Exact and Heuristic Algorithms for the Carrier-Vehicle Traveling Salesman Problem

Güneş Erdoğan* E. Alper Yıldırım†

17 June 2019, revised on 17 January 2020

Abstract

This paper presents new structural properties for the Carrier-Vehicle Traveling Salesman Problem. The authors provide a new mixed integer second order conic optimization formulation, with associated optimality cuts based on the structural properties, and an Iterated Local Search (ILS) algorithm. Computational experiments on instances from the literature demonstrate the superiority of the new formulation to the existing models and algorithms in the literature, and the high quality solutions found by the ILS algorithm.

Keywords— traveling salesman problem, multi-vehicle systems, mixed integer second order conic optimization

1 Introduction

Several complex mission planning operations necessitate the coordination and cooperation of heterogeneous vehicles with different but complementary capabilities (see, e.g., Murray (2007)). For instance, the recent technological advances in the capabilities of unmanned aerial vehicles (UAVs), also known as drones, facilitate their use together with other ground vehicles or naval vessels in order to improve the effectiveness, speed, range, safety, and cost of operations in various humanitarian, ecological, environmental, and military applications.

In this paper, we focus on a two-vehicle system consisting of a slow but large vehicle, referred to as the Carrier (e.g., a ship), with a virtually unlimited operational capability, and a faster but smaller vehicle, referred to as the Vehicle, with a limited operational capability (e.g., a helicopter or a drone). The Carrier is capable of transporting, deploying,

*School of Management, University of Bath, Claverton Down, Bath, BA2 7AY, United Kingdom. E-mail: G.Erdogan@bath.ac.uk

†School of Mathematics, The University of Edinburgh, Peter Guthrie Tait Road, Edinburgh, EH9 3FD, United Kingdom. E-mail: E.A.Yildirim@ed.ac.uk

recovering, and servicing the Vehicle. Such systems are frequently used in search-and-rescue, surveillance, monitoring, and logistics operations. In each such operation, there is a set of target points to be visited by the Vehicle for the purpose of rescuing victims, taking pictures and certain measurements, or performing delivery and pick-up operations. For each target point, the Vehicle should leave the Carrier at some take-off point, visit the target point, and return to the Carrier for servicing, refueling or recharging purposes without exhausting its limited operational capability. We focus on the problem of planning and coordinating the routes of the Carrier and the Vehicle in such a way that all of the target points are visited in minimum total time starting from an origin and ending at a prespecified destination. This problem, called the Carrier-Vehicle Traveling Salesman Problem (CVTSP), has been introduced by Garone et al. (2010b).

The CVTSP is particularly applicable in time-critical search-and-rescue operations conducted by naval security or military units as well as post-disaster operations after marine oil spills. Any improvement of the planning process would translate into more lives saved, reduced environmental damage, and decreased cost of operations.

Solving the CVTSP requires the determination of the sequence of the target points to be visited, which is inherently a combinatorial problem, and the computation of the take-off and landing points for each target point, which is a continuous problem. Indeed, under the assumption that the Vehicle and Carrier speeds are identical, the CVTSP reduces to the minimum-cost Hamiltonian path problem, or the Euclidean Traveling Salesman Problem if the origin and destination are identical. Therefore, the CVTSP is, in general, NP-hard.

We next briefly review the literature on the CVTSP. A simpler version of the CVTSP has been introduced in Garone et al. (2008), where the order of the target points to be visited is a priori fixed, referred to as the Carrier Vehicle Problem (CVP). The authors obtain closed-form solutions for some special cases with up to two target points. Building on the results from these simpler cases, they propose a heuristic for an arbitrary number of target points. The more general version of the CVTSP, in which the order of the target points is not a priori given, is studied in Garone et al. (2010b, 2011). The authors observe that the simpler version of the CVTSP with a fixed order of target points can be formulated as a convex optimization problem. They propose a two-stage heuristic method, in which a near-optimal solution to the Euclidean Traveling Salesman Problem on the target points is computed in the first stage by employing a polynomial-time approximation scheme and the CVTSP is solved by fixing this order in the second stage. For the resulting solution, they establish a worst-case approximation error bound, which depends on the approximation error in the first stage as well as some data-dependent parameters. In Garone et al. (2011), the authors propose enumerating all possible sequences of target points and solving the CVTSP with each fixed sequence for instances with up to 5 target points.

In Gambella et al. (2018), an exact solution approach based on a mixed integer second order conic (MISOC) optimization model is proposed for the CVTSP. The main algorithmic contribution of the authors is a Ranking Based Algorithm (RBA) that is based on enumeration of Hamiltonian paths starting from the origin, visiting the set of target

points, and ending at the destination. For a fixed Hamiltonian path, they compute a lower bound on the optimal value of the CVTSP. The paths are then ranked in ascending order of the lower bounds. For each path in that order, they solve the CVTSP with this fixed sequence to obtain an upper bound. The procedure terminates with an optimal solution if the upper bound matches the lower bound. The authors report optimal solutions on instances with up to 15 target points in less than an hour of CPU time.

A further extension of the CVTSP, referred to as the Generalized Carrier-Vehicle Traveling Salesman Problem (GCVTSP), is studied in Garone et al. (2014), where the Vehicle is allowed to visit multiple target points between a take-off and landing. A mixed integer nonlinear optimization model is presented and a three-stage heuristic method is proposed, which is composed of approximately solving the Euclidean TSP on the target points in order to fix the sequence in the first stage, determining the cluster of target points to be visited between each take-off and landing for the given sequence in the second stage, and computing the take-off and landing points for each cluster in the third stage. In Klauco et al. (2014), the authors formulate the GCVTSP with a fixed order of target points as a mixed integer second order conic optimization problem and report exact solutions on instances with $30 - 100$ target points in $10^3 - 10^5$ seconds.

After the initial version of this manuscript was submitted, we became aware of a related study on the Mothership and Drone Routing Problem introduced by Poikonen and Golden (2019), which is essentially the same problem as the CVTSP studied in this paper. The authors propose an exact branch-and-bound method, which is based on systematically enumerating all the visit sequences and solving a second-order cone programming problem to compute the take-off and landing points for each fixed sequence. They also propose several heuristic methods. Their exact method is able to solve randomly generated instances with up to 20 target points. They also consider the extension in which the Vehicle is allowed to visit multiple target points.

We note that the CVTSP is defined in the Euclidean plane, and there is a separate extensive literature on multi-vehicle systems defined on networks. One of the earliest such problems is the Flying Sidekick Traveling Salesman Problem introduced by Murray and Chu (2015), inspired by the drone delivery systems of logistics companies and e-tailers. Similar problems with slight variations have been studied (see, e.g., Agatz et al. (2018), Bouman et al. (2018), Ha et al. (2018), Saleu et al. (2018), Poikonen et al. (2019)). For other variants of the problem including multiple vehicles and multiple drones, we refer the reader to Murray and Raj (2020) and the references therein. In each of these variants of the problem, the landing and take-off points of the Vehicle are restricted to the set of the locations of target points or the depot (i.e., the origin in the context of the CVTSP). This assumption makes the problem amenable to a mixed integer linear programming formulation. In contrast, since the Vehicle is allowed to take-off and land at any point in the plane in the CVTSP, the synchronization of the Carrier and the Vehicle necessitates the use of nonlinear Euclidean distance constraints. As such, the CVTSP is a fundamentally different problem and its exact formulation requires a mixed integer nonlinear model (see

Section 3).

Another related problem is the Traveling Salesman Problem with Neighborhoods (TSPN), introduced in Arkin and Hassin (1994), which is concerned with finding the minimum-cost tour that visits a certain neighborhood of each target point. While certain special cases of the problem admit polynomial-time approximation schemes (see, e.g., Dumitrescu and Mitchell (2003), Mitchell (2007), Bodlaender et al. (2009), Chan and Jiang (2016)), the general version of the TSPN is NP-hard to approximate below a certain accuracy (see, e.g., de Berg et al. (2005), Safra and Schwartz (2006)). The reader is also referred to Gulczynski et al. (2006), Coutinho et al. (2016) for the related problem of close enough TSP.

Similar to Garone et al. (2010b, 2011) and Gambella et al. (2018), we focus on the variant of the CVTSP in which the order of target points is not a priori given and the Vehicle can visit one target point between each take-off and landing. The RBA of Gambella et al. (2018) can solve instances of the CVTSP with up to 15 target points to optimality, a relatively small number given the large scale of operations. For instance, following the earthquake and tsunami of 26 December 2004 in the Indian Ocean, 69 inhabited islands were impacted in the Maldives alone (see, e.g., UNEP (2005)). In this paper, we aim to develop exact and approximate solution approaches for larger instances of the CVTSP by identifying and exploiting specific structural properties of the problem. Our contributions are as follows. We establish several structural properties of the CVTSP. By exploiting these properties, we develop a new MISOC optimization model. In contrast with the optimization model of Gambella et al. (2018) that employs assignment variables for determining the visiting order of target points, our formulation relies on routing variables and subtour elimination constraints. We derive several optimality cuts arising from geometric observations. We also devise heuristic methods for computing near-optimal solutions for larger instances.

The rest of the paper is organized as follows. In Section 2, we give a formal definition of the CVTSP and identify several structural properties. We present improved and new formulations for the CVTSP in Section 3. Heuristic algorithms are presented in Section 4. Our computational experiments are reported in Section 5. Finally, we conclude the paper in Section 6.

2 Problem Definition and Structural Properties

In this section, we first give a formal definition of the CVTSP. Then, we present several structural properties that will later be utilized in our optimization formulation.

2.1 Problem Definition

As stated in the introduction, we consider a multi-vehicle system consisting of a slow Carrier with a long-range operational capability and a fast Vehicle with a limited range. The Carrier is able to transport, deploy, recover, and service the Vehicle. At the beginning

of the mission, the Vehicle is assumed to be based on the Carrier located at an origin with full operational capability. Both the Carrier and the Vehicle should return to a prespecified destination at the end of the mission. The destination may, in general, be different from the origin.

The locations are specified by the (x, y)-coordinates on the plane. The Carrier can follow any continuous trajectory without exceeding its maximum speed. The Vehicle is transported by the Carrier whenever it is not deployed. Once the Vehicle is deployed, it can follow any continuous trajectory without exceeding its maximum speed and its maximum operating time. The Vehicle should return to the Carrier before exhausting its operational capability, which we assume to be instantaneously restored as soon as the Vehicle lands on the Carrier. During the mission, the Vehicle should visit a set of target points whose locations are assumed to be known. For each target point, the Vehicle takes off from the Carrier, travels to the location of the target point, and returns to the Carrier. We assume that the service time of the Vehicle at a target point is negligible. The objective is to plan the routes of the Carrier and the Vehicle in such a way that the mission is completed as quickly as possible.

n	Number of target points
T	Set of target points ($T = \{1, \ldots, n\}$)
q_j	Coordinates of the target point j, $j = 1, \ldots, n$
V_v	Maximum speed of the Vehicle
V_c	Maximum speed of the Carrier ($V_c \leq V_v$)
a	Maximum operating time (autonomy) of the Vehicle
p_o	Coordinates of the origin
p_f	Coordinates of the destination

Table 1: Parameters of the CVTSP

The parameters of the problem are given in Table 1.

In Figure 1, we present an illustration of an optimal solution for an instance of the CVTSP with 10 target points, where take-off, target, and landing points are denoted by empty, lightly-shaded, and fully-shaded vertices, respectively. For this instance, the origin and the destination, denoted by o and f, are at the same coordinates. The parameters regarding the Carrier's speed, the Vehicle's speed, and the Vehicle's autonomy in this example are $V_c = 1$, $V_v = 5$, $a = 1$. We emphasize that the optimal solution tends to align Carrier routes between consecutive target points to line segments with similar slopes, as in the case of the part of the route visiting target points in the lower right part of Figure 1.

2.2 Structural Properties

In this section, we prove several structural properties of the CVTSP, which we will utilize to develop a stronger formulation.

Let us denote by $p_{to,k}$ and $p_{l,k}$ the coordinates of the take-off and landing points of the Vehicle before and after visiting the target point in the k^{th} order in an optimal solution

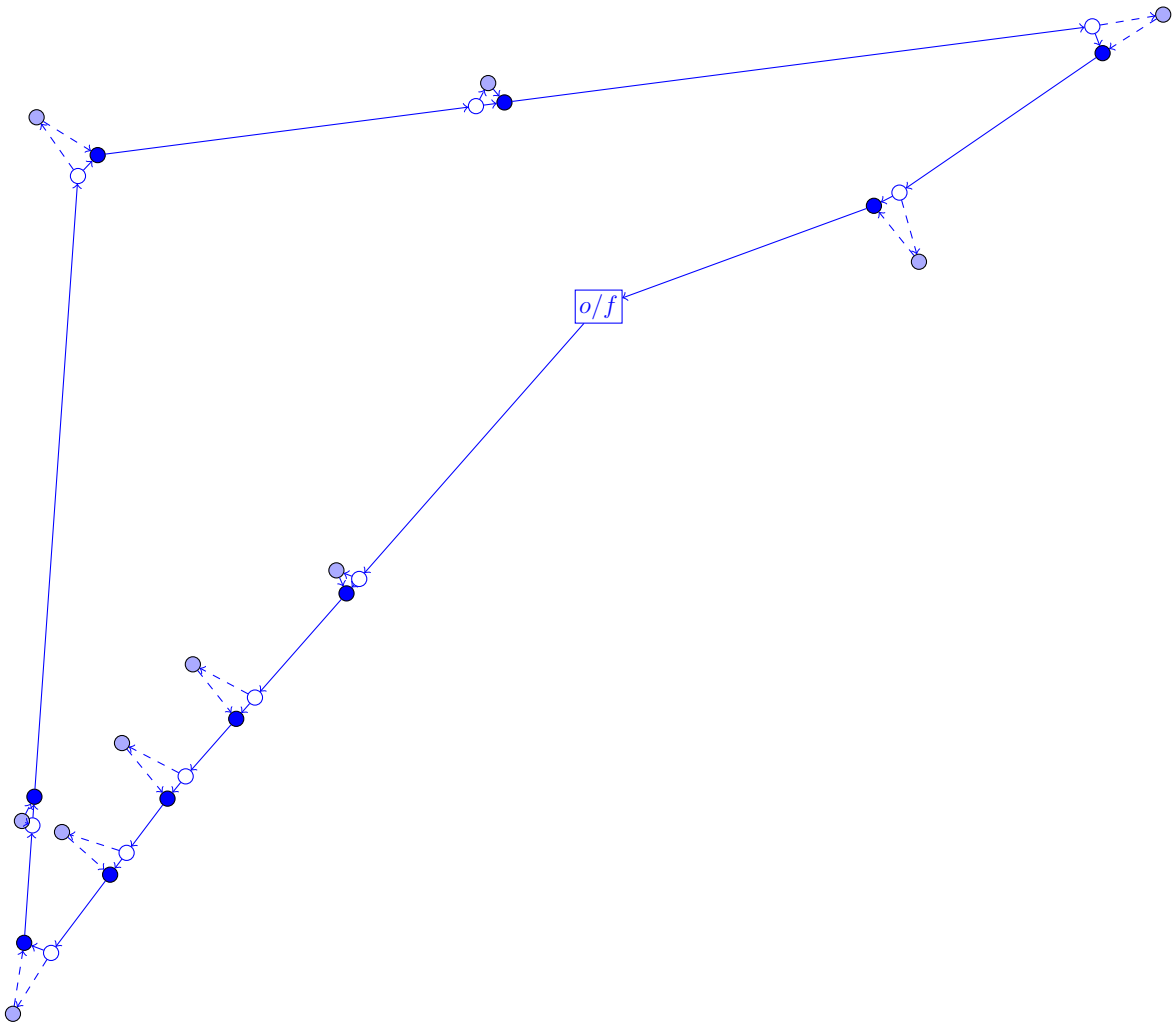

Figure 1: A sample optimal solution

of the CVTSP, respectively, where $k = 1, \ldots, n$. Therefore, the trajectory of the Carrier is given by

$$p_o, p_{to,1}, p_{l,1}, p_{to,2}, p_{l,2}, \ldots, p_{to,n}, p_{l,n}, p_f.$$

As already observed in Garone et al. (2014), during any period in which the Vehicle is transported by the Carrier, the Carrier should clearly travel along straight line segments at its maximum speed denoted by V_c. It follows that the total time during which the Carrier transports the Vehicle is given by

$$t^*_{CV} = \frac{1}{V_c}\left(\|p_o - p_{to,1}\| + \sum_{k=1}^{n-1} \|p_{l,k} - p_{to,k+1}\| + \|p_{l,n} - p_f\| \right), \tag{1}$$

where $\|\cdot\|$ denotes the Euclidean norm.
Let us now focus on the total time required to visit a target point $j \in T$. Let us define $s_{to,j}$ and $s_{l,j}$ as in Table 2, which will be used frequently in the remainder of this section.

$s_{to,j}$	Coordinates of the take-off point of the Vehicle right before visiting the target point j in an optimal solution of the CVTSP, $j \in T$
$s_{l,j}$	Coordinates of the landing point of the Vehicle right after visiting the target point j in an optimal solution of the CVTSP, $j \in T$

Table 2: Definitions of $s_{to,j}$ and $s_{l,j}$, $j \in T$

If the target point j is visited in the k^{th} order, then we have $s_{to,j} = p_{to,k}$ and $s_{l,j} = p_{l,k}$, where $j \in T$ and $k = 1, \ldots, n$. The time elapsed during the visit of the target point j is given by

$$t^*_j = \max\left\{t^c_j, t^v_j\right\}, \quad j \in T, \tag{2}$$

where t^c_j and t^v_j denote the minimum travel time of the Carrier and the Vehicle, respectively, and are given by

$$t^c_j = \frac{\|s_{to,j} - s_{l,j}\|}{V_c}, \quad t^v_j = \frac{\|s_{to,j} - q_j\| + \|q_j - s_{l,j}\|}{V_v}, \quad j \in T. \tag{3}$$

Due to the limited operational capability of the Vehicle, we have $t^*_j \leq a$, $j \in T$. We henceforth refer to t^*_j as the visit duration of target point j.
The optimal value of the CVTSP, which corresponds to the minimum completion time of the mission, is therefore given by

$$t^* = t^*_{CV} + \sum_{j \in T} t^*_j, \tag{4}$$

where t^*_{CV} and t^*_j are given by (1) and (2), respectively.
The following definition will be useful.

Definition 2.1. *Given an instance of the CVTSP, suppose that t_j^c and t_j^v are defined as in* (3)*, where $j \in T$. Then,*

(i) if $t_j^c < t_j^v$, then the Carrier is said to wait for the Vehicle at target point j,

(ii) if $t_j^v < t_j^c$, then the Vehicle is said to wait for the Carrier at target point j,

(iii) if $t_j^c = t_j^v$, then the Carrier and the Vehicle are said to be perfectly synchronized at target point j.

The next result illustrates a useful property of an optimal solution of the CVTSP.

Lemma 2.1. *In any optimal solution of the CVTSP, the Carrier never waits for the Vehicle at any target point, i.e., $t_j^v \leq t_j^c$ for each $j \in T$, where t_j^c and t_j^v are defined as in* (3)*.*

Proof. Suppose, for a contradiction, that there exists an optimal solution of the CVTSP such that the Carrier waits for the Vehicle at some target point $j \in T$, i.e., $t_j^c < t_j^v = t_j^*$ by (2). Let us denote the take-off point before visiting target point j by to and the landing point after visiting target point j by ℓ. Let us denote by i the previous point on the trajectory of the Carrier, which is either the landing point of the target point visited right before target point j or the origin. Figure 2 illustrates a schematic representation, where the solid lines represent the trajectory of the Carrier whereas the dashed lines correspond to that of the Vehicle. Note that both the take-off point and the landing point should be different from the target point j since we would otherwise have $t_j^v \leq t_j^c$ since $V_c \leq V_v$, contradicting our hypothesis.

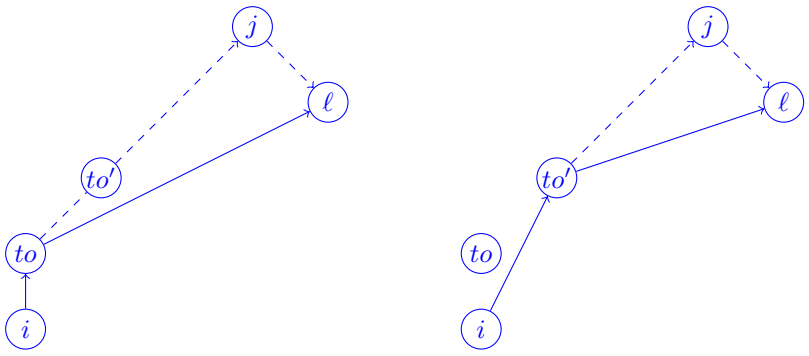

Figure 2: Illustration of the proof of Lemma 2.1

We make the following claim. There exists a point to' on the line segment between to and j such that deploying the Vehicle at to' as opposed to to strictly improves the objective function value. The coordinates of any point on the line segment between to and l, denoted by $to(\alpha)$, are given by $(1-\alpha)s_{to,j} + \alpha q_j = s_{to,j} + \alpha(q_j - s_{to,j})$ for some $\alpha \in [0,1]$, where q_j, $s_{to,j}$, and $s_{l,j}$ are defined as in Tables 1 and 2. Let us denote the travel time of the Carrier from to to $to(\alpha)$ and from $to(\alpha)$ to ℓ by $t_c(\alpha)$ and the travel time of the Vehicle

from $to(\alpha)$ to j and from j to ℓ by $t_v(\alpha)$, where $\alpha \in [0,1]$. Clearly, each of $t_c(\alpha)$ and $t_v(\alpha)$ is a continuous function of α and $t_v(\alpha)$ is a strictly decreasing function. Furthermore, $t_c(0) = t_j^c < t_v(0) = t_j^v$ and $t_c(1) \geq t_v(1)$ since $V_c \leq V_v$ and $s_{l,j} \neq q_j$. It follows that there exists $\alpha^* \in (0,1]$ such that $t_c(\alpha^*) = t_v(\alpha^*) < t_v(0) = t_j^v \leq a$ since $t_v(\alpha)$ is strictly decreasing. Denoting $to(\alpha^*)$ by to', if the Vehicle is deployed at to' instead of to, then the visit duration of target point j would be strictly smaller than t_v^j. Furthermore, note that using a shortcut from i to to' in the route of the Carrier will not increase the travel time of the Carrier by the triangle inequality. This contradicts the optimality of the original solution. □

Our next result establishes the existence of an optimal solution of the CVTSP such that the Carrier and the Vehicle are perfectly synchronized at each target point.

Proposition 2.1. *Given an instance of the CVTSP, there exists an optimal solution such that the Carrier and the Vehicle are perfectly synchronized at each target point, i.e., $t_j^v = t_j^c$ for each $j \in T$, where t_j^c and t_j^v are defined as in* (3).

Proof. Suppose that there exists an optimal solution such that the Carrier and the Vehicle are not perfectly synchronized at some target point $j \in T$. By Lemma 2.1, we have $t_j^v < t_j^c = t_j^*$, i.e., the Vehicle waits for the Carrier at target point j. Similar to the proof of Lemma 2.1, let us denote the take-off point before visiting target point j by to and the landing point after visiting target point j by ℓ. Let us denote by i the previous point on the trajectory of the Carrier, which is either the landing point of the target point visited right before target point j or the origin. In Figure 3, the solid lines represent the trajectory of the Carrier whereas the dashed lines correspond to that of the Vehicle. Note that at least one of the take-off and landing points should be different from the target point j since we would otherwise have $t_j^v = t_j^c = t_j^* = 0$, contradicting our hypothesis.

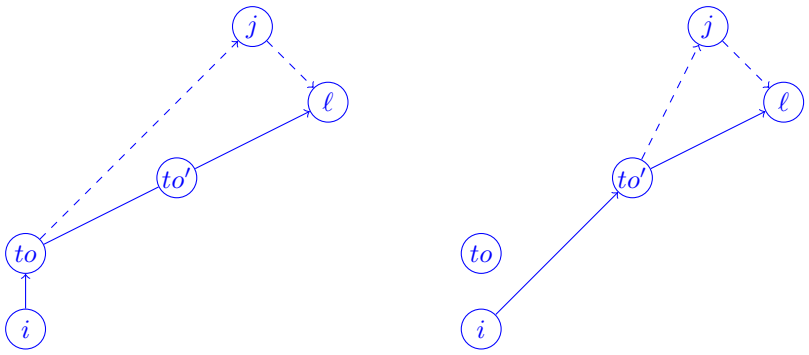

Figure 3: Illustration of the proof of Proposition 2.1

Following a similar argument as in the proof of Lemma 2.1, there exists a point to' on the line segment between to and ℓ such that the travel time of the Carrier and that of the Vehicle are identical and smaller than t_j^c if the Vehicle is deployed at to' instead of to. Finally, by the triangle inequality, we can use a short cut between i and to' in the

trajectory of the Carrier without increasing its total travel time from i to ℓ. It follows that any optimal solution in which the Carrier and the Vehicle are not perfectly synchronized at a target point can be modified without worsening the objective function value and ensuring perfect synchronization. □

Note that there may be an optimal solution of the CVTSP in which perfect synchronization may not necessarily be satisfied at a particular target point. Indeed, a close examination of the proof of Proposition 2.1 reveals that the objective function value remains the same if, for instance, the points i, to, and to' are collinear. Nevertheless, Proposition 2.1 ensures the existence of an optimal solution with perfect synchronization at *all* target points.
The next result is an immediate consequence of Proposition 2.1.

Corollary 2.1. *Given an instance of the CVTSP, there exists an optimal solution such that*

(i) the Carrier moves along straight line segments at its maximum speed V_c,

(ii) the Vehicle moves along straight line segments at its maximum speed V_v, and

(iii) the Carrier and the Vehicle are perfectly synchronized at each target point.

Next, we focus on the distance from the take-off or landing point to a target point. Denoting the visit duration of target point j by t_j^* in an optimal solution (see (2)), a naive upper bound on each of the two distances is given by $t_j^* V_v$. The next result provides a tighter upper bound.

Proposition 2.2. *Given an instance of the CVTSP, any optimal solution satisfies*

$$\max\left\{\left\|s_{to,j}-q_j\right\|,\left\|s_{l,j}-q_j\right\|\right\} \leq \frac{t_j^*\left(V_v+V_c\right)}{2} \leq \frac{a\left(V_v+V_c\right)}{2}, \quad j \in T, \tag{5}$$

where $s_{to,j}$ and $s_{l,j}$ are defined as in Table 2 and t_j^* is given by (2).

Proof. The second inequality in (5) directly follows from the inequality $t_j^* \leq a,\ j \in T$. For the first inequality, let us fix $j \in T$. By the triangle inequality,

$$\begin{aligned}
\left\|s_{to,j}-q_j\right\| &\leq \left\|s_{to,j}-s_{l,j}\right\|+\left\|s_{l,j}-q_j\right\| && (6)\\
&= t_j^c V_c + t_j^v V_v - \left\|s_{to,j}-q_j\right\| && (7)\\
&\leq t_j^* V_c + t_j^* V_v - \left\|s_{to,j}-q_j\right\|, && (8)
\end{aligned}$$

where we used (3) in the second line and (2) in the last line. It follows that $\left\|s_{to,j}-q_j\right\| \leq t_j^*\left(V_v+V_c\right)/2$. The upper bound on $\left\|s_{l,j}-q_j\right\|$ can be obtained in a similar manner. □

The next corollary establishes the existence of an optimal solution in which the distance from the take-off or landing point to each target point can be bounded below.

Corollary 2.2. *Given an instance of the CVTSP, there exists an optimal solution such that*

$$\min\left\{\left\|s_{to,j}-q_j\right\|,\left\|s_{l,j}-q_j\right\|\right\}\geq\frac{t_j^*\left(V_v-V_c\right)}{2},\quad j\in T, \tag{9}$$

where $s_{to,j}$ and $s_{l,j}$ are defined as in Table 2 and t_j^* is given by (2).

Proof. By Corollary 2.1, there exists an optimal solution of the CVTSP such that the Carrier and the Vehicle are perfectly synchronized at each target point. For such an optimal solution,

$$\left\|s_{to,j}-q_j\right\|+\left\|s_{l,j}-q_j\right\|=V_v\,t_j^*,\quad j\in T.$$

The relation (9) follows immediately from (5) and the equation above. □

We next introduce the following definition.

Definition 2.2. *Given an instance of the CVTSP, the Vehicle is said to use its full autonomy at target point j in an optimal solution if $t_j^v=a$, where $j\in T$, and t_j^v is given by (3).*

The next corollary follows from Proposition 2.2.

Corollary 2.3. *Given an instance of the CVTSP, suppose that $s_{to,j}$ and $s_{l,j}$ are defined as in Table 2, where $j\in T$. If*

$$\max\left\{\left\|s_{to,j}-q_j\right\|,\left\|s_{l,j}-q_j\right\|\right\}=\frac{a\left(V_v+V_c\right)}{2}, \tag{10}$$

then the Vehicle uses its full autonomy at target point j and the Carrier and the Vehicle are perfectly synchronized at target point j.

Proof. By Proposition 2.2, we obtain $t_j^*=a$. Since $t_j^*=\max\{t_j^v,t_j^c\}=a$ and $t_j^v\leq t_j^c$ by Lemma 2.1, we have $t_j^v\leq t_c^v=a$. If $t_j^v<t_j^c$, then, by (6)–(8), we would obtain

$$\left\|s_{to,j}-q_j\right\|<aV_c+aV_v-\left\|s_{to,j}-q_j\right\|,$$

which would imply that $\|s_{to,j}-q_j\|<a(V_v+V_c)/2$. A similar argument would yield $\|s_{l,j}-q_j\|<a(V_v+V_c)/2$, which would contradict (10). It follows that $t_j^v=t_j^c=a$, i.e., the Carrier and the Vehicle are perfectly synchronized at target point j. □

The next technical lemma establishes a gemoetric property that will be useful in the proof of the subsequent proposition.

Lemma 2.2. *Given an instance of the CVTSP, suppose that the Carrier and the Vehicle are perfectly synchronized at a target point $j\in T$. Consider the triangle with the base given by the line segment between $s_{to,j}$ and $s_{l,j}$ and the apex given by q_j, where $s_{to,j}$ and $s_{l,j}$ are defined as in Table 2 and q_j is defined as in Table 1. The height of this triangle, denoted by h_j, satisfies*

$$h_j\leq\frac{t_j^*}{2}\sqrt{V_v^2-V_c^2}, \tag{11}$$

where t_j^* is given by (2).

Proof. Suppose that the Carrier and the Vehicle are perfectly synchronized at a target point $j \in T$. If $t_j^* = 0$, then $s_{to,j} = s_{l,j} = q_j$, which implies that the triangle degenerates to a point and $h_j = 0$, clearly satisfying (11).
Suppose now that $t_j^* > 0$. Consider the triangle with with the base given by the line segment between $s_{to,j}$ and $s_{l,j}$ and the apex given by q_j. Let us denote the length of the base by α and the lengths of the other two sides by β and γ. By perfect synchronization,

$$\begin{aligned} \alpha &= V_c\, t_j^*, \\ \beta + \gamma &= V_v\, t_j^*. \end{aligned}$$

The area of this triangle is given by $(\alpha\, h_j)/2$. By Heron's formula,

$$\frac{\alpha\, h_j}{2} = \sqrt{s(s-\alpha)(s-\beta)(s-\gamma)},$$

where $s = (\alpha + \beta + \gamma)/2 = ((V_v + V_c)t_j^*)/2$. Therefore,

$$\begin{aligned} h_j^2 &= \frac{4s(s-\alpha)(s-\beta)(s-\gamma)}{\alpha^2} \\ &= \frac{4[((V_v + V_c)t_j^*)/2][((V_v - V_c)t_j^*)/2](s-\beta)(s-\gamma)}{V_c^2 (t_j^*)^2} \\ &= \frac{(V_v^2 - V_c^2)(s-\beta)(s-\gamma)}{V_c^2} \end{aligned}$$

By the triangle inequality, $s - \beta \geq 0$ and $s - \gamma \geq 0$. The last term is clearly maximized if $\beta = \gamma = (V_v\, t_j^*)/2$, which implies that

$$h_j^2 \leq \frac{(V_v^2 - V_c^2)[(V_c\, t_j^*)/2]^2}{V_c^2} = \frac{(V_v^2 - V_c^2)(t_j^*)^2}{4},$$

which establishes (11). □

The next result allows us to identify a subset of the target points at which the Vehicle uses its full autonomy in an optimal solution.

Proposition 2.3. *Given an instance of the CVTSP, if a target point j, where $j \in T$, satisfies all of the following conditions*

(i) $\|p_o - q_j\| \geq \frac{a(V_v + V_c)}{2}$,

(ii) $\|q_i - q_j\| \geq a\,(V_v + V_c)\,, \forall\ i \in T \setminus \{j\}$, *and*

(iii) $\|p_f - q_j\| \geq \frac{a(V_v + V_c)}{2}$,

then there exists an optimal solution such that the Vehicle uses its full autonomy at target point j and the Carrier and the Vehicle are perfectly synchronized at target point j.

For the sake of brevity, we provide the proof of Proposition 2.3 in the Appendix. The next result generalizes the sufficient conditions for full autonomy given in Proposition 2.3 by taking into account the visit sequence of the target points.

Proposition 2.4. *There exists an optimal solution of the CVTSP such that the Vehicle uses its full autonomy at target point j, where $j \in T$, if the Carrier travels*

(i) from the origin to target point j, and $\|p_o - q_j\| \geq \frac{a(V_v+V_c)}{2}$, or

(ii) from target point $i \in T \setminus \{j\}$ to target point j, and $\|q_i - q_j\| \geq a\,(V_v + V_c)$,

and

(iii) from target point j to target point $i \in T \setminus \{j\}$, and $\|q_i - q_j\| \geq a\,(V_v + V_c)$, or

(iv) from target point j to the destination, and $\|q_j - p_f\| \geq \frac{a(V_v+V_c)}{2}$.

Proof. We follow a similar argument as in the proof of Proposition 2.3. Under the assumption that either (i) or (ii) is satisfied and either (iii) and (iv) is satisfied, we can adjust the take-off point *to* and ℓ, if necessary (see Figures 4 and 5), so that the Vehicle uses its full autonomy without worsening the objective function value. □

3 Formulations

As stated in the introduction, the formulation of Gambella et al. (2018) employs assignment variables for sequencing target points. In this section, we present a new optimization model for the CVTSP that relies on routing variables. In addition, we utilize the structural properties presented in Section 2 to present several optimality cuts.

3.1 A Mixed Integer Nonlinear Programming Formulation

In this section, we present a new optimization model for the CVTSP. We use the same parameters presented in Table 1. Our decision variables are presented in Table 3.

x_{ij}	Binary variable which is equal to 1 if target point i is visited right before target point j and 0 otherwise, $i \in T;\ j \in T$
$x_{o,j}$	Binary variable which is equal to 1 if target point j is visited right after the origin (i.e., j is the first target point to be visited) and 0 otherwise, $j \in T$
$x_{j,f}$	Binary variable which is equal to 1 if target point j is visited right before the destination (i.e., j is the last target point to be visited) and 0 otherwise, $j \in T$
$s_{to,j}$	Coordinates of the take-off point of the Vehicle before visiting the target point j, $j \in T$
$s_{l,j}$	Coordinates of the landing point of the Vehicle after visiting the target point j, $j \in T$
ρ_{ij}	Distance traveled by the Carrier from target point i to target point j if target point j is visited right after target point i, $i \in T;\ j \in T$
$\rho_{o,j}$	Distance traveled by the Carrier from the origin to target point j if target point j is visited right after the origin, $j \in T$
$\rho_{j,f}$	Distance traveled by the Carrier from target point j to the destination if target point j is visited right before the destination, $j \in T$
t_j	Time between the take-off and landing of the Vehicle to visit target point j, $j \in T$

Table 3: Decision variables of CVTSP1

Next, we present our alternative formulation, denoted by CVTSP1:

$$\min \sum_{j\in T} t_j + \sum_{i\in T}\sum_{j\in T}(1/V_c)\rho_{ij} + \sum_{j\in T}(1/V_c)\left(\rho_{0,j}+\rho_{j,f}\right) \tag{12}$$

s.t.

$$\|s_{to,j} - s_{l,j}\| \le V_c\, t_j, \quad j \in T \tag{13}$$

$$\|s_{to,j} - q_j\| + \|s_{l,j} - q_j\| \le V_v\, t_j, \quad j \in T \tag{14}$$

$$\|s_{to,j} - q_j\| \le (1/2)\left(V_v + V_c\right)\, t_j, \quad j \in T \tag{15}$$

$$\|s_{l,j} - q_j\| \le (1/2)\left(V_v + V_c\right)\, t_j, \quad j \in T \tag{16}$$

$$\|p_o - s_{to,j}\| \le \rho_{0,j} + (1 - x_{0,j})\left(\|p_o - q_j\| + (1/2)\left(V_v + V_c\right) a\right), \quad j \in T \tag{17}$$

$$x_{o,j}\,\|p_o - q_j\| \le \rho_{0,j} + (1/2)\left(V_v + V_c\right) t_j, \quad j \in T \tag{18}$$

$$\|s_{l,j} - p_f\| \le \rho_{j,f} + (1 - x_{j,f})\left(\|q_j - p_f\| + (1/2)\left(V_v + V_c\right) a\right), \quad j \in T \tag{19}$$

$$x_{j,f}\,\|q_j - p_f\| \le \rho_{j,f} + (1/2)\left(V_v + V_c\right) t_j, \quad j \in T \tag{20}$$

$$\|s_{to,j} - s_{l,i}\| \le \rho_{ij} + (1 - x_{ij})\left(\|q_j - q_i\| + \left(V_v + V_c\right) a\right), \quad i \in T,\ j \in T \tag{21}$$

$$x_{ij}\,\|q_j - q_i\| \le \rho_{ij} + (1/2)\left(V_v + V_c\right)\left(t_i + t_j\right), \quad i \in T,\ j \in T \tag{22}$$

$$x_{jj} = 0, \quad j \in T \tag{23}$$

$$x_{o,j} + \sum_{i\in T} x_{ij} = 1, \quad j \in T \tag{24}$$

$$x_{j,f} + \sum_{k\in T} x_{jk} = 1, \quad i \in T \tag{25}$$

$$\sum_{i\in S}\sum_{j\in S} x_{ij} \le |S| - 1, \quad S \subseteq T;\ 2 \le |S| \le n \tag{26}$$

$$t_j \le a, \quad j \in T \tag{27}$$

$$\rho_{ij} \ge 0, \quad i \in T,\ j \in T \tag{28}$$

$$\rho_{o,j} \ge 0, \quad j \in T \tag{29}$$

$$\rho_{j,f} \ge 0, \quad j \in T \tag{30}$$

$$t_j \ge 0, \quad j \in T \tag{31}$$

$$x_{ij} \in \{0,1\}, \quad i \in T,\ j \in T \tag{32}$$

$$x_{o,j} \in \{0,1\}, \quad j \in T \tag{33}$$

$$x_{j,f} \in \{0,1\}, \quad j \in T. \tag{34}$$

The objective function (12) is composed of the sum of the total travel time of the Carrier along its trajectory while transporting the Vehicle and the total visit duration of the target points. The constraints (13) and (14) correspond to the visit duration of the target points. The constraints (15) and (16) directly follow from Proposition 2.2. If the target point j is the first target point to be visited (i.e., if $x_{0,j} = 1$), then $\rho_{0,j}$ corresponds to the distance traveled by the Carrier from the origin to the target point j by (17). Otherwise, $\rho_{0,j}$ can be set to zero by the triangle inequality and Proposition 2.2. The constraint (18) is

redundant if $x_{0,j} = 1$. Otherwise, it reduces to the triangle inequality between the origin, take-off point before visiting target point j, and target point j by Proposition 2.2, thereby providing a lower bound on $\rho_{0,j}$. The constraints (19) and (20) account for the last target point to be visited and are structured similarly to (17) and (18), respectively. The distance traveled by the Carrier between the landing point of target point i and the take-off point of target point j is reflected in the constraints (21) and (22) in a similar manner. The constraints (23)–(26) ensure that all target points are visited in a proper order starting from the origin and ending at the destination. Note that we employ subtour elimination constraints (26) since our model relies on routing variables x_{ij}. The maximum operating time of the Vehicle is reflected in (27). Finally, the constraints (28)–(34) specify the ranges of the decision variables.

CVTSP1 is a mixed integer second order conic optimization model consisting of $n^2 + 5n$ linear constraints, $2^n - n - 1$ subtour elimination constraints, $n^2 + 7n$ three-dimensional second order conic constraints, $n^2 + 3n$ nonnegative continuous variables, $4n$ general continuous variables, and $n^2 + 2n$ binary variables.

3.2 Optimality Cuts

In this section, we present a set of optimality cuts for CVTSP1. Let us define the following index sets:

$$\mathcal{I}_j = \{i \in \{1,\dots,n\} : \|q_i - q_j\| \geq a\,(V_v + V_c)\}, \quad j \in T, \tag{35}$$

$$\mathcal{I}_o = \{i \in \{1,\dots,n\} : \|p_o - q_i\| \geq a\,(V_v + V_c)/2\}, \tag{36}$$

$$\mathcal{I}_f = \{i \in \{1,\dots,n\} : \|p_f - q_i\| \geq a\,(V_v + V_c)/2\}. \tag{37}$$

Proposition 3.1. *Given an instance of the CVTSP, there exists an optimal solution of CVTSP1 that satisfies each of the following inequalities:*

$$t_j \geq a\left(x_{o,j} + \sum_{i\in\mathcal{I}_j}(x_{ij} + x_{ji}) + x_{j,f} - 1\right), \quad \forall j \in \mathcal{I}_o \cap \mathcal{I}_f, \tag{38}$$

$$t_j \geq a\left(\sum_{i\in\mathcal{I}_j}(x_{ij} + x_{ji}) + x_{j,f} - 1\right), \quad \forall j \in (T \setminus \mathcal{I}_o) \cap \mathcal{I}_f, \tag{39}$$

$$t_j \geq a\left(x_{o,j} + \sum_{i\in\mathcal{I}_j}(x_{ij} + x_{ji}) - 1\right), \quad \forall j \in \mathcal{I}_o \cap (T \setminus \mathcal{I}_f), \tag{40}$$

$$t_j \geq a\left(\sum_{i\in I_j}(x_{ij} + x_{ji}) - 1\right), \quad \forall j \in (T \setminus \mathcal{I}_o) \cap (T \setminus \mathcal{I}_f). \tag{41}$$

Proof. The proof directly follows from Proposition 2.4 and the constraints (24) and (25). □

For the next proposition, we introduce the auxiliary variables presented in Table 4.

$\delta_{to,j}$	Distance traveled by the Vehicle from the take-off point $s_{to,j}$ to target point j, $j \in T$
$\delta_{l,j}$	Distance traveled by the Vehicle from target point j to the landing point $s_{l,j}$ to , $j \in T$

Table 4: Auxiliary decision variables of CVTSP1

Proposition 3.2. *Given an instance of the CVTSP, there exists an optimal solution of CVTSP1 that satisfies each of the following relations:*

$$\|s_{to,j} - q_j\| \leq \delta_{to,j}, \quad j \in T, \tag{42}$$

$$\|s_{l,j} - q_j\| \leq \delta_{l,j}, \quad j \in T, \tag{43}$$

$$\delta_{to,j} + \delta_{l,j} = V_v\, t_j, \quad j \in T, \tag{44}$$

$$\delta_{to,j} \geq (1/2)(V_v - V_c)t_j, \quad j \in T, \tag{45}$$

$$\delta_{l,j} \geq (1/2)(V_v - V_c)t_j, \quad j \in T. \tag{46}$$

Proof. The inequalities (42) and (43) follow from the definitions of $\delta_{to,j}$ and $\delta_{l,j}$ in Table 4. The equality (44) follows from Corollary 2.1 and the inequalities (45) and (46) are implied by Corollary 2.2. □

Note that we refer to the relations in Propositions 3.1 and 3.2 as optimality cuts since there may exist feasible (even optimal) solutions of CVTSP1 that violate these relations. For instance, consider a CVTSP instance with one target point, where $p_o = (0,0), p_f = (4,0), q_1 = (2,0), V_c = 1, V_v = 2$, and $a = 1$. Clearly, in any optimal solution of the CVTSP, the Carrier moves along a straight line segment from the origin to the destination through the target point. As such, the Vehicle does not need to leave the Carrier. However, by Propositions 3.1 and 3.2, we can impose $t_1 = a = 1$, which implies that there are only two optimal solutions that satisfy the given optimality cuts. Either the Vehicle takes off at $(0.5, 0)$, visits the target point, and lands on the Carrier at $(1.5, 0)$, or the Vehicle takes off at $(0, 2.5)$, visits the target point, and lands on the Carrier at $(3.5, 0)$. Note that, in both optimal solutions, the Vehicle uses full autonomy, moves along straight line segments, and is perfectly synchronized with the Carrier. As illustrated by this simple example, these relations may reduce the feasible region and our results in Section 2 imply the existence of an optimal solution that satisfies each of these relations.

We have also attempted to utilize the lower bound presented by Garone et al. (2010a) and used by Gambella et al. (2018) for the case when the order of the target points is fixed. Denoting the order *ord*, and the total length of the TSP tour corresponding to *ord* as $TSP(ord)$, the lower bound is:

$$LB(ord) = \frac{TSP(ord)}{V_c} - n\frac{V_v a}{V_c} + na. \tag{47}$$

In order to generalize this lower bound, we have replaced $TSP(ord)$ with the TSP distance

of the target points based on the routing variables. The resulting optimality cut is

$$\sum_{j\in T} t_j + \sum_{i\in T}\sum_{j\in T}(1/V_c)\rho_{ij} + \sum_{j\in T}(1/V_c)\left(\rho_{0,j}+\rho_{j,f}\right) \geq \frac{\sum\limits_{i\in T}\|p_o - q_i\|\, x_{o,i} + \sum\limits_{i\in T}\sum\limits_{j\in T}\|q_i - q_j\|\, x_{i,j} + \sum_{j\in T}\|q_j - p_f\|\, x_{j,f}}{V_c} - n\frac{V_v a}{V_c} + na. \quad (48)$$

However, this optimality cut was not observed to improve the lower bound or the solution time of our models except for a subset of instances.

4 Heuristic Algorithms

In this section, we provide a greedy constructive algorithm to generate initial solutions, and an Iterated Local Search algorithm to find high quality solutions for larger instances of the CVTSP.

4.1 Constructive Algorithm

Let us start by defining a basic variant of CVTSP consisting of a single target point that we refer to as CVTSP-B. It is worth mentioning that CVTSP-B is also a special case of the CVP with only one target point. Devoid of the need to sequence the target points, CVTSP-B can be formulated as a second order conic optimization model with four variables that correspond to the coordinates of the take-off and the landing points of the Vehicle, and four constraints given by (13), (14), (15), (16). We will refer to an instance of CVTSP-B starting from q_i, visiting q_j, and ending at q_k as CVTSP-B(q_i, q_j, q_k). We also define σ as the sequence of target points to be visited in a solution, where $\sigma(i)$ corresponds to the index of the target point to be visited in the i^{th} order. Finally, for the sake of brevity, we will refer to the vector of all landing and take-off coordinates as $\mathbf{s}$.

The constructive algorithm works in a greedy fashion with one step look-ahead. It starts by selecting the first two target points to be visited, and solves a CVTSP-B to determine the take-off and landing points for the first one, where the second target point is considered to be the destination. It proceeds in this fashion by solving a series of CVTSP-B instances, until the last target point, when the final destination point is the actual destination point located at p_f. The details of the algorithm are provided as Algorithm 1.

4.2 Iterated Local Search algorithm

Iterated Local Search (ILS) is a metaheuristic algorithm that has been successfully used for solving many variants of the Traveling Salesman Problem (TSP) and the Vehicle Routing Problem (VRP). It is based on the idea of perturbing a given solution and re-optimizing through local search operators. We refer the interested reader to the comprehensive paper

Algorithm 1 Constructive heuristic

$L = \{1, ..., n\}$
$\sigma(1) = \arg\min_{l \in L} \|q_l - p_o\|$
$L = L \setminus \{\sigma(1)\}$
$\sigma(2) = \arg\min_{l \in L} \|q_l - q_{\sigma(1)}\|$
$L = L \setminus \{\sigma(2)\}$
Solve CVTSP-B$(p_o, q_{\sigma(1)}, q_{\sigma(2)})$ to determine $s_{to,\sigma(1)}$ and $s_{l,\sigma(1)}$
For $ord = 2$ **to** $|T| - 1$ **do**
    $\sigma(ord + 1) = \arg\min_{l \in L} \|q_l - q_{\sigma(ord)}\|$
    $L = L \setminus \{\sigma(ord + 1)\}$
    Solve CVTSP-B$(s_{l,\sigma(ord-1)}, q_{\sigma(ord)}, q_{\sigma(ord+1)})$ to determine $s_{to,\sigma(ord)}$ and $s_{l,\sigma(ord)}$
End For
Solve CVTSP-B$(s_{l,\sigma(n-1)}, q_{\sigma(n)}, p_f)$ to determine $s_{to,\sigma(n)}$ and $s_{l,\sigma(n)}$
Return σ and $\mathbf{s}$

by Subramanian et al. (2013). Most of the ILS implementations for the VRP use the sequence of the customers to represent solutions. We will write σ^* to denote the best known sequence, and $\mathbf{s}^*$ to denote the associated take-off and landing coordinates. Finally, we will denote the objective function value corresponding to a sequence σ and the associated take-off and landing coordinates $\mathbf{s}$ as $z(\sigma, \mathbf{s})$.

In our algorithmic design, we have opted to implement three well-known local search operators to improve the sequence, namely, 1-OPT (removing a target point in the sequence and reinserting it in a different position), 2-OPT (exchanging two carrier arcs in the route, which corresponds to reversing a section of the sequence), and 2-EXCHANGE (swapping two target points in the sequence) (Groër et al. 2010), and choose the best sequence found among them in each iteration. For 1-OPT and 2-EXCHANGE, we have implemented the operators to retain the take-off and landing coordinates of each target point, effectively preserving feasibility. In our implementation of 2-OPT, we have evaluated the resulting solutions by swapping the take-off and landing points of each target point within the reversed section. This approach has allowed us to attain better solutions while still preserving feasibility.

For any given sequence σ, it is possible to determine $\mathbf{s}$ by iteratively solving instances of CVTSP-B, similar to Algorithm 1, but without the need to determine the next target point. However, repeatedly recomputing take-off and landing points results in computational overhead, especially when implemented within a local search operator. To balance the trade-off between the computational effort and the search for better solutions, we have determined the take-off and landing points at the beginning of each local search iteration.

A crucial component of ILS is the perturbation step, where we have used the same operators to move to random solutions, and chose 1 to 3 such moves for each operator. We have set the number of iterations $k_{max} = 2|T|$, which we have observed to provide high quality solutions without requiring extensive CPU time. The details of the algorithm are presented as Algorithm 2.

Algorithm 2 Iterated Local Search

Invoke Algorithm 1 to determine σ^* and $\mathbf{s}^*$
For $k = 1$ **to** k_{max} **do**
  $\sigma = \sigma^*$; $\mathbf{s} = \mathbf{s}^*$
  If $k > 1$ **Then** perturb σ by applying random 1-OPT, 2-OPT, and 2-EXCHANGE moves.
  Do
    Solve CVTSP-B$(p_o, q_{\sigma(1)}, q_{\sigma(2)})$ to determine $s_{to,\sigma(1)}$ and $s_{l,\sigma(1)}$
    For $ord = 2$ **to** $|T| - 1$ **do**
      Solve CVTSP-B$(s_{l,\sigma(ord-1)}, q_{\sigma(ord)}, q_{\sigma(ord+1)})$ to determine $s_{to,\sigma(ord)}$ and $s_{l,\sigma(ord)}$
    End For
    Solve CVTSP-B$(s_{l,\sigma(n-1)}, q_{\sigma(n)}, p_f)$ to determine $s_{to,\sigma(n)}$ and $s_{l,\sigma(n)}$
    $\sigma' = \sigma$
    Apply local search to σ using 1-OPT, 2-OPT, and 2-EXCHANGE operators.
  While $\sigma \neq \sigma'$
  If $z(\sigma, \mathbf{s}) < z(\sigma^*, \mathbf{s}^*)$ **Then**
    $\sigma^* = \sigma$; $\mathbf{s}^* = \mathbf{s}$
  End If
End For

5 Computational Results

In our computational experiments, we used CPLEX 12.8.0 as a mixed integer second order conic optimization solver. We have conducted experiments on the nodes of the computing cluster *Balena* hosted at the University of Bath, with Intel E5-2650 v2 CPUs at a speed of 2.60 GHz. We have implemented the model of Gambella et al. (2018) on the same platform, and limited both models to use a single thread, in order to ensure a fair comparison. In all tables below, the acronyms LB and UB correspond to lower bound and upper bound, respectively.

5.1 Models

We have started by testing the model of Gambella et al. (2018), CVTSP1, as well as CVTSP1′ that includes the optimality cuts (38) - (46). All models were tested on the instance sets kindly provided by the authors of Gambella et al. (2018), and we imposed a CPU time limit of 3600 seconds.

The subtour elimination constraints (26) were implemented as *connectivity constraints* from the origin to each target point, and from each target point to the destination. Violated connectivity constraints were separated by solving a max-flow problem on the residual graph generated by the fractional x_{ij} values at each node of the branch-and-cut tree, similar to the implementation in Battarra et al. (2010). The relations (38) - (46) were directly added to the model for the target points that are *distant*, i.e. $j \in T : j \in \mathcal{I}_o, j \in \mathcal{I}_f, \mathcal{I}_j = T \setminus \{j\}$, and passed on to CPLEX as user cuts for the rest of the target points.

The instances from Gambella et al. (2018) are comprised of four families, denoted by SD, MD, LD, and VLD. In each family, instances are randomly generated in the following fashion. In SD and MD, target point coordinates are uniformly generated in $[-25, 25]$ and $[-25, 25]$, respectively. On the other hand, in LD and VLD, larger intervals of $[-50, 50]$ and $[-50, 50]$ are used to generate the target point coordinates. In the families MD and VLD, target points are chosen to satisfy $\|q_i - q_j\| \geq V_v a$ for each $i \in T$ and $j \in T$, where $i \neq j$, whereas no such restriction is imposed in the families SD and LD. Each instance is denoted by FAM_N_NUM, where FAM $\in$ {SD, MD, LD, VLD}, $N = n + 1$ with $N \in \{11, 12, \ldots, 16\}$, and NUM $\in \{1, 2, 3\}$. Therefore, a total of 72 instances were tested. In each instance, the parameters were given by $V_c = 1, V_v = 5, a = 1, p_o = [0, 0]^T$, and $p_f = [0, 0]^T$.

Detailed results are presented in Tables 5, 6, 7, and 8, where the best CPU time (in seconds) for each instance is indicated in boldface. Notably, we did not improve upon any of the upper bounds found by Gambella et al. (2018), hence we denote the upper bounds under the heading "Gambella RBA UB". For instance sets LD and VLD, CVTSP1 improves the average CPU time requirement of Gambella et al. (2018) by factors of 8.75 and 9.30, respectively. In addition, CVTSP1 successfully solves a previously unsolved instance with $|T| = 15$. Furthermore, CVTSP1$'$ augments these results to improvement factors of 17.58 and 12.78, respectively. Notably, the model of Gambella et al. (2018) is consistently faster for smaller instances, and the factors of improvement are due to instances with $|T| \in \{14, 15\}$. More modest improvements are observed for the instance sets MD and SD, with similar performances in terms of instance size. In these instance sets, CVTSP1$'$ can successfully solve 2 previously unsolved instances. We conclude that CVTSP1$'$ clearly outperforms both the MISOC model of Gambella et al. (2018) and CVTSP1 for these instances.

Table 5: Comparison of the models for SD instances

Instance	Gambella RBA UB	Gambella MISOC CPU	CVTSP1 CPU	CVTSP1$'$ CPU
SD11_1	108.76	**1.96**	41.64	17.06
SD11_2	113.92	**8.62**	91.36	68.36
SD11_3	125.19	**1.93**	28.45	15.16
SD12_1	107.60	**2.28**	20.41	12.47
SD12_2	153.94	**1.96**	58.67	23.67
SD12_3	118.61	**8.76**	32.33	22.56
SD13_1	116.12	**4.12**	37.91	22.84
SD13_2	136.86	**137.52**	689.08	919.78
SD13_3	121.47	**13.11**	84.08	36.53
SD14_1	128.14	201.05	471.73	**128.36**
SD14_2	124.66	152.53	357.42	**134.48**
SD14_3	138.07	103.62	82.94	**69.44**
SD15_1	123.67	403.20	359.25	**240.83**
SD15_2	136.10	643.02	592.08	**436.47**
SD15_3	132.72	3600.00	3600.00	3600.03
SD16_1	145.10	3600.00	379.55	**346.86**
SD16_2	155.36	3600.00	3600.00	**2228.28**
SD16_3	128.38	3600.00	3600.00	**2060.22**
Average		893.54	784.83	576.86

Table 6: Comparison of the models for MD instances

Instance	Gambella RBA UB	Gambella MISOC CPU	CVTSP1 CPU	CVTSP1′ CPU
MD11_1	146.85	**0.19**	9.42	5.28
MD11_2	132.12	**0.60**	31.58	9.53
MD11_3	133.42	**0.76**	29.16	10.86
MD12_1	157.74	**2.15**	21.27	15.86
MD12_2	165.63	**1.98**	24.02	23.69
MD12_3	121.24	**1.58**	34.77	19.03
MD13_1	150.89	**15.25**	77.94	63.75
MD13_2	130.99	**16.92**	74.94	42.75
MD13_3	150.37	**12.72**	46.08	29.38
MD14_1	146.95	**24.23**	169.53	65.61
MD14_2	163.59	**54.06**	146.58	83.95
MD14_3	153.01	**38.34**	89.48	44.95
MD15_1	168.24	1476.65	2538.06	**789.94**
MD15_2	136.94	1277.98	493.59	**234.95**
MD15_3	157.86	**624.48**	1118.97	656.53
MD16_1	166.21	3600.00	3600.00	**1716.09**
MD16_2	177.23	3232.09	786.81	**420.25**
MD16_3	64.37	3600.00	475.14	**198.94**
Average		776.67	542.63	246.19

Table 7: Comparison of the models for LD instances

Instance	Gambella RBA UB	Gambella MISOC CPU	CVTSP1 CPU	CVTSP1′ CPU
LD11_1	311.55	**0.06**	4.24	2.45
LD11_2	345.19	**0.23**	5.09	3.84
LD11_3	299.53	**0.21**	2.41	4.52
LD12_1	296.07	**0.29**	6.92	4.77
LD12_2	308.26	**0.69**	6.77	6.53
LD12_3	270.31	**0.16**	3.27	3.67
LD13_1	261.64	**1.43**	13.28	10.98
LD13_2	294.85	**2.16**	8.30	8.69
LD13_3	307.34	**2.39**	10.83	7.64
LD14_1	319.80	**5.84**	11.75	7.94
LD14_2	282.91	**23.94**	47.77	28.98
LD14_3	301.60	**5.35**	11.39	8.42
LD15_1	299.04	45.95	41.66	**28.67**
LD15_2	314.01	160.90	25.94	**25.47**
LD15_3	324.79	440.25	158.58	**74.16**
LD16_1	322.05	350.77	25.06	**13.53**
LD16_2	338.70	2726.22	267.66	**75.34**
LD16_3	353.87	2403.94	53.75	**35.42**
Average		342.82	39.15	19.50

Table 8: Comparison of the models for VLD instances

Instance	Gambella RBA UB	Gambella MISOC CPU	CVTSP1 CPU	CVTSP1′ CPU
VLD11_1	257.24	**0.31**	6.45	3.67
VLD11_2	324.75	**0.08**	4.31	4.36
VLD11_3	226.13	**0.14**	6.36	4.34
VLD12_1	326.04	**0.14**	3.52	4.69
VLD12_2	274.19	**0.21**	3.64	4.44
VLD12_3	281.94	**0.50**	11.17	4.81
VLD13_1	316.71	**4.26**	28.38	16.13
VLD13_2	239.26	**1.56**	8.56	8.08
VLD13_3	281.33	**0.98**	6.88	7.61
VLD14_1	319.63	**11.69**	20.64	26.55
VLD14_2	300.17	**2.38**	13.63	10.14
VLD14_3	280.37	**3.79**	20.56	10.5
VLD15_1	295.33	**2.18**	20.81	15.61
VLD15_2	314.70	50.91	21.13	**14.89**
VLD15_3	264.95	**5.16**	15.59	13.63
VLD16_1	379.91	624.76	46.44	**35.06**
VLD16_2	355.42	861.46	88.13	**62.16**
VLD16_3	305.99	3600.00	229.26	**157.83**
Average		287.25	30.86	22.47

We are not able to provide a direct comparison of our results with those of the RBA of Gambella et al. (2018), due to the difference of the computing platforms used. However, the CPU speed of the computer (3.10 GHz) they have used is faster than our CPU by a factor of 1.2. For the sake of brevity, we will forgo an instance-by-instance comparison and will compare only average results, as presented in Table 9. CVTSP1′ is observed to provide significant improvement factors for all instance sets, for LD and VLD in particular, despite the difference of the CPU speed. CVTSP1′ can also solve 2 instances that could not be solved by RBA.

Table 9: Comparison of the average results of the RBA of Gambella et al. (2018) and CVTSP1′

	Gambella RBA		CVTSP1′	
Instance set	CPU	Optimal	CPU	Optimal
SD	660.77	16/18	576.86	17/18
MD	551.42	17/18	246.19	18/18
LD	78.13	18/18	19.50	18/18
VLD	52.66	18/18	22.47	18/18

To explore the computational reach of CVTSP1′, we have generated larger instances using the same generation scheme, for $|T| \in \{16, 17, 18, 19, 20, 25, 30, 35, 40, 45, 50\}$. These instances will be made available by the first author upon request. Table 10 presents the results for $|T| \in \{16, 17, 18, 19, 20\}$, where a CPU time limit of 7200 was used. CVTSP1 and CVTSP1′ have successfully solved 59 out of 60 instances, extending the computational reach of exact algorithms from $|T| = 15$ to $|T| = 20$. It can be observed from Table 10 that the lower bounds provided by CVTSP1′ are significantly stronger than those of the MISOC model of Gambella et al. (2018). CVTSP1′ outperforms CVTSP1 for 49 out of

60 instances in terms of CPU time. Unfortunately, we cannot provide a comparison with RBA due to the lack of the source code of the authors' implementation.

Table 10: Comparison of the models for the new instances with $|T| \in \{16, 17, 18, 19, 20\}$

	Gambella MISOC				CVTSP1				CVTSP1′			
Instance	UB	LB	Gap (%)	CPU	UB	LB	Gap (%)	CPU	UB	LB	Gap (%)	CPU
SD17_1	295.33	234.76	25.80	7200.00	285.58	285.58	0.00	44.07	285.58	285.58	0.00	**36.98**
SD17_2	357.74	204.54	74.90	7200.00	341.49	341.49	0.00	215.62	341.49	341.49	0.00	**190.38**
SD17_3	286.28	155.62	83.96	7200.00	270.31	270.31	0.00	123.38	270.31	270.31	0.00	**107.42**
SD18_1	351.79	222.06	58.42	7200.00	329.47	329.47	0.00	**95.45**	329.47	329.47	0.00	104.69
SD18_2	338.97	206.07	64.49	7200.00	314.49	314.49	0.00	72.70	314.49	314.49	0.00	**58.15**
SD18_3	381.83	179.94	112.20	7200.00	341.80	341.80	0.00	397.79	341.80	341.80	0.00	**262.13**
SD19_1	390.30	191.75	103.55	7200.00	359.70	359.70	0.00	1567.84	359.70	359.70	0.00	**1327.38**
SD19_2	372.80	179.37	107.84	7200.00	312.15	312.15	0.00	64.26	312.15	312.15	0.00	**51.94**
SD19_3	388.31	212.70	82.57	7200.00	379.31	379.31	0.00	830.60	379.31	379.31	0.00	**429.47**
SD20_1	396.96	166.23	138.80	7200.00	365.93	365.93	0.00	2845.95	365.93	365.93	0.00	**2317.80**
SD20_2	366.12	223.80	63.59	7200.00	355.71	355.71	0.00	**1029.80**	355.71	355.71	0.00	2857.17
SD20_3	431.74	144.93	197.90	7200.00	341.76	341.76	0.00	694.78	341.76	341.76	0.00	**403.03**
SD21_1	434.67	209.44	107.54	7200.00	371.82	371.82	0.00	157.74	371.82	371.82	0.00	**78.72**
SD21_2	419.55	133.77	213.65	7200.00	373.80	373.80	0.00	4217.53	373.80	373.80	0.00	**2723.54**
SD21_3	350.28	102.39	242.11	7200.00	350.28	350.28	0.00	**3471.80**	350.28	350.28	0.00	3850.27
MD17_1	340.54	243.18	40.04	7200.00	322.21	322.21	0.00	**32.13**	322.21	322.21	0.00	44.80
MD17_2	375.15	272.96	37.44	7200.00	350.08	350.08	0.00	189.15	350.08	350.08	0.00	**91.71**
MD17_3	405.54	200.93	101.83	7200.00	356.27	356.27	0.00	39.80	356.27	356.27	0.00	**33.61**
MD18_1	379.83	211.10	79.93	7200.00	346.43	346.43	0.00	176.01	346.43	346.43	0.00	**122.22**
MD18_2	341.80	217.00	57.52	7200.00	335.27	335.27	0.00	63.81	335.27	335.27	0.00	**60.85**
MD18_3	439.54	213.39	105.98	7200.00	376.27	376.27	0.00	117.60	376.27	376.27	0.00	**107.19**
MD19_1	426.68	146.58	191.09	7200.00	364.45	364.45	0.00	329.67	364.45	364.45	0.00	**157.89**
MD19_2	424.95	197.06	115.64	7200.00	370.01	370.01	0.00	216.08	370.01	370.01	0.00	**119.73**
MD19_3	424.84	158.78	167.56	7200.00	345.01	345.01	0.00	114.86	345.01	345.01	0.00	**82.63**
MD20_1	421.84	174.40	141.89	7200.00	341.91	341.91	0.00	498.29	341.91	341.91	0.00	**249.25**
MD20_2	359.41	153.24	134.54	7200.00	332.47	332.47	0.00	**633.07**	332.47	332.47	0.00	676.15
MD20_3	399.58	176.75	126.07	7200.00	342.91	342.91	0.00	191.52	342.91	342.91	0.00	**128.94**
MD21_1	451.89	109.98	310.88	7200.00	363.66	363.66	0.00	1075.81	363.66	363.66	0.00	**792.71**
MD21_2	418.01	114.14	266.23	7200.00	344.70	344.70	0.00	1632.04	344.70	344.70	0.00	**463.34**
MD21_3	455.66	86.31	427.95	7200.00	363.71	363.71	0.00	466.48	363.71	363.71	0.00	**442.70**
LD17_1	391.63	217.67	79.92	7200.00	380.11	380.11	0.00	178.32	380.11	380.11	0.00	**125.73**
LD17_2	358.52	234.40	52.95	7200.00	323.68	323.68	0.00	59.74	323.68	323.68	0.00	**46.76**
LD17_3	391.46	155.23	152.18	7200.00	349.20	349.20	0.00	103.31	349.20	349.20	0.00	**56.40**
LD18_1	394.76	164.37	140.16	7200.00	329.43	329.43	0.00	35.29	329.43	329.43	0.00	**27.52**
LD18_2	342.50	142.67	140.06	7200.00	287.78	287.78	0.00	**63.96**	287.78	287.78	0.00	92.67
LD18_3	389.83	188.46	106.85	7200.00	368.89	368.89	0.00	**124.69**	368.89	368.89	0.00	160.26
LD19_1	348.81	166.71	109.23	7200.00	313.50	313.50	0.00	778.58	313.50	313.50	0.00	**520.81**
LD19_2	384.38	157.53	144.01	7200.00	359.53	359.53	0.00	**125.51**	359.53	359.53	0.00	141.38
LD19_3	338.75	133.64	153.48	7200.00	323.57	323.57	0.00	1290.56	323.57	323.57	0.00	**647.55**
LD20_1	407.15	152.59	166.82	7200.00	376.88	376.88	0.00	990.95	376.88	376.88	0.00	**521.94**
LD20_2	389.09	191.69	102.98	7200.00	334.67	334.67	0.00	**1549.34**	334.67	334.67	0.00	1630.62
LD20_3	382.88	149.82	155.56	7200.00	334.96	334.96	0.00	2330.24	334.97	334.96	0.00	**1299.39**
LD21_1	495.10	177.07	179.61	7200.00	400.73	400.73	0.00	534.71	400.73	400.73	0.00	**321.13**
LD21_2	408.63	134.42	203.99	**7200.00**	348.67	339.25	2.78	**7200.00**	348.67	337.64	3.27	**7200.00**
LD21_3	463.03	118.78	289.81	7200.00	365.98	365.98	0.00	1754.99	365.98	365.98	0.00	**1434.91**
VLD17_1	363.22	261.95	38.66	7200.00	337.94	337.94	0.00	108.42	337.94	337.94	0.00	**94.53**
VLD17_2	342.21	204.69	67.18	7200.00	303.03	303.03	0.00	31.74	303.03	303.03	0.00	**30.62**
VLD17_3	398.66	253.57	57.22	7200.00	371.44	371.44	0.00	36.58	371.44	371.44	0.00	**32.65**
VLD18_1	359.39	228.17	57.51	7200.00	326.79	326.79	0.00	44.47	326.79	326.79	0.00	**25.95**
VLD18_2	369.31	204.31	80.76	7200.00	344.19	344.19	0.00	412.83	344.19	344.19	0.00	**389.31**
VLD18_3	358.17	185.15	93.45	7200.00	320.31	320.31	0.00	60.54	320.31	320.31	0.00	**40.90**
VLD19_1	399.65	190.05	110.29	7200.00	337.27	337.27	0.00	113.66	337.27	337.27	0.00	**68.31**
VLD19_2	399.02	138.14	188.85	7200.00	314.04	314.04	0.00	71.67	314.04	314.04	0.00	**64.92**
VLD19_3	425.96	187.38	127.33	7200.00	392.18	392.18	0.00	974.66	392.18	392.18	0.00	**559.74**
VLD20_1	387.35	115.88	234.28	7200.00	362.99	362.99	0.00	875.97	362.99	362.99	0.00	**576.73**
VLD20_2	435.12	154.86	180.97	7200.00	355.56	355.56	0.00	163.72	355.56	355.56	0.00	**133.79**
VLD20_3	381.21	92.50	312.13	7200.00	310.86	310.86	0.00	348.91	310.86	310.86	0.00	**168.59**
VLD21_1	466.43	115.97	302.19	7200.00	388.97	388.97	0.00	**1442.15**	388.97	388.97	0.00	3804.03
VLD21_2	505.43	144.73	249.22	7200.00	387.86	387.86	0.00	1342.39	387.86	387.86	0.00	**1224.05**
VLD21_3	419.35	113.95	268.03	7200.00	337.64	337.64	0.00	4076.81	337.64	337.64	0.00	**1944.37**

Finally, we have compared the models with the branch-and-bound algorithm of Poikonen and Golden (2019) on the instances generated by the authors. These instances are distinctly different in nature, where the speed of the vehicle is only twice of the carrier i.e. $V_v = 2$, but the autonomy of the vehicle is $a = 20$. For these instances, both the MISOC of Gambella et al. (2018) and the branch-and bound algorithm of Poikonen and Golden (2019) can outperform CVTSP1′, as the results in Table 11 show. Each row of Table 11 corresponds to the average result of 25 instances for the corresponding instance type and size. We think that there are two possible reasons for the degradation of the performance of CVTSP1′ on these instances. First, the sets (35) - (37) are potentially smaller, thereby

reducing the number of optimality cuts in Proposition 5. Second, the big-M constraints (17), (19), and (21) in CVTSP1′ are considerably weaker due to the large value of a, which presumably leads to looser relaxations. Notably, (48) improved the performance of CVTSP1′ for these instances.

Table 11: Computational results for the instances of Poikonen and Golden (2019)

		Poikonen branch-and-bound	Gambella MISOC		CVTSP1′	
Instance type	Instance size	CPU	CPU	Gap (%)	CPU	Gap (%)
Regular	10	1.54	54.56	0.00	1724.13	0.71
	15	18.8	5897.45	7.78	7200.00	19.46
	20	700.22	7200.00	48.13	N/A	N/A
Clustered	10	12.585	31.04	0.00	2713.42	0.00
	15	559.557	3939.31	1.35	7200.00	22.45
	20	N/A	7200.00	16.10	N/A	N/A

5.2 Iterated Local Search

We have then proceeded to test the performance of the ILS algorithm, by testing it on the same instance set performing 10 experiments per instance. Tables 12, 13, 14, and 15 report the average results for the instances with $|T| \in \{10, 11, 12, 13, 14, 15, 20\}$, for which we have strong lower bounds provided by CVTSP1′. The average CPU time requirement per instance and the average CPU requirement by instance size are reported under column headings 'CPU' and 'Average CPU', respectively. The average optimality gap, computed as the ratio of the difference between the upper and lower bounds divided by the lower bound, is observed to have an overall average of 0.47%. The only exception is the instance SD15_3, for which none of the three models or RBA could provide an exact solution. The performance is observed to be particularly high for the instance set VLD, for which the target points are more sparsely distributed.

For larger instances with $|T| \in \{25, 30, 35, 40, 45, 50\}$, we have computed the deviation of the results found by the ILS algorithm from the Best Known Solution (BKS) value to observe the robustness of the performance, the details of which are presented in Tables 16, 17, 18, and 19. The overall deviation is computed as 0.83%, quite uniformly distributed among the four instance sets.

6 Conclusions

We have studied the CVTSP and provided structural properties that pertain to optimal solutions, i.e., perfect synchronization of the Carrier and the Vehicle, the maximum feasible take-off and landing distances for the Vehicle at the target points, and the conditions under which the Vehicle would utilize all of its autonomy in an optimal solution. We have presented a new mixed integer second order conic optimization model, based on the properties and subtour elimination constraints from the Traveling Salesman Problem literature, and augmented it with optimality cuts. Extensive computational experiments have demonstrated the superiority of our model and that it is capable of solving instances

Table 12: Results of the ILS algorithm for instance set SD

Instance	Average UB	LB	Gap (%)	CPU	Average CPU
SD11_1	109.75	108.76	0.91	13.97	
SD11_2	114.48	113.92	0.49	11.65	
SD11_3	125.59	125.19	0.32	13.78	13.13
SD12_1	108.60	107.60	0.93	17.61	
SD12_2	154.38	153.94	0.29	15.77	
SD12_3	120.29	118.61	1.42	15.63	16.34
SD13_1	116.49	116.12	0.32	22.67	
SD13_2	138.31	136.86	1.06	19.87	
SD13_3	122.24	121.47	0.64	21.02	21.19
SD14_1	131.14	128.14	2.34	27.40	
SD14_2	127.65	124.66	2.40	25.95	
SD14_3	139.71	138.07	1.19	28.21	27.18
SD15_1	126.63	123.67	2.40	33.97	
SD15_2	136.66	136.10	0.41	34.29	
SD15_3	134.94	118.36	14.01	31.07	33.11
SD16_1	145.96	145.10	0.59	42.62	
SD16_2	156.08	155.35	0.47	41.78	
SD16_3	130.34	125.70	3.69	41.77	42.06
SD17_1	285.98	285.58	0.14	48.74	
SD17_2	342.38	341.49	0.26	47.47	
SD17_3	271.94	270.31	0.60	47.94	48.05
SD18_1	330.23	329.47	0.23	53.57	
SD18_2	314.78	314.49	0.09	54.78	
SD18_3	342.28	341.80	0.14	56.31	54.89
SD19_1	360.07	359.70	0.10	64.78	
SD19_2	312.38	312.15	0.07	68.55	
SD19_3	379.66	379.31	0.09	62.67	65.33
SD20_1	369.76	365.93	1.05	70.73	
SD20_2	356.34	355.71	0.18	73.30	
SD20_3	342.08	341.76	0.09	78.34	74.12
SD21_1	373.95	371.82	0.57	86.82	
SD21_2	375.07	373.80	0.34	88.77	
SD21_3	351.42	350.28	0.33	81.52	85.70

Table 13: Results of the ILS algorithm for instance set MD

Instance	Average UB	LB	Gap (%)	CPU	Average CPU
MD11_1	147.00	146.85	0.10	15.28	
MD11_2	132.35	132.12	0.17	14.75	
MD11_3	133.57	133.42	0.11	14.19	14.74
MD12_1	157.95	157.74	0.13	19.57	
MD12_2	165.77	165.63	0.09	17.75	
MD12_3	121.59	121.24	0.29	18.19	18.50
MD13_1	151.37	150.89	0.32	21.62	
MD13_2	131.55	130.99	0.43	20.83	
MD13_3	150.72	150.37	0.23	21.16	21.20
MD14_1	147.74	146.95	0.54	28.04	
MD14_2	164.19	163.59	0.37	26.55	
MD14_3	153.43	153.01	0.27	26.85	27.15
MD15_1	168.83	168.24	0.35	32.74	
MD15_2	137.60	136.94	0.48	36.28	
MD15_3	159.92	157.86	1.30	32.83	33.95
MD16_1	167.31	166.21	0.66	40.93	
MD16_2	177.55	177.23	0.18	42.46	
MD16_3	165.29	164.37	0.56	42.69	42.03
MD17_1	322.40	322.21	0.06	45.95	
MD17_2	350.77	350.08	0.20	43.49	
MD17_3	356.45	356.27	0.05	53.48	47.64
MD18_1	346.81	346.43	0.11	55.20	
MD18_2	337.24	335.27	0.59	53.48	
MD18_3	376.45	376.27	0.05	59.86	56.18
MD19_1	364.90	364.45	0.12	63.02	
MD19_2	370.42	370.01	0.11	64.07	
MD19_3	346.49	345.01	0.43	61.58	62.89
MD20_1	342.21	341.91	0.09	73.70	
MD20_2	334.46	332.47	0.60	71.52	
MD20_3	343.11	342.91	0.06	72.88	72.70
MD21_1	364.25	363.66	0.16	83.03	
MD21_2	345.80	344.70	0.32	75.82	
MD21_3	364.37	363.71	0.18	87.88	82.24

Table 14: Results of the ILS algorithm for instance set LD

Instance	Average UB	LB	Gap (%)	CPU	Average CPU
LD11_1	311.58	311.55	0.01	14.39	
LD11_2	345.24	345.19	0.01	13.62	
LD11_3	299.61	299.53	0.03	14.69	14.23
LD12_1	296.65	296.07	0.20	17.95	
LD12_2	308.49	308.26	0.08	18.31	
LD12_3	270.53	270.31	0.08	20.75	19.00
LD13_1	261.88	261.64	0.09	25.64	
LD13_2	294.95	294.85	0.03	24.86	
LD13_3	307.60	307.34	0.08	24.42	24.97
LD14_1	320.01	319.79	0.07	31.40	
LD14_2	283.52	282.91	0.21	27.45	
LD14_3	302.12	301.60	0.17	29.13	29.33
LD15_1	299.21	299.04	0.06	35.16	
LD15_2	314.16	314.01	0.05	34.59	
LD15_3	325.45	324.79	0.20	35.19	34.98
LD16_1	322.23	322.05	0.06	42.72	
LD16_2	339.07	338.70	0.11	41.74	
LD16_3	354.81	353.87	0.27	41.23	41.90
LD17_1	380.68	380.11	0.15	51.61	
LD17_2	323.98	323.68	0.09	46.56	
LD17_3	349.45	349.20	0.07	48.22	48.80
LD18_1	329.68	329.43	0.08	56.64	
LD18_2	287.95	287.78	0.06	53.28	
LD18_3	369.57	368.89	0.18	61.41	57.11
LD19_1	313.90	313.50	0.13	68.34	
LD19_2	359.92	359.53	0.11	69.25	
LD19_3	323.89	323.57	0.10	64.60	67.40
LD20_1	377.91	376.88	0.27	71.18	
LD20_2	337.06	334.67	0.71	72.63	
LD20_3	335.74	334.96	0.23	79.21	74.34
LD21_1	401.85	400.73	0.28	88.64	
LD21_2	351.03	339.25	3.47	87.91	
LD21_3	366.23	365.98	0.07	81.49	86.01

Table 15: Results of the ILS algorithm for instance set VLD

Instance	Average UB	LB	Gap (%)	CPU	Average CPU
VLD11_1	257.27	257.24	0.01	15.14	
VLD11_2	324.77	324.75	0.01	14.73	
VLD11_3	226.26	226.13	0.06	14.45	14.77
VLD12_1	326.07	326.04	0.01	19.86	
VLD12_2	274.23	274.19	0.01	19.64	
VLD12_3	282.00	281.94	0.02	19.67	19.72
VLD13_1	316.98	316.71	0.09	26.37	
VLD13_2	239.43	239.26	0.07	25.09	
VLD13_3	281.45	281.33	0.04	24.19	25.21
VLD14_1	319.81	319.63	0.06	28.45	
VLD14_2	300.42	300.17	0.08	29.17	
VLD14_3	280.54	280.37	0.06	31.60	29.74
VLD15_1	295.47	295.33	0.05	40.02	
VLD15_2	314.76	314.70	0.02	39.65	
VLD15_3	265.10	264.95	0.06	37.86	39.18
VLD16_1	380.38	379.91	0.12	39.63	
VLD16_2	355.56	355.42	0.04	44.67	
VLD16_3	306.50	305.99	0.17	42.74	42.35
VLD17_1	338.34	337.94	0.12	48.95	
VLD17_2	303.26	303.03	0.08	55.69	
VLD17_3	372.21	371.44	0.21	48.08	50.91
VLD18_1	327.38	326.79	0.18	56.74	
VLD18_2	345.38	344.19	0.34	57.23	
VLD18_3	322.60	320.31	0.72	59.15	57.71
VLD19_1	338.40	337.27	0.34	64.46	
VLD19_2	317.19	314.04	1.00	61.92	
VLD19_3	392.62	392.18	0.11	67.67	64.68
VLD20_1	364.09	362.99	0.30	75.65	
VLD20_2	355.93	355.56	0.10	70.45	
VLD20_3	311.30	310.86	0.14	70.05	72.05
VLD21_1	391.84	388.97	0.74	82.31	
VLD21_2	390.47	387.86	0.67	81.50	
VLD21_3	338.73	337.64	0.32	98.19	87.33

Table 16: Results of the ILS algorithm for larger SD instances

Instance	Average UB	BKS	Deviation (%)	CPU	Average CPU
SD26_1	395.23	394.99	0.06	158.57	
SD26_2	414.79	414.76	0.01	131.13	
SD26_3	380.52	380.22	0.08	129.30	139.67
SD31_1	410.29	410.26	0.01	203.37	
SD31_2	356.06	352.96	0.88	206.11	
SD31_3	394.38	389.35	1.29	211.65	207.04
SD36_1	484.09	482.62	0.30	289.00	
SD36_2	414.94	414.88	0.01	286.66	
SD36_3	392.26	387.95	1.11	287.85	287.84
SD41_1	488.67	482.19	1.34	366.71	
SD41_2	456.02	454.54	0.33	359.86	
SD41_3	456.87	455.94	0.20	363.10	363.23
SD46_1	512.96	507.26	1.12	474.81	
SD46_2	424.76	419.41	1.28	453.13	
SD46_3	516.00	509.85	1.21	470.62	466.19
SD51_1	453.78	452.13	0.37	593.26	
SD51_2	454.77	450.80	0.88	548.01	
SD51_3	533.79	525.36	1.60	536.61	559.29

Table 17: Results of the ILS algorithm for larger MD instances

Instance	Average UB	BKS	Deviation (%)	CPU	Average CPU
MD26_1	346.37	346.20	0.05	144.93	
MD26_2	419.88	417.45	0.58	140.36	
MD26_3	414.23	412.64	0.38	134.66	139.98
MD31_1	428.15	425.52	0.62	219.14	
MD31_2	415.73	414.11	0.39	199.21	
MD31_3	481.61	473.49	1.72	227.13	215.16
MD36_1	432.28	427.53	1.11	282.85	
MD36_2	425.04	421.18	0.92	294.72	
MD36_3	378.05	370.40	2.07	271.26	282.94
MD41_1	455.85	453.11	0.60	358.58	
MD41_2	434.06	433.82	0.06	358.43	
MD41_3	479.18	477.83	0.28	376.38	364.46
MD46_1	474.25	473.14	0.24	457.93	
MD46_2	487.96	486.95	0.21	449.78	
MD46_3	535.48	531.40	0.77	454.13	453.95
MD51_1	480.95	474.10	1.44	570.00	
MD51_2	500.74	497.72	0.61	561.57	
MD51_3	525.46	507.20	3.60	545.16	558.91

Table 18: Results of the ILS algorithm for larger LD instances

Instance	Average UB	BKS	Deviation (%)	CPU	Average CPU
LD26_1	365.45	362.71	0.75	138.11	
LD26_2	391.60	389.64	0.50	140.18	
LD26_3	371.35	371.09	0.07	137.37	138.55
LD31_1	413.12	412.80	0.08	202.15	
LD31_2	384.43	384.37	0.02	205.03	
LD31_3	427.43	427.17	0.06	197.59	201.59
LD36_1	455.68	451.19	0.99	272.50	
LD36_2	445.28	438.95	1.44	269.33	
LD36_3	483.67	474.67	1.90	278.45	273.43
LD41_1	462.42	462.34	0.02	355.77	
LD41_2	497.90	485.73	2.51	359.05	
LD41_3	494.67	492.17	0.51	359.88	358.23
LD46_1	495.06	488.51	1.34	457.03	
LD46_2	497.09	494.68	0.49	448.37	
LD46_3	544.76	538.40	1.18	452.11	452.50
LD51_1	529.29	508.23	4.14	580.37	
LD51_2	509.13	502.14	1.39	554.65	
LD51_3	498.74	496.39	0.47	560.40	565.14

Table 19: Results of the ILS algorithm for larger VLD instances

Instance	Average UB	BKS	Deviation (%)	CPU	Average CPU
VLD26_1	347.11	346.91	0.06	142.73	
VLD26_2	415.93	415.47	0.11	133.54	
VLD26_3	387.97	386.78	0.31	137.14	137.80
VLD31_1	411.53	408.62	0.71	213.70	
VLD31_2	410.16	407.70	0.60	222.12	
VLD31_3	403.32	403.29	0.01	203.76	213.20
VLD36_1	426.76	422.42	1.03	278.67	
VLD36_2	416.46	415.81	0.16	291.84	
VLD36_3	408.44	408.37	0.02	302.75	291.09
VLD41_1	466.11	461.83	0.93	366.44	
VLD41_2	426.43	423.05	0.80	347.59	
VLD41_3	458.87	453.22	1.25	360.06	358.03
VLD46_1	439.61	428.10	2.69	420.14	
VLD46_2	497.73	493.09	0.94	428.39	
VLD46_3	482.46	479.86	0.54	439.63	429.39
VLD51_1	504.40	498.77	1.13	561.99	
VLD51_2	469.78	467.14	0.57	541.90	
VLD51_3	505.94	494.39	2.34	536.31	546.74

with up to 20 target points. We have also provided an ILS algorithm that can provide near-optimal solutions within 10 minutes of CPU time for instances with up to 50 target points.

Acknowledgement

We thank Claudio Gambella for providing the benchmark problem instances. This study was partially supported by University of Bath International Research Funding Scheme. This support is gratefully acknowledged.

Appendix

Proof of Proposition 3

We now provide the proof of Proposition 3.

Proof. Assume that the conditions $(i) - (iii)$ hold for some target point $j \in T$, but the Vehicle does not use its full autonomy at target point j in any optimal solution. By Proposition 2.1, we can assume that the Carrier and the Vehicle are perfectly synchronized at all target points. Suppose that $s_{to,j}$ and $s_{l,j}$ are defined as in Table 2. Proposition 2.2 and the contrapositive of Corollary 2.3 imply that

$$\max\left\{\left\|s_{to,j} - q_j\right\|, \left\|s_{l,j} - q_j\right\|\right\} < \frac{a\left(V_v + V_c\right)}{2}.$$

It follows by (i) and (iii) that $s_{to,j} \neq p_o$ and $s_{l,j} \neq p_f$, respectively. Furthermore, we claim that there does not exist a target point $j' \in T \setminus \{j\}$ such that $s_{to,j} = s_{l,j'}$ or $s_{l,j} = s_{to,j'}$. Suppose, for a contradiction, that $s_{to,j} = s_{l,j'}$ for some $j' \in T \setminus \{j\}$. Then, by Proposition 2.2, we should have

$$\left\|s_{to,j} - q_{j'}\right\| = \left\|s_{l,j'} - q_{j'}\right\| \leq \frac{a\left(V_v + V_c\right)}{2}.$$

Then, by the triangle inequality,

$$\left\|q_j - q_{j'}\right\| \leq \left\|q_j - s_{to,j}\right\| + \left\|s_{to,j} - q_{j'}\right\| < \frac{a\left(V_v + V_c\right)}{2} + \frac{a\left(V_v + V_c\right)}{2} = a\left(V_v + V_c\right), \quad (49)$$

which contradicts the condition (ii). A similar argument can be employed if $s_{l,j} = s_{to,j'}$ for some $j' \in T \setminus \{j\}$.

We therefore make the following claim: If the Vehicle does not use full autonomy at a target point j that satisfies the conditions $(i) - (iii)$, then the solution can be modified so that the Vehicle uses full autonomy without compromising optimality.

Suppose that target point j satisfies the conditions $(i) - (iii)$ and, for simplicity, let us denote the take-off and landing points by to and ℓ, respectively. Let us denote the point

before to and the point after ℓ on the trajectory of the Carrier by i and k, respectively. Consider a circle of radius $a(V_v + V_c)/2$ centered at target point j. In Figures 4 and 5, the blue solid lines represent the trajectory of the Carrier whereas the blue dashed lines correspond to that of the Vehicle. By Corollary 2.3, both to and ℓ should be strictly inside this circle whereas neither of the points i and k can lie strictly inside this circle by the previous argument.

There are two cases:

Case 1: Suppose that the points i, to, ℓ, and k are all collinear (see Figure 4). In this case, suppose that to is replaced by to' located on the boundary of the circle and ℓ is replaced by ℓ', which is located at a distance of $V_c\, a$ from to' so that the travel time of the Carrier is exactly equal to a (see the first illustration in Figure 4). Denoting the coordinates of to' and ℓ' by $s_{to',j}$ and $s_{\ell',j}$, respectively, the triangle inequality yields

$$\big\|q_j - s_{\ell',j}\big\| \geq \big\|s_{to',j} - q_j\big\| - \big\|s_{to',j} - s_{\ell',j}\big\| = \frac{a(V_v + V_c)}{2} - V_c\, a = \frac{a(V_v - V_c)}{2},$$

which implies that the travel time of the Vehicle from to' to q_j and from q_j to ℓ' satisfies

$$\frac{\big\|s_{to',j} - q_j\big\| + \big\|q_j - s_{\ell',j}\big\|}{V_v} \geq \frac{a(V_v + V_c) + a(V_v - V_c)}{2\, V_v} = a.$$

Therefore, if the travel time of the Vehicle is equal to a, then we obtain an optimal solution with perfect synchronization since the mission completion time remains the same and the Vehicle uses its full autonomy, which establishes our claim.

If the travel time of the Vehicle is strictly greater than a, then we move the pair to' and ℓ' towards the inside of the circle while maintaing the distance of $V_c\, a$ between them. Consider the case in which the triangle with the base given by the line segment between to' and ℓ' and the apex given by j forms an isosceles triangle (see the second illustration in Figure 4). By Lemma 2.2, the height of the triangle with the base given by the line segment between to and ℓ and the apex given j satisfies

$$h_j \leq \frac{t_j^*}{2}\sqrt{V_v^2 - V_c^2} < \frac{a}{2}\sqrt{V_v^2 - V_c^2},$$

where the last inequality is due to our assumption that the Vehicle does not use its full autonomy. Clearly, the triangle with the base given by the line segment between to' and ℓ' and the apex given by j also has height h_j. Denoting the travel time of the Vehicle from to' to j and from j to ℓ' by t, we obtain

$$\frac{V_v^2\, t^2}{4} = h_j^2 + \frac{V_c^2 a^2}{4} < \frac{a^2(V_v^2 - V_c^2)}{4} + \frac{V_c^2 a^2}{4} = \frac{V_v^2\, a^2}{4}$$

by the Pythagorean theorem, which implies that $t < a$. Since the travel time of the Vehicle changes continuously as to' moves away from the boundary of the circle, we conclude that there exists a pair of points to^* and ℓ^* such that the travel time of each of the Carrier and the Vehicle is equal to a. Therefore, we obtain an optimal solution with perfect

synchronization since the mission completion time remains the same and the Vehicle uses its full autonomy.

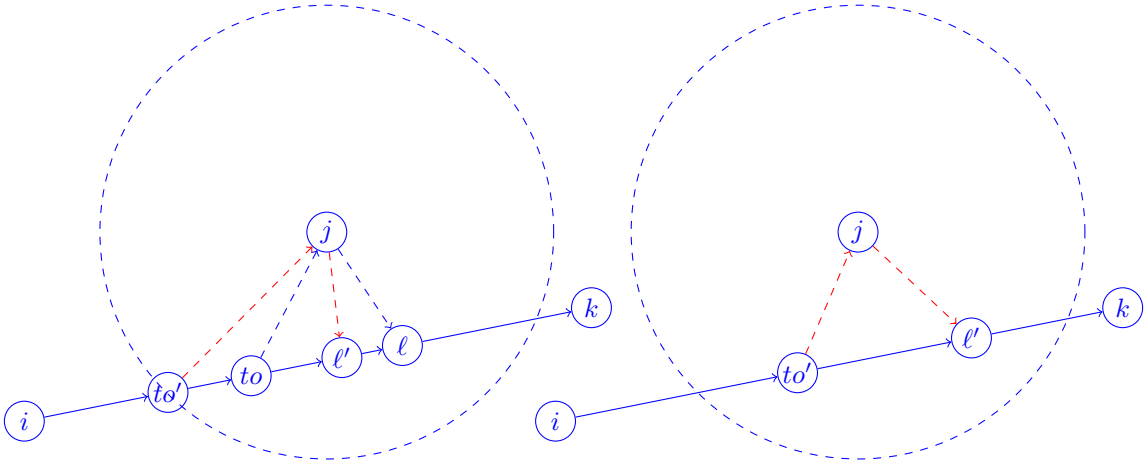

Figure 4: Illustration of Case 1 in the proof of Proposition 2.3

Case 2: Suppose that the points i, to, ℓ, and k are not collinear. In this case, we claim that the travel time of the Carrier between to and ℓ is equal to a (i.e., $t_j^c = a$) in every optimal solution. Suppose, for a contradiction, that there exists an optimal solution such that $t_j^c < a$. By Lemma 2.1, we obtain $t_j^v \leq t_j^c < a$. There are two subcases:

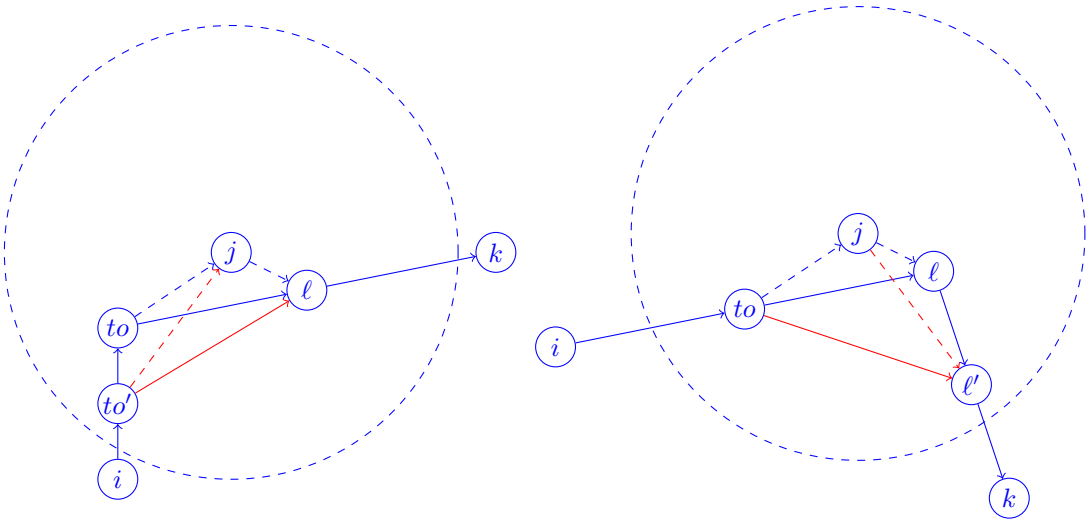

Figure 5: Illustration of Case 2 in the proof of Proposition 2.3

Case 2a: Suppose that i, to, and ℓ are not collinear. Then, we can replace to by a point to' on the incoming path of the Carrier which is closer to the boundary of the circle (see the first illustration in Figure 5) while ensuring that the maximum of the travel time of the Vehicle and the Carrier is less than or equal to a. Therefore, we retain feasibility of the solution. By the triangle inequality, the travel time of the Carrier between i and ℓ is strictly improved since i, to, and ℓ are not collinear, which contradicts the optimality of the original solution. Therefore, the travel time of the Carrier between to and ℓ is equal

to a in every optimal solution. By Proposition 2.2, there exists an optimal solution with perfect synchronization, i.e., there exists an optimal solution such that $t_v^j = t_c^j = a$. The assertion follows.

Case 2b: Suppose that to, ℓ, and k are not collinear. Then, we can replace ℓ by a point ℓ' on the outgoing path of the Carrier which is closer to the boundary of the circle (see the second illustration in Figure 5). A similar argument as in Case 2a yields a contradiction. By invoking Proposition 2.2 once again, we complete the proof. □

References

Niels A. H. Agatz, Paul Bouman, and Marie Schmidt. Optimization approaches for the traveling salesman problem with drone. *Transportation Science*, 52(4):965–981, 2018. doi: 10.1287/trsc.2017.0791. URL `https://doi.org/10.1287/trsc.2017.0791`.

E. M. Arkin and R. Hassin. Approximation algorithms for the geometric covering salesman problem. *Discrete Applied Mathematics*, 55(3):197–218, 1994.

M Battarra, G. Erdoğan, G. Laporte, and D. Vigo. The traveling salesman problem with pickups, deliveries, and handling costs. *Transportation Science*, 44(3):383–399, 2010.

H. L. Bodlaender, C. Feremans, Al. Grigoriev, E. Penninkx, R. Sitters, and T. Wolle. On the minimum corridor connection problem and other generalized geometric problems. *Computational Geometry*, 42(9):939–951, 2009.

Paul Bouman, Niels A. H. Agatz, and Marie Schmidt. Dynamic programming approaches for the traveling salesman problem with drone. *Networks*, 72(4):528–542, 2018. doi: 10.1002/net.21864. URL `https://doi.org/10.1002/net.21864`.

T.-H. Hubert Chan and Shaofeng H.-C. Jiang. Reducing curse of dimensionality: Improved PTAS for TSP (with neighborhoods) in doubling metrics. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 754–765, 2016.

Walton Pereira Coutinho, Roberto Quirino do Nascimento, Artur Alves Pessoa, and Anand Subramanian. A branch-and-bound algorithm for the close-enough traveling salesman problem. *INFORMS Journal on Computing*, 28(4):752–765, 2016. doi: 10.1287/ijoc.2016.0711. URL `https://doi.org/10.1287/ijoc.2016.0711`.

M. de Berg, J. Gudmundsson, M. J. Katz, C. Levcopoulos, M. H. Overmars, and A. F. van der Stappen. TSP with neighborhoods of varying size. *Journal of Algorithms*, 57(1):22–36, 2005.

A. Dumitrescu and J. S. B. Mitchell. Approximation algorithms for TSP with neighborhoods in the plane. *Journal of Algorithms*, 48(1):135–159, 2003.

Claudio Gambella, Andrea Lodi, and Daniele Vigo. Exact solutions for the carrier-vehicle traveling salesman problem. *Transportation Science*, 52(2):320–330, 2018. doi: 10.1287/trsc.2017.0771. URL `https://doi.org/10.1287/trsc.2017.0771`.

E. Garone, R. Naldi, A. Casavola, and E. Frazzoli. Cooperative path planning for a class of carrier-vehicle systems. In *Proceedings of the 47th IEEE Conference on Decision and Control, CDC 2008, December 9-11, 2008, Cancún, Mexico*, pages 2456–2462, 2008.

E. Garone, R. Naldi, A. Casavola, and E. Frazzoli. Planning algorithms for a class of heterogeneous multi-vehicle systems. In *Proceedings of the 8th IFAC Symposium on Nonlinear Control Systems, September 1-3, 2010, University of Bologna, Italy*, pages 969–974, 2010a.

E. Garone, R. Naldi, A. Casavola, and E. Frazzoli. Cooperative mission planning for a class of carrier-vehicle systems. In *Proceedings of the 49th IEEE Conference on Decision and Control, CDC 2010, December 15-17, 2010, Atlanta, Georgia, USA*, pages 1354–1359, 2010b.

E. Garone, R. Naldi, and A. Casavola. Traveling salesman problem for a class of carrier-vehicle systems. *Journal of Guidance, Control, and Dynamics*, 34(4):1272–1276, 2011.

E. Garone, J.-F. Determe, and R. Naldi. Generalized traveling salesman problem for carrier-vehicle systems. *Journal of Guidance, Control, and Dynamics*, 37(3):776–774, 2014.

C. Groër, B. Golden, and E. Wasil. A library of local search heuristics for the vehicle routing problem. *Mathematical Programming Computation*, 2(2):79–101, 2010.

Damon J. Gulczynski, Jeffrey W. Heath, and Carter C. Price. *The Close Enough Traveling Salesman Problem: A Discussion of Several Heuristics*, pages 271–283. Springer US, Boston, MA, 2006. ISBN 978-0-387-39934-8. doi: 10.1007/978-0-387-39934-8_16. URL `https://doi.org/10.1007/978-0-387-39934-8_16`.

Quang Minh Ha, Yves Deville, Quang Dung Pham, and Minh Hong H. On the min-cost traveling salesman problem with drone. *Transportation Research Part C: Emerging Technologies*, 86:597 – 621, 2018. ISSN 0968-090X. doi: https://doi.org/10.1016/j.trc.2017.11.015. URL `http://www.sciencedirect.com/science/article/pii/S0968090X17303327`.

M. Klauco, S. Blazek, M. Kvasnica, and M. Fikar. Mixed-integer SOCP formulation of the path planning problem for heterogeneous multi-vehicle systems. In *European Control Conference, ECC 2014, Strasbourg, France, June 24-27, 2014*, pages 1474–1479, 2014.

J. S. B. Mitchell. A PTAS for TSP with neighborhoods among fat regions in the plane. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pages 11–18, 2007.

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.

Chase C. Murray and Ritwik Raj. The multiple flying sidekicks traveling salesman prob-

lem: Parcel delivery with multiple drones. *Transportation Research Part C: Emerging Technologies*, 110:368 – 398, 2020. ISSN 0968-090X. doi: https://doi.org/10.1016/j.trc.2019.11.003. URL http://www.sciencedirect.com/science/article/pii/S0968090X19302505.

R. M. Murray. Recent research in cooperative control of multivehicle systems. *Journal of Dynamic Systems, Measurement, and Control*, 129(5):571–583, 2007.

Stefan Poikonen and Bruce Golden. The mothership and drone routing problem. *INFORMS Journal on Computing*, 2019. doi: 10.1287/ijoc.2018.0879. URL https://doi.org/10.1287/ijoc.2018.0879.

Stefan Poikonen, Bruce L. Golden, and Edward A. Wasil. A branch-and-bound approach to the traveling salesman problem with a drone. *INFORMS Journal on Computing*, 31(2):335–346, 2019. doi: 10.1287/ijoc.2018.0826. URL https://doi.org/10.1287/ijoc.2018.0826.

S. Safra and O. Schwartz. On the complexity of approximating TSP with neighborhoods and related problems. *Computational Complexity*, 14(4):281–307, 2006.

Raïssa G. Mbiadou Saleu, Laurent Deroussi, Dominique Feillet, Nathalie Grangeon, and Alain Quilliot. An iterative two-step heuristic for the parallel drone scheduling traveling salesman problem. *Networks*, 72(4):459–474, 2018. doi: 10.1002/net.21846. URL https://doi.org/10.1002/net.21846.

A. Subramanian, E. Uchoa, and L.S. Ochi. A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research*, 40(10):2519–2531, 2013.

UNEP. After the tsunami: Rapid environmental assessment, 2005. URL http://wedocs.unep.org/handle/20.500.11822/8372.