CIRRELT-2025-22

An Exact Linearization-Based Refinement Algorithm to the Carrier-Vehicle Traveling Salesman Problem

Bruno S. Vieira
Enzo M. Frazzon
Leandro C. Coelho

July 2025

An Exact Linearization-Based Refinement Algorithm to the Carrier-Vehicle Traveling Salesman Problem

Bruno S. Vieira[2], Enzo M. Frazzon[3], Leandro C. Coelho[1]*

1. Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), School of Management, Université du Québec à Montréal
2. Department of Industrial Engineering, Instituto Federal de Santa Catarina, Brazil
3. Department of Industrial Engineering, Universidade Federal de Santa Catarina, Brazil

Abstract. Traditional single-vehicle systems often fail to provide efficient solutions in applications such as rescue operations, post-disaster relief, environmental mapping, and last-mile delivery. Coordinated systems composed of two echelons and heterogeneous vehicles with complementary capabilities offer a promising alternative. This paper focuses on the Carrier-Vehicle Traveling Salesman Problem, which involves a slow carrier vehicle (mothership) transporting a fast vehicle (e.g., a drone) that must takeoff from and return to the mothership to serve a set of customers. We propose a Mixed-Integer Nonlinear Programming formulation and an approximation that can be modeled as a Mixed-Integer Linear Programming model. An exact algorithm is proposed that corrects the approximation error of the nonlinear term. Combined with a set of valid inequalities, our branch-and-cut algorithm guarantees optimal solutions for the original nonlinear problem. We evaluate our algorithm on a wide range of benchmark instances from the literature, ranging from 10 to 200 customers. Our method outperforms all previous exact and heuristic approaches in the literature in terms of objective value or computational time. Moreover, we are the first to solve all benchmark instances with up to 25 customers optimally, and notably, finding a proven optimal solution for an instance with 35 customers, the largest one solved to optimality in the literature.

Keywords: Carrier-Vehicle Traveling Salesman Problem, Drone, Exact algorithm, Mixed-Integer Linear programming.

Acknowledgments: This project was partly supported by the Instituto Federal de Santa Catarina (IFSC) (BSV) and by the Canadian Natural Sciences and Engineering Research Council (NSERC) under grand 2025-03964. This support is greatly acknowledged.

---

* Corresponding author: leandro.coelho@fsa.ulaval.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2025

1 Introduction

Technological improvement has significantly transformed our social, economic, and personal lives. The use of unmanned aerial vehicles (UAVs), such as drones, exemplifies this transformation in environmental, defense, and civil applications, including soil and air quality monitoring, border surveillance, delivery to war zones, humanitarian logistics, and disaster management [2, 16, 18]. Although drone deliveries are not yet widely used, many companies such as Amazon, Google, and FedEx have carried out pilot tests, and this technology is likely to become more common in the near future [3]. These technological advances, on the other hand, impose operational challenges. In some applications, such as rescue operations, post-disaster relief, or environmental mapping, traditional single-vehicle systems may not yield the best solutions, which has motivated research involving multi-vehicle systems [5, 14, 19]. The same applies to last-mile delivery, which has also necessitated new perspectives on how transportation is carried out. Recent research has explored the possibility of utilizing secondary routes, potentially via drones [8, 17].

The cooperation and coordination of heterogeneous vehicles with complementary capabilities is an alternative with great potential [13]. Specifically, a two-vehicle system known as the carrier-vehicle has garnered attention in the past decade due to the development of planning and control algorithms, which are characterized by their broad applicability and simplicity of representation [5]. This system comprises a slow vehicle, commonly referred to as a carrier, which typically operates with unconstrained autonomy and transports a fast vehicle with limited autonomy. The fast vehicle performs deliveries or other activities to some targets, aiming to minimize the total travel time or distance. This problem was presented by Garone et al. [6], who studied the case in which the carrier moves freely in the Cartesian space, as in applications involving a ship carrying a helicopter. The authors then defined two variants: one in which the customer visiting sequences are predefined and the model must only prescribe the locations of the takeoff and landing points; in the second variant, called the Carrier/Carried Traveling Salesman Problem (CCTSP), the visiting sequence must also be determined. Later, this problem was referred to as the Carrier-Vehicle TSP (CVTSP) [4, 5] and the Mothership and Drone Routing Problem [15].

Gambella et al. [5] proposed a mixed-integer second-order conic programming (MISOCP) model and a ranking-based solution algorithm to solve new instances with up to 15 customers. Erdoğan and Yıldırım [4] presented some valid inequalities and improvements to that model and optimally solved instances with up to 20 customers. They also proposed an Iterated Local Search algorithm to provide solutions for instances with up to 50 customers. Poikonen and Golden [15] have also optimally solved instances with up to 20 customers using an MISOCP approach.

A variant of this problem, arising in truck transportation, was introduced by Murray and Chu [12] and is called the Flying Sidekick TSP. The difference is that takeoff and landing points can only occur at customers or the depot. A mixed-integer linear programming (MILP) formulation was introduced, and four heuristics were tested on 72 instances with 10 customer locations. In none of the instances was the MILP formulation able to find a proven optimal solution. Other problems involving drone routing combined with other vehicles can be found in Macrina et al. [11]. These problems, defined on a graph with limited takeoff and landing locations, are arguably simpler, as the distances between all locations are known a priori. In the CVTSP studied in this paper, the carrier vehicle travels on a continuous space; hence, the locations for takeoff and landing are decision variables, and thus, the distances to be traveled by the drone are also decision variables.

Several others have explored problems that share similarities with the CVTSP, particularly those involving the coordination between a mothership and aerial vehicles. In Li et al. [9], the authors address the Joint Vessel-UAV Routing Problem, where multiple ships, each equipped with a UAV, are tasked with inspecting offshore oil platforms. Although the problem bears conceptual resemblance to the CVTSP, its operational context and constraints differ significantly. Similarly, Zhuge et al. [20] consider the use of UAVs launched from ships for monitoring emissions, but the mothership's route is fixed over discrete points, in contrast with the continuous planning space considered in our problem. Amorosi et al. [1] study a problem involving the routing of multiple drones from a mothership, focusing on covering arcs rather than nodes. Finally, Irawan et al. [7] examine a continuous location and routing problem in offshore wind farms, where multiple delivery and retrieval trips are required for maintenance tasks, and route durations depend on the time needed for both maintenance and travel. While each of these contributions offers valuable insights into hybrid vehicle coordination, the present study differs in that it addresses a continuous-space variant and focuses on exact optimization.

In this paper, a two-vehicle system consists of a single slow mothership carrying a fast vehicle, such as a drone, which takes off from the mothership and must make deliveries to a set of customers before returning to the mothership. The objective is to minimize the total travel time of the mothership. We employ a nonlinear model and propose an approximate linearization cast as an MILP model. We introduce an exact algorithm based on branch-and-cut that iteratively refines any approximation errors and finds proven optimal solutions for the original problem, or provides a valid lower bound on the objective function value, in addition to valid solutions.

The results of our exact method outperform all previous exact and heuristic approaches in the literature, either in terms of objective value or computational time, across a wide range of instances from the literature. We solve to optimality all benchmark instances with up to 25 customers, and notably, prove an optimal solution for one instance with 35 customers, the largest known instance solved to optimality in the literature. We show that the linearization and correction procedure ensures convergence to globally optimal solutions and preserves the validity of the lower bound with respect to the original MINLP model. Finally, a set of valid inequalities is introduced based on the drone-mothership coordination, which is proved to be highly effective in improving computational performance and tightening bounds throughout the branch-and-cut process.

The remainder of this paper is organized as follows. In Section 2, we provide a formal description of the problem along with mathematical models, including the nonlinear and linearized formulations. Our main algorithmic developments are presented in Section 3. We describe the computational experiments used to validate our procedure on benchmark instances from the literature in Section 4, and our conclusions follow in Section 5.

2 Problem description and mathematical formulations

The problem consists of a single slow carrier, referred to as a mothership, which moves freely in the Cartesian space with speed V^m, carrying a fast vehicle, such as a drone, which takes off from the mothership and travels at speed $V^d > V^m$. The drone must make deliveries to a set $\mathcal{C}$ of customers and return to the mothership. The mothership must then return to its home position, node 0, also included in the set $\mathcal{C}$. The drone's travel distance and time must be less than its autonomy K and T, respectively. After each drone's return, it is assumed to be ready for a new trip with full autonomy. We define an event $e \in \mathcal{E}$ when the drone takes off ($e = 0$) and lands ($e = 1$).

The problem consists of determining the mothership's route and the locations of the events to visit all customers to minimize the total delivery time z. Formally, each customer $i \in \mathcal{C}$ is located at (X_i, Y_i) and the event of the drone to depart towards and return from customer i happens at location (x_{ie}, y_{ie}). We define a binary variable $u_{ieje'}$ equal to one if and only if events e of customer i and e' of customer j happen consecutively. The distances between two consecutive events of the mothership are measured by the variable $d^m_{ieje'}$, which is positive only when $u_{ieje'}$ is equal to 1. Distances between the mothership and customers, traveled by the drone, are represented by d^d_{ie}. Due to the distance equations, the model becomes an MINLP, which will be linearized and solved as an MILP. We present the MINLP formulation and its corresponding MILP model in the following sections. The MINLP notation is summarized in Table 1.

Sets	
$\mathcal{C}$	Set of customers plus origin/destination point.
$\mathcal{E}$	Set of events for each customer (landing ($e = 1$), takeoff ($e = 0$))
Parameters	
V^d	Drone speed
V^m	Mothership speed
K	Maximum drone distance per round trip
X_i, Y_i	Coordinates of customer i
Variables	
x_{ie}, y_{ie}	Coordinates of event e of customer i
$u_{ieje'}$	Binary variables that indicate if event e of customer i is followed by event e' of customer$j \geq i$
$d^m_{ieje'}$	Continuous variables representing the distance traveled by the mothership between event e of customer i and event e' of customer j
d^d_{ie}	Continuous variables representing the distance traveled by the drone between customer i and event e
z	Total travel time of the mothership

Table 1: Notation used in the MINLP model

2.1 Mixed-integer nonlinear programming formulation

The MINLP model (P) can be formulated as follows:

$$(P) \quad min \quad z = \frac{1}{V^m_{max}} \sum_{i\in\mathcal{C}} \sum_{j\in\mathcal{C}} \sum_{e\in\mathcal{E}} \sum_{\substack{e'\in\mathcal{E}: \\ (j>i\wedge e'\neq e)\ \vee \\ (j=i\ \wedge\ e'>e)}} d^m_{ieje'} \tag{1}$$

subject to

$$\sum_{e\in\mathcal{E}} d^d_{ie} \leq K \quad i \in \mathcal{C} \tag{2}$$

$$\sum_{e\in\mathcal{E}} d^d_{0e} = 0 \tag{3}$$

$$d^m_{ieje'} \geq \sqrt{(x_{ie} - x_{je'})^2 + (y_{ie} - y_{je'})^2} - (1 - u_{ieje'})M$$
$$i, j \in \mathcal{C}, e, e' \in \mathcal{E} : (j > i \wedge e' \neq e) \vee (j = i \wedge e' > e) \tag{4}$$

$$d^d_{ie} \geq \sqrt{(x_{ie} - X_i)^2 + (y_{ie} - Y_i)^2} \quad e \in \mathcal{E}, i \in \mathcal{C} \tag{5}$$

$$\sum_{j\in\mathcal{C}} \sum_{\substack{e'\in\mathcal{E}: \\ j>i \vee \\ (j=i \wedge e'>e)}} u_{ieje'} + \sum_{j\in\mathcal{C}} \sum_{\substack{e'\in\mathcal{E}: \\ j<i \vee \\ (j=i \wedge e'>e)}} u_{je'ie} = 2 \quad i \in \mathcal{C}, e \in \mathcal{E} \tag{6}$$

$$u_{i0i1} = 1 \quad i \in \mathcal{C} \tag{7}$$

$$\sum_{i\in\mathcal{S}} \sum_{e\in\mathcal{E}} \sum_{j\in\mathcal{S}} \sum_{\substack{e'\in\mathcal{E}: \\ j>i \vee \\ (j=i \wedge e'>e)}} u_{ieje'} \leq |\mathcal{S}| - 1 \quad \mathcal{S} \subset \mathcal{C}, |\mathcal{S}| \geq 2 \tag{8}$$

$$\frac{d^m_{i0i1}}{V^m_{max}} = \frac{\sum_{e\in\mathcal{E}} d^d_{ie}}{V^d_{max}} \quad i \in \mathcal{C}. \tag{9}$$

$$u_{ieje'} \in \{0, 1\} \quad i, j \in \mathcal{C}, e, e' \in \mathcal{E} :$$
$$(j > i \wedge e' \neq e) \vee (j = i \wedge e' > e). \tag{10}$$

The objective function (1) minimizes the total travel time. Constraints (2) ensure that each drone's round trip respects its autonomy. Equation (3) defines that the mothership departs from the origin and returns to the destination. Constraints (4) define the distances between two events of the mothership. Constraints (5) are analogous to (4), but define distances between the location of an event e and a customer i. Constraints (6) are the degree equations, which guarantee that every event is linked to two other events. Constraints (7) guarantee that after a takeoff to a customer, there will always be a landing from the same customer. Inequalities (8) are the subtour elimination constraints. Constraints (9) derive from Erdoğan and Yıldırım [4], who have proven that in an optimal solution, the carrier and the drone travel at their maximum speed and meet synchronously, that is, neither of them waits for the other. Thus, the distances covered by the mothership and the drone are proportional to their speeds. Constraints (10) define the binary nature of the u variables.

2.2 Mixed-integer linear programming formulation

The model's nonlinearities occur due to the distance constraints associated with $d^m_{ieje'}$ and d^d_{ie}. Next, we demonstrate how to linearize constraints (4) and (5). Figure 1 illustrates the situation at hand.

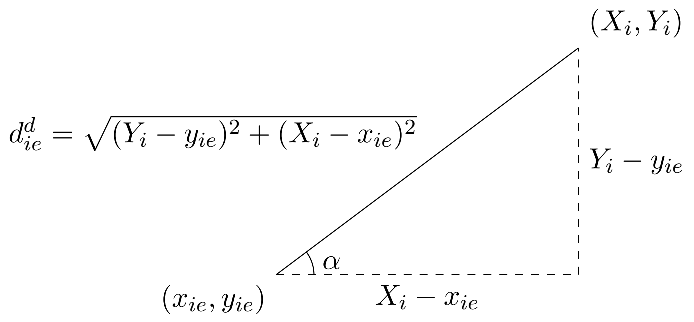


Figure 1: Representation of the variables d^d_{ie}

In the example of Figure 1, we can write $Y_i - y_{ie}$ and $X_i - x_{ie}$ as follows:

$$\begin{aligned} d^d_{ie} \sin\alpha &= Y_i - y_{ie} \\ d^d_{ie} \cos\alpha &= X_i - x_{ie}. \end{aligned} \tag{11}$$

We can also write the fundamental trigonometric identity, multiplied by d^d_{ie} on both sides of the equality, and develop as follows:

$$\begin{aligned} (\sin^2\alpha + \cos^2\alpha) d^d_{ie} &= d^d_{ie} \\ d^d_{ie} \sin\alpha \, \sin\alpha + d^d_{ie} \cos\alpha \, \cos\alpha &= d^d_{ie}. \end{aligned} \tag{12}$$

Using (11) on (12) we obtain:

$$d^d_{ie} = (Y_i - y_{ie}) \sin\alpha + (X_i - x_{ie}) \cos\alpha. \tag{13}$$

Finally, we can define:

$$\begin{aligned} d^d_{ie} &= \sqrt{(Y_i - y_{ie})^2 + (X_i - x_{ie})^2} \\ &= (Y_i - y_{ie}) \sin\alpha + (X_i - x_{ie}) \cos\alpha. \end{aligned} \tag{14}$$

It turns out that the values of variables x_{ie} and y_{ie} are not known, and therefore, we also do not know the angle α. However, we can ensure that if the angle is different from its correct value, the value computed using the right-hand side of equation (14) will underestimate the distance value, according to the following proposition.

Proposition 1: For any value of α, the following inequality holds:

$$\begin{aligned} &\sqrt{(Y_i - y_{ie})^2 + (X_i - x_{ie})^2} = d^d_{ie} \\ &\geq (Y_i - y_{ie}) \sin\alpha + (X_i - x_{ie}) \cos\alpha. \end{aligned} \tag{15}$$

Proof: To demonstrate that the maximum value of the right-hand side of the inequality is at most equal to the left-hand side of the inequality, we differentiate it with respect to α and set it to zero. To prove that it is a maximum point, we differentiate it again with respect to α and verify that the value is negative. Differentiating the right-hand side of the inequality with respect to α, we have:

$$\begin{aligned} &\frac{d}{d\alpha}\big[(Y_i - y_{ie}) \sin\alpha + (X_i - x_{ie}) \cos\alpha\big] \\ &= (Y_i - y_{ie}) \cos\alpha - (X_i - x_{ie}) \sin\alpha = 0. \end{aligned}$$

Dividing both sides by $\cos\alpha$ and solving for α:

$$(Y_i - y_{ie}) - \frac{(X_i - x_{ie}) \sin\alpha}{\cos\alpha} = 0$$
$$(X_i - x_{ie}) \tan\alpha = (Y_i - y_{ie}).$$
$$\begin{aligned} \alpha &= \arctan\left(\frac{Y_i - y_{ie}}{X_i - x_{ie}}\right) = \arcsin\left(\frac{Y_i - y_{ie}}{d^d_{ie}}\right) \\ &= \arccos\left(\frac{X_i - x_{ie}}{d^d_{ie}}\right). \end{aligned}$$

Then, we can write the right side of inequality (15) as:

$$\begin{aligned} &(Y_i - y_{ie}) \sin\arcsin\left(\frac{Y_i - y_{ie}}{d^d_{ie}}\right) \\ &+ (X_i - x_{ie}) \cos\arccos\left(\frac{X_i - x_{ie}}{d^d_{ie}}\right). \end{aligned}$$

Therefore, considering that $d^d_{ie} = \sqrt{(Y_i - y_{ie})^2 + (X_i - x_{ie})^2}$, a critical point of the right-hand side of (15) is:

$$\frac{(Y_i - y_{ie})^2 + (X_i - x_{ie})^2}{d^d_{ie}} = \sqrt{(Y_i - y_{ie})^2 + (X_i - x_{ie})^2},$$

which is equal to the left-hand side of inequality (15). It then remains to prove that this critical point is a maximum point. To do this, we show that the second derivative of the right-hand side of (15) is negative for any value of α:

$$\begin{aligned}
&\frac{d^2}{d\alpha^2}\left[(Y_i - y_{ie})\sin\alpha + (X_i - x_{ie})\cos\alpha\right] \\
&= -\left[(Y_i - y_{ie})\sin\alpha + (X_i - x_{ie})\cos\alpha\right] \\
&\leq 0.
\end{aligned}$$

Using (14), we get:

$$-\sqrt{(Y_i - y_{ie})^2 + (X_i - x_{ie})^2} \leq 0.$$

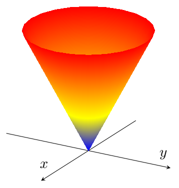
Showing that inequality (15) holds for any value of α. □

Corollary 1: *As a result of Proposition 1, we can conclude that it is possible to define as many inequalities* (15) *as desired, with different values of α for the same distance variable, without making the problem infeasible or losing optimality.*

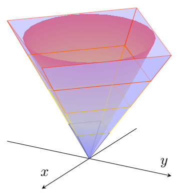
In fact, the more inequalities are defined (i.e., with several values of α), the closer the linearization gets to the original function. When the inequality is written for the correct angle α, it computes the exact distance; for all other values of α, the inequality provides an underapproximation of the distance value.

We can now define the MILP notation, which uses sets to indicate the angles of the linearization planes used to approximate each distance variable. We define d^{d*}_{ie} and $d^{m*}_{ieje'}$ as the linearized distance variables, to approximate d^{d}_{ie} and $d^{m}_{ieje'}$ of the original model. Let $\mathcal{L}_{ie}$ ($\mathcal{L}_{ieje'}$) be the set of linearization planes for each distance variable d^{d*}_{ie} ($d^{m*}_{ieje'}$). Let A_l be the angle in degrees of inequality l. In Figure 2, we illustrate this linearization. In Figure 2a, we have the representation of the continuous nonlinear distance constraints, such as modeled exactly by constraints (4) and (5). As inequalities define these constraints, the distance variables can take on any value within the internal region of the shape. To see this, take any point in the (x, y) horizontal plane, and move upwards parallel to the z axis until the cone boundary is reached. The distance from the (x, y) plane where $z = 0$ to the boundary of the cone determines the correct distance value. By continuing to move upwards (inside the cone), the distance only increases, so the objective function ensures that the minimum possible value is used and that the distance to the boundary of the cone is considered.

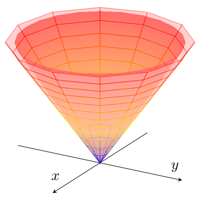
In Figures 2b and 2c, we show the domain of linearized distance variables, after the inclusion of $|\mathcal{L}| = 4$ and $|\mathcal{L}| = 12$ inequalities. Each inequality represents a plane that touches a single straight line of the original conic figure and limits the domain of the linearized variables. Observe that from the (x, y) plane where $z = 0$ and up until a plane defined by one inequality represents a lower bound on the actual distance. As more planes are added, the distance approximation is refined and gets closer to the actual distance. For a given location (i.e., a given (x, y) coordinate), our procedure described in Section 3 determines the plane defined by the exact angle required to compute the distance exactly.



(a) Example of d^{d}_{ie} variable



(b) Example of d^{d*}_{ie} variable, with $|\mathcal{L}| = 4$



(c) Example of d^{d*}_{ie} variable, with $|\mathcal{L}| = 12$

Figure 2: Linear and nonlinear representations of distance variables

With this understanding, we can establish another consequence of Proposition 1.

Corollary 2: *The optimal solution value of the MILP model is also a lower bound to that of the MINLP model.*

The MILP formulation is similar to the MINLP model, except that (1)–(5) must be replaced respectively by (16)–(20), defined next. In particular, constraints (19) and (20) are those that linearize model (P) using l inequalities for each distance variable, exploiting the observation from Corollary 1. All extra notation required for the MILP model is presented in Table 2. The MILP model (L) is then:

$$(L) \quad min \quad z = \frac{1}{V^{m}_{max}} \sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{C}} \sum_{e \in \mathcal{E}} \sum_{\substack{e' \in \mathcal{E}: \\ (j>i \wedge e' \neq e) \vee \\ (j=i \, \wedge \, e'>e)}} d^{m*}_{ieje'} \tag{16}$$

subject to

Sets	
$\mathcal{B}$	Set of customer pairs whose distance between them is less than $T(V^d_{\max} + V^m_{\max})$
$\mathcal{L}_{ie}$, $\mathcal{L}_{ieje'}$	Set of angles for inequalities (19) and (20)
Parameters	
A_l	Angle of linearization plane l tangent to the nonlinear distance curve, underestimating it
D_{ij}	Distance between nodes i and j, $D_{ij} = \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2}$
T	Maximum drone flying time per trip, $T = \frac{K}{V^d_{max}}$
Variables	
$d^{m*}_{ieje'}$	Underestimated distance between event e of customer i and event e' of customer j
d^{d*}_{ie}	Underestimated drone travel distance between the mothership and event e of customer i
t_i	Drone travel time to and from customer i

Table 2: MILP notation used in the model

$$\sum_{e \in \mathcal{E}} d^{d*}_{ie} \leq K \quad i \in \mathcal{C} \tag{17}$$

$$\sum_{e \in \mathcal{E}} d^{d*}_{0e} = 0 \tag{18}$$

$$\begin{aligned} d^{m*}_{ieje'} \geq & \cos\left(\frac{\pi A_l}{180}\right)(x_{ie} - x_{je'}) + \sin\left(\frac{\pi A_l}{180}\right)(y_{ie} - y_{je'}) \\ & - (1 - u_{ieje'})M \quad l \in \mathcal{L}_{ieje'}, (e, e') \in \mathcal{E} \\ & (j > i \wedge e' \neq e) \vee (j = i \wedge e' > e), i, j \in \mathcal{C} \end{aligned} \tag{19}$$

$$\begin{aligned} d^{d*}_{ie} \geq & \cos\left(\frac{\pi A_l}{180}\right)(x_{ie} - X_i) + \sin\left(\frac{\pi A_l}{180}\right)(y_{ie} - Y_i) \\ & l \in \mathcal{L}_{ie}, e \in \mathcal{E}, i \in \mathcal{C}. \end{aligned} \tag{20}$$

The objective function (16) and constraints (17) and (18) are equivalent to those of model (P). Constraints (19) define the underestimated distances between two linked points of the mothership, considering the angles at which some planes are tangent to the distance function. Constraints (20) are analogous to (19), but define underestimated distances between the location of an event e and a customer i.

2.3 Valid inequalities

Erdoğan and Yıldırım [4] have defined tighter bounds for the d^d variables, which we describe in (22) and (23). To help understand these inequalities, we define t_i in (21) as the travel time of the drone to and from customer i.

$$t_i = T\frac{\sum_{e\in\mathcal{E}} d^d_{ie}}{K} \quad i \in \mathcal{C} \tag{21}$$

$$d^d_{ie} \leq \frac{t_i(V^d_{max} + V^m_{max})}{2} \quad i \in \mathcal{C}, e \in \mathcal{E} \tag{22}$$

$$d^d_{ie} \geq \frac{t_i(V^d_{max} - V^m_{max})}{2} \quad i \in \mathcal{C}, e \in \mathcal{E}. \tag{23}$$

Since the distances are symmetric, the direction of the route in which customers are visited yields two solutions of the same cost; the models presented above already exploit this fact in the definition of the u variables, and our constraints consider only half of the symmetric graph. However, there is still another important symmetry that can be considered. It is always possible to reverse all takeoff and landing points and obtain an equivalent solution. This symmetry can be broken by:

$$\sum_{j\in\mathcal{C}\setminus\{0\}} j u_{01j0} \geq \sum_{k\in\mathcal{C}\setminus\{0\}} k u_{00k1}. \tag{24}$$

The next set of valid inequalities is based on the idea that for a predetermined sequence of customers, the total distance traveled by the drone, whether transported by the mothership or flying alone, must be greater than or equal to the total distance traveled by a TSP tour with this same sequence of customers. Figure 3 highlights this property, making it easy to see that the route taken by the drone, whether alone or transported, is not shorter than the corresponding TSP tour. Mathematically, this can be expressed as follows:

$$\begin{aligned}&\sum_{i\in\mathcal{C}}\sum_{j\in\mathcal{C}}\sum_{e\in\mathcal{E}}\sum_{\substack{e'\in\mathcal{E}:\\ j>i\ \wedge(e'\neq e)}} d^m_{ieje'} + \sum_{i\in\mathcal{C}}\sum_{e\in\mathcal{E}} d^d_{ie} \geq \\ &\sum_{i\in\mathcal{C}}\sum_{j\in\mathcal{C}}\sum_{e\in\mathcal{E}}\sum_{\substack{e'\in\mathcal{E}:\\ j>i\ \wedge(e'\neq e)}} u_{ieje'} D_{ij}.\end{aligned} \tag{25}$$

This idea can be refined by increasing the right-hand side of the inequality as follows. For each pair of connected customers, if the distance between them exceeds D_{ij}, the excess is added to the right-hand side of the expression. To model this, we define penalty variables $p_{ieje'}$, which capture this excess via inequalities (26):

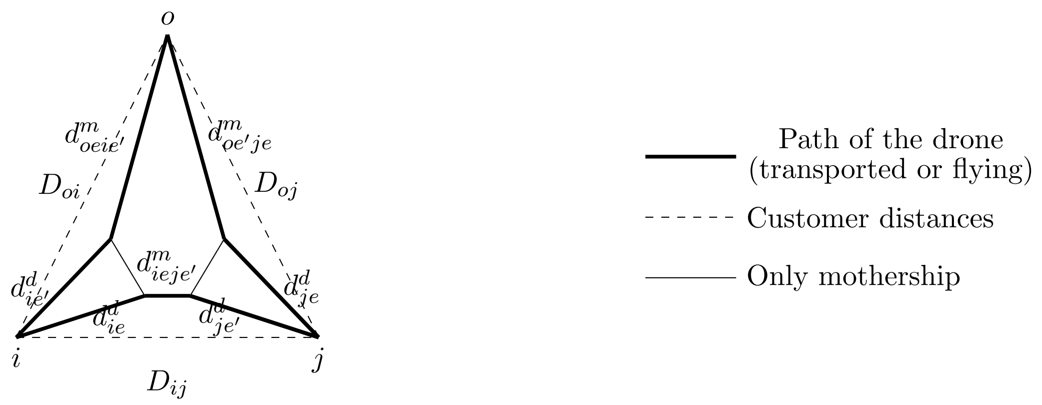


Figure 3: Example with two customers and one depot

$$\begin{aligned} p_{ieje'} \geq & d^d_{ie} + d^d_{je'} + d^m_{ieje'} - D_{ij} - (1 - u_{ieje'})M \\ & (e, e') \in \mathcal{E} : (j > i \land e' \neq e), i, j \in \mathcal{C}. \end{aligned} \tag{26}$$

We could introduce the sum of these $p_{ieje'}$ variables on the right-hand side of inequality (25). However, it will prove convenient to define a variable to represent this sum. We denote this variable by g, representing the total penalty associated with the pairs of connected customers, as defined in inequality (27). We then include this variable in inequality (25), resulting in (28).

$$g \geq \sum_{i\in\mathcal{C}} \sum_{j\in\mathcal{C}} \sum_{e\in\mathcal{E}} \sum_{\substack{e'\in\mathcal{E}: \\ j>i \ \land(e'\neq e)}} p_{ieje'} \tag{27}$$

$$\begin{aligned} & \sum_{i\in\mathcal{C}} \sum_{j\in\mathcal{C}} \sum_{e\in\mathcal{E}} \sum_{\substack{e'\in\mathcal{E}: \\ j>i \ \land(e'\neq e)}} d^m_{ieje'} + \sum_{i\in\mathcal{C}} \sum_{e\in\mathcal{E}} d^d_{ie} \geq \\ & \sum_{i\in\mathcal{C}} \sum_{j\in\mathcal{C}} \sum_{e\in\mathcal{E}} \sum_{\substack{e'\in\mathcal{E}: \\ j>i \ \land(e'\neq e)}} u_{ieje'} D_{ij} + g. \end{aligned} \tag{28}$$

This idea can be further refined by introducing variables that capture the penalties associated with three customers visited consecutively, with j being the one in the middle. Figure 4 illustrates this configuration.

Observe that $d^m_{ieje'} + d^m_{jeje'} + d^m_{jeke'} \geq D_{ik}$. To transform the path represented on the left side of the inequality into the path of interest, that is, one in which the drone is either transported or flying alone, we subtract $d^m_{jeje'}$ and add the sum of d^d_{je} and $d^d_{je'}$ to both sides, resulting in:

$$d^m_{ieje'} + d^d_{je} + d^d_{je'} + d^m_{jeke'} \geq D_{ik} + d^d_{je}. + d^d_{je'} - d^m_{jeje'}.$$

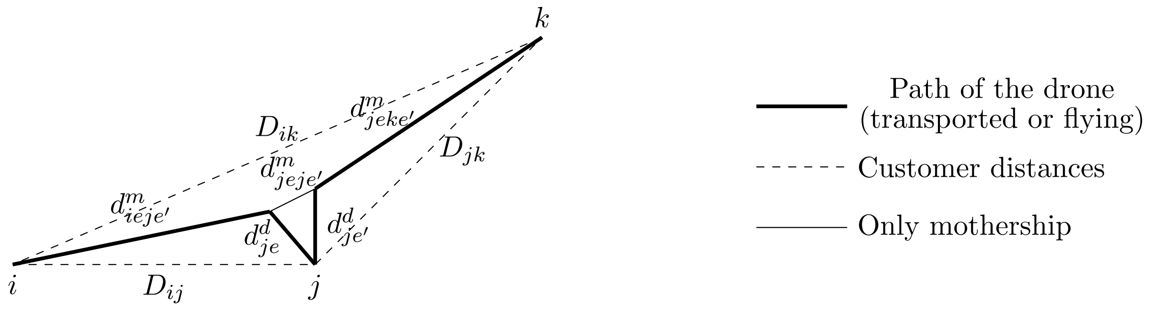


Figure 4: Example of three connected customers

Since there is a proportional relationship between $d^m_{jeje'}$ and the sum of d^d_{je} and $d^d_{je'}$ (see equations (9)), we can reformulate the right-hand side as follows:

$$d^m_{ieje'} + d^d_{je} + d^d_{je'} + d^m_{jeke'} \geq D_{ik} + d^m_{jeje'} \left(\frac{V^d_{max}}{V^m_{max}} - 1 \right).$$

Since the goal is to increase the right-hand side of inequality (28), which already considers D_{ij} and D_{jk}, we must subtract these constants to isolate the penalty δ_{ijk} associated with the three customers. Note that the greater the ratio between V^d_{max} and V^m_{max}, the greater the penalty tends to be, and the stronger the inequality becomes.

$$\begin{aligned} \delta_{ijk} \geq & D_{ik} + d^m_{jeje'} \left(\frac{V^d_{max}}{V^m_{max}} - 1 \right) - D_{ij} - D_{jk} \\ & - (2 - u_{ieje'} - u_{je'ke})M. \end{aligned} \tag{29}$$

We now introduce the variables δ_{ijk} for a set $\mathcal{R}$ of customer tuples with positive penalties for a given solution, and link their sum to the previously defined penalty variable g, as shown in (30).

$$g \geq \sum_{(i,j,k) \in \mathcal{R}} \delta_{ijk}. \tag{30}$$

The summation can be performed as long as penalties associated with the same edges are not included more than once. This means that, in any given solution, whether fractional or integer, a customer serving as the central node in one tuple cannot appear as either the central or an extreme node in another tuple. However, a customer positioned as an extreme node in one tuple may reappear as an extreme node in another. Thus, for any fractional or integer solution, we can evaluate which customer tuples generate positive penalties and solve a knapsack model to determine which tuples should be selected. Each tuple is defined by three indices: two extremes i and k, and a central customer j. A binary variable p_{ijk} indicates whether the tuple (i, j, k) is selected. Each tuple has an associated penalty δ_{ijk}, treated here as a parameter whose value is determined by inequalities (29). The resulting model (H) is then:

$$(H) \quad \max \sum_{(i,j,k)\in\mathcal{R}} p_{ijk}\delta_{ijk} \tag{31}$$

$$|\mathcal{R}| \sum_{\substack{(j=m)| \\ (i,j,k)\in\mathcal{R}}} p_{ijk} + \sum_{\substack{(i=m\vee k=m) \\ |(i,j,k)\in\mathcal{R}}} p_{ijk} \leq |\mathcal{R}| \quad \forall m \in \mathcal{C} \tag{32}$$

$$p_{ijk} \in \{0,1\}, \quad \forall (i,j,k) \in \mathcal{R}. \tag{33}$$

The objective function (31) maximizes the total penalty associated with the selected tuples. Constraints (32) ensure that each customer appears as a central node in at most one chosen tuple and does not simultaneously appear as an extreme node; at the same time, if a customer is not selected as a central node, they are allowed to appear as an extreme node in multiple tuples. Constraints (33) define the binary domain of the variables.

3 Solution algorithm

The model defined by constraints (6)–(10) and (16)–(20) cannot be fed into a general MILP solver as there is an exponential number of subtour elimination constraints (8) and an infinite number of distance constraints (19) and (20) as the values taken by the angle α are continuous. To this end, we develop an exact branch-and-cut algorithm where these constraints must be dynamically generated throughout the search process, only when they are found to be violated. Initially, none of these constraints are generated, and we solve the model by branch-and-bound. At a generic node of the search tree, the linear program containing the model with a subset of the subtour elimination constraints is solved. A search for violated inequalities (8) is performed. The ones found to be violated are dynamically generated and added to the model being solved, which is then reoptimized. The process continues until an integer solution without subtours is found.

After eliminating subtours, the algorithm corrects distance errors due to the linearization. To better understand this part of the algorithm, it is helpful to observe that every feasible solution of $u_{ieje'}$ variables for model (L) is also feasible for model (P). Observe that only constraints (2) and (3) limit the distance variables in model (P). Given that variables d_{ie}^{d*} in the linear formulation can assume values smaller than those of the d_{ie}^{d} variables in the nonlinear one, there may be a solution of the linear model

(L) that violates constraints (2) in model (P). However, given that there are no constraints on the maximum distance travelled by the mothership, it is always possible to increase its distance and get closer to each client, as much as necessary, until the variables d_{ie}^{d*} respect the range constraints of the drones.

Moreover, every feasible solution of x_{ie} and y_{ie} variables for model (P) is also feasible for model (L). Again, the constraints that limit x_{ie} and y_{ie} are those related to the autonomy of the drones. Indeed, all distances ($d_{ie}^{d}, d_{ie}^{d*}, d_{ie}^{m}$, and d_{ie}^{m*}) are functions of x_{ie} and y_{ie}; moreover, as seen before, $d_{ie}^{d} \geq d_{ie}^{d*}$ and $d_{ie}^{m} \geq d_{ie}^{m*}$. Then, the domains of variables x_{ie} and y_{ie} in the linear model necessarily include their domains in the nonlinear model. Hence, any solution for x_{ie} and y_{ie} in the nonlinear model is feasible to the linear model as it respects the drone autonomy constraints.

Upon finding an integer solution without subtours, the algorithm begins correcting the error caused by linearization. The values of the variables $u_{ieje'}$ of the linear model (L) are fixed, and model (P) is solved. From the values obtained for the variables x_{ie}, y_{ie}, d_{ie}^{d}, and $d_{ieje'}^{m}$, new constraints (19) and (20) are defined in model (L) with the angles A_l obtained from model (P). If the new objective value of the solution of model (L) does not converge to a value sufficiently close to that of model (P), it means that model (L) found an intermediate solution and the process is iterated by inserting the constraints associated with the angles from the current solution of model (L). The complete procedure is described in Algorithm 1.

4 Computational experiments

This section describes the extensive computational experiments conducted to evaluate our algorithm on established benchmark sets and compare it with competing exact and heuristic algorithms from the literature. Our algorithms were implemented in C++ and used Gurobi 12.0.2 as the solver. We run our tests on a machine equipped with two Intel Xeon 6548N processors clocked at 2.80 GHz and up to 128 GB of RAM. We allowed 1 hour of runtime for all instances except those with 200 customers, for which we ran for 2 hours.

We used all benchmark instances available from the works of Gambella et al. [5], Poikonen and Golden [15], and Erdoğan and Yıldırım [4], varying from 10 to 200 locations. The details of the instances are described next. Erdoğan and Yıldırım [4] used CPLEX 12.8.0 as a mixed-integer second-order conic optimization solver in their exact approaches and conducted experiments on the nodes of the Balena

Algorithm 1 Exact branch-and-cut algorithm

```
1: **Input:** $A_l$, percentage of fractional nodes to which subtour elimination constraints are added,
   percentage of fractional nodes to which penalty constraints are added
2: Begin optimization of model (L) with linearizations in angles $A_l$ for each distance variable
3: **while** there are nodes to be explored **do**
4:     **if** some $u_{ieje'}$ is fractional **then**
5:         With a given probability, separate subtour elimination constraints (8)
6:         With a given probability, find positive penalties $\delta_{ijk}$ and solve model (H) to separate constraints (30)
7:     **else if** the solution contains subtours **then**
8:         Add subtour elimination constraints (8)
9:     **else**
10:        Solve model (P) with variables $u_{ieje'}$ fixed from model (L)
11:        Define new linear cutting planes at the angles defined by the solution of model (P)
12:        **if** the objective value of model (P) is smaller than the incumbent of the model (L) **then**
13:            Provide the solution from model (P) to model (L)
14:        **else if** the solution of model (L) does not converge to that of model (P) **then**
15:            Define new linear cutting planes at the angles defined by the solution of model (P)
16:        **end if**
17:    **end if**
18: **end while**
19: **Return** the best found solution
```

computing cluster hosted at the University of Bath (Intel Xeon E5-2650 v2 CPUs @ 2.60 GHz). They implemented the model of Gambella et al. [5] on the same platform and limited both models to use a single thread. Poikonen and Golden [15] conducted their experiments on an Intel i7-6700 @ 3.40 GHz with 16 GB of RAM.

Based on preliminary experiments, we initiate model (L) with $|\mathcal{L}| = 4$ and $A_l = 0, 90, 180, 270$ for each distance variable. Subtour elimination constraints were separated for 10% of the fractional nodes using the CVRPSep package [10]. Model (H) for the knapsack problem was invoked for 2% of the fractional nodes for the instances of Gambella et al. [5] and Erdoğan and Yıldırım [4], and not used for the Poikonen and Golden [15] instances. The reason is the higher ratio between V^d_{max} and V^m_{max} observed in the Gambella et al. [5] and Erdoğan and Yıldırım [4] instances, which explain the performance gains of constaints (30) in those cases.

4.1 Impact of Valid Inequalities

To evaluate the impact of each group of valid inequalities (VIs) on the performance of our algorithm, we conducted a series of tests in which different versions of the model were applied to a diverse set of instances. These versions are built cumulatively, allowing us to isolate the individual contribution of each family of VIs. Recall that constraints (21)–(23) are derived from the literature, while the others are original from this work. Table 3 summarizes the performance of each version of the model across selected instances, reporting objective value (z), lower bound (LB), optimality gap, and computational time. The results illustrate the progressive improvement in solution quality and computational efficiency as more refined VIs are incorporated.

Table 3 presents a comparison of five versions of our algorithm, progressively incorporating VIs. A clear trend can be observed: the computational performance improves consistently up to VI (28), as stronger formulations lead to faster convergence and tighter bounds. However, the difference after the inclusion of inequalities (29) and (30) is more nuanced. For the Poikonen and Golden instances (i.e., those with prefix PK), this inclusion performs slightly worse in terms of total time. For the small- and mid-sized instances from Gambella et al. and Erdoğan and Yıldırım (up to 45 customers), the results are very similar between the two versions, with a marginal drop in performance in some cases. In contrast, for the largest Erdoğan and Yıldırım instances (with 50 customers), the inclusion of (29) and (30) yields a clear improvement, achieving lower gaps and better bounds. This indicates that the additional inequalities introduced are particularly effective for larger instances and when the ratio between V^d_{max} and V^m_{max} is greater.

4.2 Instances of Gambella et al. [5]

The instances proposed by Gambella et al. [5] are small and contain between 10 and 15 targets. For all of them, $V^d_{max} = 5$, $V^m_{max} = 1$, and $T = 1$ are defined. These instances are organized into four sets (SD, MD, LD, VLD), which are grouped into two categories based on the spatial distribution of customers. In the first group (SD and MD), customer coordinates are sampled uniformly within a $[-25, 25]^2$ square. The MD set differs from SD by enforcing a minimum distance between customers. The second group (LD and VLD) uses a wider $[-50, 50]^2$ region. Again, VLD differs from LD by introducing a minimum inter-customer distance. Optimal solutions are known for most instances in this set. Erdoğan and Yıldırım [4] compared the performance of their MISOCP model (called CVTSP1') with the original MISOCP formulation (called MISOCP) from Gambella et al. [5]. We report the numbers provided by Erdoğan and Yıldırım [4].

Table 3: Performance comparison of different versions of our algorithm.

Instance	Model (L)				Model (L) +(21)–(23)				Model (L)+(21)–(25)				Model (L)+(21)–(28)				Model (L)+(21)–(30)			
	z	LB	Gap (%)	Time (s)	z	LB	Gap (%)	Time (s)	z	LB	Gap (%)	Time (s)	z	LB	Gap (%)	Time (s)	z	LB	Gap (%)	Time (s)
PK10_10	223.39	223.39	0.00	37.85	223.39	223.39	0.00	34.00	223.39	223.39	0.00	10.65	223.39	223.39	0.00	3.51	223.39	223.39	0.00	4.54
PK15_5	242.04	242.04	0.00	3170	242.04	242.04	0.00	1835	242.04	242.03	0.00	152.12	242.04	242.03	0.00	8.31	242.04	242.03	0.00	13.47
PK20_0	282.56	164.39	41.82	3600	280.53	164.40	41.39	3600	280.53	258.74	7.77	3600	280.53	280.53	0.00	129.01	280.53	280.53	0.00	379.08
SD11_1	108.76	108.76	0.00	0.63	108.76	108.76	0.00	0.71	108.76	108.76	0.00	0.35	108.76	108.76	0.00	0.41	108.76	108.76	0.00	0.38
VLD16_3	306.00	305.99	0.00	2.57	306.00	306.00	0.00	2.02	306.00	305.99	0.00	0.92	306.00	306.00	0.00	0.75	306.00	305.99	0.00	0.84
LD20_1	376.88	376.88	0.00	7.00	376.88	376.88	0.00	8.45	376.88	376.88	0.00	2.60	376.88	376.87	0.00	1.00	376.88	376.87	0.00	0.99
SD26_1	393.55	393.55	0.00	133.87	393.55	393.55	0.00	150.46	393.55	393.55	0.00	31.43	393.55	393.55	0.00	16.75	393.55	393.55	0.00	18.34
SD36_1	477.51	432.26	9.48	3600	477.51	436.48	8.59	3600	477.51	443.60	7.10	3600	477.51	454.41	4.84	3600	477.51	451.38	5.47	3600
MD46_1	470.34	396.94	15.61	3600	470.28	398.49	15.26	3600	470.34	415.17	11.73	3600	470.28	437.45	6.98	3600	470.28	432.97	7.93	3600
LD51_1	504.69	400.79	20.59	3600	504.69	406.64	19.43	3600	504.69	429.89	14.82	3600	504.69	446.001	11.63	3600	504.60	452.47	10.33	3600
LD51_3	494.31	326.85	33.88	3600	494.31	350.30	29.13	3600	493.20	422.69	14.30	3600	493.21	442.816	10.22	3600	493.21	445.52	9.67	3600
MD51_2	497.64	408.37	17.94	3600	494.21	410.15	17.01	3600	495.86	429.16	13.45	3600	495.52	446.847	9.82	3600	494.70	444.31	10.18	3600
SD51_1	449.02	339.75	24.34	3600	448.45	344.39	23.20	3600	448.49	367.33	18.10	3600	448.45	383.401	14.50	3600	448.45	393.60	12.23	3600
Average	**371.28**	**316.92**	**12.59**	**2196.27**	**370.81**	**320.11**	**11.85**	**2094.64**	**370.86**	**339.78**	**6.71**	**1953.70**	**370.83**	**349.39**	**4.46**	**1673.83**	**370.76**	**350.11**	**4.29**	**1693.67**

While these instances are small, in a few cases, optimality has not been proven in the literature. For example, instance SD15_3 reached the time limit (3600 seconds) without a proven optimal solution in the earlier works. Our algorithm, however, successfully proved optimality for all instances (it took less than 15s for SD15_3), as reported in Table 4.

Table 4: Comparison of the performances of competing exact methods on the Gambella et al. [5] instances.

Instance	Optimal UB	MISOCP [5] Time (s)	CVTSP1' [4] Time (s)	Our algorithm Time (s)
SD11_1	108.757	1.96	17.06	0.47
SD11_2	113.919	8.62	68.36	0.83
SD11_3	125.186	1.93	15.16	0.41
MD11_1	146.851	0.19	5.28	0.22
MD11_2	132.116	0.60	9.53	0.24
MD11_3	133.416	0.76	10.86	0.35
LD11_1	311.549	0.06	2.45	0.16
LD11_2	345.195	0.23	3.84	0.15
LD11_3	299.534	0.21	4.52	0.19
VLD11_1	257.237	0.31	3.67	0.33
VLD11_2	324.747	0.08	4.36	0.13
VLD11_3	226.129	0.14	4.34	0.18
SD12_1	107.603	2.28	12.47	0.37
SD12_2	153.936	1.96	23.67	0.35
SD12_3	118.606	8.76	22.56	0.64
MD12_1	157.736	2.15	15.86	0.41
MD12_2	165.631	1.98	23.69	0.34
MD12_3	121.237	1.58	19.03	0.41
LD12_1	296.068	0.29	4.77	0.24
LD12_2	308.263	0.69	6.53	0.32
LD12_3	270.310	0.16	3.67	0.18
VLD12_1	326.036	0.14	4.69	0.18
VLD12_2	274.192	0.21	4.44	0.18
VLD12_3	281.940	0.50	4.81	0.22
SD13_1	116.117	4.12	22.84	0.35
SD13_2	136.855	137.52	919.78	1.59
SD13_3	121.473	13.11	36.53	0.54
MD13_1	150.894	15.25	63.75	0.64
MD13_2	130.994	16.92	42.75	0.39
MD13_3	150.368	12.72	29.38	0.47
LD13_1	261.637	1.43	10.98	0.33
LD13_2	294.855	2.16	8.69	0.26
LD13_3	307.339	2.39	7.64	0.28
VLD13_1	316.709	4.26	16.13	0.5
VLD13_2	239.259	1.56	8.08	0.23

Table 4 – Continued

Instance	Optimal UB	MISOCP [5] Time (s)	CVTSP1' [4] Time (s)	Our algorithm Time (s)
VLD13_3	281.329	0.98	7.61	0.27
SD14_1	128.137	201.05	128.36	0.76
SD14_2	124.663	152.53	134.48	0.94
SD14_3	138.071	103.62	69.44	0.52
MD14_1	146.951	24.23	65.61	0.47
MD14_2	163.589	54.06	83.95	0.53
MD14_3	153.006	38.34	44.95	0.43
LD14_1	319.795	5.84	7.94	0.36
LD14_2	282.915	23.94	28.98	0.44
LD14_3	301.595	5.35	8.42	0.54
VLD14_1	319.628	11.69	26.55	0.49
VLD14_2	300.168	2.38	10.14	0.49
VLD14_3	280.366	3.79	10.5	0.21
SD15_1	123.668	403.20	240.83	0.76
SD15_2	136.101	643.02	436.47	0.95
SD15_3	132.717	3600	3600.03	14.59
MD15_1	168.238	1476.65	789.94	1.29
MD15_2	136.935	1277.98	234.95	0.72
MD15_3	157.864	624.48	656.53	0.79
LD15_1	299.037	45.95	28.67	0.48
LD15_2	314.008	160.9	25.47	0.39
LD15_3	324.791	440.25	74.16	0.82
VLD15_1	295.334	2.18	15.61	0.45
VLD15_2	314.700	50.91	14.89	0.35
VLD15_3	264.946	5.16	13.63	0.33
SD16_1	145.096	3600.00	346.86	0.84
SD16_2	155.355	3600.00	2228.28	2.62
SD16_3	128.385	3600.00	2060.22	2.19
MD16_1	166.214	3600.00	1716.09	1.82
MD16_2	177.235	3232.09	420.25	0.94
MD16_3	164.369	3600.00	198.94	0.82
LD16_1	322.053	350.77	13.53	0.3
LD16_2	338.704	2726.22	75.34	0.67
LD16_3	353.867	2403.94	35.42	0.69
VLD16_1	379.909	624.76	35.06	0.49
VLD16_2	355.422	861.46	62.16	0.52
VLD16_3	305.995	3600	157.83	0.79
	Average	**575.07**	**216.26**	**0.75**

In addition to comparing our approach with exact methods, we also evaluated it against the heuristic algorithms proposed by the same authors. Table 5 presents this comparison, showing the upper bounds and times obtained by the Iterated Local Search (ILS) of Erdoğan and Yıldırım [4], the ranking-based solution (RBA) algorithm of Gambella et al. [5], and our exact method. The results show that, despite being an exact approach, our method achieved equal or better upper bounds in all instances while requiring significantly less computational time in most cases.

Table 5: Comparison of the performances of heuristics and our algorithm on the Gambella et al. [5] instances.

Instance	ILS [4]		RBA [5]		Our algorithm	
	UB	Time (s)	UB	Time (s)	UB	Time (s)
SD11_1	108.76	9.11	108.76	2.52	108.76	0.47
SD11_2	113.92	7.95	113.92	15.19	113.92	0.83
SD11_3	125.19	9.24	125.19	3.29	125.19	0.41
MD11_1	146.85	9.49	146.85	0.31	146.85	0.22
MD11_2	132.12	9.59	132.12	0.65	132.12	0.24
MD11_3	133.48	9.18	133.42	1.11	133.42	0.35
LD11_1	311.55	10.62	311.55	0.08	311.55	0.16
LD11_2	345.20	10.15	345.19	0.34	345.20	0.15
LD11_3	299.53	10.05	299.53	0.41	299.53	0.19
VLD11_1	257.24	10.51	257.24	0.38	257.24	0.33
VLD11_2	324.75	11.13	324.75	0.12	324.75	0.13
VLD11_3	226.13	10.92	226.13	0.21	226.13	0.18
SD12_1	107.60	12.44	107.60	2.96	107.60	0.37
SD12_2	154.02	12.25	153.94	2.24	153.94	0.35
SD12_3	118.61	12.47	118.61	12.80	118.61	0.64
MD12_1	157.85	13.31	157.74	2.44	157.74	0.41
MD12_2	165.74	13.51	165.63	2.44	165.63	0.34
MD12_3	121.24	12.45	121.24	2.05	121.24	0.41
LD12_1	296.07	12.55	296.07	0.25	296.07	0.24
LD12_2	308.26	12.59	308.26	0.85	308.26	0.32
LD12_3	270.31	14.08	270.31	0.22	270.31	0.18
VLD12_1	326.04	14.59	326.04	0.18	326.04	0.18
VLD12_2	274.19	15.67	274.19	0.30	274.19	0.18
VLD12_3	281.94	13.48	281.94	0.55	281.94	0.22
SD13_1	116.12	17.61	116.12	4.33	116.12	0.35
SD13_2	137.06	14.29	136.86	211.11	136.86	1.59
SD13_3	121.47	17.04	121.47	13.78	121.47	0.54
MD13_1	151.11	15.54	150.89	12.68	150.89	0.64
MD13_2	131.35	16.53	130.99	14.32	130.99	0.39
MD13_3	150.37	16.10	150.37	10.28	150.37	0.47
LD13_1	261.64	17.24	261.64	1.33	261.64	0.33
LD13_2	294.85	16.99	294.85	2.15	294.86	0.26
LD13_3	307.34	17.32	307.34	1.36	307.34	0.28
VLD13_1	316.74	19.59	316.71	4.30	316.71	0.50
VLD13_2	239.26	18.62	239.26	1.60	239.26	0.23
VLD13_3	281.33	18.24	281.33	1.03	281.33	0.27

Table 5 – Continued

Instance	ILS [4]		RBA [5]		Our algorithm	
	UB	Time (s)	UB	Time (s)	UB	Time (s)
SD14_1	128.14	21.65	128.14	199.36	128.14	0.76
SD14_2	124.66	19.45	124.66	123.07	124.66	0.94
SD14_3	138.07	20.05	138.07	56.12	138.07	0.52
MD14_1	146.95	19.52	146.95	20.23	146.95	0.47
MD14_2	163.59	18.82	163.59	36.10	163.59	0.53
MD14_3	153.01	19.44	153.01	24.68	153.01	0.43
LD14_1	319.80	21.41	319.80	4.29	319.80	0.36
LD14_2	282.92	19.64	282.91	17.30	282.92	0.44
LD14_3	301.60	22.22	301.60	2.93	301.60	0.54
VLD14_1	319.63	23.04	319.63	6.89	319.63	0.49
VLD14_2	300.17	25.53	300.17	2.28	300.17	0.49
VLD14_3	280.37	25.28	280.37	2.77	280.37	0.21
SD15_1	125.79	24.03	123.67	277.22	123.67	0.76
SD15_2	136.64	25.18	136.10	259.27	136.10	0.95
SD15_3	134.37	23.38	132.72	3,600.00	132.72	14.59
MD15_1	168.24	23.17	168.24	1,468.48	168.24	1.29
MD15_2	136.94	23.15	136.94	478.55	136.94	0.72
MD15_3	157.86	23.72	157.86	413.02	157.86	0.79
LD15_1	299.04	26.32	299.04	28.30	299.04	0.48
LD15_2	314.01	24.94	314.01	20.97	314.01	0.39
LD15_3	324.79	25.70	324.79	152.35	324.79	0.82
VLD15_1	295.33	28.47	295.33	1.90	295.33	0.45
VLD15_2	314.70	30.90	314.70	10.64	314.70	0.35
VLD15_3	264.95	29.80	264.95	5.90	264.95	0.33
SD16_1	145.10	32.21	145.10	673.47	145.10	0.84
SD16_2	155.58	30.37	155.36	2,837.07	155.36	2.62
SD16_3	128.38	31.21	128.38	3,600.00	128.38	2.19
MD16_1	166.21	34.78	166.21	3,600.00	166.21	1.82
MD16_2	177.50	30.46	177.23	1,176.74	177.23	0.94
MD16_3	164.37	29.75	164.37	2,661.46	164.37	0.82
LD16_1	322.05	31.75	322.05	43.64	322.05	0.30
LD16_2	338.70	31.20	338.70	104.96	338.70	0.67
LD16_3	354.89	29.38	353.87	1,024.65	353.87	0.69
VLD16_1	379.91	31.04	379.91	65.06	379.91	0.49
VLD16_2	355.42	35.41	355.42	121.26	355.42	0.52
VLD16_3	306.32	31.14	305.99	722.48	305.99	0.79
Average	**221.41**	**19.80**	**221.30**	**335.74**	**221.30**	**0.75**

These results demonstrate a clear advantage of our exact algorithm. While previous models, namely, the MISOCP model of Gambella et al. [5] and the CVTSP1' model of Erdoğan and Yıldırım [4], struggled with several instances (with some exceeding 3600 seconds), our approach consistently solved all instances to optimality (most of them in under 3 seconds). The average runtime of our method was only 0.75 seconds, compared to 216.26 seconds for CVTSP1' and 575.07 seconds for the MISOC model. These strong results highlight the effectiveness of the proposed formulation and its potential for scaling to larger problem instances.

4.3 Instances of Erdoğan and Yıldırım [4]

Erdoğan and Yıldırım [4] generated larger instances using the same scheme of Gambella et al. [5]. The new instances contain from 16 to 50 customers. For those with fewer than 20 customers, Erdoğan and Yıldırım [4] tested their method as well as that of Gambella et al. [5]. For instances with 25 to 50 customers, Erdoğan and Yıldırım [4] only used the ILS procedure. In what follows, we compare the results of our exact algorithm against all these results.

4.3.1 Instances with 16–20 customers

For the instances with 16–20 customers, the results of Table 6 confirm the superiority of our method. The model of Gambella et al. [5] could not prove optimality for any of the 60 instances in this set, reaching the 2-hour time limit for all instances with an average gap of 142.13%. The model proposed by Erdoğan and Yıldırım [4] performs significantly better, finding an optimal solution for all but one instance, with an average runtime of 695 seconds and an average gap of 0.05%. Our method quickly proved optimality for all instances with an average time of 1.45 seconds.

Table 6: Comparison of the performances of competing exact methods on the Erdoğan and Yıldırım [4] instances with 16 to 20 customers.

	MISOCP [5]				CVTSP1' [4]				Our algorithm			
Instance	UB	LB	Gap (%)	Time (s)	UB	LB	Gap (%)	Time (s)	UB	LB	Gap (%)	Time (s)
SD17_1	295.33	234.76	25.80	7,200	285.58	285.58	0.00	36.98	285.59	285.59	0.00	0.53
SD17_2	357.74	204.54	74.90	7,200	341.49	341.49	0.00	190.38	341.49	341.49	0.00	0.70
SD17_3	286.28	155.62	83.96	7,200	270.31	270.31	0.00	107.42	270.31	270.31	0.00	0.77
SD18_1	351.79	222.06	58.42	7,200	329.47	329.47	0.00	104.69	329.47	329.47	0.00	0.76
SD18_2	338.97	206.07	64.49	7,200	314.49	314.49	0.00	58.15	314.49	314.49	0.00	0.82
SD18_3	381.83	179.94	112.20	7,200	341.80	341.80	0.00	262.13	341.80	341.80	0.00	1.21
SD19_1	390.30	191.75	103.55	7,200	359.70	359.70	0.00	1,327.38	359.70	359.70	0.00	2.56
SD19_2	372.80	179.37	107.84	7,200	312.15	312.15	0.00	51.94	312.15	312.15	0.00	0.46
SD19_3	388.31	212.70	82.57	7,200	379.31	379.31	0.00	429.47	379.31	379.31	0.00	1.11
SD20_1	396.96	166.23	138.80	7,200	365.93	365.93	0.00	2,317.80	365.93	365.93	0.00	2.46
SD20_2	366.12	223.80	63.59	7,200	355.71	355.71	0.00	2,857.17	355.71	355.71	0.00	1.65

Table 6 – Continued

	MISOCP [5]				CVTSP1' [4]				Our algorithm			
Instance	UB	LB	Gap (%)	Time (s)	UB	LB	Gap (%)	Time (s)	UB	LB	Gap (%)	Time (s)
SD20_3	431.74	144.93	197.90	7,200	341.76	341.76	0.00	403.03	341.76	341.76	0.00	1.15
SD21_1	434.67	209.44	107.54	7,200	371.82	371.82	0.00	78.72	371.82	371.82	0.00	0.90
SD21_2	419.55	133.77	213.65	7,200	373.80	373.80	0.00	2,723.54	373.80	373.80	0.00	3.08
SD21_3	350.28	102.39	242.11	7,200	350.28	350.28	0.00	3,850.20	350.28	350.28	0.00	3.48
MD17_1	340.54	243.18	40.04	7,200	322.21	322.21	0.00	44.80	322.21	322.21	0.00	0.46
MD17_2	375.15	272.96	37.44	7,200	350.08	350.08	0.00	91.71	350.08	350.08	0.00	0.85
MD17_3	405.54	200.93	101.83	7,200	356.27	356.27	0.00	33.61	356.27	356.27	0.00	0.42
MD18_1	379.83	211.10	79.93	7,200	346.43	346.43	0.00	122.22	346.43	346.43	0.00	0.71
MD18_2	341.80	217.00	57.52	7,200	335.27	335.27	0.00	60.85	335.27	335.27	0.00	1.09
MD18_3	439.54	213.39	105.98	7,200	376.27	376.27	0.00	107.19	376.28	376.28	0.00	0.59
MD19_1	426.68	146.58	191.09	7,200	364.45	364.45	0.00	157.89	364.45	364.45	0.00	0.71
MD19_2	424.95	197.06	115.64	7,200	370.01	370.01	0.00	119.73	370.01	370.01	0.00	0.54
MD19_3	424.84	158.78	167.56	7,200	345.01	345.01	0.00	82.63	345.01	345.01	0.00	0.93
MD20_1	421.84	174.40	141.89	7,200	341.91	341.91	0.00	249.25	341.91	341.91	0.00	0.99
MD20_2	359.41	153.24	134.54	7,200	332.47	332.47	0.00	676.15	332.47	332.47	0.00	1.96
MD20_3	399.58	176.75	126.07	7,200	342.91	342.91	0.00	128.94	342.91	342.91	0.00	0.69
MD21_1	451.89	109.98	310.88	7,200	363.66	363.66	0.00	792.71	363.66	363.66	0.00	2.76
MD21_2	418.01	114.14	266.23	7,200	344.70	344.70	0.00	463.34	344.70	344.70	0.00	1.22
MD21_3	455.66	86.31	427.95	7,200	363.71	363.71	0.00	442.70	363.71	363.71	0.00	2.90
LD17_1	391.63	217.67	79.92	7,200	380.11	380.11	0.00	125.73	380.11	380.11	0.00	1.61
LD17_2	358.52	234.40	52.95	7,200	323.68	323.68	0.00	46.76	323.68	323.68	0.00	0.55
LD17_3	391.46	155.23	152.18	7,200	349.20	349.20	0.00	56.40	349.20	349.20	0.00	0.50
LD18_1	394.76	164.37	140.16	7,200	329.43	329.43	0.00	27.52	329.43	329.43	0.00	0.57
LD18_2	342.50	142.67	140.06	7,200	287.78	287.78	0.00	92.67	287.78	287.78	0.00	0.55
LD18_3	389.83	188.46	106.85	7,200	368.89	368.89	0.00	160.26	368.89	368.89	0.00	0.88
LD19_1	348.81	166.71	109.23	7,200	313.50	313.50	0.00	520.81	313.50	313.50	0.00	1.11
LD19_2	384.38	157.53	144.01	7,200	359.53	359.53	0.00	141.38	359.53	359.53	0.00	0.60
LD19_3	338.75	133.64	153.48	7,200	323.57	323.57	0.00	647.55	323.57	323.57	0.00	1.20
LD20_1	407.15	152.59	166.82	7,200	376.88	376.88	0.00	521.94	376.88	376.88	0.00	0.99
LD20_2	389.09	191.69	102.98	7,200	334.67	334.67	0.00	1,630.62	334.67	334.67	0.00	1.76
LD20_3	382.88	149.82	155.56	7,200	334.97	334.96	0.00	1,299.39	334.96	334.96	0.00	1.39
LD21_1	495.10	177.07	179.61	7,200	400.73	400.73	0.00	321.13	400.73	400.73	0.00	1.54
LD21_2	408.63	134.42	203.99	7,200	348.67	337.64	3.27	7,200.00	348.67	348.67	0.00	16.22
LD21_3	463.03	118.78	289.81	7,200	365.98	365.98	0.00	1,434.91	365.98	365.98	0.00	1.76
VLD17_1	363.22	261.95	38.66	7,200	337.94	337.94	0.00	94.53	337.94	337.94	0.00	0.64
VLD17_2	342.21	204.69	67.18	7,200	303.03	303.03	0.00	30.62	303.03	303.03	0.00	0.46
VLD17_3	398.66	253.57	57.22	7,200	371.44	371.44	0.00	32.65	371.44	371.44	0.00	0.42
VLD18_1	359.39	228.17	57.51	7,200	326.79	326.79	0.00	25.95	326.79	326.79	0.00	0.54
VLD18_2	369.31	204.31	80.76	7,200	344.19	344.19	0.00	389.31	344.19	344.19	0.00	1.29
VLD18_3	358.17	185.15	93.45	7,200	320.31	320.31	0.00	40.90	320.31	320.31	0.00	0.51
VLD19_1	399.65	190.05	110.29	7,200	337.27	337.27	0.00	68.31	337.27	337.27	0.00	0.58
VLD19_2	399.02	138.14	188.85	7,200	314.04	314.04	0.00	64.92	314.04	314.04	0.00	0.51

Table 6 – Continued

	MISOCP [5]				CVTSP1' [4]				Our algorithm			
Instance	UB	LB	Gap (%)	Time (s)	UB	LB	Gap (%)	Time (s)	UB	LB	Gap (%)	Time (s)
VLD19_3	425.96	187.38	127.33	7,200	392.18	392.18	0.00	559.74	392.18	392.18	0.00	1.21
VLD20_1	387.35	115.88	234.28	7,200	362.99	362.99	0.00	576.73	362.99	362.99	0.00	1.08
VLD20_2	435.12	154.86	180.97	7,200	355.56	355.56	0.00	133.79	355.56	355.56	0.00	0.90
VLD20_3	381.21	92.50	312.13	7,200	310.86	310.86	0.00	168.59	310.86	310.86	0.00	1.22
VLD21_1	466.43	115.97	302.19	7,200	388.97	388.97	0.00	3,804.03	388.97	388.97	0.00	1.97
VLD21_2	505.43	144.73	249.22	7,200	387.86	387.86	0.00	1,224.05	387.86	387.86	0.00	4.02
VLD21_3	419.35	113.95	268.03	7,200	337.64	337.64	0.00	1,944.37	337.64	337.64	0.00	1.70
Average	**392.08**	**175.33**	**142.13**	**7,200.00**	**345.63**	**345.44**	**0.05**	**695.97**	**345.63**	**345.63**	**0.00**	**1.45**

In addition to the comparison with exact methods, we also evaluated the performance of our algorithm against the ILS of Erdoğan and Yıldırım [4] for the same set of instances. Table 7 presents this comparison. Although ILS is relatively fast and produces good-quality solutions, our exact method consistently achieved either better or equivalent upper bounds for all instances, and did so in significantly less time on average (1.45 seconds versus 52.35 seconds). This result reinforces the efficiency and robustness of our approach, even when compared with state-of-the-art heuristics.

Table 7: Comparison of the performances of heuristics and our algorithm on the [4] instances with 16 to 20 customers.

Instance	ILS [4]		Our algorithm	
	UB	Time (s)	UB	Time (s)
SD17_1	285.66	36.03	285.59	0.53
SD17_2	341.49	40.97	341.49	0.70
SD17_3	270.31	37.01	270.31	0.77
SD18_1	329.47	46.34	329.47	0.76
SD18_2	314.49	44.48	314.49	0.82
SD18_3	342.03	44.04	341.80	1.21
SD19_1	360.02	48.37	359.70	2.56
SD19_2	312.15	54.76	312.15	0.46
SD19_3	379.31	52.82	379.31	1.11
SD20_1	365.93	56.73	365.93	2.46
SD20_2	355.92	53.10	355.71	1.65
SD20_3	341.76	62.59	341.76	1.15
SD21_1	371.82	71.05	371.82	0.90
SD21_2	375.85	66.34	373.80	3.08
SD21_3	350.93	65.70	350.28	3.48
MD17_1	322.21	38.20	322.21	0.46
MD17_2	350.08	36.55	350.08	0.85
MD17_3	356.27	40.59	356.27	0.42

Table 7 – Continued

Instance	ILS		Our Method	
	UB	Time (s)	UB	Time (s)
MD18_1	346.47	46.71	346.43	0.71
MD18_2	335.27	42.63	335.27	1.09
MD18_3	376.41	45.62	376.28	0.59
MD19_1	364.45	48.23	364.45	0.71
MD19_2	370.01	51.52	370.01	0.54
MD19_3	345.01	48.04	345.01	0.93
MD20_1	341.91	59.70	341.91	0.99
MD20_2	332.49	54.35	332.47	1.96
MD20_3	342.91	60.96	342.91	0.69
MD21_1	363.77	69.91	363.66	2.76
MD21_2	344.71	69.76	344.70	1.22
MD21_3	363.71	74.47	363.71	2.90
LD17_1	380.11	38.45	380.11	1.61
LD17_2	323.68	37.38	323.68	0.55
LD17_3	349.20	36.23	349.20	0.50
LD18_1	329.43	45.54	329.43	0.57
LD18_2	287.78	41.00	287.78	0.55
LD18_3	369.28	40.96	368.89	0.88
LD19_1	313.64	52.77	313.50	1.11
LD19_2	359.53	51.09	359.53	0.60
LD19_3	323.57	47.94	323.57	1.20
LD20_1	376.88	57.20	376.88	0.99
LD20_2	334.67	58.10	334.67	1.76
LD20_3	334.96	60.60	334.96	1.39
LD21_1	400.73	78.02	400.73	1.54
LD21_2	349.17	72.73	348.67	16.22
LD21_3	366.11	66.89	365.98	1.76
VLD17_1	337.94	39.93	337.94	0.64
VLD17_2	303.03	39.06	303.03	0.46
VLD17_3	371.44	41.58	371.44	0.42
VLD18_1	326.80	45.35	326.79	0.54
VLD18_2	344.79	44.39	344.19	1.29
VLD18_3	320.31	50.47	320.31	0.51
VLD19_1	337.27	51.07	337.27	0.58
VLD19_2	314.04	51.60	314.04	0.51
VLD19_3	392.18	50.39	392.18	1.21
VLD20_1	362.99	57.22	362.99	1.08
VLD20_2	355.60	59.94	355.56	0.90
VLD20_3	310.86	57.39	310.86	1.22
VLD21_1	389.40	63.77	388.97	1.97
VLD21_2	387.86	72.35	387.86	4.02
VLD21_3	337.64	63.90	337.64	1.70

Average	**345.73**	**52.35**	**345.63**	**1.45**

4.3.2 Instances with 25–50 customers

For the 72 instances containing between 25 and 50 customers, our algorithm proves optimality for 23, one of which of 35 customers. This is the first time an instance of this size has had a proven optimal solution. The average gaps for instances with 25, 30, 35, 40, 45, and 50 customers are respectively 0.0%, 0.53%, 4.06%, 7.37%, 8.39%, and 11.33%, which is significantly better than the competing ILS, as shown in Table 8.

The ILS of Erdoğan and Yıldırım [4] was executed 10 times for each instance. The average and best results of the 10 runs are shown alongside the average runtime for all 10 executions. We then show the time if took our algorithm to reach the average UB of Erdoğan and Yıldırım [4]; then, we show the final results for our algorithm including the UB, LB, gap, and time. Our algorithm has matched or improved the BKS in 69 out of 72 instances. The final results of ILS report an average time of 322 seconds to reach the average value 441.86, while our algorithm reaches this average UB in 77.95 seconds.

Table 8: Comparison of the performances of heuristics and our algorithm on the Erdoğan and Yıldırım [4] instances with 25 to 50 customers.

	ILS [4]			Our algorithm				
Instance	Average UB	BKS	Time (s)	Time to reach average UB (s)	UB	LB	Gap (%)	Total time (s)
SD26_1	393.56	393.56	114.25	0.38	393.55	393.55	0.00	27.30
SD26_2	413.12	413.12	109.32	7.03	413.12	413.12	0.00	7.03
SD26_3	378.69	378.69	112.36	0.25	377.82	377.82	0.00	103.09
SD31_1	408.99	408.99	186.34	0.41	406.69	399.67	1.73	3,600.53
SD31_2	352.59	352.59	157.93	0.91	348.83	332.69	4.63	3,600.70
SD31_3	390.10	388.68	161.96	3.22	388.68	388.68	0.00	79.64
SD36_1	479.52	479.52	240.47	19.70	477.51	452.73	5.19	3,600.38
SD36_2	413.77	413.77	234.43	1.88	412.96	392.63	4.92	3,600.17
SD36_3	384.90	384.90	246.15	81.32	384.89	360.77	6.27	3,600.98
SD41_1	485.61	485.61	328.54	2.41	478.33	445.41	6.88	3,602.17
SD41_2	452.43	452.43	329.00	113.14	452.43	410.11	9.35	3,601.00
SD41_3	455.08	455.08	365.14	3.33	452.86	429.46	5.17	3,600.60
SD46_1	504.77	504.77	460.09	177.94	504.61	472.10	6.44	3,601.45
SD46_2	417.48	417.48	456.20	187.08	416.15	367.92	11.59	3,600.66
SD46_3	509.55	509.55	466.12	75.37	506.30	470.34	7.10	3,602.90

Table 8 − Continuation

	ILS [4]			Our algorithm				
Instance	Average UB	BKS	Time (s)	Time to reach average UB (s)	UB	LB	Gap (%)	Total time (s)
SD51_1	449.58	449.58	611.47	17.85	448.45	394.07	12.13	3,601.13
SD51_2	448.58	448.58	620.98	–	448.73	376.23	16.16	3,600.53
SD51_3	512.18	512.18	631.85	1.07	508.49	461.89	9.16	3,600.14
MD26_1	345.22	345.22	122.42	5.04	345.22	345.22	0.00	5.04
MD26_2	417.27	416.73	104.45	5.12	416.73	416.73	0.00	42.88
MD26_3	413.35	412.09	109.62	10.26	412.09	412.09	0.00	37.93
MD31_1	424.24	424.24	173.22	40.36	424.24	424.24	0.00	289.39
MD31_2	413.55	413.47	171.95	3.65	413.47	413.47	0.00	1,151.87
MD31_3	473.10	473.10	172.87	6.00	473.10	473.10	0.00	335.48
MD36_1	431.14	431.14	244.57	0.67	426.02	426.02	0.00	3,086.68
MD36_2	419.99	419.99	246.84	26.55	419.85	409.54	2.45	3,600.26
MD36_3	375.21	375.21	225.14	0.68	368.41	346.91	5.83	3,601.39
MD41_1	448.76	448.76	313.00	158.04	448.15	419.22	6.46	3,600.87
MD41_2	427.43	427.43	326.27	318.00	427.43	398.06	6.87	3,600.20
MD41_3	474.80	474.80	338.76	49.02	472.19	441.66	6.46	3,601.36
MD46_1	475.46	475.46	447.51	4.83	470.28	432.49	8.04	3,601.26
MD46_2	485.85	485.85	442.42	15.34	481.06	437.60	9.03	3,602.33
MD46_3	527.79	527.79	449.39	–	527.92	480.48	8.98	3,602.25
MD51_1	471.55	471.55	648.69	112.66	467.44	415.31	11.15	3,600.07
MD51_2	498.48	498.48	602.42	89.12	493.47	444.71	9.88	3,602.33
MD51_3	509.48	504.62	631.86	89.56	504.62	457.46	9.35	3,601.08
LD26_1	363.06	362.36	116.08	0.90	362.36	362.36	0.00	22.41
LD26_2	387.36	387.36	111.56	0.23	387.36	387.35	0.00	6.78
LD26_3	371.37	369.81	109.64	0.77	369.81	369.81	0.00	16.34
LD31_1	412.46	412.32	158.96	24.65	412.32	412.32	0.00	315.12
LD31_2	382.81	382.81	162.85	0.40	382.71	382.71	0.00	202.81
LD31_3	426.00	426.00	165.45	0.31	426.00	426.00	0.00	571.70
LD36_1	447.29	447.29	233.80	26.15	447.29	428.28	4.25	3,600.77
LD36_2	436.95	436.95	240.79	472.75	436.95	416.68	4.64	3,600.64
LD36_3	473.32	473.32	231.24	6.00	473.32	466.30	1.48	3,600.61
LD41_1	460.80	460.80	329.50	38.01	460.10	438.24	4.75	3,600.57
LD41_2	482.70	482.70	327.44	1.00	482.70	459.56	4.79	3,600.29
LD41_3	491.53	491.53	343.40	8.44	481.59	458.33	4.83	3,600.74
LD46_1	483.54	483.54	442.82	35.00	483.54	445.44	7.88	3,602.98
LD46_2	488.92	488.92	451.89	26.79	486.61	449.95	7.53	3,601.42
LD46_3	541.70	541.70	455.37	313.69	537.06	508.82	5.26	3,600.53
LD51_1	504.69	504.69	606.34	1,883.00	504.69	453.04	10.23	3,603.78
LD51_2	498.95	498.95	672.29	113.14	497.88	438.83	11.86	3,602.71
LD51_3	493.20	493.20	650.29	–	493.69	444.93	9.88	3,602.32
VLD26_1	344.97	344.97	106.89	2.75	344.70	344.70	0.00	55.71
VLD26_2	417.36	415.45	110.17	2.51	414.97	414.97	0.00	191.67

Table 8 – Continuation

	ILS [4]			Our algorithm				
Instance	Average UB	BKS	Time (s)	Time to reach average UB (s)	UB	LB	Gap (%)	Total time (s)
VLD26_3	386.05	386.05	109.15	0.24	386.03	386.03	0.00	17.44
VLD31_1	407.16	407.16	166.33	1.02	406.19	406.19	0.00	252.69
VLD31_2	407.27	407.27	170.40	6.44	407.27	407.27	0.00	69.17
VLD31_3	401.43	401.43	181.80	0.40	401.43	401.43	0.00	42.84
VLD36_1	421.04	421.04	226.01	14.33	421.04	406.17	3.53	3,601.10
VLD36_2	415.35	415.35	232.88	0.54	414.01	395.19	4.55	3,600.78
VLD36_3	407.88	406.44	234.82	1.80	406.21	383.55	5.58	3,601.30
VLD41_1	460.33	459.25	331.93	33.88	459.25	433.20	5.67	3,601.58
VLD41_2	420.32	420.32	338.05	55.31	419.15	381.09	9.08	3,601.23
VLD41_3	450.69	450.69	323.29	242.89	450.69	418.24	7.20	3,600.82
VLD46_1	425.48	425.48	446.74	42.85	423.65	373.84	11.76	3,601.89
VLD46_2	490.50	490.50	440.85	0.50	490.50	453.52	7.54	3,602.02
VLD46_3	473.92	473.92	477.84	41.42	472.23	427.30	9.51	3,600.56
VLD51_1	494.95	494.95	626.69	38.07	493.89	447.64	9.37	3,602.51
VLD51_2	464.93	464.93	662.18	188.45	460.22	398.90	13.33	3,602.32
VLD51_3	494.45	494.45	640.56	125.86	488.39	422.57	13.48	3,600.93
Average	**441.86**	**441.65**	**322.78**	**77.95**	**440.25**	**416.45**	**5.00**	**2,547.11**

4.4 Instances of Poikonen and Golden [15]

We have also conducted experiments on the diverse instance set of Poikonen and Golden [15]. There are two groups of instances: one in which customer locations are uniformly distributed, and the other in which customers are clustered. These 325 instances vary in size according to the number of customers, ranging from 10 up to 200 target points. For all of them $V^d_{max} = 2$, $V^m_{max} = 1$, and $T = 20$, which are considerably different from the previous instances.

Table 9 reports the objective values and computation times for the uniformly distributed instances, obtained by the exact and heuristic methods proposed by Poikonen and Golden [15] (a second-order conic programming approach combined with a branch-and-bound technique as an exact algorithm, a greedy sequence heuristic (GS), a greedy sequence with local search heuristic (GSLS), and a partial solve with greedy insert heuristic (PSGI)). Only the GS algorithm provided solutions for all instances. We also report detailed results for our algorithm.

As shown in Table 9, our algorithm consistently achieves superior or equivalent performance across almost all uniformly distributed instances. For small instances (e.g., 10 or 15 locations), our method

Table 9: Comparison between the methods of Poikonen and Golden [15] and our algorithm for the uniformly distributed instances

Size	Poikonen and Golden [15]								Our algorithm		
	Exact		GS		GSLS		PSGI				
	UB	Time (s)	UB	Time (s)	UB	Time (s)	UB	Time (s)	UB	Gap (%)	Time (s)
10	213.74	1.54	214.54	0.009	214.40	2.15	215.14	0.30	213.74	0.00	1.14
15	240.74	18.8	245.20	0.014	243.60	6.81	248.74	1.02	240.74	0.00	22.15
20	252.23	700.22	260.78	0.024	258.52	17.28	263.65	3.33	260.37	0.00	183.78
30	-	-	302.82	0.048	301.28	55.34	299.91	36.00	290.59	6.45	3303
50	-	-	371.10	0.157	369.22	293.98	-	-	354.77	14.80	3602
100	-	-	511.60	1.331	-	-	-	-	498.59	20.41	3609
200	-	-	698.99	16.80	-	-	-	-	696.05	22.52	7220

matches the optimal objective values reported by Poikonen and Golden [15], with comparable or slightly higher computational times. For larger instances (more than 30 locations), where Poikonen and Golden [15] did not report results for the exact method due to time limitations, our method continues to demonstrate competitive performance, achieving better solutions than all heuristic approaches.

The only case in which our objective value is slightly higher than that reported by the exact method of Poikonen and Golden [15] occurs for instances with 20 locations (260.37 vs. 252.23). We discussed with these authors, but the details of the executions are no longer available. One explanation could be that the results reported come from a subset of instances, possibly excluding those for which optimal solutions could not be found within the 1000-second time limit, rather than the full benchmark set of 20 instances. Despite this gap, comparing the runtimes of our algorithm on these instances against theirs, we observe that our runtimes are almost four times shorter.

Table 10 presents the results for the clustered instances. As can be observed, our method outperforms the approaches proposed by the authors in nearly all instance sizes. The only exception is the 15-customer instances where our method required more time than the exact method of Poikonen and Golden [15]. We achieved an identical UB but did not prove optimality for all instances of this group, resulting in a very small gap of 0.42%. For all other larger instance sizes, our method achieved significantly better upper bounds. Notably, for the instance with 100 customers, our method again performed significantly better than theirs. However, we suspect that they do not report the correct values, as they are identical to those reported for the uniform instances of the same size. This suggests a possible transcription error during data compilation.

Table 10: Comparison between the methods of Poikonen and Golden [15] and our algorithm for the clustered instances

Size	Poikonen and Golden [15]								Our algorithm		
	Exact		GS		GSLS		PSGI				
	UB	Time (s)	UB	Time (s)	UB	Time (s)	UB	Time (s)	UB	Gap (%)	Time (s)
10	242.106	12.585	245.518	0.026	244.982	2.335	243.187	0.365	242.106	0	2.77
15	252.732	559.557	258.422	0.037	257.782	7.115	254.932	1.53	252.732	0.42	1036
20	-	-	265.013	0.065	264.707	16.69	264.356	14.495	260.066	7.90	3463.96
30	-	-	277.63	0.163	277.003	55.342	-	-	272.167	13.32	3600
50	-	-	298.854	0.341	298.142	229.996	-	-	292.921	15.95	3600
100	-	-	511.596	1.331	-	-	-	-	339.949	21.47	3600

Erdoğan and Yıldırım [4] also tested their exact models as well as that of Gambella et al. [5] on the instances proposed by Poikonen and Golden [15]. Table 11 presents a comparison of the results obtained by these methods and our algorithms. These results highlight the robustness and scalability of our method, which not only provides optimality guarantees for the smaller instances but also outperforms the state-of-the-art heuristics for the largest ones.

Table 11: Comparison of exact methods on the instances of Poikonen and Golden [15]

Type	Size	Poikonen and Golden [15]		MISOCP [5]		CVTSP1' [4]		Our algorithm	
		Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)
Uniform	10	0.00	1.54	0.00	54.56	0.71	1724.13	0.00	1.14
	15	0.00	18.80	7.78	5897.45	19.46	7200.00	0.00	22.15
	20	0.00	700.22	48.13	7200.00	-	-	0.00	183.78
Clustered	10	0.00	12.59	0.00	31.04	0.00	2713.42	0.00	2.77
	15	0.00	559.56	1.35	3939.31	22.45	7200.00	0.37	1036.08
	20	-	-	16.10	7200.00	-	-	7.90	3463.96

5 Conclusions

In this paper we have proposed a new exact algorithm for the Carrier-Vehicle Traveling Salesman Problem (CVTSP). Our method is based on linearizing and approximating a nonlinear model, which, when combined with an iterative correction procedure, guarantees optimality for the original nonlinear problem. The algorithm operates based on a branch-and-cut scheme, which is accelerated by existing and new valid inequalities.

Extensive computational experiments have demonstrated the effectiveness of our approach. Compared to state-of-the-art exact and heuristic methods, our algorithm showed significantly superior performance across a wide range of benchmark instances. For small instances, including those proposed by Gambella et al. [5] and Erdoğan and Yıldırım [4], our method consistently achieved optimal solutions in a fraction of the time required by competing exact and heuristic algorithms. For mid-sized instances (16–25 customers), our algorithm outperformed all other exact and heuristics methods in terms of runtime while maintaining solution quality. Moreover, for large-scale instances (up to 50 customers), our method matched or improved the best-known solutions in most cases, providing tight lower bounds – a key advantage over heuristics. We proved optimality for 23 out of 72 instances of these sizes for the first time, while Erdoğan and Yıldırım [4] did not prove an optimal solution for any of these instances. Moreover, we proved optimality for an instance with 35 locations for the first time.

Our algorithm also produced strong results for the large instances of Poikonen and Golden [15]. For instances with up to 15 customers, our algorithm matched the optimal solutions obtained by their exact approach. For larger instances (30 to 200 customers), our method consistently outperformed all heuristic procedures presented by Poikonen and Golden [15], highlighting its robustness and scalability.

These results confirm the potential of our algorithm to serve not only as an exact method for solving CVTSP instances but also as a tool for evaluating the quality of the many heuristic solutions available in the literature.

References

[1] L. Amorosi, J. Puerto, and C. Valverde. A multiple-drone arc routing and mothership coordination problem. *Computers & Operations Research*, 159:106322, 2023.

[2] L. Babel. Curvature-constrained traveling salesman tours for aerial surveillance in scenarios with obstacles. *European Journal of Operational Research*, 262(1):335–346, 2017.

[3] Y. Y. Chan, K. K. Ng, T. Wang, K. K. Hon, and C.-H. Liu. Near time-optimal trajectory optimisation for drones in last-mile delivery using spatial reformulation approach. *Transportation Research Part C: Emerging Technologies*, 171:104986, 2025.

[4] G. Erdoğan and E. A. Yıldırım. Exact and heuristic algorithms for the carrier-vehicle traveling salesman problem. *Transportation Science*, 55(1):101–121, 2021.

[5] C. Gambella, A. Lodi, and D. Vigo. Exact solutions for the carrier-vehicle traveling salesman problem. *Transportation Science*, 52(2):320–330, 2018.

[6] E. Garone, R. Naldi, A. Casavola, and E. Frazzoli. Cooperative mission planning for a class of carrier-vehicle systems. In *49th IEEE Conference on Decision and Control (CDC)*, pages 1354–1359. IEEE, 2010.

[7] C. A. Irawan, S. Salhi, and H. K. Chan. A continuous location and maintenance routing problem for offshore wind farms: Mathematical models and hybrid methods. *Computers & Operations Research*, 144:105825, 2022.

[8] P. Kitjacharoenchai, B.-C. Min, and S. Lee. Two echelon vehicle routing problem with drones in last mile delivery. *International Journal of Production Economics*, 225:107598, 2020.

[9] Y. Li, S. Wang, S. Zhou, and Z. Wang. A mathematical formulation and a tabu search heuristic for the joint vessel-UAV routing problem. *Computers & Operations Research*, 169:106723, 2024.

[10] J. Lysgaard, A. N. Letchford, and R. W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100(2):423–445, 2004.

[11] G. Macrina, L. P. Pugliese, F. Guerriero, and G. Laporte. Drone-aided routing: A literature review. *Transportation Research Part C: Emerging Technologies*, 120:102762, 2020.

[12] C. C. Murray and A. G. Chu. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies*, 54: 86–109, 2015.

[13] R. M. Murray. Recent research in cooperative control of multivehicle systems. *Journal of Dynamic Systems, Measurement, and Control*, 129:571, 2007.

[14] B. E. Oruc and B. Y. Kara. Post-disaster assessment routing problem. *Transportation Research Part B: Methodological*, 116:76–102, 2018.

[15] S. Poikonen and B. L. Golden. The mothership and drone routing problem. *INFORMS Journal on Computing*, 32(2):249–262, 2020.

[16] F. Ramadhan, C. A. Irawan, S. Salhi, and Z. Cai. The truck traveling salesman problem with drone and boat for humanitarian relief distribution in flood disaster: Mathematical model and solution methods. *European Journal of Operational Research*, 322(1):270–291, 2025.

[17] F. Senna, L. C. Coelho, R. Morabito, and P. Munari. An exact method for a last-mile delivery routing problem with multiple deliverymen. *European Journal of Operational Research*, 317(2): 550–562, 2024.

[18] Y. Yin, L. Qing, D. Wang, T. C. E. Cheng, and J. Ignatius. Exact solution method for vehicle-and-drone cooperative delivery routing of blood products. *Computers & Operations Research*, 164:106559, 2024.

[19] G. Zhang, N. Zhu, S. Ma, and J. Xia. Humanitarian relief network assessment using collaborative truck-and-drone system. *Transportation Research Part E: Logistics and Transportation Review*, 152:102417, 2021.

[20] D. Zhuge, J. Du, L. Zhen, S. Wang, and P. Wu. Ship emission monitoring with a joint mode of motherships and unmanned aerial vehicles. *Computers & Operations Research*, 179:107012, 2025.