Vali, M., and Salimifard, K. (2017, August). A constraint programming approach for solving multiple traveling salesman problem. In The Sixteenth International Workshop on Constraint Modelling and Reformulation (pp. 1-17).

[9] Oberlin, P., Rathinam, S., Darbha, S. (2009, June). A transformation for a multiple depot, multiple traveling salesman problem. In 2009 American Control Conference (pp. 2636-2641). IEEE.

[10] Wang, Z., Fang, X., Li, H., and Jin, H. (2020). An improved partheno-genetic algorithm with reproduction mechanism for the multiple traveling salesperson problem. IEEE Access, 8, 102607-102615.

[11] Pandiri, V., and Singh, A. (2015). Two metaheuristic approaches for the multiple traveling salesperson problem. Applied Soft Computing, 26, 74-89.

[12] Bérczi, K., Mnich, M., and Vincze, R. (2023). Approximations for many-visits multiple traveling salesman problems. Omega, 116, 102816.

[13] Shirafkan, M. T., Seidgar, H., and Javadian, N. (2018). A new mathematical model for non-fixed destination multi-depot multiple travelling salesmen with time window problem. International Journal of Services and Operations Management, 31(4), 530-538.

[14] Lu, L. C., and Yue, T. W. (2019). Mission-oriented ant-team ACO for min–max MTSP. Applied Soft Computing, 76, 436-444.

[15] Ghafurian, S., and Javadian, N. (2011). An ant colony algorithm for solving fixed destination multi-depot multiple traveling salesmen problems. Applied Soft Computing, 11(1), 1256-1262.

[16] Parsons, S. (2005). Ant Colony Optimization by Marco Dorigo and Thomas Stützle, MIT Press, 305 pp., ISBN 0-262-04219-3. The Knowledge Engineering Review, 20(1), 92-93.

[17] Dorigo, M., Gambardella, L. M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, **1**(1), 53–66 (1997).

[18] Reinelt, G.: TSPLIB-A traveling salesman problem library. *ORSA Journal on Computing*, **3**(4), 376–384 (1991).