Optimization Approaches for the Traveling Salesman Problem with Drone

Niels Agatz Paul Bouman Marie Schmidt

June 27, 2016

Abstract

The fast and cost-efficient home delivery of goods ordered online is logistically challenging. Many companies are looking for new ways to cross the last-mile to their customers. One technology-enabled opportunity that recently has received much attention is the use of a drone to support deliveries. An innovative last-mile delivery concept in which a truck collaborates with a drone to make deliveries gives rise to a new variant of the traveling salesman problem (TSP) that we call the TSP with drone. In this paper, we model this problem as an IP and develop several fast route first-cluster second heuristics based on local search and dynamic programming. We prove worst-case approximation ratios for the heuristics and test their performance by comparing the solutions to the optimal solutions for small instances. In addition, we apply our heuristics to several artificial instances with different characteristics and sizes. Our experiments show that substantial savings are possible with this concept in comparison to truck-only delivery.

1 Introduction

In an effort to provide faster and more cost-efficient delivery for goods ordered online, companies are looking for new technologies to bridge the last-mile to their customers. One

	speed	weight	capacity	range
drone	high	light	one	short
truck	low	heavy	many	long

Table 1: Complementary features truck and drone

technology-driven opportunity that has recently received much attention is the deployment of unmanned aerial vehicles or drones to support parcel delivery. An important advantage of a delivery drone as compared to a regular delivery vehicle is that it can operate without a costly human pilot. Another advantage is that a drone is fast and can fly over congested roads without delay.

Several companies, including Amazon, Alibaba and Google, are currently running practical trails to investigate the use of drones for parcel delivery [Popper, 2013]. These trails typically involve multi-propeller drones that can carry parcels of approximately 2 kilograms over a range of 20 kilometers. There are examples of drones that are already used for deliveries in practice, albeit solely in a non-urban environment. DHL Parcel, for instance, recently started operating a drone delivery service to deliver medications and other urgently needed goods to one of Germany's North Sea islands [Hern, 2014]. In this example, the drone flies automated but still has to be continuously monitored. Aeronautics experts expect that drones will be able to fly autonomously and safely in urban environments within the next few years, based on rapid advances in obstacle detection and avoidance technology [Nicas and Bensinger, 2015].

While a drone is fast and relatively inexpensive in terms of costs per mile, there are also some inherent limitations to its use. The size of the drone puts an upper limit on the size of the parcels it can carry. This means that a drone has to return to the depot after each delivery, which is not very efficient. Since it is battery-powered, the range is likely to remain limited as compared to a regular, fuel-based, delivery vehicle. A regular delivery truck, on the other hand, has a long range and can carry many parcels but is also heavy and slow. Table 1 summarizes the complementary features of the truck and the drone.

One way to extend the effective range and capacity of a drone is to let it collaborate with a delivery vehicle. AMP Electric Vehicles is working with the University of Cincinnati Department of Aerospace Engineering on a drone that would be mounted on the top of its electric-powered trucks to help the truck driver make deliveries [Wohlsen, 2014]. In this system, the delivery truck and the drone collaboratively serve all customers. While the delivery truck moves between different customer locations to make deliveries, the drone simultaneously serves another set of customer locations, one by one, returning to the truck after each delivery to pickup another parcel.

From a transportation planning perspective, this innovative new concept gives rise to several relevant planning problems. Even for a single truck and a single drone, the problem involves both assignment decisions and routing decisions. Assignment decisions to determine which vehicle, drone or truck, will serve which customers, and routing decisions to determine in which sequence the customers assigned to each vehicle are visited. We call this variant of the traveling salesman problem (TSP) the TSP with Drone (TSP-D). Figure 1 provides an illustration of a small example in which five customer locations need to be served from the depot. We see that by serving two customer locations with the drone instead of the truck, we can reduce the distance traveled by the truck. By this parallelization of different delivery tasks, we can also reduce the total time required to serve all customers.

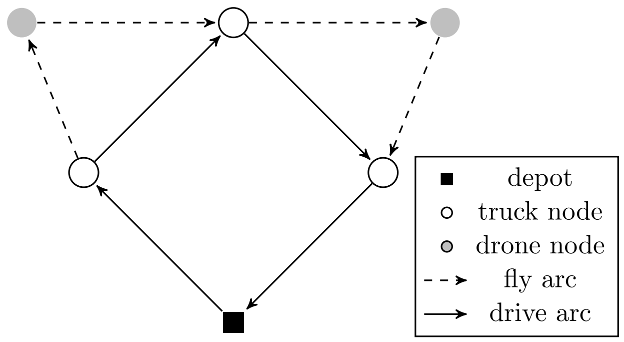


Figure 1: An example TSP-D solution

Since this is a new phenomenon, there is a need for new models and innovative algorithms to help to better understand the related planning problems and potential benefits. This paper aims to fill this gap by developing a new integer programming formulation as well as several fast route first-cluster second heuristics based on local search and dynamic programming for the TSP-D. We prove worst-case approximation ratios for the heuristics and test their performance by comparing the solutions to the optimal solutions for small instances. In addition, we apply our heuristics to several artificial instances with different characteristics and sizes to investigate the benefits of truck and drone delivery concept over truck-only delivery for different problem instances.

The main contributions of this paper are: (1) a new IP model that is able to solve instances of reasonable size to optimality. (2) several new heuristics that are very fast and provide good solutions, even for large instances. In particular, we provide a dynamic programming algorithm that is able to find the best assignment of truck and drone deliveries for any given delivery sequence. (3) a theoretical analysis of our heuristics by giving a (worst-case) approximation guarantee, and an experimental comparison of the different variants. We are the first to compare heuristic solutions to exact solutions of the TSP-D, whereas earlier work compared heuristic solutions for the TSP-D to exact solutions of the TSP. (4) a numerical study to assess the performance of a drone delivery system in various customer densities, geographic distributions and drone speeds.

The remainder of this paper is organized as follows. In Section 2, we present the related literature. In Section 3, we formally define the problem. In Section 4, we provide some theoretical insights that help us to develop suitable solution approaches. In Section 5, we provide an integer programming formulation and propose several heuristics in Section 6. In Section 7, we discuss the results from our computational experiments. Finally, we conclude with some directions for future research in Section 8.

2 Related Literature

There is an extensive body of research on the traveling salesman problem (TSP). For an excellent overview of the recent advancements in this area see Applegate et al. [2011].

One variant of the TSP that is conceptually related to the TSP-D is the covering salesman problem (CSP) as introduced by Current and Schilling [1989]. The CSP aims to find the shortest tour of a subset of given nodes such that every node that is not on the tour is within a predefined covering distance of a node on the tour. Current and Schilling [1989] propose a heuristic solution procedure that is based on the set covering problem. Several generalizations and special cases of the CSP have been studied in the literature, see e.g. Gulczynski et al. [2006], Shuttleworth et al. [2008], Golden et al. [2012], Behdani and Smith [2014]. Similar to the CSP, in the TSP-D the truck does not have to visit all nodes. However, in the TSP-D it is not enough to pass nodes that are not visited by the truck within a certain distance threshold, but drone visits have to be scheduled for each such node, and truck and drone need to be synchronized.

From this perspective, the TSP-D can be considered to fall in the class of vehicle routing problems that require synchronization between vehicles [Drexl, 2012]. The TSP-D shares some aspects with the truck and trailer routing problem (TTRP). In this problem, a vehicle composed of a truck with a detachable trailer serves the demand of a set of customers reachable by truck and trailer or only by the truck without the trailer. Several heuristics based on the cluster first-route second principle have been proposed to solve the TTRP, including Tabu search [Scheuerer, 2006] and simulated annealing [Lin et al., 2009]. We are only aware of one paper that uses a route first-cluster second procedure by Villegas et al. [2011]. Furthermore, for the TTRP some exact approaches based on branch-and-cut [Drexl, 2014] and branch-and-price [Drexl, 2011] have been studied. The main difference between the TTRP and the TSP-D is that both the truck and the drone can separately serve customer locations in the TSP-D, while the trailer cannot serve customers without the truck in the TTRP.

We are aware of two other papers that consider a combined truck and drone delivery system. Murray and Chu [2015] investigate the "Flying Sidekick Traveling Salesman Problem" which is very similar to the TSP-D. To solve the problem, they propose a mixed integer programming formulation for two variants of the problem and consider two simple heuristics which we compare to our heuristic approaches in our computational experiments.

Wang et al. [2016] derive a number of worst-case results for the vehicle routing problem with drones, in which a fleet of trucks equipped with drones delivers packages to customers.

Apart from the application of the drone as a delivery vehicle, there are several works on operational aspects in using drones for military and civil surveillance tasks, see, e.g., Evers et al. [2014], Kim et al. [2014], Valavanis and Vachtsevanos [2015].

3 Problem Definition

The TSP-D can be modeled in a graph $G = (V, E)$, where node v_0 represents the depot and N nodes $v_1, \ldots, v_N$ serve as the customer locations. Let $c(e) = c(v_i, v_j)$ and $c^d(e) = c^d(v_i, v_j)$ be the travel time between v_i and v_j of respectively the truck and the drone. The drone is typically faster than the truck, i.e. $c^d(e) \leq c(e)$. The reason for this is that the drone may not need to follow the road network but may be able to use short-cuts. Moreover, even if the drone is restricted to follow the same road network as the truck due to safety and privacy regulations, in contrast to the truck, the drone is not affected by congestion. Note that the definition of network distances as driving or flying times implies that the *triangle inequality* holds for both c and c^d.

The objective of the TSP-D is to find the shortest tour, in terms of time, to serve all customer locations by either the truck or the drone. We let $V^t \subseteq V$ be the set of nodes which need to be served by the truck since they are not suitable for drone delivery.

Throughout our analysis, we make the following three assumptions:

1. The drone has unit-capacity and has to return to the truck after each delivery.

2. The pickup of parcels from the truck can only take place at the customer locations. That is, the drone can only land on and depart from the truck while it is parked at a customer location or the depot.

3. To simplify notation we furthermore assume that the pickup and delivery of packages by both vehicles can be neglected. We also assume that recharging time of the drone can be neglected, e.g. by swapping batteries. We discuss how this assumption can be relaxed in Section 8.

In case that battery life of the drone is limited we can specify a certain maximum flying distance d^{max} (and a corresponding flight time of t^{max}) in each flight.

A *solution* to the TSP-D is hence a truck route $\mathcal{R} = (r_0 = v_0, r_1, \ldots, r_n = v_0)$ from v_0 to v_0 together with a drone route $\mathcal{D} = (d_0 = v_0, d_1, \ldots, d_m = v_0)$, where all nodes $v \in V^t$ need to be contained in $\mathcal{R}$. The drone route describes the full path of the drone, including all customers that are visited by both truck and drone.

We distinguish between the following types of nodes.

Drone node: a node that is visited by the drone separated from the truck

Truck node: a node that is visited by the truck separated from the drone

Combined node: a node that is visited by both truck and drone

To compute the time needed for the pair of tours $(\mathcal{R}, \mathcal{D})$, we cannot simply sum up truck and/ or drone driving times since the two vehicles need to be synchronized, i.e., they have to wait for each other at the combined nodes. Therefore, it is beneficial to introduce the concept of an *operation*.

An operation k consists of two combined nodes, called *start node* and *end node*, at most one drone node, and a non-negative number of truck nodes. If the operation contains a drone node, the drone departs from the truck at the start node, then serves the drone node and meets up with the truck again at the end node. The truck can travel directly from the start node to the end node or can visit any number of truck nodes in between. Moreover, the truck can also wait at the start node for the drone to return. In this case, the start node is equal to the end node. If the operation does not contain a drone node, the operation consists of just one edge between start node and end node. The truck rides from start node to end node while the drone stays parked on the truck.

Hence, an operation can be described as a pair of subtours $(\mathcal{R}_k, \mathcal{D}_k)$, where $\mathcal{R}_k$ is the subsequence of $\mathcal{R}$ starting at the kth combined node of $\mathcal{R}$ and up to and including the $(k+1)$th combined node of $\mathcal{R}$. Similarly $\mathcal{D}_k$ is the subsequence of $\mathcal{D}$ from the kth combined node of $\mathcal{D}$ up to and including the $(k+1)$th combined node of $\mathcal{D}$. We denote by $c(\mathcal{R}_k) := \sum_{e \in \mathcal{R}_k} c(e)$ and

$c^d(\mathcal{D}_k) := \sum_{e\in\mathcal{R}_d} c^d(e)$ the driving time/flying time on the subtours $\mathcal{R}_k$ and $\mathcal{D}_k$, respectively. Note that the subsequence of the drone contains at most two edges for any feasible solution because we assume that at most one drone delivery can take place within an operation.

To evaluate the time duration of a solution $(\mathcal{R}, \mathcal{S})$ to the TSP-D, it is convenient to regard $(\mathcal{R}, \mathcal{S})$ as a sequence of operations $(o_1, o_2, \dots, o_l)$ in the above-described way. In the case that the drone is at least as fast as the truck between every pair of nodes, the time duration of an operation o_k can be computed as $t(o_k) = \max\{c(\mathcal{R}_k), c^d(\mathcal{D}_k)\}$. In the case that the drone can be slower than the truck, we have

$$t(o_k) = \begin{cases} \max\{c(\mathcal{R}_k), c^d(\mathcal{D}_k)\} & \text{if } o_k \text{ contains a drone node,} \\ c(\mathcal{R}_k) & \text{if } o_k \text{ does not contain a drone node.} \end{cases} \tag{1}$$

Finally, the overall time needed to serve all customers using TSP-D tour $(\mathcal{R}, \mathcal{D})$ is

$$t(\mathcal{R}, \mathcal{D}) := \sum_{k\in K} t(o_k).$$

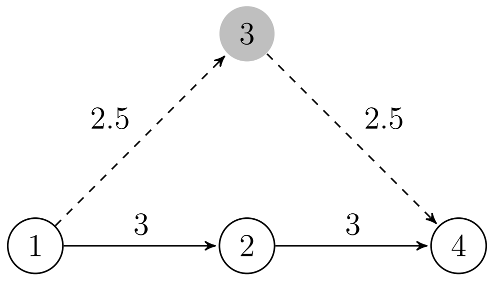


Figure 2: An operation with a combined node serving as start node (1), a combined node serving as end node (4), a drone node (3) and a truck node (2)

Figure 2 provides an illustrative example of an operation, where the dashed arcs correspond to the path of the drone and the solid arcs to the path of the truck. The numbers above the arcs represent the travel time of the truck and the drone. This operation has node 1 as the start node, node 3 as the drone node and node 2 as a truck node. The time that it takes to complete this operation is $\max(6, 5) = 6$ and the drone will need to wait for the truck at node 4.

4 Theoretical Insights

Since the TSP-D generalizes the NP-hard Hamiltonian Cycle Problem [Karp, 1972], the TSP-D is NP-hard. In the following, we describe some insights into the TSP-D, which help us to develop suitable solution algorithms in the subsequent sections.

To specify bounds and approximation results, we express the maximum speed-up that can be gained by using the drone instead of the truck between two nodes v_i and v_j by α. That is $\alpha := \max_{\{v_i,v_j\}\in E} \frac{c(v_i,v_j)}{c^d(v_i,v_j)}$. In this Section, we provide the proofs for the case that the drone is at least as fast as the truck, that is $c^d(v_i, v_j) \leq c^d(v_i, v_j)$ for all nodes v_i, v_j to avoid unnecessary notation. However, the results also hold if the drone speed is lower than the truck speed for some pair of nodes, see the appendix for details.

In Section 4.1, we start by analyzing how much we can gain by delivering goods with truck and drone instead of only the truck. Afterwards, in Section 4.2, we prove a lower bound on the optimal solution value of the TSP-D, and show that TSP algorithms can be used as approximation algorithms for the TSP-D.

Finally, in Section 4.3, we we state some properties of optimal TSP-D solutions that we use for our integer programming formulation in Section 5.

4.1 Comparing truck-and-drone delivery to truck-only delivery

The use of a drone in combination with a truck allows for the parallelization of different delivery operations and can thereby reduce the total time required to serve all customers.

Consider the example depicted in Figure 3 with depot node v_0 and two customer nodes v_1 and v_2 that need to be served from a central depot location and could both be served by the drone, i.e., $V^t = \emptyset$. For this example we assume that $c^d(e) = \frac{1}{\alpha}c(e)$ for all edges e and a given α.

The truck driving time between v_0 and v_1 is 1, the truck driving time between v_0 and v_2 is α, and the truck driving time between v_1 and v_2 is $1+\alpha$. The optimal solution to the TSP-D is to serve v_1 with the truck and v_2 with the drone, which leads to a total time consumption of 2. However, if only the truck can be used, i.e., if we solve a TSP, the truck would serve both customers one after the other, leading to a total time consumption of $2 + 2\alpha$. This provides savings in the total service time of a factor $1 + \alpha$.

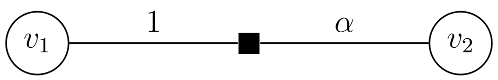


Figure 3: Two customers (circle) that need to be served from the depot (rectangle)

Indeed, we show in Theorem 4.1 that the maximum savings which can be obtained by employing a drone are of factor $1 + \alpha$.

To prove Theorem 4.1, we observe that under the assumption that the drone is at least as fast as the truck, for any TSP-D tour $(\mathcal{R}, \mathcal{D})$ it holds that

$$t(\mathcal{R}, \mathcal{D}) \geq \max\{c(\mathcal{R}), c^d(\mathcal{D})\}, \tag{2}$$

i.e., the tour duration of the TSP-D tour is always at least as long as the driving time of the truck, and at least as long as the flying time of the drone. Furthermore, we can specify the following lower bound on the length of the drone tour in an optimal TSP-D tour.

Lemma 4.1. *Let $(\mathcal{R}, \mathcal{D})$ be a TSP-D tour. Then*

$$c(\mathcal{D}) \geq 2 \cdot \sum_{v \in V_\mathcal{D} \setminus V_\mathcal{R}} \min_{w \in V_\mathcal{R}} c(v, w). \tag{3}$$

Proof.

$$c(\mathcal{D}) = \sum_{e \in E_\mathcal{D}} c(e) \geq \sum_{e \in E_\mathcal{D} \setminus E_\mathcal{R}} c(e) \geq 2 \cdot \sum_{v \in V_\mathcal{D} \setminus V_\mathcal{R}} \min_{w \in V_\mathcal{R}} c(v, w).$$

The last inequality holds because every operation contains at most one drone node, i.e., the drone visits a combined node in $V_\mathcal{R} \cap V_\mathcal{D}$ right before and right after visiting a drone node in $V_\mathcal{D} \setminus V_\mathcal{R}$. □

Theorem 4.1. *An optimal solution to the TSP is a $(1 + \alpha)$-approximation to the TSP-D.*

Proof. Here, we prove this result only for the case $c^d(v_i, v_j) \leq c(v_i, v_j)$ for all nodes v_i, v_j. A proof for the general setting is given in the Appendix. Let $(\mathcal{R}^*, \mathcal{D}^*)$ be an optimal solution to the TSP-D and $\mathcal{R}^{\text{TSP}}$ be an optimal solution to the TSP.

From $(\mathcal{R}^*, \mathcal{D}^*)$ we can construct a TSP tour $\mathcal{R}$ in the following way: Start with $\mathcal{R}^*$. For every drone node $v \in V_{\mathcal{D}^*}$, we pick the node $w \in V_{\mathcal{R}^*}$ which is closest to v with respect to the driving time c and insert arcs (w, v) and (v, w) into the tour. Then we have

$$c(\mathcal{R}^{\text{TSP}}) \leq c(\mathcal{R}) = c(\mathcal{R}^*) + 2 \cdot \sum_{v \in V_{\mathcal{D}^*} \setminus V_{\mathcal{R}^*}} \min_{w \in V_{\mathcal{R}^*}} c(v, w).$$

Using (3) we conclude

$$c(\mathcal{R}^{\text{TSP}}) \leq c(\mathcal{R}^*) + c(\mathcal{D}^*). \tag{4}$$

Furthermore, using (2) we obtain

$$t(\mathcal{R}^*, \mathcal{D}^*) \geq \max\{c(\mathcal{R}^*), c^d(\mathcal{D}^*)\} \geq \max\{c(\mathcal{R}^*), \frac{1}{\alpha} c(\mathcal{D}^*)\}, \tag{5}$$

where the last inequality follows from the definition of α.

Combining (4) and (5) we obtain

$$t(\mathcal{R}^*, \mathcal{D}^*) \geq \max\{c(\mathcal{R}^{\text{TSP}}) - c(\mathcal{D}^*), \frac{1}{\alpha} c(\mathcal{D}^*)\} \geq \frac{c(\mathcal{R}^{\text{TSP}})}{1 + \alpha}.$$

The last inequality follows from the fact that $\max_{y \in \mathbb{R}^+}\{c(\mathcal{R}^*) - y, \frac{1}{\alpha} y\}$ is minimal if $c(\mathcal{R}^*) - y = \frac{1}{\alpha} y$. We conclude that

$$t(\mathcal{R}^{\text{TSP}}, \mathcal{R}^{\text{TSP}}) = c(\mathcal{R}^{\text{TSP}}) \leq (1 + \alpha) t(\mathcal{R}^*, \mathcal{D}^*).$$

□

This result proves that the maximum gain we can obtain by employing truck and drone together is of factor $1 + \alpha$. On the other hand, the result also proves that for instances with a fixed bound α the TSP-D is constant-factor approximable in polynomial time, since any γ approximation algorithm for the TSP yields a $\gamma(1+\alpha)$-approximation algorithm for the TSP-D. We can, e.g., conclude that the minimum spanning tree (MST) algorithm [Rosenkrantz et al., 1977] with $\gamma = 2$ provides a $2 + 2\alpha$ approximation of the TSP-D and that Christofides algorithm [Christofides, 1976] with $\gamma = 1.5$ provides a $1.5 + 1.5\alpha$ approximation. However, based on the bound from Lemma 4.2 in the next section, we can obtain an even better approximation guarantee for the MST algorithm and Christofides algorithm.

4.2 A lower bound and an approximation result for the TSP-D

Based on Lemma 4.1 we can prove a lower bound on the time duration of an optimal solution to the TSP-D, which is an (α-dependent) multiple of the size of a minimum spanning tree in the considered network.

Lemma 4.2. *Let $(\mathcal{R}^*, \mathcal{D}^*)$ be an optimal solution to the TSP-D and let $T = (V_T, E_T)$ be a minimum spanning tree in G. Then we have*

$$t(\mathcal{R}^*, \mathcal{D}^*) \geq \frac{2}{2+\alpha} c(T).$$

Proof. Here, we prove this result only for the case $c^d(v_i, v_j) \leq c(v_i, v_j)$ for all nodes v_i, v_j. A proof for the general setting is given in the Appendix.

Based on the node sets $V_{\mathcal{R}^*}$ of $\mathcal{R}^*$ and $V_{\mathcal{D}^*}$ of $\mathcal{D}^*$ we construct a spanning tree T' of G as follows: first, we remove one edge from $\mathcal{R}^*$ to obtain a spanning tree $T_{\mathcal{R}^*}$ of the node set $V_{\mathcal{R}^*}$. Then we connect all nodes v from the node set $V_{\mathcal{D}^*} \setminus V_{\mathcal{R}^*}$ to the closest node in $V_{\mathcal{R}^*}$. Hence

$$c(T') = c(T_{\mathcal{R}^*}) + \sum_{v \in V_{\mathcal{D}^*} \setminus V_{\mathcal{R}^*}} \min_{w \in V_{\mathcal{R}^*}} c(v, w). \tag{6}$$

Using (3), $c(T_{\mathcal{R}^*}) \leq c(\mathcal{R}^*)$, and (6) we obtain

$$\begin{aligned}
t(\mathcal{R}^*, \mathcal{D}^*) &\geq \max\{c(\mathcal{R}^*), c^d(\mathcal{D}^*)\} && (7)\\
&\geq \max\{c(\mathcal{R}^*), \frac{1}{\alpha} c(\mathcal{D}^*)\} && (8)\\
&\geq \max\{c(T_{\mathcal{R}^*}), \frac{2}{\alpha} \sum_{v \in V_{\mathcal{D}^*} \setminus V_{\mathcal{R}^*}} \min_{w \in V_{\mathcal{R}^*}} c(v, w)\} && (9)\\
&= \max\{c(T_{\mathcal{R}^*}), \frac{2}{\alpha}(c(T') - c(T_{\mathcal{R}^*}))\} && (10)\\
&\geq \frac{2}{2+\alpha} c(T') && (11)\\
&\geq \frac{2}{2+\alpha} c(T) && (12)
\end{aligned}$$

for the minimum spanning tree T of G. Hereby, inequality (11) holds because $\max_{x \in \mathbb{R}^+}\{x, \frac{2}{\alpha}(c(T') - x)\}$ is minimal when $x = \frac{2}{\alpha}(c(T') - x)$. □

The result of Lemma 4.2 gives us a second (and better) approximation result for solving the TSP-D with the MST algorithm, than the one provided at the end of Section 4.1.

Theorem 4.2. *A solution $(\mathcal{R}, \mathcal{R})$, consisting of a truck tour $\mathcal{R}$ constructed with the minimum spanning tree heuristic for the TSP and no drone deliveries, is a $(2+\alpha)$-approximation for the TSP-D.*

Proof. The length of a tour $\mathcal{R}$ constructed with the minimum spanning tree heuristic for the TSP is $2c(T)$ (for T being a minimum spanning tree). Using the bound from Lemma 4.2, we conclude that this is a

$$\frac{2c(T)}{\frac{2}{2+\alpha}c(T)} = 2 + \alpha$$

approximation. $\square$

Of course, this approximation guarantee holds as well for TSP heuristics which improve on the minimum spanning tree heuristic, i.e., which are guaranteed to achieve a worst-case objective value $< 2c(T)$, as, e.g., Christofides algorithm.

4.3 Properties of optimal solutions

We conclude this section with some insights in the characteristics of the optimal solutions of a TSP-D that are relevant for the IP formulation in the next section.

Observation 4.1. *An optimal solution to the TSP-D may require that a combined node is visited more than once.*

To see this, consider the example given in Figure 4 with six customer locations, where each arc has the same distance, and where the drone is twice as fast as the truck. As before, the dashed arcs correspond to the path of the drone and the solid arcs to the path of the truck.

In this example, it is optimal for the truck to first travel to node (1), then to node (2), and then back to node (1) before returning to the depot. Since the drone is twice as fast as

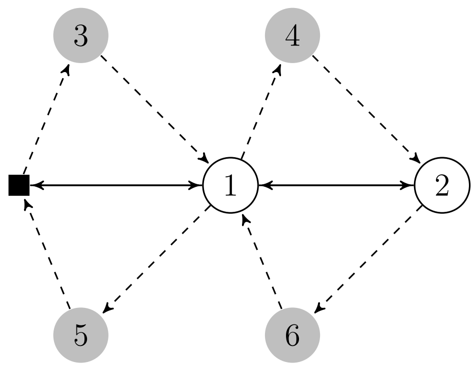


Figure 4: A solution with four drone nodes (3,4,5,6) and two combined nodes (1,2) in which node (1) is visited twice.

the truck there are no waiting times in this solution. The reason that it may be beneficial for the truck to visit a node twice is that a truck can visit a node either to serve a customer or to supply the drone with another parcel.

On the other hand it is easy to show that it is never necessary to visit a truck node or a drone node again.

Observation 4.2. *To each instance of the TSP-D there exists an optimal solution such that each drone node and each truck node is visited only once.*

To see this, consider an optimal solution $(\mathcal{R}, \mathcal{D})$ of an instance of the TSP-D such that node v occurs twice and that at least in one of these occurrences it is not used as a combined node. Then (since we have the triangle inequality), removing v on its first occurrence as a non-combined node leads to a solution with at most the same objective value.

5 An Integer Programming Formulation

We let O denote the set of *feasible* operations, where c_o denotes the costs of operation $o \in O$. An operation is feasible if it contains at most one drone flight, two combined nodes and a non-negative number of truck nodes. However, it is easy to take into account other practical restrictions on the path of the drone and/ or truck, e.g. maximum flight distance d^{max} and time t^{max}, nodes that cannot be visited by the drone V^t, maximum waiting times, etc. The x

variables indicate whether operation o is chosen ($x_o = 1$) or not ($x_o = 0$). We let $O^-(v) \subset O$ represent the set of operations with start node v, $O^+(v) \subset O$ the set of operations with end node v, and $O(v) \subset O$ the set of all operations that contain node v. Analogously, for each set of nodes $S \subset V$ we define $O^-(S)$ to be the set of all operations with start node in S and end node in $V \setminus S$. For each set of nodes $S \subset V$ define $O^+(S)$ to be the set of all operations with end node in S and start node in $V \setminus S$. Let y_v be an auxiliary variable that indicates whether node v is chosen as a start node in at least one operation. This results in the following IP formulation:

$$\min \sum_{o \in O} c_o x_o \tag{13}$$

$$\text{such that} \sum_{o \in O(v)} x_o \geq 1 \qquad \forall v \in V \tag{14}$$

$$\sum_{o \in O^+(v)} x_o \leq n \cdot y_v \qquad \forall v \in V \tag{15}$$

$$\sum_{o \in O^+(v)} x_o = \sum_{o \in O^-(v)} x_o \qquad \forall v \in V \tag{16}$$

$$\sum_{o \in O^+(S)} x_o \geq y_v \qquad \forall S \subset V \setminus \{v_0\}, v \in S \tag{17}$$

$$\sum_{o \in O^+(v_0)} x_o \geq 1 \tag{18}$$

$$y_{v_0} = 1 \tag{19}$$

$$x_o \in \{0, 1\} \qquad \forall o \in O \tag{20}$$

$$y_v \in \{0, 1\} \qquad \forall i \in V \tag{21}$$

The objective function (13) minimizes the costs of the tour, which are the sum of the costs of the operations chosen. Constraints (14) ensure that all nodes are covered. Due to constraints (15), y_v is set to 1 if at least one chosen operation uses v as a start node. The left-hand side of (15) is at most n because each operation must contain at least one previously unvisited node in any optimal solution.

Considering operations as arcs from their start node to their end node, constraints (16-18) ensure that the chosen operations $O' := \{o \in O : x_o = 1\}$ span an Eulerian graph G'

such that an Eulerian cycle in this graph represents a feasible truck-and-drone tour $(\mathcal{R}, \mathcal{D})$. More formally, we can define G' as a multigraph (V', E') with $V' := \{v \in V : y_v = 1\}$ and$e' = (v_i, v_j)$ for each operation $o \in O'$, start node v_i and end node v_j. We can see that the graph G' is connected due to constraints (17) and (18)) and is Eulerian due to constraints (16). To obtain a truck-and-drone tour from the IP solution, we simply need to compute an Eulerian cycle $(e_1, e_2, \ldots, e_{|O|})$ in G'. Constraint (19) ensures that this tour starts (and ends) at the depot. Constraints (20) and (21) force the variables x_o and y_v to be binary.

6 Heuristic Approaches

We use a route first-cluster second procedure (see, e.g., Beasley [1983]) to find a solution $(\mathcal{R}, \mathcal{D})$ as follows:

1. We construct a truck tour $\mathcal{R}_1$ which contains *all* nodes. Note that $(\mathcal{R}_1, \mathcal{R}_1)$ is already a solution to the TSP-D (where all nodes are combined nodes, i.e., the drone does not deliver any parcels). This is detailed in Section 6.1.

2. We split tour $\mathcal{R}_1$ into drone tour $\mathcal{D}$ and truck tour $\mathcal{R}$, leading to a solution $(\mathcal{R}, \mathcal{D})$ of the TSP-D. This is described in Section 6.2.

6.1 Finding a truck tour

As a first step of our heuristic approach, we assume that the drone does not make any deliveries and construct a tour for the truck only, which visits all customers. This is a (regular) traveling salesman problem. Although the traveling salesman problem is NP-hard [Garey and Johnson, 1979], it is a well-researched problem and instances of up to several hundreds of nodes can be solved quickly with a state of the art TSP solver such as Concorde [Applegate et al., 2001]. Let $\mathcal{R}_{TSP}$ be an optimal solution to the TSP and let $(\mathcal{R}^*, \mathcal{D}^*)$ be an optimal solution to the TSP-D. Then according to Theorem 4.1 we have

$$t(\mathcal{R}_{TSP}, \mathcal{R}_{TSP}) \leq (1 + \alpha) t(\mathcal{R}^*, \mathcal{D}^*).$$

Note that the approach to find an optimal TSP tour first and then solve the truck and drone partitioning for this tour does in general not lead to an optimal solution of the TSP-D. This justifies the use of any TSP heuristic to construct an initial tour during our experiments. See Laporte [1992] for an overview of different heuristics and approximation algorithms. As a fast alternative to the optimal tour, we use Kruskal's minimum spanning tree algorithm to construct a TSP tour in time $O(|E| \log |E|)$ for general graphs [Kruskal, 1956]. It can be used to obtain a minimum spanning tree for a Euclidean graph in expected time $O(|V| \log |V|)$ using a Delaunay Triangulation [Shamos, 1978], circumventing the need to consider all entries in the distance matrix. After computing a minimum spanning tree T for G we can report the nodes in T according to a depth first preorder traversal starting at the depot. This yields a tour $\mathcal{R}_{MST}$. Since it is well known that this tour is a 2-approximation for the optimal TSP tour [Rosenkrantz et al., 1977], Theorem 4.2 implies

$$t(\mathcal{R}_{MST}, \mathcal{R}_{MST}) \leq (2 + \alpha) t(\mathcal{R}^*, \mathcal{D}^*).$$

6.2 Constructing a TSP-D tour from a TSP tour

Based on the truck tour $\mathcal{R}_1 := \mathcal{R}_{TSP}$ or $\mathcal{R}_1 := \mathcal{R}_{MST}$ we construct a solution $(\mathcal{R}, \mathcal{D})$ by assigning some nodes as drone nodes. Both approaches have in common that the order in which the nodes are visited remains unchanged with respect to $\mathcal{R}_1$, i.e. both $\mathcal{R}$ and $\mathcal{D}$ are subtours of $\mathcal{R}_1$. We propose two different approaches: in Section 6.2.1 we describe a fast greedy heuristic, in Section 6.2.2 we propose an exact partitioning algorithm based on dynamic programming which is slower but finds an optimal solution among all solutions which leave the sequence of nodes visited unchanged.

6.2.1 A greedy partitioning heuristic

We first propose a greedy heuristic to partition $\mathcal{R}_1$ into a subtour for the drone and a subtour for the truck. During the heuristic, each node $v_1, v_2, \ldots, v_N$ is assigned a label which is dynamically updated. Besides the labels *truck*, *drone*, and *combined*, which indicate truck, drone, and combined nodes, we have an additional label *simple* for yet unprocessed nodes.

Initially all nodes are assigned the *simple* label. In every step of the heuristic, the number of *simple* nodes decreases by at least one and the algorithm terminates when there are no *simple* nodes left.

After termination of the heuristic, the subtour for the truck $\mathcal{R}$ can be computed by removing all nodes with the *drone* label from $\mathcal{R}_1$. The subtour for the drone $\mathcal{D}$ can be computed by removing all nodes with label *truck* from $\mathcal{R}_1$. At each step of the heuristic, either one of the following operations is performed on a node with the *simple* label:

MakeFly Assigns the *drone* label to node r_i and assigns the *combined* label to its two adjacent nodes r_{i-1} en r_{i+1}. This operation can only be performed on a node has both a predecessor and a successor in $\mathcal{R}_1$ (i.e. it is not at the boundaries of $\mathcal{R}_1$). The savings of performing this operation on node r_i in $\mathcal{R}_1$ are $c(r_{i-1}, r_i) + c(r_i, r_{i+1}) - \max\left\{c^d(r_{i-1}, r_i) + c^d(r_i, r_{i+1}), c(r_{i-1}, r_{i+1})\right\}$

PushLeft Assigns the *combined* label to node r_i and assigns the *truck* label to node r_{i-1}. This operation can only be performed if node r_{i-1} has a *combined* label. The savings are a bit more tricky to compute: the costs for the truck within the operation are increased by $c(r_{i-1}, r_i)$ and the costs for the drone within the operation change by $c^d(r_f, r_{i-1}) - c^d(r_f, r_i))$, where r_f refers to the drone node that is currently visited by the drone prior to the combined node r_{i-1}. The maximum of both the new truck and the new drone costs need to be compared to the old maximum in order to compute the savings, but with proper administration these can be computed in constant time.

PushRight The same operation as *PushLeft*, but then performed with the node right of the current node.

With clever administration, it is possible to update the possible savings of each node in constant time. Combining this with a priority queue to find the best operation, it is easy to see that this heuristic can be run in $O(n \log n)$ time.

Constraints on the operations such as a limited distance that can be covered by the drone or a limited drone radius can be included by forbidding operations which are not feasible according to these constraints. In this case, the nodes on which none of the above operations can be performed, should be assigned the combined label.

6.2.2 An exact partitioning algorithm

We now describe an exact algorithm to partition $\mathcal{R}_1$ into a a truck tour $\mathcal{R}$ and a drone tour $\mathcal{D}$ based on dynamic programming. To simplify notation, we rename the nodes in $\mathcal{R}_1$ in consecutive order, i.e., $\mathcal{R}_1 = (r_0, r_1, r_2, \ldots, r_{n+1})$ with $r_0 = r_{n+1} = v_0$. Similar to the greedy partitioning heuristic described in Section 6.2.1, the exact partitioning algorithm does not change the order of the nodes in $\mathcal{R}_1$, but simply decides which of the nodes are to be visited by the truck and which by the drone. Therefore, the nodes in each operation in a solution found by the exact partitioning algorithm form a consecutive subsequence $(r_i, r_{i+1}, \ldots, r_j)$ of $\mathcal{R}_1$. Hence, an operation can be defined by a triplet (i, j, k), with start node r_i, end node r_j and drone node r_k with $i < k < j$, where we write $k = -1$ if the operation does not contain a drone node.

The basic algorithm can be summarized as follows:

1. We compute the time duration $T(i, j, k)$ of each operation (i, j, k) by

   $$T(i,j,k) = \max\{c^d(r_i, r_k) + c^d(r_k + r_j), \sum_{l=i}^{k-2} c(r_l, r_{l+1}) + \sum_{l=k+1}^{j-1} c(r_l, r_{l+1})\}.$$

   For an operation without drone node (which, by definition, has no truck nodes), we set

   $$T(i, i+1, -1) = c(r_i, r_{i+1}).$$

   If a certain operation is infeasible, e.g., due to a maximum on the flying distance of the drone, we can set its time duration to ∞.

2. For each consecutive subsequence $(r_i, r_{i+1}, \ldots, r_j)$ of $\mathcal{R}_1$ we compute the time duration of the best operation covering this subsequence:

$$T(i,j) := \min_{k=i+1}^{j-1} T(i,j,k). \tag{22}$$

To track the corresponding TSP-D tour, we define a matrix M and store the best operation found in (22) in $M(r_i, r_j)$.

3. Based on these pre-computations, for each node we compute the minimum time $V(i)$ to reach this node by truck (while serving all preceding nodes by truck or drone) by the following recursive formula: We set

$$V(0) = 0,$$
$$V(i) = \min_{k=0}^{i-1} [V(k) + T(k,i)].$$

To store the sequence of combined nodes, we store in $P(i)$ an element from $\operatorname{argmin}_{k=0}^{i-1}[V(k)+ T(k,i)]$.

We can now reconstruct the tour based on P and M.

In its present form, the above-described algorithm requires the computation of a time duration for all possible operations whose nodes form an uninterrupted subsequence of $\mathcal{R}_1$. There are $O(n^3)$ such operations. The naive computation of their length takes $O(n)$ each, which would mean a running time of $O(n^4)$ for the first step of the algorithm. However, by using a simple trick when computing $T(k,i,j)$ (described in the Appendix) it is possible to do the computations of the length of all operations in $O(n^3)$, and hence, to obtain a total running time of $O(n^3)$.

By construction of the described algorithm we obtain the following lemma.

Lemma 6.1. *For a given truck route $\mathcal{R}_1$, the solution $(\mathcal{R}^*, \mathcal{D}^*)$ constructed by the exact partitioning algorithm has minimal time duration among all solutions $(\mathcal{R}, \mathcal{D})$ to the TSP-D with $\mathcal{R}$ and $\mathcal{D}$ both subsequences of $\mathcal{R}_1$. This solution can be found in time $O(n^3)$.*

6.3 Iterative improvement procedures

Both the greedy and the exact partitioning algorithm rely on an initial tour $\mathcal{R}_1$. However, a good initial tour may not necessarily lead to a good tour for the truck and drone. Since our partitioning methods are fast given an initial TSP tour, we can start from a number of initial tours to see if we can find better solutions. We adopt several local search heuristics which modify $\mathcal{R}_1$ and run the different drone assignment methods to obtain the quality of the modifications. We consider various simple neighborhoods: one where we swap two nodes in $\mathcal{R}_1$, i.e. a 2-point move (2p),one where we remove two edges in $\mathcal{R}_1$ and replace them with two new edges resulting in a different tour, i.e. a 2-opt move, and one where we relocate a node in in $\mathcal{R}_1$ to a new position, i.e. a 1-point move (1p). Moreover, we also consider a neighborhood that combines all the different moves above (all). Since we are interested in the potential gains that we can achieve with such techniques, we try all possible moves and take the best one during each iteration. The pseudocode is given in Algorithm 1.

All the proposed neighbourhoods are of size $O(n^2)$. Since running the greedy partitioning heuristic takes $O(n \log n)$ time, a single iteration of a gp heuristic takes $O(n^3 \log n)$ time. The exact partitioning algorithm takes $O(n^3)$ time, and thus a single iteration of an ep heuristic takes $O(n^5)$ time.

7 Computational Study

In this section, we present the results of a computational study on various randomly generated TSP-D instances with different characteristics. The objective of this study is to (i) assess the performance of the different heuristics and (ii) investigate the benefits of combining a truck and a drone in different settings.Since we are primarily interested in assessing the performance of our methods, we chose to run our experiments on the most challenging instances in which there are no limitations on the range of the drone ($t^{\max} = \infty$) and all locations can be visited by both the drone and the truck ($V^t = \emptyset$). Practical restrictions can be easily added and decrease the running time of our algorithms. Heuristics that quickly find

```
**Data:** An initial tour $\mathcal{R}_1$, a partitioning algorithm $f$, and a neighborhood function $N$
**Result:** A locally optimal tour for a neighborhood function
$\mathcal{R}'_1 \leftarrow \mathcal{R}_1$ ;
$i \leftarrow$ True ;
while $i$ do
    $i \leftarrow$ False ;
    $M \leftarrow N(\mathcal{R}_1)$ ;
    for $m \in M$ do
        Modify $\mathcal{R}_1$ according to move $m$ ;
        if $f(\mathcal{R}_1) < f(\mathcal{R}'_1)$ then
            $\mathcal{R}'_1 \leftarrow \mathcal{R}_1$ ;
            $i \leftarrow$ True ;
        end
        Modify $\mathcal{R}_1$ according to the inverse of move $m$;
    end
    if $i$ then
        $\mathcal{R}_1 \leftarrow \mathcal{R}'_1$ ;
    end
end
return $\mathcal{R}_1$ ;
```

Algorithm 1: Iterative improvement procedure

good solutions in the most challenging cases will likely work even better in the restrictive setting.

Our experiments were executed within a virtual machine with 2GB of virtual RAM running a 64-bit version of xubuntu 15.04 on a VirtualBox 5.0.10 hypervisor with Windows 7 as the host OS. The hardware configuration of the host consisted of an Intel Core i7-4770 CPU with 16GB of RAM. The TSP solutions were obtained by calling the Concorde TSP Solver, one of the most advanced and fastest TSP solvers using branch-and-cut [Applegate et al., 2001]. CPLEX 12.6.1 was used to solve the IP formulation and as an LP solver for Concorde. Our other code was implemented in Java running on the OpenJDK `1.8.0_45` runtime environment. For obtaining an MST solution we applied the Delaunay Triangulation code in the JTS Topology Suite library, version 1.13.

7.1 Instance generation

For the sake of simplicity, we generate locations on a plane and assume that the travel time of the truck between any pair of locations is proportional to the Euclidean distance. The instances generated for our experiments are released as Bouman et al. [2015].

In the first type of instance, the *uniform instances* we draw the x and y coordinates for every location independently and uniformly from $\{0, 1, \ldots, 100\}$. For the second type of instance, the *1-center* instances, for each location we draw an angle a from $[0, 2\pi]$ uniformly and a distance r from a normal distribution with mean 0 and standard deviation 50. The x coordinate of the location is computed as $r \cos a$ and the y coordinate as $r \sin a$. This way, we have locations close to the center $(0, 0)$ with higher probability than in the uniform case and as a result these instances have a higher probability to mimic a circular city center than a uniform instance. Finally, we have the *2-center* instances, which are generated in the same way as the *1-center* instances, but every location is translated by 200 distance units over the x axis with probability $\frac{1}{2}$. These instance have a greater probability to mimic a city which has two centers at $(0, 0)$ and $(200, 0)$. In all cases, the first location generated is chosen as depot.

In the default setting we assume that the drone is twice as fast as the truck on all edges, i.e. $\alpha = \frac{c(e)}{c^d(e)} = 2$, but we also experiment with the situation where the drone and truck have equal speed ($\alpha = 1$) and the situation where the drone is three times as fast as the truck ($\alpha = 3$).

7.2 Comparison of the heuristics

To asses the quality of our proposed heuristics, we compare their solution values to the optimal solution values obtained by using the IP formulation as presented in Section 5 for several small instances. As discussed in Section 6, we use two methods to create an initial truck-only tour, i.e. the optimal TSP tour and the MST heuristic tour, and two methods to assign drone nodes, i.e. greedy partitioning (gp) and exact partitioning (ep). This gives rise to four basic versions of our heuristics: MST-gp, MST-ep, TSP-gp and TSP-ep. Applying the iterative improvement procedures on each of these approaches gives sixteen versions in total. As an additional benchmark, we also implemented the local search algorithm that was introduced by Murray and Chu [2015], where the search space consists of solutions which specify both the truck and drone tours. We use the optimal TSP tour as a starting tour (TSP-MC). Moreover, we report the solution when using our exact partitioning (ep) approach starting from the sequence of the TSP-MC tour and call this TSP-MC-ep. Note that the TSP-MC neighborhood is defined by truck-and-drone tours which results in an $\Omega(n^3)$ running time of a single iteration. Assuming that we need $O(n)$ time to evaluate the costs of a truck-and-drone tour, a single iteration of the TSP-MC heuristic takes $O(n^4)$ time.

We compare solution quality based on three statistics: the average percentage deviation from the optimal solution, the maximum percentage deviation from the optimal solution and the number of instances for which the optimal solution is obtained.

7.2.1 Base comparison

In the base case, with $\alpha = 2$, Theorem 4.2 and Theorem 4.1 guarantee that we find at least a 4-approximation with the MST-based heuristics, and at least a 3-approximation with the

TSP-based ones. However, our experiments show that the solutions found heuristically are usually much better than these theoretical worst-case bounds. Table 2 provides the results for 10 randomly generated instances with 10 nodes of each instance type. The solution times for the IP for all 10 node instances are between 5.7 seconds and 38.7 seconds.(Note that for larger instances the enumeration of all operations quickly becomes a lot more time and memory consuming, i.e. instances with 12 nodes take more than two hours to solve and required around 2 GB of RAM.)

The table shows that there are big variations in solution quality between the twenty heuristic variations. The TSP-ep-all heuristic performs best, with a maximum optimality gap of 4.6% and achieving optimality in 16 of the 30 instances. On the other end of the spectrum, the MST-gp performs very poor, with a maximum gap of 47.1% and achieving optimality in none of the 30 instances. Also the TSP-MC heuristic does not perform well with an average gap between 16.4% and 22.0%. A possible reason for this is that the heuristic may commit to long drone flights with many truck nodes early on, thereby blocking additional drone flights later. Interestingly, we can achieve significant improvements when running the exact partitioning approach using the TSP-MC sequence as the initial tour (TSP-MC-ep). Since the worst case running time of a single exact partition run is $O(n^3)$ and a single iteration of the TSP-MC heuristic takes at least $\Omega(n^3)$ time, we can obtain solutions that are at least as good and often better without an increase in worst-case running time by combining the approaches.

For the twenty route first-cluster second heuristics, we can explain the variations in solution quality by looking at three aspects: the initial tour, the partitioning method and the improvement procedure. We start by comparing the heuristics without an iterative improvement procedure. First, we observe that starting with the optimal TSP tour generally provides better results than starting with the longer MST tour, i.e. a difference of up to 11.5 p.p. in average solution value. Second, we see that the exact partitioning approach consistently outperforms the greedy heuristic. This is as expected. The difference in the optimality gap between the solutions of the two drone assignment methods ranges between 0.1 p.p. and 7.4 p.p. on average. Finally, we see that the improvement procedures are very

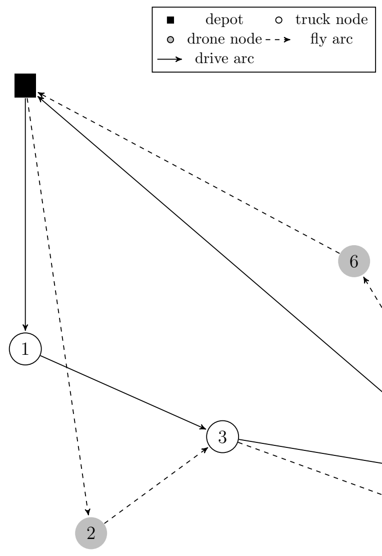
	Uniform			*1-center*			*2-center*		
	Δ (%)			Δ (%)			Δ (%)		
	avg	max	# opt	avg	max	# opt	avg	max	# opt
MST-gp	19.6	35.6	0/10	27.4	40.9	0/10	25.6	47.1	0/10
MST-ep	18.1	35.6	0/10	23.2	40.9	0/10	23.2	45.8	0/10
TSP-gp	16.1	30.1	0/10	22.6	32.9	0/10	18.5	24.9	0/10
TSP-ep	16.0	23.3	0/10	15.2	32.9	0/10	12.7	19.7	0/10
MST-gp-1p	5.2	10.5	1/10	8.5	17.7	0/10	7.9	11.3	0/10
MST-ep-1p	2.2	9.3	1/10	2.1	11.8	3/10	3.5	10.2	2/10
TSP-gp-1p	2.8	6.2	0/10	6.8	16.5	0/10	6.1	18.4	1/10
TSP-ep-1p	1.8	6.2	2/10	1.7	9.7	3/10	4.2	17.4	3/10
MST-gp-2p	4.7	11.5	1/10	8.1	14.1	1/10	8.0	12.0	0/10
MST-ep-2p	2.6	9.3	2/10	1.6	6.0	3/10	3.1	7.2	1/10
TSP-gp-2p	2.7	6.0	0/10	8.2	17.1	0/10	8.0	17.4	1/10
TSP-ep-2p	1.3	4.1	2/10	2.5	6.0	2/10	2.3	8.3	4/10
MST-gp-2opt	3.2	11.0	2/10	7.5	15.3	0/10	5.8	10.4	0/10
MST-ep-2opt	2.4	10.1	3/10	2.1	9.7	4/10	2.7	10.2	4/10
TSP-gp-2opt	2.9	6.2	2/10	7.8	16.9	0/10	6.2	18.4	0/10
TSP-ep-2opt	1.5	6.2	4/10	3.0	9.7	4/10	4.3	17.4	2/10
MST-gp-all	2.0	6.4	1/10	6.3	13.8	0/10	5.7	10.4	0/10
MST-ep-all	0.7	2.7	4/10	0.4	2.0	6/10	2.8	7.8	2/10
TSP-gp-all	1.6	3.1	1/10	5.2	13.7	0/10	5.8	17.4	1/10
TSP-ep-all	0.4	2.3	6/10	1.1	4.6	5/10	1.3	4.2	5/10
TSP-MC	20.1	36.0	0/10	22.0	38.7	0/10	16.4	28.6	0/10
TSP-MC-ep	13.2	34.9	0/10	14.9	31.4	0/10	11.0	23.0	1/10

Table 2: Comparison of heuristic solutions to the optimal IP solutions, uniform, $N = 10$, $\alpha = 2$, averaged over 10 instances

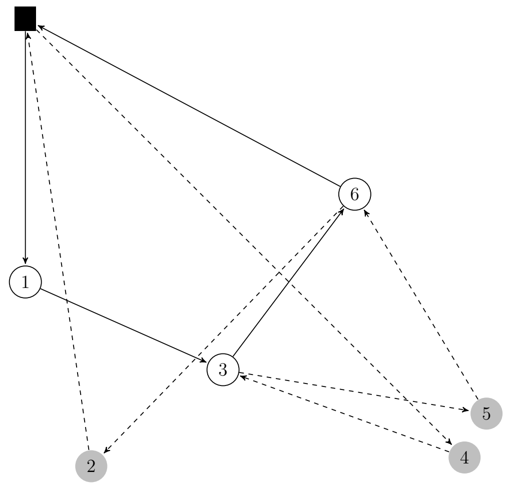
beneficial. In particular, we observe a decrease of the gap with the optimal solution by up to 15% over the heuristics without improvement procedure. This suggests that to fully exploit the benefits of the drone, we should adapt the truck route to facilitate good drone flights. This is illustrated in Figure 5 that shows the solution of the TSP-ep heuristic and an optimal solution for a specific uniform instance with six nodes. The numbers in the nodes indicate the sequence in which they are visited in the optimal TSP tour. We observe that visiting both node 4 and 5 by the drone leads to time savings of 17% but requires a different route sequence than in an optimal TSP tour. While the TSP-ep solution is associated with very long waiting times for the drone, we see that changing the route sequence allows us to make better use of the drone in the optimal solution.

To provide some more insight into the solutions obtained by the different methods, we compare them based on the following characteristics: the number of drone nodes, the number of truck nodes, the travel distance of the truck and the drone and the waiting time of both the drone and the truck. The waiting times represent the time that the drone (truck) has to wait before meeting up with the truck (drone) again. We normalize the distances by dividing them by the total length of the optimal truck-only TSP tour. The waiting time is expressed as a percentage of the total time, i.e. the objective value. We also provide the total time, our objective value, which we state as a percentage of the total time of an optimal TSP tour.

Table 3 provides the averages based on 10 instances with uniformly distributed locations for four variants of our heuristic that start from the TSP tour. (Since the results for the individual iterative improvement procedures are similar we report only the results for one of them, i.e. the swap move.) We observe that the number of drone flights is very similar in the different solutions. There is, however, more variation in the number of truck nodes in the different solutions. The number of truck nodes is higher for the solutions with exact partitioning than for the solutions of the greedy heuristic. Looking at the TSP-based heuristics only, better solutions seem to be associated with more work in parallel and thus more simultaneous deliveries. Focusing on the travel distances in the optimal solution, we see that the distance traveled by the truck is about 30% less than in the truck-only TSP. At the same time, we see that the drone travels approximately twice the distance of the truck. This is



(a) TSP-ep solution, objective value = 117, waiting: truck = 0%; drone = 42%



(b) Optimal MIP solution, objective value = 100, waiting: truck = 6%; drone = 2.5%

Figure 5: Example TSP-D solutions, uniform, $N = 6$, $\alpha = 2$

	Time	*# nodes*		*Distance*		*Waiting*	
	total	truck	drone	truck	drone	truck	drone
IP	100	2.3	4.0	69.3	124.8	2.3	5.1
TSP-gp	116.1	0.5	4.1	82.8	87.8	0.2	27.1
TSP-ep	116.0	1.7	4.1	82.8	98.0	0.1	33.5
TSP-gp-2p	102.7	1.4	4.1	71.9	118.3	1.6	12.9
TSP-ep-2p	101.3	2.2	3.8	70.2	120.7	2.4	10.0
TSP-gp-all	101.6	1.2	4.2	70.3	120.0	2.5	11.1
TSP-ep-all	100.4	2.2	3.9	69.5	125.8	2.4	9.0

Table 3: Characteristics of TSP-D solutions, uniform, $N = 10$, $\alpha = 2$, averaged over 10 instances

intuitive since we assume a drone that is twice as fast as the truck. That is, to maximize the utilization of both the truck and the drone (and minimize waiting times), the drone should cover twice the distance of the truck. In the solutions of poor quality, the drone covers less distance and has longer waiting times than in good solutions.

7.2.2 Comparison of the heuristics for different drone speeds

Up to now, we have considered a drone that is twice as fast as the truck. In the following experiments, we vary the relative drone speeds α between one and three. A relative speed of one means that the drone travels at the same speed as the truck. Figure 6 shows the quality of the TSP-ep, TSP-ep-swap and the TSP-ep-all heuristic for these different drone speeds on the instance with uniformly distributed customer locations. The results show that all the heuristics perform better for smaller α-values than for larger ones.

We also see that the advantage of trying different initial tours as basis for the solution which we gain when using TSP-ep-swap or TSP-ep-all instead of TSP-ep is more prominent at higher relative drone speeds. The reason for this is that when the speed of the drone increases, it is beneficial to cover more distance with the drone, making the truck tour less

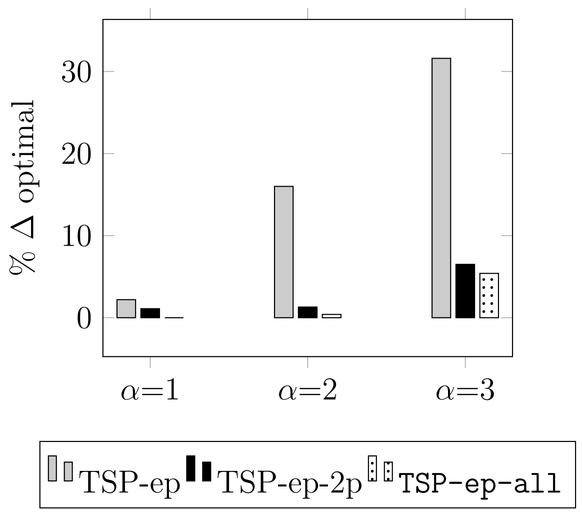


Figure 6: Comparison of performance for different speeds, uniform, $N = 10$, averaged over 10 instances

relevant. This is illustrated in Figure 7, which plots the sum of the distance of the drone and the truck, where a value of 100 represents the distance of the optimal truck-only tour. The figure also shows that the total combined distance increases with a faster drone. This relates to the fact that the drone has to go back and forth to the truck for each delivery. Note that in practice we do not expect the drone to reach speeds that are more than twice the speed of the truck. This means that for realistic drone speeds our heuristics provide very good solutions.

7.3 Benefits of combining a truck and a drone

Figure 8 presents box plots of the ratio of optimal objective values $\frac{Z(TSP)}{Z(TSP-D)}$ over 10 randomly generated instances of each type. We observe that the combination of a truck and a drone leads to substantial time savings, on average between a factor 1.4 and 2. The results suggest that adding a drone is especially beneficial if demand is clustered around one or two centers. This is intuitive since we may save a lot of time by serving points outside the center with the drone.

To study the potential gains in larger problem instances, we use the route first-cluster second heuristics to solve the TSP-D as we cannot solve these to optimality with our IP

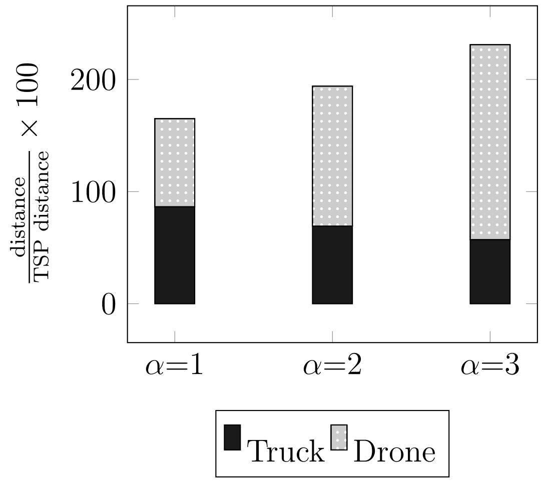


Figure 7: The travel distances of the truck and the drone, respectively, in the optimal solutions, uniform, $N = 10$, averaged over 10 instances

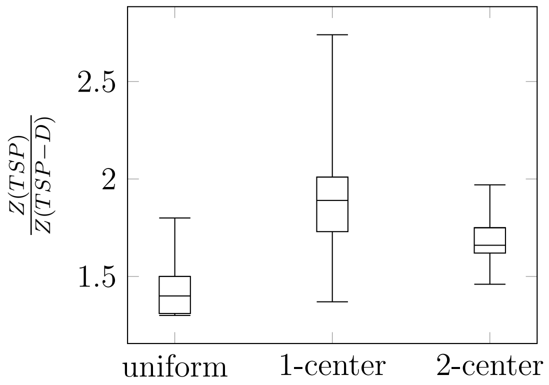


Figure 8: TSP versus TSP-D, uniform, $N = 10$, $\alpha = 2$, averaged over 10 instances

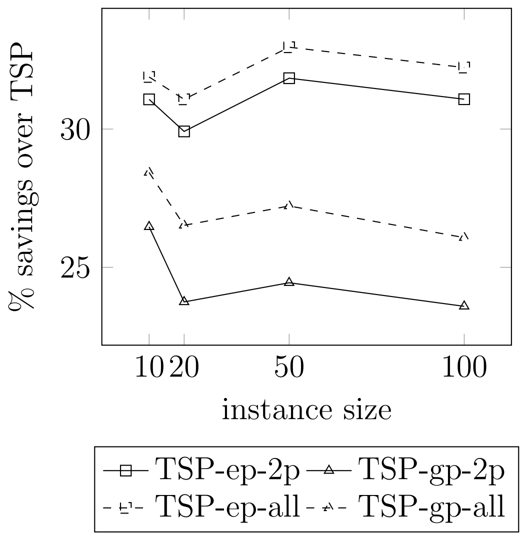


Figure 9: Comparison of heuristics, uniform, $\alpha = 2$, averaged over 10 instances

models.

The heuristics have shown to provide good solutions for smaller instances so we believe it is appropriate to use them to investigate the value of drones in larger instances.

In the case of uniformly distributed delivery locations, we expect the relative savings of the optimal truck-drone combination over the optimal truck-only solution to be similar for different instance sizes. The results in Figure 9 support this observation. The figure shows the gain over the truck-only TSP tour for the four versions of the heuristic with improvement procedure. We observe that using the exact partitioning consistently outperforms the greedy heuristic. The heuristics that use the exact partitioning (TSP-ep) show consistent savings of about 30% and the greedy heuristic (TSP-gp) of approximately 20%. It is encouraging to see that the average performance of the heuristics does not deteriorate with the size of the instance.

While TSP-ep-all consistently outperforms the other heuristics, it has a longer solution run time. Figure 10 shows the run times of the different heuristics (see the Appendix for the same figure using a log scale) . We see that the greedy approaches (TSP-gp-swap and TSP-gp-all) are extremely fast and solve all the instances with 100 nodes within a few seconds. The run time of heuristics that use the exact partitioning approach, on the other hand,

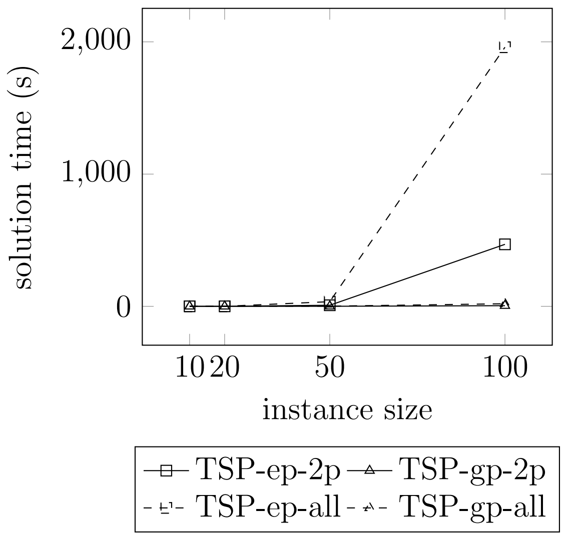


Figure 10: Comparison of solution times, uniform, $\alpha = 2$, averaged over 10 instances

increase rapidly with the number of nodes.

8 Conclusions

In this paper, we study the combination of a truck and a drone in a last-mile delivery setting. We theoretically and empirically show that substantial savings are possible in such a combined system as compared to the truck only solution. Moreover, we present several fast heuristics that provide solutions that are close to optimal.

As this is one of the first papers to address the collaborative use of a truck and a drone, we see many potential areas for future research. One challenging area of future research is to develop exact solution approaches that can provide optimal solutions for medium sized problems with reasonable solution times. A natural extension is to consider a setting with multiple trucks and multiple drones. Another extension would be to consider the possibility to recharge the drone on the truck during operations. If the drone fully recharges after each delivery and charging can only take place while the truck is parked at certain locations then it can easily be incorporated in our operations. However, it is less clear how to model a more flexible recharging policy as this requires monitoring the battery life over multiple operations.

References

David L. Applegate, Robert E. Bixby, Vašek Chvátal, and William J. Cook. TSP cuts which do not conform to the template paradigm. In *Computational Combinatorial Optimization*, pages 261–303. Springer, 2001.

David L. Applegate, Robert E. Bixby, Vašek Chvátal, and William J. Cook. *The traveling salesman problem: a computational study*. Princeton University Press, 2011.

John E Beasley. Route first - cluster second methods for vehicle routing. *Omega*, 11(4): 403–408, 1983.

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

Paul Bouman, Niels Agatz, and Marie Schmidt. Instances for the tsp with drone, July 2015. URL `http://dx.doi.org/10.5281/zenodo.22245`.

Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, DTIC Document, 1976.

John R. Current and David A. Schilling. The covering salesman problem. *Transportation science*, 23(3):208–213, 1989.

Michael Drexl. Branch-and-price and heuristic column generation for the generalized truck-and-trailer routing problem. *Revista de Métodos Cuantitativos para la Economía y la Empresa*, 12:5–38, 2011.

Michael Drexl. Synchronization in vehicle routing-a survey of VRPs with multiple synchronization constraints. *Transportation Science*, 46(3):297–316, 2012.

Michael Drexl. Branch-and-cut algorithms for the vehicle routing problem with trailers and transshipments. *Networks*, 63(1):119–133, 2014.

Lanah Evers, Twan Dollevoet, Ana Isabel Barros, and Herman Monsuur. Robust uav mission planning. *Annals of Operations Research*, 222(1):293–315, 2014.

Michael R Garey and David S Johnson. *Computers and intractability: a guide to NP-completeness*. WH Freeman New York, 1979.

Bruce Golden, Zahra Naji-Azimi, S Raghavan, Majid Salari, and Paolo Toth. The generalized covering salesman problem. *INFORMS Journal on Computing*, 24(4):534–553, 2012.

Damon J Gulczynski, Jeffrey W Heath, and Carter C Price. The close enough traveling salesman problem: A discussion of several heuristics. In *Perspectives in Operations Research*, pages 271–283. Springer, 2006.

A. Hern. DHL launches first commercial drone 'Parcelcopter' delivery service, 2014.

Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger, editors, *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Springer US, 1972. ISBN 978-1-4684-2003-6.

Min-Hyuk Kim, Hyeoncheol Baik, and Seokcheon Lee. Response threshold model based uav search planning and task allocation. *Journal of Intelligent & Robotic Systems*, 75(3-4): 625–640, 2014.

Joseph B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.

Gilbert Laporte. The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2):231–247, 1992.

Shih-Wei Lin, F Yu Vincent, and Shuo-Yan Chou. Solving the truck and trailer routing problem based on a simulated annealing heuristic. *Computers & Operations Research*, 36 (5):1683–1692, 2009.

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

J Nicas and G. Bensinger. Delivery drones hit bumps on path to doorstep, 2015.

B. Popper. UPS researching delivery drones that could compete with Amazon's Prime Air, 2013.

Daniel J Rosenkrantz, Richard E Stearns, and Philip M Lewis, II. An analysis of several heuristics for the traveling salesman problem. *SIAM journal on computing*, 6(3):563–581, 1977.

Stephan Scheuerer. A tabu search heuristic for the truck and trailer routing problem. *Computers & Operations Research*, 33(4):894–909, 2006.

Michael Ian Shamos. *Computational geometry.* PhD thesis, Yale University, 1978.

Robert Shuttleworth, Bruce L Golden, Susan Smith, and Edward Wasil. Advances in meter reading: Heuristic solution of the close enough traveling salesman problem over a street network. In *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 487–501. Springer, 2008.

Kimon P Valavanis and George J Vachtsevanos. Uav applications: Introduction. In *Handbook of Unmanned Aerial Vehicles*, pages 2639–2641. Springer, 2015.

Juan G Villegas, Christian Prins, Caroline Prodhon, Andrés L Medaglia, and Nubia Velasco. A grasp with evolutionary path relinking for the truck and trailer routing problem. *Computers & Operations Research*, 38(9):1319–1334, 2011.

Xingyin Wang, Stefan Poikonen, and Bruce Golden. The vehicle routing problem with drones: several worst-case results. *Optimization Letters*, pages 1–19, 2016.

M. Wohlsen. The next big thing you missed: Amazon's delivery drones could work. they just need trucks, 2014.

Appendix

8.1 Proofs of theoretical results for the general case

In Section 4 we proved the results for the case that the drone is as least as fast as the truck between all pairs of nodes. In this section we will show how these results can be transferred to the general case where the drone is not necessarily faster than the truck but can also be slower between some or all pairs of nodes.

Consider a TSP-D tour $(\mathcal{R}, \mathcal{D})$. When the drone is slower than the truck on some edges, inequality (2) does not necessarily hold any more. We denote by K' the index set of the operations in which the drone leaves the truck and observe that

$$t(\mathcal{R}, \mathcal{D}) \geq \max\{c(\mathcal{R}), \sum_{k \in K'} c^d(\mathcal{D}_k)\}. \tag{23}$$

Similarly to Lemma 4.1 we can show the stronger inequality from Lemma 8.1

Lemma 8.1. *Let $(\mathcal{R}, \mathcal{D})$ be a TSP-D tour and denote by K' the index set of the operations in which the drone leaves the truck. Then*

$$\sum_{k \in K'} c^d(\mathcal{D}_k) \geq 2 \cdot \sum_{v \in V_\mathcal{D} \setminus V_\mathcal{R}} \min_{w \in V_\mathcal{R}} c(v, w). \tag{24}$$

Proof. Again, because in every operation there is at most one drone node, we know that the drone visits a combined node in $V_\mathcal{R} \cap V_\mathcal{D}$ right before and right after visiting a drone node in $V_\mathcal{D} \setminus V_\mathcal{R}$. Hence we have

$$\sum_{k \in K'} c^d(\mathcal{D}_k) = \sum_{e \in E_\mathcal{D} \setminus E_\mathcal{R}} c(e) \geq 2 \cdot \sum_{v \in V_\mathcal{D} \setminus V_\mathcal{R}} \min_{w \in V_\mathcal{R}} c(v, w). \tag{25}$$

□

Proof. Proof of Theorem 4.1 Let $(\mathcal{R}^*, \mathcal{D}^*)$ be an optimal solution to the TSP-D and $\mathcal{R}^{\text{TSP}}$ be an optimal solution to the TSP.

From $(\mathcal{R}^*, \mathcal{D}^*)$ construct a TSP tour $\mathcal{R}$ like in the proof for the special case in Section 4: Start with $\mathcal{R}^*$. For every drone node $v \in V_{\mathcal{D}^*}$, we pick the node $w \in V_{\mathcal{R}^*}$ which is closest to v with respect to the driving time c and insert arcs (w, v) and (v, w) into the tour and obtain

$$c(\mathcal{R}^{\text{TSP}}) \leq c(\mathcal{R}) = c(\mathcal{R}^*) + 2 \cdot \sum_{v \in V_{\mathcal{D}^*} \setminus V_{\mathcal{R}^*}} \min_{w \in V_{\mathcal{R}^*}} c(v, w).$$

We proceed analogously to the proof for the special case in Section 4: Using (24) we conclude

$$c(\mathcal{R}^{\text{TSP}}) \leq c(\mathcal{R}^*) + c(\mathcal{D}_d^*). \tag{26}$$

On the other hand, from (23) we obtain

$$t(\mathcal{R}^*, \mathcal{D}^*) \geq \max\{c(\mathcal{R}^*), \sum_{k \in K'} c^d(\mathcal{D}_k)\} \geq \max\{c(\mathcal{R}^*), \frac{1}{\alpha} \sum_{k \in K'} c^d(\mathcal{D}_k)\}, \tag{27}$$

where the last inequality follows from the definition of α. Finally combining (26) and (27)

$$t(\mathcal{R}^*, \mathcal{D}_d^*) \geq \max\{c(\mathcal{R}^{\text{TSP}}) - \sum_{k \in K'} c^d(\mathcal{D}_k), \frac{1}{\alpha} \sum_{k \in K'} c^d(\mathcal{D}_k)\} \geq \frac{c(\mathcal{R}^{\text{TSP}})}{1 + \alpha}.$$

We conclude that

$$t(\mathcal{R}^{\text{TSP}}, \mathcal{R}^{\text{TSP}}) = c(\mathcal{R}^{\text{TSP}}) \leq (1 + \alpha) t(\mathcal{R}^*, \mathcal{D}^*).$$

□

Proof. Proof of Lemma 4.2 for the general case We proceed analogously to the proof in Section 4: Based on the node sets $V_{\mathcal{R}^*}$ of $\mathcal{R}^*$ and $V_{\mathcal{D}^*}$ of $\mathcal{D}^*$ we construct a spanning tree T' of G as in the proof for the special case in Section 4: first, we remove one edge from $\mathcal{R}^*$ to obtain a spanning tree $T_{\mathcal{R}^*}$ of the node set $V_{\mathcal{R}^*}$. Then we connect all nodes v from the node set $V_{\mathcal{D}^*} \setminus V_{\mathcal{R}^*}$ to the closest node in $V_{\mathcal{R}^*}$. Hence

$$c(T') = c(T_{\mathcal{R}^*}) + \sum_{v \in V_{\mathcal{D}^*} \setminus V_{\mathcal{R}^*}} \min_{w \in V_{\mathcal{R}^*}} c(v, w). \tag{28}$$

Using (24), $c(T_{\mathcal{R}^*}) \le c(\mathcal{R}^*)$, and (28) we obtain

$$\begin{aligned}
t(\mathcal{R}^*, \mathcal{D}^*) &\ge \max\{c(\mathcal{R}^*), \sum_{k \in K'} c^d(\mathcal{D}_k)\} \qquad (29)\\
&= \max\{c(\mathcal{R}^*), \sum_{e \in E_{\mathcal{D}^*} \setminus E_{\mathcal{R}^*}} c^d(e)\} \qquad (30)\\
&\ge \max\{c(\mathcal{R}^*), \frac{1}{\alpha} \sum_{e \in E_{\mathcal{D}^*} \setminus E_{\mathcal{R}^*}} c(e)\} \qquad (31)\\
&\ge \max\{c(T_{\mathcal{R}^*}), \frac{2}{\alpha} \sum_{v \in V_{\mathcal{D}^*} \setminus V_{\mathcal{R}^*}} \min_{w \in V_{\mathcal{R}^*}} c(v,w)\} \qquad (32)\\
&= \max\{c(T_{\mathcal{R}^*}), \frac{2}{\alpha}(c(T') - c(T_{\mathcal{R}^*}))\} \qquad (33)\\
&\ge \frac{2}{2+\alpha} c(T') \qquad (34)\\
&\ge \frac{2}{2+\alpha} c(T) \qquad (35)
\end{aligned}$$

for the minimum spanning tree T of G. Hereby, inequality (34) holds because $\max_{x \in \mathbb{R}^+}\{x, \frac{2}{\alpha}(c(T') - x)\}$ is minimal if $x = \frac{2}{\alpha}(c(T') - x)$. $\square$

Then Theorem 4.2 follows as detailed in the proof in Section 4.

It is easy to check that all results from Section 4 are still valid if non-zero recharging times and pickup and delivery times *of the drones* are considered, since these would not affect the time duration of the tour $(\mathcal{R}^{\text{TSP}}, \mathcal{R}^{\text{TSP}})$ but would slow down the duration of each TSP-D tour which uses the drone.

Speed-up for the exact partitioning algorithm from Section 6.2.2

To compute $T(i,j,k)$ in time $O(n^3)$ we proceed as follows. For each k compute

- $\texttt{SUM}(k-1, k+1, k) := c(r_{k-1}, r_{k+1})$ and $T_k(k-1, k+1) = \max\{[c^d(r_{k-1}, r_k) + c^d(r_k, r_{k+1})], \texttt{SUM}(k-1, k$
- for each $i < k-1$: $\texttt{SUM}(i, k+1, k) = c(r_i, r_{i+1}) + \texttt{SUM}(i+1, k+1, k)$ and $T(i, k+1, k) = \max\{[c^d(r_i, r_k) + c^d(r_k, r_{k+1})], \texttt{SUM}(i, k+1, k)\}$,
- for each $i < k-1$, $j > k+1$: $\texttt{SUM}(i,j,k) = c(r_{j-1}, r_j) + \texttt{SUM}(i, j-1, k)$ and $T(i,j,k) = \max\{\alpha[c^d(r_i, r_k) + c^d(r_k, r_j)], \texttt{SUM}(i,j,k)\}$,

Hence we find $T(i, j, k)$ in computation time $O(n^3)$.

Furthermore, we can speed up our algorithm by not computing all $T(i, j, k)$, but only the ones for operations which can be part of an optimal solution: It is easy to see (from the triangle inequality) that if

$$\texttt{SUM}(i, j, k) \geq [c^d(r_i, r_k) + c^d(r_k, r_j)] \tag{36}$$

it follows that

$$\begin{aligned} T(i, j+1, k) =& \max\{[c^d(r_i, r_k) + c^d(r_k + r_j)], \texttt{SUM}(i, j, k) + c(r_i, r_j)] \\ =& \max\{[c^d(r_i, r_k) + c^d(r_k + r_j)], \texttt{SUM}(i, j, k)] + c(r_i, r_j) \\ =& T(k, i, j) + T(j, j+1, -1) \end{aligned}$$

and by induction the same holds for all $j' > j$. With a symmetric argument, we obtain the same results for all $i' < i$. Hence, as soon as for a fixed k and some i, j we have reached (36) we do not have to compute 'larger' operations and can move on to the next k.

Log scale figure from Section 7.3

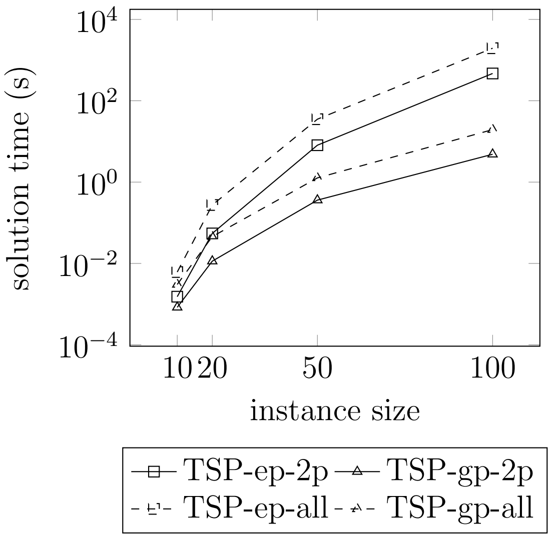


Figure 11: Comparison of solution times, uniform, $\alpha = 2$, averaged over 10 instances