Upper Bound Computation for the Multiple Close-Enough Traveling Salesman Problem

Francesco Carrabs[a], Raffaele Cerulli[b], Ciriaco D'Ambrosio[c] and Gabriele Murano[d]
Department of Mathematics, University of Salerno, Giovanni Paolo II, 132, 84084, Fisciano(SA), Italy
{fcarrabs, raffaele, cdambrosio, gmurano}@unisa.it

Keywords: Close-Enough, Multiple Traveling Salesman Problem, Neighborhoods, Drones.

Abstract: This paper addresses the Multiple Close-Enough Traveling Salesman Problem, a variant of the Close-Enough Traveling Salesman Problem, where multiple vehicles are used to visit a given number of points. A vehicle visits a point if it passes through the neighborhood set of that point. The goal of the problem is to minimize the longest of the defined routes. We face the problem by using a discretization schema that reduces it to the Multiple Generalized Traveling Salesman Problem for which we propose a Mixed Integer Linear Programming formulation. The use of discretization schema introduces a discretization error that makes the optimal solution value of our model an upper bound of the optimal solution value of the Multiple Close-Enough Traveling Salesman Problem. Moreover, we apply a graph reduction algorithm to remove arcs and nodes without changing the optimal solution of the problem. We provide proof of the correctness of this algorithm, and we show that it significantly reduces the size of the instances tested. We verified the performance of our model on the benchmark instances of Close-Enough Traveling Salesman Problem.

1 INTRODUCTION

In this paper we address a variant of the Close-Enough Traveling Salesman Problem (CETSP) called the Multiple Close-Enough Traveling Salesman Problem (mCETSP). Given a set of target points in a Euclidean space, where each target has a neighborhood represented by a circular compact region centered on the target, the CETSP consists of finding a minimum length tour that starts and ends at a depot and intersects each neighborhood once. The variant mCETSP that, to the best of our knowledge, we address for the first time in the literature, considers m vehicles and the neighborhoods are represented by a set of discretization points and it consists in finding m tours (or routes), one for each vehicle, such that each neighborhood is intersected by al least one tour. The goal of the problem is to build the m tours so that the length of the longest one is minimized.

The mCETSP has several practical applications in the context of Unmanned Aerial Vehicles (UAV) in particular when the use of multiple drones is required. For example, in precision agriculture, a fleet of drones is tasked with flying over specific areas of interest (see Figure 1), gathering, from the sensors on the ground, essential information on parameters such as soil moisture, plant health, and other key indicators that influence agricultural decisions. In such a context, it becomes essential to plan the routes of the drones so that every area of interest is covered and the time to gathering the information is minimized. A comprehensive review of UAV applications in Smart Farming, along with the critical role of route planning for these vehicles, can be found in (Dolias et al., 2022).

The CETSP was defined and addressed for the first time in (Gulczynski et al., 2006). The authors de-

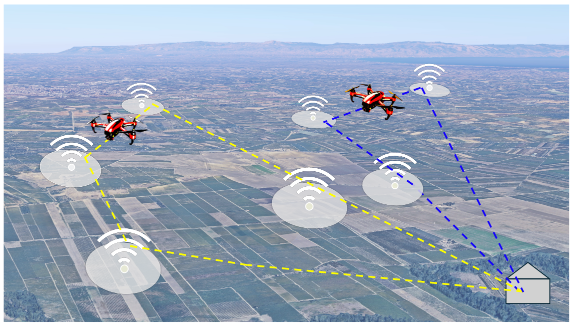
Figure 1: Collecting information from ground sensors using drones.

[a] https://orcid.org/0000-0003-2187-8624
[b] https://orcid.org/0000-0002-3277-6802
[c] https://orcid.org/0000-0003-1274-6144
[d] https://orcid.org/0009-0009-1238-1545

Carrabs, F., Cerulli, R., D'Ambrosio, C. and Murano, G.
Upper Bound Computation for the Multiple Close-Enough Traveling Salesman Problem.
DOI: 10.5220/0013377900003893
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 14th International Conference on Operations Research and Enterprise Systems (ICORES 2025)*, pages 186-195
ISBN: 978-989-758-732-0; ISSN: 2184-4372
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

velop and test some heuristics on several test cases. Later, in (Dong et al., 2007) based on a mixed-integer nonlinear program (MINLP), the authors propose a clustering-based algorithm and a convex hull-based algorithm, to find near-optimal solutions. (Mennell, 2009) and (Mennell et al., 2011) propose a heuristic algorithms based on Steiner zones, that is, the nonempty zones obtained by the intersections of the neighborhood sets, and this idea is applied also in the paper of (Wang et al., 2019), while (Yuan et al., 2007) introduces a first effective evolutionary approach. More recent and sophisticated approaches to the CETSP have been presented by (Behdani and Smith, 2014), (Coutinho et al., 2016), (Carrabs et al., 2017a; Carrabs et al., 2017b), (Yang et al., 2018) and (Carrabs et al., 2020). In (Behdani and Smith, 2014) the authors narrow the search space, proving that all optimal solutions can be described by a finite set of segments whose endpoints lie on the boundary of the disks representing the neighbourhoods of the targets. They present a mixed-integer programming (MIP) model for the CETSP based on a discretization scheme. The MIP model offers both lower and upper bounds for the optimal tour length, based on the granularity of the discretization. Furthermore, they propose valid inequalities along with two alternative formulations that further enhance the lower bounds and the resolution of the original problem. In (Coutinho et al., 2016), the authors propose an exact algorithm based on branch-and-bound and a Second-Order Cone Programming (SOCP) formulation. The proposed algorithm is the first method that provides exact optimal solutions for the CETSP in a finite number of steps. A main contribution to CETSP is represented by the discretization scheme proposed in (Carrabs et al., 2017b), which suggests discretizing not the outer circumference of a disk, but an inner circumference with a radius equal to the apothem of the regular polygon inscribed within the circumference, with the number of sides equal to the number of points used for discretization. In addition the article proposes a graph reduction algorithm (eliminating redundant edges) that significantly reduces the problem size. In (Carrabs et al., 2017a) the authors present an improved version of the discretization scheme proposed in (Carrabs et al., 2017b) and propose a new heuristic approach that is able to compute tight bounds for the problem. (Yang et al., 2018) develops an heuristic that combines a genetic algorithm with a particle swarm optimization. The computational results show that this heuristic is effective on the instances proposed by (Mennell, 2009). In (Carrabs et al., 2020), the authors propose a metaheuristic called (lb/ub)Alg to compute both upper and lower bounds on the optimal solution for the CETSP. This metaheuristic employs an innovative strategy to discretize the neighborhoods of the targets, minimizing discretization error, and applies the Carousel Greedy Algorithm to progressively select neighborhoods to add to the partial solution until a feasible solution is obtained. Very recently, in (Lei and Hao, 2024) the authors propose an effective memetic algorithm that integrates a carefully designed crossover operator and an effective local optimization procedure with original search operators. This algorithm is competitive with the others proposed in the literature and provides 30 new upper bounds. Finally, (Cariou et al., 2024) explores optimal route planning for UAVs used to collect data from IoT-based agricultural sensors. The study models sensor communication ranges as hemispheres and tackles the CETSP to establish efficient UAV trajectories.

In this paper, we address the *m*CETSP, a variant of the CETSP, which consists of finding *m* routes such that: i) each route starts and ends in the depot; ii) each neighborhood is crossed by at least one route. The problem aims to minimize the maximum route length among the *m* routes defined. Indeed, this goal better suits the characteristics of the real application since we want to gain the information from the sensors as soon as possible and this is done by using the drones simultaneously. However, the total time required to complete this task is not given by the sum of the length of the routes but from the longest one among them. Due to the complexity of this problem, we face here a discretized version of *m*CETSP, named *m*GTSP in which a set of discretization points are used to represent each neighborhood. The only difference between the two problems is that for *m*GTSP the routes are built by using only the discretization points of the neighborhoods whereas for *m*CETSP any point of the neighborhood can be used. Obviously, as a side effect of using the discretization, the optimal solution value of *m*GTSP will be an upper bound of the optimal solution value of *m*CETSP. The idea to discretize the neighborhoods was already successfully applied for the CETSP too and it allowed us to obtain tight upper bounds of the optimal solution. For *m*GTSP we will provide a mixed integer linear programming model. Unfortunately, from a literature review, we found out that the min-max version of problems similar to *m*GTSP, like the *m*TSP, is usually more complicated to solve with respect to the classical version in which the goal is to minimize the total distance travelled. To the best of our knowledge, *m*GTSP has not been previously studied in the literature.

The contribution of the current work can be summarized as follows:

- We propose a Mixed Integer Programming (MIP) formulation for *m*GTSP having a polynomial number of constraints;
- In order to reduce the size of the instances, we applied a graph reduction algorithm already tested for the discretized version of CETSP by (Carrabs et al., 2017b). In particular, we provide here a correctness proof of this algorithm, to prove that it works for the *m*GTSP problem too, despite in this last problem there are more routes and a different objective function.
- We provide a comprehensive numerical analysis of our proposed model, testing it on benchmark instances with a time limit of 1 hour.

The remainder of this paper is organized as follows. In Section 2, we introduce terminology and notation to be used throughout the paper and the formal definition of *m*GTSP. In Section 3 we present our MIP model for solving the problem. Computational results as reported in Section 4. Finally, conclusions are provided in Section 5.

2 DEFINITIONS AND NOTATION

Let N be a set of points in a two-dimensional plane, with $|N| = n$, and let 0 denote the *depot* point. The elements of N will be referred to as the *target points*. Each target point v is associated with a circumference C_v having center v and radius r_v. The neighborhood $N(v)$ of v consists of all points that are inside or on C_v. Without loss of generality, we assume that $0 \notin N(v)$ for all $v \in N$. The *m*CETSP consists of finding m routes such that: i) each route starts and ends in the depot; ii) each neighborhood is crossed by at least one route. The length of a route T is defined as the total sum of the edge lengths that compose T. We denote this length as $c(T)$. The goal of the problem is to build the m tours so that the length of the longest one is minimized. Let us define the *turn points* as the points of a tour where a direction change occurs. Any tour can be uniquely identified through its turn points. Due to the limited battery capacity of the drones, a maximum route distance (or travel time) $T^{\max}$ is imposed. In what follows, the terms maximum route distance and maximum drone travel time will be used interchangeably. The same is done for the terms tour and route. In Figure 2, a solution for the *m*CETSP problem is illustrated. The instance includes seven target points $\{v_1, v_2, \ldots, v_7\}$, with their corresponding neighborhoods represented by disks centered at these points. The points $\{p_1, p_2, \ldots, p_5\}$ are the turn points and , together with the depot 0, define the two tours: $T_1 =< 0, p_1, p_2, 0 >$ and $T_2 =< 0, p_3, p_4, p_5, 0 >$, depicted as dashed blue lines. In the following, we denote a solution of *m*CETSP as the set of its tours $\{T_1, \ldots, T_m\}$. Therefore, the solution of the example is denoted by $\{T_1, T_2\}$.

2.1 The Discretized Version of *m*CETSP

The number of feasible solutions for the *m*CETSP is infinite because there are infinitely many possible turn points in each neighborhood that can be used to build the tours. However, the number of turn points in any feasible solution is finite (see Figure 2). To formulate the *m*CETSP as an integer linear programming problem, each neighborhood $N(v)$ is discretized into a fixed number k of discretization points. We denote the set of discretization points for neighborhood $N(v)$ as $\widehat{N}(v)$, and the union of all discretization points for the instance as $\widehat{N} = \bigcup_{v=1}^{n} \widehat{N}(v)$.

Moreover, let $\mathcal{T}(i)$ be a function that, given a discretization point $i \in \widehat{N}$, returns the target point v, if $i \in \widehat{N}(v)$, or the depot 0 if i is the depot. Formally, $\mathcal{T} : \widehat{N} \cup \{0\} \to N \cup \{0\}$

$$\mathcal{T}(i) = \begin{cases} v \in N : i \in \widehat{N}(v), & \text{if } i \neq 0, \\ 0, & \text{if } i = 0. \end{cases}$$

The discretized version of the *m*CETSP corresponds to a variant of the well-known Generalized Traveling Salesman Problem (GTSP) with m vehicles instead of a single one. For this reason, we denote the discretized version of the *m*CETSP as *m*GTSP. To obtain a feasible solution for the *m*GTSP, it is sufficient to ensure that at least one discretization point in each neighborhood is visited by one of m routes.

The discretization points positioning in the neighborhoods is crucial to obtain high-quality solutions for the *m*CETSP. Indeed, the gap between the optimal solution values of *m*CETSP and *m*GTSP depends on this positioning. It has been proved, in the literature (Behdani and Smith, 2014), that the optimal solution of the CETSP places the turn points of the routes on the circumferences of the neighborhoods. Hence, the most intuitive strategy would involve placing the discretization points along the circumferences of the neighborhoods. Such a scheme, referred to as the *Perimetral Discretization Scheme (PD)*, divides each circumference C_v associated with a neighborhood $N(v)$ into k equal circular arcs. The discretization points are positioned at the endpoints of these arcs. In Figure 3(a) is shown the Perimetral Discretization Schema with three discretization points, d_1, d_2 and d_3, and the relative circular arcs $\widehat{d_1, d_2}$, $\widehat{d_2, d_3}$ and $\widehat{d_3, d_1}$.

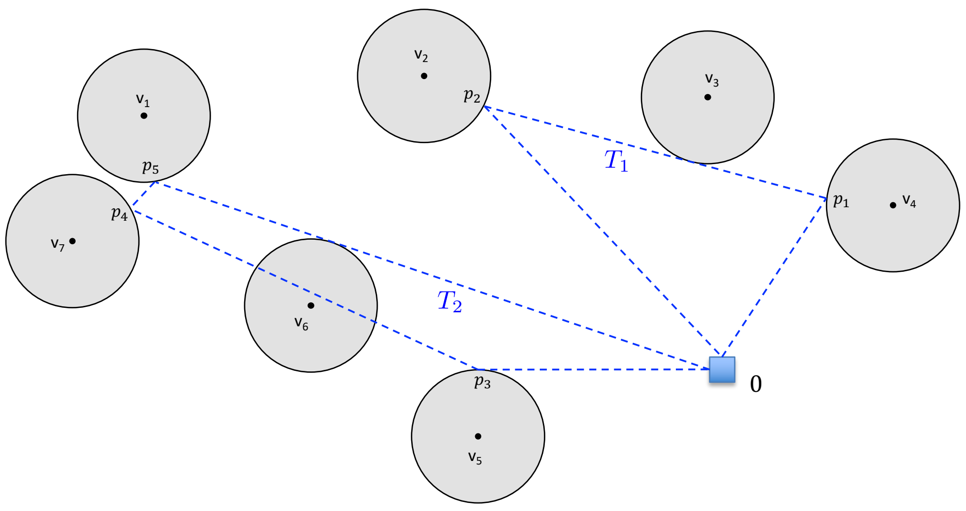

Figure 2: An example of feasible solution of *m*CETSP problem with seven target points and two routes.

(Carrabs et al., 2017b) proved that better results can be achieved by positioning these discretization points within the neighborhoods rather than on their circumferences. This scheme, named *Internal Point Discretization Schema* (IP), works as follows. Given the number of discretization points k, IP divides C_v into k equal circular arcs and places a discretization point at the midpoint of the chord corresponding to each arc. Figure 3(b) shows the IP schema for $k = 3$.

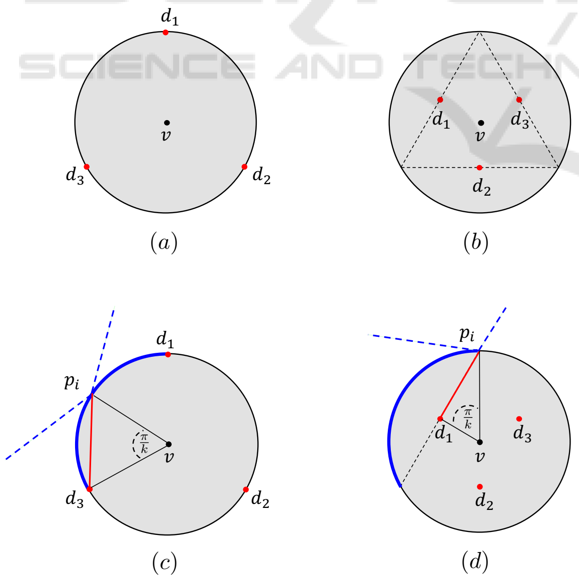

Figure 3: (a) Perimetral and (b) Internal Point Discretization schemas for $k = 3$. Discretization error for (c) Perimetral schema and (d) Internal Point schema.

(Carrabs et al., 2017b) defined the discretization error ε_v, carried out by a tour T on a neighborhood $\hat{N}(v)$, as two times the maximum distance between the turn point p_i of T on C_v and the discretization point of $\hat{N}(v)$ closest to p_i. Figure 3(c) shows the discretization error for the PD schema. If T intersects $N(v)$ on the circular arc $\widehat{d_1,d_3}$ then the maximum distance occurs when p_i is in the middle of this circular arc and it is represented by the red segment $\overline{d_3,p_i}$ (or $\overline{d_1,p_i}$). From trigonometry we derive that the length of segment $\overline{d_3,p_i}$ is $c(\overline{d_3,p_i}) = 2r_v \sin(\frac{\pi}{2k})$ and $\varepsilon_v = 4r_v \sin(\frac{\pi}{2k})$.

For the IP schema, the maximum distance occurs when p_i is on an extreme of the circular arc and it is represented by the red segment $\overline{d_1,p_i}$ in Figure 3(d). By trigonometry we have that $c(\overline{d_1,p_i}) = r_v \sin(\frac{\pi}{k})$ and $\varepsilon_v = 2r_v \sin(\frac{\pi}{k})$. Based on trigonometric considerations, this implies that the discretization error carried out by the IP schema is lower than the one carried out by the PD schema, in particular for small values of k. For this reason, we adopt the IP discretization schema in this work.

The *m*GTSP is defined on a complete directed graph $G = (V,A)$, where the node set is $V = \{0 \cup \widehat{N}\}$ and the arc set is $A = \{(i,j) : i \in V, j \in V, i \neq j, \mathcal{T}(i) \neq \mathcal{T}(j)\}$. Figure 4 illustrates an instance of the *m*GTSP with $k = 3$. The black points $\{v_1, v_2, \ldots, v_7\}$ represent the target points, while the red points indicate the discretization points. In this example, three discretization points are used to discretize each neighborhood. The depot is denoted as 0. The two tours T_1 and T_2, shown as dashed blue lines, correspond to the solution $\{T_1, T_2\}$ for the *m*CETSP, while the two tours $\widehat{T}_1$ and $\widehat{T}_2$, depicted as red dotted lines, represent the solution $\{\widehat{T}_1, \widehat{T}_2\}$ for the *m*GTSP. Let us denote by $\ell(T)$ the length of a tour T. The solution value of *m*CETSP is equal to $\ell(T_2)$ while the one of *m*GTSP is $\ell(\widehat{T}_2)$. It is easy to see that $\ell(\widehat{T}_2) > \ell(T_2)$ and this occurs be-

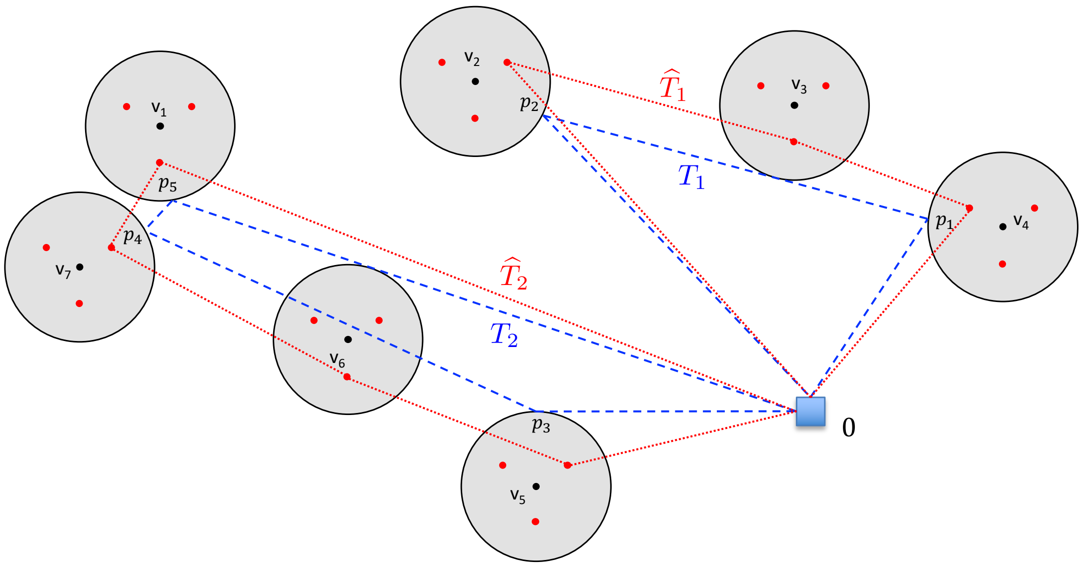

Figure 4: Comparison between the solutions of *m*CETSP (in blue) and *m*GTSP problems (in red).

cause the use of the discretization points limits the number of routes that the algorithms can build. For this reason, the optimal solution value we obtain by solving *m*GTSP is an upper bound to the optimal solution value of *m*CETSP.

3 A MATHEMATICAL FORMULATION FOR *m*GTSP

In this section, we present a mixed integer linear programming formulation for the *m*GTSP obtained by adapting the formulation proposed by (Bianchessi et al., 2018) for the Team Orienteering Problem. The formulation involves four types of variables: x_{ij}, y_i, w_{ij}, and z. The meaning of these variables is defined in the following.

$$x_{ij} = \begin{cases} 1 & \text{if the arc } (i,j) \text{ is selected,} \\ 0 & \text{otherwise.} \end{cases}$$

The x variables are used to describe the routes assigned to the drones.

$$y_i = \begin{cases} 1 & \text{if node } i \text{ is visited,} \\ 0 & \text{otherwise.} \end{cases}$$

The y variables state what are the nodes visited by drones. These variables are necessary to assure that each target point is covered by the routes.

$$w_{ij} \geq 0$$

The w variables are continuous variables that state the arrival time to the node j of a drone coming from node i. This variable is used to calculate the length of individual tours. Finally, variable z represents the length of the maximum route inside the solution, and the goal of the model is to minimize the value of this variable.

Following the formulation proposed by (Bianchessi et al., 2018), we introduce a second (dummy) depot that we denote by $\bar{0}$. Both depots, 0 and $\bar{0}$, are located in the same position of the plane. Therefore, from now on, the graph $G = (V, A)$ has the node set $V = \{0 \cup \bar{0} \cup \widehat{N}\}$ and the arc set is $A = \{(i,j) : i \in V, j \in V, i \neq j, \mathcal{T}(i) \neq \mathcal{T}(j)\}$ with $\mathcal{T}(\bar{0}) = \bar{0}$.

In order to present our mathematical formulation for *m*GTSP, we need the following notation.

- $T^{\max}$: Maximum length of a route;
- m: Number of routes;
- t_{ij}: Travel time from node i to j, where $t_{0\bar{0}} = 0$ and $t_{ii} = 0$;
- $t^0_{ij} = t_{0i} + t_{ij}$;
- $T^{\max}_{j\bar{0}} = T^{\max} - t_{j\bar{0}}$;
- $\delta^+(S) = \{(i,j) \in A : i \in S, j \notin S\}$: Set of arcs leaving the set S, with $S \subseteq \widehat{N} \cup \{0, \bar{0}\}$;
- $\delta^-(S) = \{(i,j) \in A : i \notin S, j \in S\}$: Set of arcs entering the set S, with $S \subseteq \widehat{N} \cup \{0, \bar{0}\}$.

The mathematical formulation of the *m*GTSP is the following:

$$(\textbf{MIP}) \quad min \ z \tag{1}$$

$$\sum_{j \in \widehat{N}} x_{0j} = \sum_{i \in \widehat{N}} x_{i\bar{0}} = m \tag{2}$$

$$\sum_{(j,i)\in\delta^-(i)} x_{ji} = \sum_{(i,j)\in\delta^+(i)} x_{ij} = y_i \quad i \in \widehat{N} \tag{3}$$

$$\sum_{i\in\widehat{N}(v)} y_i = 1 \quad v \in N \tag{4}$$

$$w_{0j} = t_{0j}x_{0j} \quad j \in \widehat{N} \tag{5}$$

$$\sum_{(i,j)\in\delta^+(i)} w_{ij} - \sum_{(j,i)\in\delta^-(i)} w_{ji} = \sum_{(i,j)\in\delta^+(i)} t_{ij}x_{ij} \quad i \in \widehat{N} \tag{6}$$

$$w_{ij} \le T^{\max}_{j\bar{0}} x_{ij} \quad (i,j) \in A \setminus \{(0,\bar{0})\} \tag{7}$$

$$w_{ij} \ge t^0_{ij} x_{ij} \quad (i,j) \in A \setminus \{(0,\bar{0})\} \tag{8}$$

$$z \ge w_{i\bar{0}} \quad i \in \widehat{N} \tag{9}$$

$$y_i \in \{0,1\} \quad i \in \widehat{N} \tag{10}$$

$$w_{ij} \ge 0 \quad i,j \in \widehat{N} \tag{11}$$

$$x_{ij} \in \{0,1\} \quad (i,j) \in A \setminus \{(0,\bar{0})\} \tag{12}$$

- The objective function (1) aims to minimize the variable z that represents the length of the maximum route inside the solution;
- Constraint (2) impose that exactly m routes have to start from the depot 0 and end into the depot $\bar{0}$. This assure that all the required routes are defined and that each route starts and ends into the depot;
- Constraint (3) assure that a discretization node i is entered and leaved exactly once if it is visited (i.e., if $y_i = 1$);
- Constraint (4) ensures that exactly one discretization point in the neighborhood $\widehat{N}(v)$ of each target point v is visited. Since in our problem, we use the Euclidean distances, the triangle inequality holds. This means that always exists an optimal solution to the problem that visits one point for each neighborhood.
- Constraint (5) assigns to variable w_{0j} the arrival time of a route into node j coming directly from depot 0.
- Constraint (6) ensures that the w_{ij} variables are correctly updated according to the arcs selected (i.e. $x_{ij} = 1$).
- Constraint (7) sets an upper limit on the duration of each route.
- Constraint (8) sets a lower bound on the values of w_{ij}, ensuring w_{ij} represents the arrival time at node j if $x_{ij} = 1$, otherwise $w_{ij} = 0$.
- Constraint (9) ensures that the variable z equals the length of the maximum route.
- Finally, (10)-(12) are variables definition constraints.

To enhance the strength of the model, an additional constraint can be introduced to limit the overall duration of all routes combined:

$$\sum_{(i,j)\in A\setminus(0,\bar{0})} t_{ij}x_{ij} \le mT^{\max} \tag{13}$$

In our model, the T^{max} parameter plays a fundamental role in determining the feasibility of solutions, as it represents an upper limit on the maximum route duration for each drone. In order to compute an upper limit of the optimal solution value, we take a feasible solution T of the CETSP and add to its value $c(T)$ the discretization error $\varepsilon_v = 2r_v \sin\left(\frac{\pi}{k}\right)$ carried out for each neighborhood $\hat{N}(v)$ inside a route. Formally:

$$T^{\max} = \left\lceil c(T) + \sum_{v\in N} \varepsilon_v \right\rceil \tag{14}$$

It is worth noting that the ε_v value depends on the r_v value and, in general, the neighborhoods could have different radii. Since T_{max} is used as a bigM value inside the model, we would to minimize its value to improve the performance of the model. To this end, we observe that each of the m routes must contain at least one discretization point. This means that, the maximum number of discretization points inside a route for mGTSP, is equal to $|N|-(m-1)$. Therefore, the total discretization error, carried out in any route, can be computed by adding the $|N|-(m-1)$ largest ε_v values. By denoting with N_m the set of $|N|-(m-1)$ largest ε_v values, we update the formula to compute T_{max} as follow:

$$T^{\max} = \left\lceil c(T) + \sum_{v\in N_m} \varepsilon_v \right\rceil \tag{15}$$

3.1 Graph Reduction Algorithm (GRA)

In order to speed up the computation of the upper bound, the *graph reduction algorithm* (GRA) proposed in (Carrabs et al., 2017b), is applied on $G = (V,A)$ to remove nodes and arcs that will not contribute to the optimal solution. Furthermore, we provide a proof of correctenss of GRA in the context of mGTSP, as GRA was originally proposed for a discretized version of CETSP while in our case we have m routes and a different objective function.

GRA works as follows. Let v be a target point and let $i \in V$ and $j \in V$ be two points such that $\mathcal{T}(i) \neq \mathcal{T}(j) \neq v$ (see Figure 5). According to the Euclidean distance, the algorithm computes the shortest path from i to j passing through r, for each point

$r \in \widehat{N}(v)$. Notice that these shortest paths are always composed of two edges (i,r) and (r,j). Among all the shortest paths computed, the algorithm saves the shortest one, which we denote from now on as $S_{(i,v,j)}$, and it marks as necessary its two edges and the point of $\widehat{N}(v)$ belong to it. This computation is repeated for each target point $v \in N$ and for each pair of points $i \in V$ and $j \in V$ with $\mathcal{T}(i) \neq \mathcal{T}(j) \neq v$.

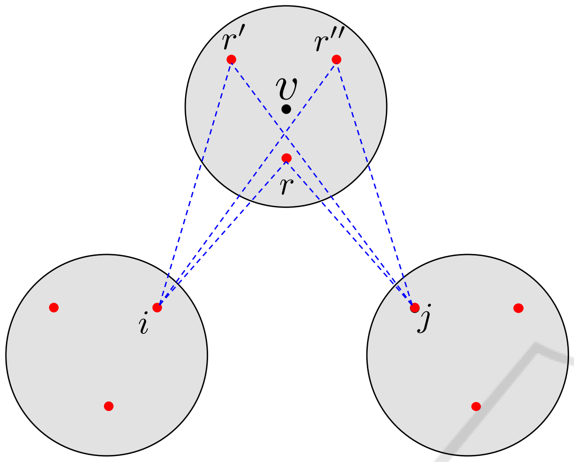

Figure 5: Example of shortest paths computed using Graph Reduction Algorithm.

At the end of the computation, GRA removes all the edges and nodes not marked, that is, the ones that do not belong to any shortest path $S_{(i,v,j)}$. Notice that the depots are marked by default because they are necessary for the construction of the routes. In the following, with a little abuse of notation, we use the ℓ function to denote the length of paths too. Let us consider the example depicted in Figura 5. GRA computes the three shortest paths $< i,r,j >$, $< i,r',j >$ and $< i,r'',j >$. Since, among these three paths, the shortest one is $< i,r,j >$, the algorithm marks the arcs (i,r), (r,j) and the node r. Moreover, $S_{(i,v,j)} =< i,r,j >$. Theorem 1 proves that by applying GRA to the graph, the optimal solution of mGTSP does not change.

Theorem 1. *Let $S = \cup_{i\in V, j\in V, v\in N, \mathcal{T}(i)\neq\mathcal{T}(j)\neq v} S_{(i,v,j)}$ be the set of shortest paths saved by GRA algorithm. Given a target point $v \in N$, any edge $(x,y) \in A$, incident to $\widehat{N}(v)$ and not belonging to any shortest path in S, either does not belong to the optimal solution $\{\widehat{T}_1^*, \widehat{T}_2^*, \dots, \widehat{T}_m^*\}$ or there is an alternative optimal solution without this arc.*

Proof. Let us suppose that the arc (x,y) belongs to a tour $\widehat{T}_h^*$ of the optimal solution, with $x \in \widehat{N}(v)$, and let q be the point that coming just before x in $\widehat{T}_h^*$. There are the following two cases to consider:

1. *$\widehat{T}_h^*$ is the longest tour of the optimal solution and the other tours are shorter than it.*
   In this case we have that $\ell(\widehat{T}_h^*) > \ell(\widehat{T}_p^*)$, with $p = \{1,\dots,m\} \setminus \{h\}$, and the optimal solution value is equal to $\ell(\widehat{T}_h^*)$. We have to consider the following subcases:
   - $\ell(< q,x,y >) > \ell(S_{(q,v,y)})$.
     Since the arc (x,y) do not belong to $S_{(q,v,y)}$ than there exists another discretization point $r \in \widehat{N}(v)$ such that the path $< q,r,y >$ is shorter than $< q,x,y >$. This means that, by replacing $< q,x,y >$ with $< q,r,y >$ in $\widehat{T}_h^*$, we obtain a new shortest tour $\widehat{T}_{h'}^*$ and then a new solution $\{\widehat{T}_1^*, \widehat{T}_2^*, \dots, \widehat{T}_{h'}^*, \dots, \widehat{T}_m^*\}$ better than the optimal one. A contradiction.
   - $\ell(< q,x,y >) = \ell(S_{(q,v,y)})$.
     This situation occurs when $S_{(q,v,y)}$ can be build by using more different discretization points of $\widehat{N}(v)$. Let us suppose that $r \in \widehat{N}(v)$ is another discretization point of $\widehat{N}(v)$ such that $\ell(< q,x,y >) = \ell(< q,r,y >)$. This means that, by replacing $< q,x,y >$ with $< q,r,y >$ in $\widehat{T}_h^*$, we obtain another tour $\widehat{T}_{h'}^*$ with $\ell(\widehat{T}_h^*) = \ell(\widehat{T}_{h'}^*)$ and then $\{\widehat{T}_1^*, \widehat{T}_2^*, \dots, \widehat{T}_{h'}^*, \dots, \widehat{T}_m^*\}$ is an alternative optimal solution that does not contain the arc (x,y).
2. *$\widehat{T}_p^*$ is the longest tour of the optimal solution with $p \neq h$.*
   In this case we have that $\ell(\widehat{T}_h^*) \leq \ell(\widehat{T}_p^*)$. Moreover, since (x,y) does not belonging to any shortest path in S then $\ell(< q,x,y >) \geq \ell(S_{(q,v,y)})$. As a consequence, by replacing $< q,x,y >$ with $< q,r,y >$ in $\widehat{T}_h^*$, we obtain a new tour $\widehat{T}_{h'}^*$ with $\ell(\widehat{T}_{h'}^*) \leq \ell(\widehat{T}_h^*)$. This means that the solution $\{\widehat{T}_1^*, \widehat{T}_2^*, \dots, \widehat{T}_{h'}^*, \dots, \widehat{T}_m^*\}$ have the same cost of the optimal one and then it is an alternative optimal solution that does not contain the arc (x,y).

□

4 COMPUTATIONAL TESTS

In this section, we describe the results of our MIP model for mGTSP obtained during our computational test phase. The model was coded in C++ and solved using the IBM ILOG CPLEX 22.1.1 solver. A time limit of one hour, a thread limit of 4, a memory limit of 16 GB were imposed. Moreover, the relative and absolute MIP gap tolerance are set to 1e-7. All remaining CPLEX parameters were left to their default value. All tests were performed on an OSX platform (iMac 2020), running on an Intel Core i9-10910 processor clocked at 3.6 GHz with 64 GB of RAM.

The computational tests are carried out on 360 instances having 6, 8, 10 and 12 target points and three radius values: 0.25, 0.50, and 1.00. There are 30 instances for each combination of the number of target points and the radius values. This is a subset of the instances proposed by (Behdani and Smith, 2014) for the CETSP problem. For this set of instances, the optimal values of the CETSP solutions are available. So, since it is the smallest possible value as $c(T)$, we use it to calculate T^{max} as described in Section 2. It is worth noting that in CETSP all the neighborhoods must be covered by a single route whereas in *m*GTSP the same task is carried out using *m* routes. Therefore, it could be natural to think that the optimal solution value of CETSP is an upper bound of the optimal solution value of *m*GTSP but it's not always like this. Indeed, the use of the discretization points, in *m*GTSP, generates a discretization error, for each neighborhood, that must be added to the cost of the routes and then the optimal solution value of CETSP may be lower than the one of *m*GTSP.

Due to the complexity of *m*GTSP and to limit the size of the instances, we use three discretization points (that is $k = 3$) for each neighborhood. This means that for a problem with 10 target nodes, the model will solve an instance with $3 \times 10 + 2 = 32$ nodes. We start our computational study by evaluating the effectiveness of the GRA algorithm. In Figure 6 the percentages of nodes and arcs removed by the GRA algorithm are shown. The four scenarios, with 6, 8, 10 and 12 target points and radii 0.25, 0.5 and 1, are reported on the x-axis, each one composed of thirty instances. These results highlight the effectiveness of GRA which removes always more than 14% of nodes and 32% of edges. In more detail, the percentage of nodes removed ranges from 14.6%, in the scenario with 12 target points and a radius equal to 1, up to 27%, in the scenario with 6 target points and a radius equal to 0.25. A similar trend is observed for the arcs too but with higher percentage values. Indeed, the percentage of arcs removed ranges from 32.4%, in the scenario with 12 target points and a radius equal to 1, to 53.3%, in the scenario with 6 target points and a radius equal to 0.25. Figure 6 shows a clear trend in which GRA is more effective in the instances with the smaller radius and smallest number of targets in both the removal of nodes and arcs. Since the new graph produced by GRA contains a much smaller number of nodes and edges, the MIP model requires less computational time to find the optimal solution.

In Table 1 we evaluate the performance of the MIP model. Each row of the table contains values computed on a test scenario composed of 30 instances. The first four columns of the table report the characteristics of the instances that is: the length of the radius (*Radius*), the number of target points (*Target*), the number of nodes (*Nodes*) and arcs (*Arcs*). The next five columns report, for $m = 2$, the average objective value (*Obj*), the average computational time (*Time*), in seconds, the number of optimal (*#Opt*) and feasible (*#Feas*) solutions found by model and the average of the optimality gap (*Gap%*). The last five columns report the same information but with $m = 3$. Table 1 shows that, for $m = 2$, MIP finds the optimal solution on 291 out of 360 instances within the time limit of 1 hour. In particular, all the small instances with 20 and 26 nodes are solved optimally in less than 40 seconds whatever is the radius value. On the larger instances, with 32 and 38 nodes, MIP finds the optimal solution on 111 out of 180 instances. The optimality gap percentage is at most 2.19% for the instances with 32 nodes and lower than 9% for the instances with 38 nodes. These optimality gaps show that, in the largest instances with 38 nodes, the incumbent solution found by the model, is not too far from the optimal one when two drones are used. Finally, the average computational time is lower than 2450 seconds. The performance of the model worsens when we increase the number of drones used to 3. Indeed, MIP finds the optimal solution on 230 out of 360 instances for $m = 3$. In particular, all the small instances with 20 and 26 nodes are solved to optimality but the computational time grows up to 73 seconds. On the larger instances, with 32 and 38 nodes, MIP finds the optimal solution on 50 out of 180 instances and the optimality gap percentage reaches approximately 8% for the instances with 32 nodes, and 21% for the instances with 38 nodes. Finally, the average computational time grows up to 3290 seconds. These results highlight that the number of routes used impacts both the quality of the final solution found by the model and the computational time required to obtain it.

5 CONCLUSION

In this article, we studied the Multiple Close-Enough Traveling Salesman Problem (*m*CETSP), a variant of the Close-Enough Traveling Salesman Problem (CETSP), where multiple vehicles are used to visit a given number of target points. Due to the complexity of the problem we adopted a discretization schema that reduced *m*CETSP to the Multiple Generalized Traveling Salesman Problem (*m*GTSP). For this dis-

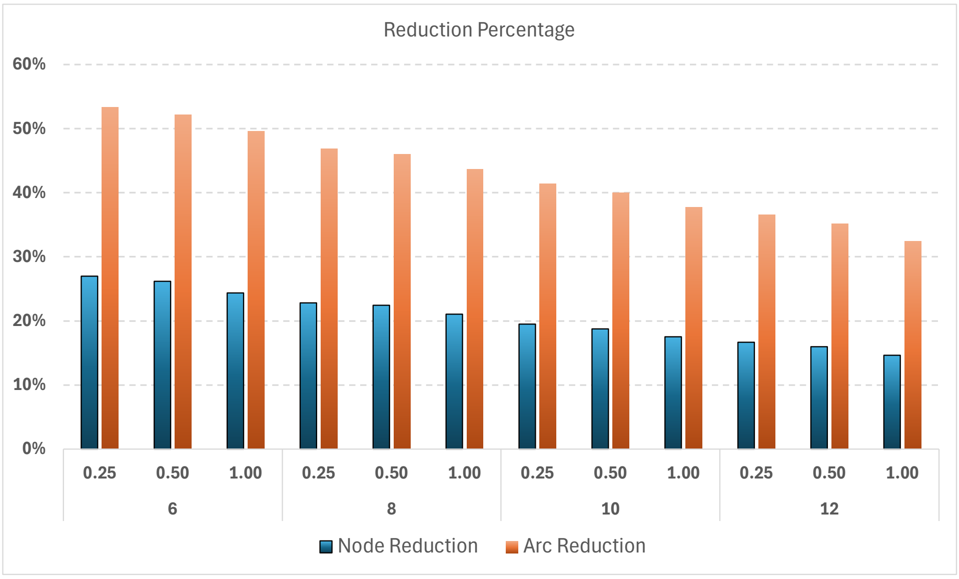

Figure 6: Percentage of nodes and arcs removed by GRA algorithm.

Table 1: Test Results of the MIP model.

				2 vehicles					3 vehicles				
Radius	**Target**	**Nodes**	**Arcs**	**Obj**	**Time**	**#Opt**	**#Feas**	**Gap%**	**Obj**	**Time**	**#Opt**	**#Feas**	**Gap%**
0.25	6	20	306	22.85	0.30	30	0	0.00%	21.37	0.40	30	0	0.00%
	8	26	552	25.52	18.94	30	0	0.00%	23.53	40.60	30	0	0.00%
	10	32	870	26.80	859.84	26	4	1.04%	24.57	2162.57	15	15	6.79%
	12	38	1260	28.70	2353.16	13	17	6.92%	26.20	3182.36	3	27	19.23%
0.50	6	20	306	22.53	0.40	30	0	0.00%	21.14	0.45	30	0	0.00%
	8	26	552	25.17	24.80	30	0	0.00%	23.30	46.33	30	0	0.00%
	10	32	870	26.38	978.08	24	6	1.54%	24.28	2264.88	14	16	7.65%
	12	38	1260	28.24	2442.80	13	17	7.47%	25.90	3089.85	3	27	19.95%
1.00	6	20	306	21.93	0.65	30	0	0.00%	20.69	0.56	30	0	0.00%
	8	26	552	24.53	39.83	30	0	0.00%	22.84	72.76	30	0	0.00%
	10	32	870	25.60	1123.15	23	7	2.19%	23.76	2438.22	13	17	8.27%
	12	38	1260	27.41	2441.69	12	18	8.88%	25.32	3288.45	2	28	21.68%

cretized version, we proposed a mixed integer linear Programming formulation which provides an upper bound for the optimal solution value of *m*CETSP, given the discretization error introduced. Additionally, we applied a graph reduction algorithm (GRA) to reduce the problem size, and we provided a proof of its correctness as the algorithm was originally proposed for a discretized version of CETSP. The computation tests performed on benchmark instances of the CETSP highlights i) the effectiveness of GRA in significantly reducing the number of nodes and arcs, and ii) the impact of the number of routes on the performance on the mathematical model. A potential direction for future research is to improve the mathematical model and develop effective metaheuristics.

ACKNOWLEDGEMENTS

We acknowledge financial support under the National Recovery and Resilience Plan (NRRP), Mission 4, Component 2, Investment 1.1, Call for tender No. 1409 published on 14.9.2022 by the Italian Ministry of University and Research (MUR), funded by the European Union – NextGenerationEU– Project Title “Smart Agriculture by Collaborative Robots Swarm” – CUP P2022E38SJ - Grant Assignment Decree No. 1379 adopted on 01.09.2023 by the Italian Ministry of Ministry of University and Research (MUR).

REFERENCES

Behdani, B. and Smith, J. C. (2014). An integer-programming-based approach to the close-enough traveling salesman problem. *INFORMS Journal on Computing*, 26(3):415 – 432.

Bianchessi, N., Mansini, R., and Speranza, M. G. (2018). A branch-and-cut algorithm for the team orienteering problem. *International Transactions in Operational Research*, 25(2):627–635.

Cariou, C., Moiroux-Arvis, L., Bendali, F., and Mailfert, J. (2024). Optimal route planning of an unmanned aerial vehicle for data collection of agricultural sensors. In *IEEE INFOCOM 2024-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6. IEEE.

Carrabs, F., Cerrone, C., Cerulli, R., and D'Ambrosio, C. (2017a). Improved upper and lower bounds for the close enough traveling salesman problem. In Au, M. H. A., Castiglione, A., Choo, K.-K. R., Palmieri, F., and Li, K.-C., editors, *Green, Pervasive, and Cloud Computing*, pages 165–177, Cham. Springer International Publishing.

Carrabs, F., Cerrone, C., Cerulli, R., and Gaudioso, M. (2017b). A novel discretization scheme for the close enough traveling salesman problem. *Computers and Operations Research*, 78:163 – 171.

Carrabs, F., Cerrone, C., Cerulli, R., and Golden, B. (2020). An adaptive heuristic approach to compute upper and lower bounds for the close-enough traveling salesman problem. *INFORMS Journal on Computing*, 32(4):1030–1048.

Coutinho, W. P., Nascimento, R. Q. d., Pessoa, A. A., and Subramanian, A. (2016). A branch-and-bound algorithm for the close-enough traveling salesman problem. *INFORMS Journal on Computing*, 28(4):752–765.

Dolias, G., Benos, L., and Bochtis, D. (2022). On the routing of unmanned aerial vehicles (uavs) in precision farming sampling missions. In *Information and Communication Technologies for Agriculture—Theme III: Decision*, pages 95–124. Springer.

Dong, J., Yang, N., and Chen, M. (2007). Heuristic approaches for a tsp variant: The automatic meter reading shortest tour problem. *Extending the horizons: Advances in computing, optimization, and decision technologies*, pages 145–163.

Gulczynski, D. J., Heath, J. W., and Price, C. C. (2006). The close enough traveling salesman problem: A discussion of several heuristics. *Perspectives in Operations Research: Papers in Honor of Saul Gass' 80 th Birthday*, pages 271–283.

Lei, Z. and Hao, J.-K. (2024). An effective memetic algorithm for the close-enough traveling salesman problem. *Applied Soft Computing*, 153:111266.

Mennell, W., Golden, B., and Wasil, E. (2011). A steiner-zone heuristic for solving the close-enough traveling salesman problem. In *2th INFORMS computing society conference: operations research, computing, and homeland defense*.

Mennell, W. K. (2009). *Heuristics for solving three routing problems: Close-enough traveling salesman problem, close-enough vehicle routing problem, and sequence-dependent team orienteering problem*. University of Maryland, College Park.

Wang, X., Golden, B., and Wasil, E. (2019). A Steiner zone variable neighborhood search heuristic for the close-enough traveling salesman problem. *Computers & Operations Research*, 101(1):200–219.

Yang, Z., Xiao, M.-Q., Ge, Y.-W., Feng, D.-L., Zhang, L., Song, H.-F., and Tang, X.-L. (2018). A double-loop hybrid algorithm for the traveling salesman problem with arbitrary neighbourhoods. *European Journal of Operational Research*, 265(1):65–80.

Yuan, B., Orlowska, M., and Sadiq, S. (2007). On the optimal robot routing problem in wireless sensor networks. *IEEE transactions on knowledge and data engineering*, 19(9):1252–1261.