Transportation Science

Optimization Approaches for the Traveling Salesman Problem with Drone

Niels Agatz, Paul Bouman, Marie Schmidt

Optimization Approaches for the Traveling Salesman Problem with Drone

Niels Agatz,[a] Paul Bouman,[b] Marie Schmidt[a]

[a] Rotterdam School of Management, Erasmus University, 3062 PA Rotterdam, Netherlands; [b] Econometric Institute, Erasmus University, 3062 PA Rotterdam, Netherlands
Contact: nagatz@rsm.nl, http://orcid.org/0000-0003-3514-201X (NA); bouman@ese.eur.nl, http://orcid.org/0000-0003-4893-4083 (PB); schmidt2@rsm.nl, http://orcid.org/0000-0001-9563-9955 (MS)

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 drones 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 integer program 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 compared to truck-only delivery.

Supplemental Material: The online appendix is available at https://doi.org/10.1287/trsc.2017.0791.

1. Introduction

In an effort to provide faster and more cost-efficient delivery of goods ordered online, companies are looking for new technologies to bridge the last mile to their customers. One 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 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 trials to investigate the use of drones for parcel delivery (Popper 2013). These trials typically involve multipropeller drones that can carry parcels of approximately two kilograms over a range of 20 kilometers. There are examples of drones that are already used for deliveries in practice, albeit solely in nonurban environments. 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 is 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 decade, 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. Furthermore, a drone cannot carry more than one package; thus, it has to return to the depot after each delivery. Since it is battery powered, the range is likely to remain limited 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 top of its electric-powered trucks to help the truck driver make deliveries (Wohlsen 2014). The 2016 Mercedes-Benz concept vehicle called the Vision Van also includes roof-top drones. 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, returning to the truck after each delivery to pick up another parcel.

Table 1. Complementary Features of Using a Drone or a Truck for Delivery

	Speed	Weight	Capacity	Range
Drone	High	Light	One	Short
Truck	Low	Heavy	Many	Long

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 determine which vehicle, drone or truck will serve which customers, and routing decisions determine the sequence in which 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.

Since this is a new phenomenon, there is a need for new models and innovative algorithms to help better understand the related planning problems and potential benefits. This paper aims to fill this gap by developing a new integer programming (IP) 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 delivery with truck and drone over truck-only delivery for various problem instances.

The main contributions of this paper are as follows: (1) We develop a new IP model that is able to solve

instances of up to 12 nodes to optimality, while earlier research did not report optimal solutions. (2) We provide 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) We conduct 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) We conduct 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 in Section 6, we propose several heuristics. 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. 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, for example, Gulczynski, Heath, and Price (2006); Shuttleworth et al. (2008); Golden et al. (2012); and 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 the 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

Figure 1. An Example of a TSP-D Solution

- ■ Depot
- ○ Truck node
- ● Drone node
- - -► Fly arc
- —► Drive arc
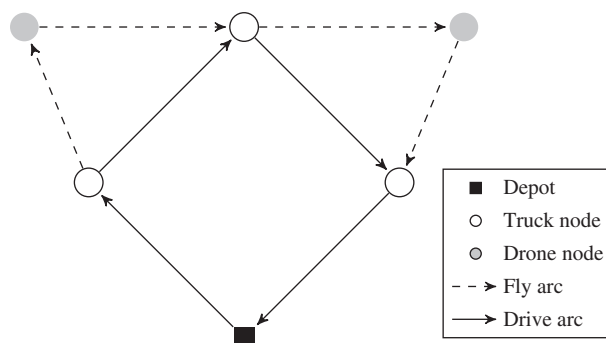
vehicle composed of a truck with a detachable trailer serves the demand of a set of customers reachable by truck and trailer or only by the truck without the trailer. Several heuristics based on the cluster-first, route-second principle have been proposed to solve the TTRP, including tabu search (Scheuerer 2006) and simulated annealing (Lin, Vincent, and Chou 2009). We are only aware of one paper that uses a route-first, cluster-second procedure, that 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.

The single truck and trailer routing problem with satellite depots is a special case of the truck and trailer routing problem in which all customers can be visited only by the truck without the trailer. Villegas et al. (2011) introduce a heuristic to solve this problem. More recently, Belenguer et al. (2016) provided an exact solution procedure based on a branch-and-cut algorithm that enabled them to solve problems with up to 50 customer locations and 10 satellite depots.

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 in Section 7.

Since Murray and Chu (2015) did not obtain optimal solutions using their model formulation, even for instances with only six nodes, we present a different formulation to model the TSP-D as an integer program.

Even though our formulation has an exponential number of variables in the number of customers, it is more similar to the strong formulation for the regular TSP used by Applegate et al. (2011). As a result, we were able to solve instances with up to 12 customers to optimality. Furthermore, we believe our formulation to be a good starting point for exact approaches where variables are generated on the fly, for example, branch-and-cut-and-price approaches.

Wang, Poikonen, and Golden (2017) study a vehicle routing problem with drones in which a fleet of trucks, each equipped with a number of drones, delivers packages to customers. Their main assumptions on drone capacity, pickup, and delivery are identical to our assumptions. In contrast to our paper, they do not develop approaches to find truck-and-drone tours, but instead focus on deriving worst case bounds for ratios of the total delivery time of different delivery options.

Their results imply that for our setting, the truck-only TSP tour takes at most $1 + \alpha$ as long as the combined truck-and-drone tour, where the speed of the drone is α times the speed of the truck. In Theorem 1 we extend this result to the setting where drones follow a different distance metric than the truck.

Apart from the application of the drone as a delivery vehicle, there are several works on the operational aspects of using drones for military and civil surveillance tasks; see, for example, Evers et al. (2014); Kim, Baik, and Lee (2014); and 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 times between v_i and v_j of the truck and the drone, respectively.

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 shortcuts. Moreover, even if the drone is restricted to following the same road network as the truck because of safety and privacy regulations, 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 that 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 take place only at the customer locations; that is, the drone can land on and depart from the truck only while it is parked at a customer location or the depot.

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

In the case where battery life of the drone is limited, we can specify a certain maximum flying distance, which we will refer to as drone range $d^{\max}$ (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 $\mathscr{D} = (d_0 = v_0, d_1, \ldots, d_m = v_0)$, where all nodes $v \in V^t$ need to be contained in $\mathscr{R}$. 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 $(\mathscr{R}, \mathscr{D})$, 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 the *start node* and *end node*; at most one drone node; and a nonnegative 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 remain stationary 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 the start node and the end node. The truck rides from the start node to the end node while the drone stays parked on the truck. We call such an operation *basic*.

Hence, an operation can be described as a pair of subtours $(\mathscr{R}_k, \mathscr{D}_k)$, where $\mathscr{R}_k$ is the subsequence of $\mathscr{R}$ starting at the kth combined node of $\mathscr{R}$ and up to and including the $(k+1)$th combined node of $\mathscr{R}$. Similarly, $\mathscr{D}_k$ is the subsequence of $\mathscr{D}$ from the kth combined node of $\mathscr{D}$ up to and including the $(k+1)$th combined node of $\mathscr{D}$. We denote by $c(\mathscr{R}_k) := \sum_{e \in \mathscr{R}_k} c(e)$ and $c^d(\mathscr{D}_k) := \sum_{e \in \mathscr{D}_k} c^d(e)$ the driving time/flying time on the subtours $\mathscr{R}_k$ and $\mathscr{D}_k$, 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 $(\mathscr{R}, \mathscr{D})$ to the TSP-D, it is convenient to regard $(\mathscr{R}, \mathscr{D})$ as a sequence of operations $(o_1, o_2, \ldots, o_l)$ 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

Figure 2. An Operation with a Combined Node Serving as the Start Node (1), a Combined Node Serving as the End Node (4), a Drone Node (3), and a Truck Node (2)
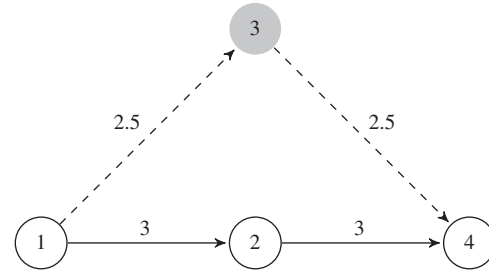
$t(o_k) = \max\{c(\mathscr{R}_k), c^d(\mathscr{D}_k)\}$. In the case that the drone can be slower than the truck, we have

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

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

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

Figure 2 provides an illustrative example of an operation, where the dashed arcs correspond to the path of the drone and the solid arcs correspond to the path of the truck. The numbers above the arcs represent the travel times 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 the 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 Sections 5 and 6.

To specify bounds and approximation results, we express the maximum speedup that can be gained by using the drone instead of the truck between two nodes v_i and v_j by α, that is, $\alpha := \max_{\{v_i, v_j\} \in E}\{c(v_i, v_j)/c^d(v_i, v_j)\}$. In this section, we provide the proofs for the case where the drone is at least as fast as the truck, that is, $c^d(v_i, v_j) \le c(v_i, v_j)$ 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 online appendix for details.

In Section 4.1, we start by analyzing how much we can gain by delivering goods with the truck and drone instead of only the truck. In Section 4.2, we prove a lower bound on the optimal solution value

Figure 3. Two Customers (Circle) That Need to Be Served from the Depot (Rectangle)

of the TSP-D and show that TSP algorithms can be used as approximation algorithms for the TSP-D. Note that under the assumption that the truck travel time between each pair of nodes is a multiple of the drone travel time between these nodes, these two results follow immediately from the findings in Wang, Poikonen, and Golden (2017), where bounds for delivery with multiple trucks and multiple drones are given. We show that these results extend to the case of different distance metrics for the drone and truck.

Finally, in Section 4.3, 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., $V^t = \varnothing$. For this example, we assume that $c^d(e) = (1/\alpha)c(e)$ for all edges e and a given positive 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 serve 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, leading to a total time consumption of $2 + 2\alpha$. This provides savings in the total service time of a factor $1 + \alpha$.

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

To prove Theorem 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

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

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, denoting by $V_{\mathcal{R}}$ and $V_{\mathcal{D}}$ the node sets of truck tour $\mathcal{R}$ and drone tour $\mathcal{D}$, we can specify the following lower bound on the length of the drone tour in an optimal TSP-D tour.

Lemma 1. *Let* $(\mathcal{R}, \mathcal{D})$ *be a TSP-D tour. Then*

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

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 $V_{\mathcal{R}} \cap V_{\mathcal{D}}$ before and after visiting a drone node in $V_{\mathcal{D}} \setminus V_{\mathcal{R}}$. □

Theorem 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 online appendix. Let $(\mathcal{R}^*, \mathcal{D}^*)$ be an optimal solution to the TSP-D, and let $\mathcal{R}^{\mathrm{TSP}}$ 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 $w \in V_{\mathcal{R}^*}$ that 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

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

Using (3) we conclude

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

Furthermore, using (2) we obtain

$$\begin{aligned} t(\mathcal{R}^*, \mathcal{D}^*) &\geq \max\{c(\mathcal{R}^*), c^d(\mathcal{D}^*)\} \\ &\geq \max\left\{c(\mathcal{R}^*), \frac{1}{\alpha}c(\mathcal{D}^*)\right\}, \end{aligned} \qquad (5)$$

where the last inequality follows from the definition of α.

Combining (4) and (5) we obtain

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

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

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

This result proves that the maximum gain we can obtain by employing the truck and drone together is of factor $1 + \alpha$, and we have seen in the example in Figure 3 that this bound is tight. Note that for the case where truck speed and drone speed are related as $c^d(e) = \alpha c(e)$ for all edges e, this result is a special case of Theorem 4 in Wang, Poikonen, and Golden (2017), which

provides a worst case analysis for settings where m trucks and k drones deliver goods simultaneously.

The result of Theorem 1 also shows that for instances with a fixed bound α, the TSP-D is a 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, for example, conclude that the minimum spanning tree (MST) based approximation algorithm (Rosenkrantz, Stearns, and Lewis 1977) with $\gamma = 2$ provides a $2 + 2\alpha$ approximation of the TSP-D, and that the Christofides (1976) algorithm with $\gamma = 1.5$ provides a $1.5 + 1.5\alpha$ approximation. However, based on the bound from Lemma 2, we can obtain an even better approximation guarantee for the MST algorithm and the Christofides (1976) algorithm.

4.2. A Lower Bound and an Approximation Result for the TSP-D

Based on Lemma 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 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 online 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 $V_{\mathcal{D}^*} \backslash V_{\mathcal{R}^*}$ to the closest node in $V_{\mathcal{R}^*}$. Hence

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

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

$$t(\mathcal{R}^*, \mathcal{D}^*) \geq \max\{c(\mathcal{R}^*), c^d(\mathcal{D}^*)\} \quad (7)$$

$$\geq \max\left\{c(\mathcal{R}^*), \frac{1}{\alpha} c(\mathcal{D}^*)\right\} \quad (8)$$

$$\geq \max\left\{c(T_{\mathcal{R}^*}), \frac{2}{\alpha} \sum_{v \in V_{\mathcal{D}^*} \backslash V_{\mathcal{R}^*}} \min_{w \in V_{\mathcal{R}^*}} c(v, w)\right\} \quad (9)$$
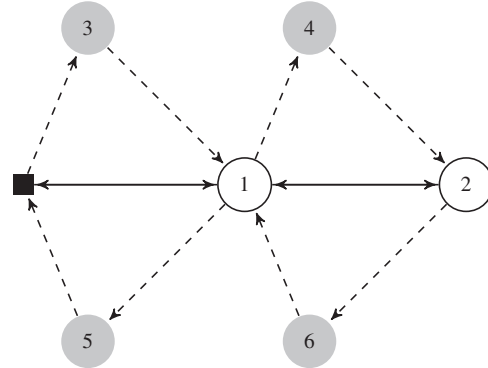
$$= \max\left\{c(T_{\mathcal{R}^*}), \frac{2}{\alpha}(c(T') - c(T_{\mathcal{R}^*}))\right\} \quad (10)$$

$$\geq \frac{2}{2+\alpha} c(T') \quad (11)$$

$$\geq \frac{2}{2+\alpha} c(T) \quad (12)$$

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

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 result of Lemma 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 2. *A solution $(\mathcal{R}, \mathcal{R})$, consisting of a TSP tour $\mathcal{R}$ constructed with the minimum spanning tree heuristic 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 2, we conclude that this is a

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

approximation. $\square$

Of course, this approximation guarantee holds as well for TSP heuristics that improve on the minimum spanning tree heuristic, i.e., that are guaranteed to achieve a worst case objective value $< 2c(T)$, as, for example, the Christofides (1976) algorithm.

4.3. Properties of Optimal Solutions

We conclude this section with some insights into the characteristics of the optimal solutions of a TSP-D that are relevant for the IP formulation in Section 5.

Observation 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 correspond 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 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 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 noncombined 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 $c_o := t(o)$ denotes the costs of operation $o \in O$. In the basic version of the problem, an operation is feasible if it contains at most one drone flight, two combined nodes, and a nonnegative number of truck nodes. However, it is easy to take into account other practical restrictions on the path of the drone and/or truck, for example, 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, $O^+(v) \subset O$ represent the set of operations with end node v, and $O(v) \subset O$ represent 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 the start node in S and the 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 the end node in S and the 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:

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

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

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

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

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

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

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

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

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

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. Because of 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 a Eulerian graph G' such that a 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 edges $e' = (v_i, v_j)$ for each operation $o \in O'$ with a start node v_i and an 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 a 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.

To reduce the number of variables, it is possible to restrict ourselves to operations where the truck travels the shortest tour that covers all of the truck nodes. If we have two operations with the same start node, end node, truck nodes, and drone nodes, we only need to consider the operations with the lowest cost. However, it is still required to consider an exponential number of operations even if we remove the dominated ones.

6. Heuristic Approaches

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

1. We construct a TSP tour $\mathcal{R}'$ that contains *all* nodes. Note that $(\mathcal{R}', \mathcal{R}')$ 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 the tour $\mathcal{R}'$ into the drone tour $\mathcal{D}$ and the 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 TSP Tour

As a first step of our heuristic approach, we assume that the drone does not make any deliveries on its own and construct a tour for truck and drone together that 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

hundred nodes can be solved quickly with a state-of-the-art TSP solver such as Concorde (Applegate et al. 2001). Let $\mathcal{R}_{\text{TSP}}$ 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 1, we have

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

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 the different heuristics and approximation algorithms. As a fast alternative to the optimal tour, we use Kruskal's (1956) minimum spanning tree algorithm to construct a TSP tour in time $O(|E|\log|E|)$ for general graphs. 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 $\mathcal{R}_{\text{MST}}$. Since it is well known that this tour is a 2-approximation for the optimal TSP tour (Rosenkrantz, Stearns, and Lewis 1977), Theorem 2 implies

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

6.2. Constructing a TSP-D Tour from a TSP Tour

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

To simplify notation, in both approaches we rename the nodes in $\mathcal{R}'$ in consecutive order, i.e., $\mathcal{R}' = (r_0, r_1, r_2, \ldots, r_{n+1})$ with $r_0 = r_{n+1} = v_0$.

6.2.1. A Greedy Partitioning Heuristic.
We first propose a greedy heuristic to partition the TSP tour $\mathcal{R}'$ into a subtour for the drone and a subtour for the truck, which is summarized in Algorithm 1. We start with the TSP-D tour $(\mathcal{R}', \mathcal{R}')$, that is, the tour where every node

is visited by a truck and drone. This tour consists of $n + 1$ basic operations.

Algorithm 1 (Pseudocode Greedy Partitioning Heuristic).
Data: A TSP tour $(r_0, \ldots, r_{n+1})$
Result: A feasible TSP-D tour $(\mathcal{R}, \mathcal{D})$
Initialize labels as *simple*;
Initialize MakeFlySavings as described in (22);
Initialize PushLeftSavings and PushRightSavings
    as $-\infty$;
while *There is a node with label* simple *and finite*
    savings **do**
    Execute feasible action with highest possible
      savings;
    Update labels and savings as described;
end
Create TSP-D tour by removing all nodes with
    drone label from truck tour, and all nodes with *truck*
    label from drone tour;
return TSP-D tour.

Each step of the heuristic consists of combining two operations o^1 and o^2 into one operation o^3 using one of the *actions* described below. Among all feasible actions, we choose the one that yields the highest time savings. These can be computed based on the time duration of the two initial operations o^1 and o^2 and the combined operation o^3 as $t(o^1) + t(o^2) - t(o^3)$.

The greedy algorithm assigns labels to the nodes to indicate their role in the TSP-D tour. As long as a node is incident to two basic operations, it is assigned the label *simple*. When combining operations, nodes are assigned the labels *truck*, *drone*, and *combined*, which indicate their role in the newly built (nonbasic) operation. We can easily obtain the corresponding TSP-D tour from a sequence of labels: we obtain the truck tour by removing all nodes with label *drone* from $\mathcal{R}'$ and the drone tour by removing all nodes with label *truck* from $\mathcal{R}'$. We only allow actions that ensure that this indeed gives a well-defined and feasible TSP-D tour.

We initialize all nodes as *simple* except for the depot, which is required to be a *combined* node.

At each step of the heuristic, one of the following actions is performed on a node with the *simple* label:

MakeFly. This action can be performed on all nodes with label *simple* except for the depot. It assigns the *drone* label to node r_i. The two adjacent nodes, r_{i-1} and r_{i+1}, are assigned the label *combined*. In other words, we combine the two basic operations that proceed and succeed r_i, i.e., for which r_i is the start and end node. In the basic version of the problem, this action is feasible for each simple node. If not every node can be visited by the drone, of if there are limitations on the maximum flight distance, we set the savings of the corresponding operation to $-\infty$. The savings of the *MakeFly* action on

node r_i can be computed as

$$MakeFlySavings(r_i) = \underbrace{c(r_{i-1}, r_i)}_{t(o^1)} + \underbrace{c(r_i, r_{i+1})}_{t(o^2)}$$
$$- \underbrace{\max\{c^d(r_{i-1}, r_i) + c^d(r_i, r_{i+1}), c(r_{i-1}, r_{i+1})\}}_{t(o^3)}. \quad (22)$$

PushLeft. This action can be performed only if node r_{i-1} has a *combined* label. It assigns the *combined* label to node r_i and assigns the *truck* label to node r_{i-1}. In other words, we combine the two operations preceding r_i: the nonbasic operation ending in r_{i-1} and the basic operation consisting of r_{i-1} and r_i.

PushRight. This is the same action as *PushLeft*, but performed with the successor of the current node. It can be performed only if node r_{i+1} has a *combined* label.

Since when starting the heuristic all nodes are labeled as *simple*, *PushLeft* and *PushRight* actions are not possible, so they are initialized with costs $-\infty$. In later steps of the heuristic, when a node receives the label *combined* (which can happen because of a *MakeFly*, *PushLeft*, or *PushRight* action), the savings by *PushLeft* and *PushRight* actions for adjacent nodes with the label *simple* need to be updated. A naive recomputation of $t(o^1) + t(o^2) - t(o^3)$ would need $O(n)$ time, since the number of nodes in nonbasic operations can be $O(n)$. However, by keeping track of time duration $t(o)$, driving time $d(o)$, flying time $f(o)$, and flight node r^o of each nonsimple operation, we can compute the savings in constant time as

$$PushLeftSavings(r_i) = t(o^1) + \underbrace{c(r_{i-1}, r_i)}_{t(o^2)}$$
$$- \underbrace{\max\{d(o^3) + c(r_{i-1}, r_i), f(o^3) - c^d(r^{o^3}, r_{i-1}) + c^d(r^{o^3}, r_i)\}}_{t(o^3)}$$

for a *PushLeft* action on node r_i, and analogously, for a *PushRight* action on r_i, as

$$PushRightSavings(r_i) = \underbrace{c(r_i, r_{i+1})}_{t(o^1)} + t(o^2)$$
$$- \underbrace{\max\{c(r_i, r_{i+1}) + d(o^3), f(o^3) - c^d(r_{i+1}, r^{o^3}) + c^d(r_i, r^{o^3})\}}_{t(o^3)}.$$

Also, for *PushLeft* and *PushRight* savings, if there is a restriction on the feasibility of operations, for example, a limited drone distance, savings of actions that violate these constraints are set to $-\infty$.

Since in each action the label of at least one node is set from *simple* to *truck*, *drone*, or *combined*, the algorithm stops after at most n iterations. In each iteration, only labels of adjacent nodes are updated in constant time. Applying a priority queue to find the best operation, this heuristic can be run in $O(n \log n)$ time. See the online appendix for details.

6.2.2. An Exact Partitioning Algorithm. We now describe an exact algorithm to partition $\mathcal{R}'$ into a truck tour $\mathcal{R}$ and a drone tour $\mathcal{D}$ based on dynamic programming. Similar to the greedy partitioning (gp) heuristic described in Section 6.2.1, the exact partitioning algorithm does not change the order of the nodes in $\mathcal{R}'$, 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 $\mathcal{R}'$. Hence, an operation can be defined by a triplet (i, j, k), with a 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

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

For a basic operation, we set

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

If a certain operation is infeasible, for example, because of a maximum on the flying distance of the drone, we set its time duration to ∞.

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

$$T(i, j) := \min_{k=i+1}^{j-1} T(i, j, k). \quad (23)$$

To track the corresponding TSP-D tour, we define a matrix M and store the best operation found in (23) in $M(r_i, r_j)$.

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

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

To store the sequence of combined nodes, we store in $P(i)$ an element from $\arg\min_{k=0}^{i-1}[V(k) + T(k, i)]$.
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 $\mathcal{R}'$. 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 online appendix), it is possible to do the computations of the lengths 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 3. *For a given TSP route $\mathcal{R}'$, 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 $\mathcal{R}'$. This solution can be found in time $O(n^3)$.*

6.3. Iterative Improvement Procedure

Algorithm 2 (Iterative Improvement Procedure).
Data: An initial tour $\mathcal{R}'$, a partitioning algorithm f, and a neighborhood function N
Result: A locally optimal tour for a neighborhood function
$\mathcal{R}'_1 \leftarrow \mathcal{R}'$;
$i \leftarrow$ True;
while i **do**
  $\quad i \leftarrow$ False;
  $\quad M \leftarrow N(\mathcal{R}')$;
  $\quad$**for** $m \in M$ **do**
  $\quad\quad$Modify $\mathcal{R}'$ according to move m;
  $\quad\quad$**if** $f(\mathcal{R}') < f(\mathcal{R}'_1)$ **then**
  $\quad\quad\quad \mathcal{R}'_1 \leftarrow \mathcal{R}'$;
  $\quad\quad\quad i \leftarrow$ True;
  $\quad\quad$**end**
  $\quad\quad$Modify $\mathcal{R}'$ according to the inverse of move m;
  $\quad$**end**
  $\quad$**if** i **then**
  $\quad\quad \mathcal{R}' \leftarrow \mathcal{R}'_1$;
  $\quad$**end**
end
return $\mathcal{R}'$.

Both the greedy and exact partitioning algorithms rely on an initial tour $\mathcal{R}'$. 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 whether we can find better solutions. We adopt several local search heuristics that modify $\mathcal{R}'$ 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 $\mathcal{R}'$, i.e., a two-point move (2p); one where we remove two edges in $\mathcal{R}'$ and replace them with two new edges, resulting in a different tour, i.e., a 2-opt move (2opt); and one where we relocate a node in

$\mathcal{R}'$ to a new position, i.e., a one-point move (1p). Moreover, we also consider a neighborhood that combines all of 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 2.

All of the proposed neighborhoods 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. In the main set of experiments, we assume that the drone has an unlimited range ($t^{\max} = \infty$) and that all locations can be visited by both the drone and the truck ($V^t = \varnothing$). However, we also run some experiments in which we add restrictions on the drone range and the nodes that the drone can visit. These experiments are discussed in Sections 7.2.4 and 7.3.3.

Our experiments were executed within a virtual machine with 2 GB 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 operating system. The hardware configuration of the host consisted of an Intel Core i7-4770 CPU with 16 GB 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. To obtain an MST solution, we applied the Delaunay triangulation code in the JTS Topology Suite library, version 1.13. The source code of the data structures and algorithms used in our experiments are released as Bouman (2018).

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 in Bouman, Agatz, and Schmidt (2018).

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 is computed 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 instances 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 the depot.

In the default setting we assume that the drone is twice as fast as the truck on all edges, i.e., $\alpha = c(e)/c^d(e) = 2$, 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 assess 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 and exact partitioning. 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 16 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 approach starting from the sequence of the TSP-MC tour and call this the 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 $\alpha = 2$, Theorems 1 and 2 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. For each instance I and each heuristic H applied to that instance, we compute the optimality gap as

$$\Delta = \frac{\text{objective value heuristic} - \text{optimal objective value}}{\text{optimal objective value}}.$$

We report the average gap and the maximum gap in percentages, as well as the number of instances that were solved to optimality. The solution times for the IP for all 10 node instances are between 5.7 seconds and 38.7 seconds. (Note that the number of possible operations, which are used as variables in our integer program, increases exponentially with the instance size and quickly becomes much more time and memory consuming; e.g., instances with 12 nodes take more than two hours to solve and require around 2 GB RAM.)

The table shows that there are big differences in solution quality between the 20 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 heuristic performs very poorly, with a maximum gap of 47.1% and achieving optimality in none of the 30 instances. The TSP-MC heuristic does not perform well either, with an average gap between 16.4% and 22.0%. A possible reason for this is that the TSP-MC 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 20 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

Table 2. Comparison of Heuristic Solutions to the Optimal IP Solutions, Uniform, $n = 10$, and $\alpha = 2$, Averaged Over 10 Instances

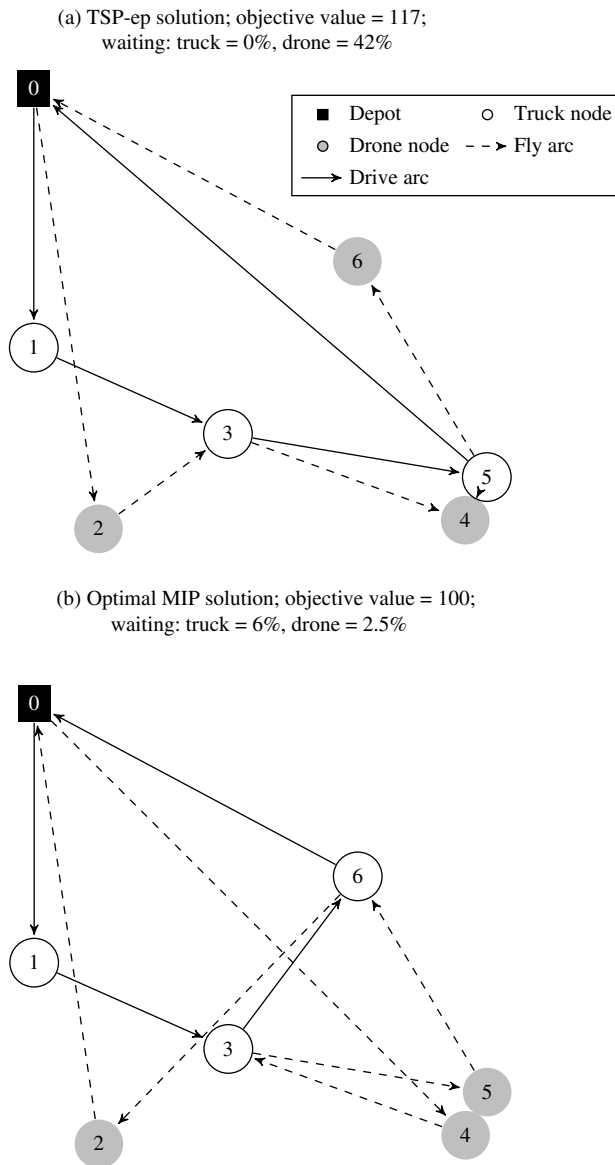|	Uniform			1-center			2-center		
	Δ (%)			Δ (%)			Δ (%)		
	Avg	Max	# opt	Avg	Max	# opt	Avg	Max	# opt
MST-gp	19.6	35.6	0/10	27.4	40.9	0/10	25.6	47.1	0/10
MST-ep	18.1	35.6	0/10	23.2	40.9	0/10	23.2	45.8	0/10
TSP-gp	16.1	30.1	0/10	22.6	32.9	0/10	18.5	24.9	0/10
TSP-ep	16.0	23.3	0/10	15.2	32.9	0/10	12.7	19.7	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

of up to 11.5 pp (percentage point) 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 pp and 7.4 pp on average. Finally, we see that the improvement procedures are very beneficial. In particular, we observe a decrease of the gap with the optimal solution by up to 15% over the heuristics without the 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, which shows the solution of the TSP-ep heuristic and an optimal solution for a specific 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 nodes 4 and 5 by the drone leads to a 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 distances of the truck and the drone, and the waiting times 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-D 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 the results for only one of them, i.e., the two-point 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%

Figure 5. Example of TSP-D Solutions, Uniform, $|V| = 6$, and $\alpha = 2$

(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%


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 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.
To this point, 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

Table 3. Characteristics of TSP-D Solutions, Uniform, $n = 10$, and $\alpha = 2$, Averaged Over 10 Instances

	Time	No. of 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

Figure 6. Comparison of Performance for Different Speeds, Uniform, $n = 10$, Averaged Over 10 Instances
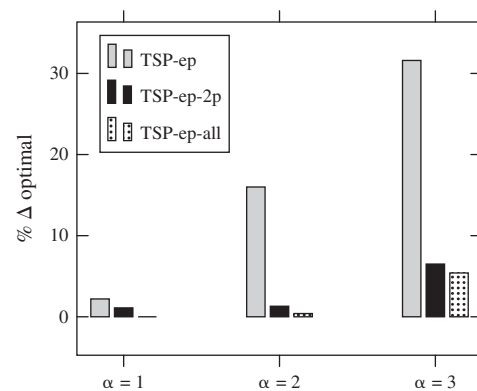
shows the quality of the TSP-ep, TSP-ep-2p, and TSP-ep-all heuristics for these different drone speeds on the instance with uniformly distributed customer locations. The results show that all of the heuristics perform better for smaller α values than for larger ones.

We also see that the advantage of trying different initial tours as the basis for the solution that we gain when using TSP-ep-2p or TSP-ep-all instead of TSP-ep is more prominent at higher relative drone speeds. The reason for this is that, while with low drone speed the drone can deliver only to nodes that are close to the truck route without causing long waiting times, when the speed of the drone is high, the drone will be able to cover more distance, and thus can be used to deliver to nodes further away from the truck route. Hence, using a good TSP solution as a base route leads to solutions with high drone waiting times and does not use the full potential of simultaneous delivery. This is illustrated in Figure 7, which plots the sum of the distances 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.

Figure 7. The Travel Distances of the Truck and the Drone, Respectively, in the Optimal Solutions, Uniform, $n = 10$, Averaged Over 10 Instances
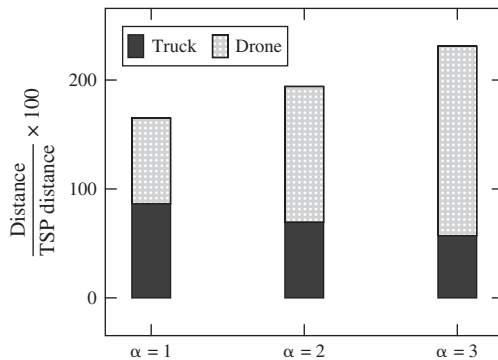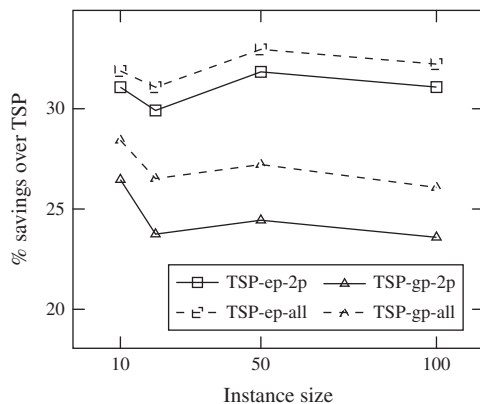

Figure 8. Comparison of Heuristics, Uniform, $\alpha = 2$, Averaged Over 10 Instances

Figure 9. Comparison of Solution Times, Uniform, $\alpha = 2$, Averaged Over 10 Instances

shows the run times of the different heuristics. We see that the greedy approaches (TSP-gp-2p and TSP-gp-all) are extremely fast and solve all of the instances with 100 nodes within a few seconds. The run time of heuristics that use the exact partitioning approach, on the other hand, increase rapidly with the number of nodes.

7.2.4. Comparison of the Heuristics for Problems with Drone Restrictions. In this section, we study the performance of the different heuristics on a set of instances that incorporate several restrictions on the drone flights. In one set of experiments, we impose various limits on the range of the drone as a percentage of the farthest distances between the nodes in the instance. In a second set of experiments, we consider different numbers of customer locations that cannot be served by the drone, i.e., between 10% and 100% nondrone nodes. We use the same 10 instances as before and randomly select 10 different subsets for each number of nondrone nodes for each instance. To model the restrictions, we change the distance function c^d such that flights to nondrone nodes and a sequence of two flights that violate the drone range become ∞.

Figure 10 plots the optimality gap for both sets of experiments for the different TSP-based heuristics. We see that the exact partitioning approaches consistently outperform the greedy methods and that the improvement procedure that combines different move neigborhoods (all) performs better than the simple two-point move neighborhood (2p). As expected, we see that the performances of the heuristics converge for the more constrained settings. In Figure 10(a), we see that the performance of all heuristics improves with the number of nondrone nodes, i.e., when the number of feasible operations decreases, which was to be expected.

The pattern we see in Figure 10(b) is less clear. Here, the drone range is decreased gradually from 200% (no restrictions) to 0% (no drone flights allowed). A drone range of 100% implies that we allow only operations where the travel time of the drone does not
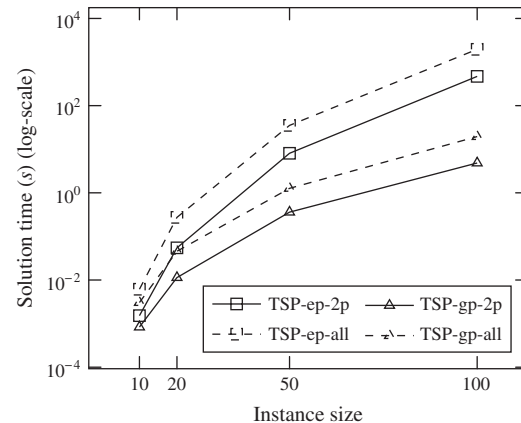
7.2.3. Comparison of the Heuristics for Larger Problems. To study the potential gains in larger problem instances, we use the route-first, cluster-second heuristics to solve the TSP-D, as we cannot solve these to optimality with our IP models. The heuristics provide good solutions for smaller instances, 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 8 support this observation. The figure shows the gain over the truck-only TSP tour for the four versions of the heuristic with the improvement procedure. We observe that using the exact partitioning consistently outperforms using the greedy heuristic. The heuristics that use the exact partitioning (TSP-ep) show consistent savings of about 30%, and those using the greedy heuristic (TSP-gp), savings 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 9

Figure 10. Comparison of Heuristics, Uniform, $n = 10$, and $\alpha = 2$, Averaged Over (a) 10 Instances and (b) 100 Instances
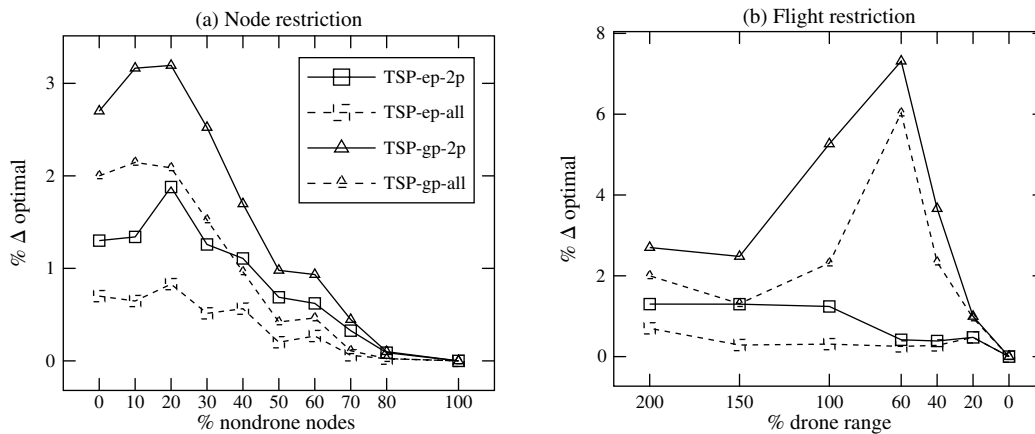

exceed the longest travel time between a single pair of locations, i.e., $\max_{v,u \in V} c^d(v, u)$. For the exact partitioning approaches, we observe a very similar pattern as in Figure 10(a), which was to be expected since a decrease in drone range leads to a decreased number of feasible operations. However, the performance of the greedy heuristics deteriorates with a decreasing flight range up to a 60% range but then improve with further restrictions. One potential explanation for this behavior is the way that the greedy algorithm generates operations. The greedy algorithm generates more complex operations from "triangle" operations, which consist only of two combined nodes and a drone node. If the flight distance of a "triangle" operation is higher than the flight range, the operation is regarded as infeasible and will not be considered by the greedy algorithm. Consequently, the potentially feasible and good complex operations that would have been generated from the triangle operation will not be generated.

7.3. Impact of Different Parameters on System Performance

In this section, we study the impact of characteristics of different parameters on the performance of a combined truck-and-drone delivery system. This will create important insights to companies that are experimenting with this innovative new mode of transportation. We evaluate the performance by comparing the completion time of the truck-and-drone system with the truck-only TSP. In all experiments, we use instances with 100 nodes and apply the TSP-ep-all heuristic to solve the TSP-D.

7.3.1. Distribution of Service Locations. Figure 11 presents box plots of the savings of the solutions of the TSP-D over the truck-only TSP over 10 randomly generated instances of each instance type as introduced in Section 7.1. We observe that the combination of a truck and a drone leads to substantial time savings, on average, between 30% and 38%. The results suggest that adding a drone is especially beneficial if demand

Figure 11. TSP vs. TSP-D, $n = 100$, and $\alpha = 2$; Box Plot for 10 Instances
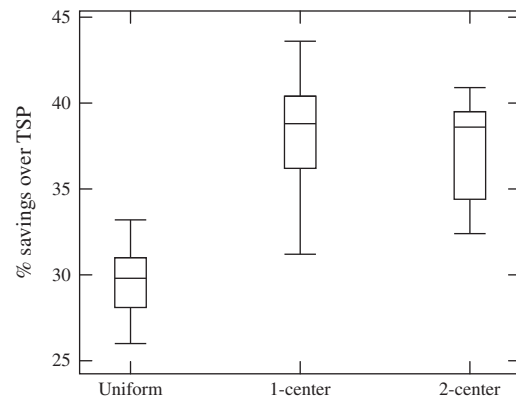
is clustered around one or two centers. This is intuitive since we may save a lot of time by serving points outside the center with the drone.

7.3.2. Drone Speed. In this section, we examine the impact of the speed of the drone relative to that of the truck on the performance of a combined truck-and-drone delivery system. As discussed previously, the drone is typically faster than the truck as it can take shortcuts and is not affected by congestion. Table 4 provides the results for three different relative speeds of the drone, i.e., equal speed ($\alpha = 1$), twice as fast ($\alpha = 2$), and three times as fast ($\alpha = 3$). In addition to the savings over the truck-only TSP, we also provide the number of nodes per operation, the distance per stop, and the utilization of the truck and the drone, i.e., its travel time as a percentage of the total time.

The table shows that the benefits of a combined truck-and-drone delivery system increase with the relative speed of the drone. In line with this observation, we see that the number of drone deliveries increases with the speed of the drone. On the contrary, the number of simultaneous deliveries by the truck and the drone, as indicated by the number of truck nodes per operation (i.e., drone flight), decreases with the speed of the

Table 4. Characteristics of TSP-D Solutions for Different Relative Drone Speeds (α), TSP-ep-All, Uniform, and $n = 100$, Averaged Over 10 Instances

	Savings (%)	No. of nodes per operation			Distance per stop		Utilization (%)	
		Truck	Drone	Combined	Truck	Drone	Truck	Drone
$\alpha = 1$	17.3	3.6	1.0	1.0	1.0	4.4	99.3	94.0
$\alpha = 2$	29.5	0.6	1.0	1.0	1.1	3.1	98.9	83.1
$\alpha = 3$	34.1	0.4	1.0	1.0	1.1	3.5	99.2	74.0

drone. The truck makes, on average, 3.6 simultaneous deliveries at lower speeds but less than one simultaneous delivery at higher speeds. This is intuitive since it is easier for a faster drone to catch up with the truck before the truck makes another delivery. The average distances per stop suggest that the truck serves the locations that are close to each other, while the drone serves the locations that are further away. This effect is especially pronounced in the setting where the drone has the same speed as the truck. As expected, we see that the utilization of the drone reduces as it becomes faster, as it has to wait for the truck more often.
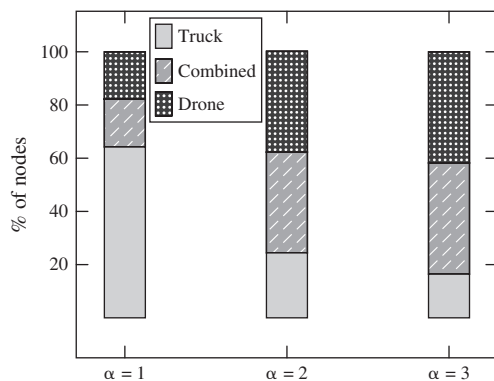
To compare the structure of the solutions in more detail, Figure 12 shows the number of truck, combined, and drone nodes for the different speeds. The figure illustrates that the number of drone nodes increases

Figure 12. The Number of Truck, Combined, and Drone Nodes, TSP-ep-All, Uniform, and $n = 100$, Averaged Over 10 Instances

with the drone speed. We particularly see that 41.8% of the customer locations are visited by the drone in the fastest setting ($\alpha = 3$). Overall, we observe that the number of combined nodes is roughly equal to the number of drone nodes in the different solutions. This indicates that the truck never waits for the drone at the start node of an operation and that there are very few operations without a drone node.

7.3.3. Drone Range. In this section, we investigate the impact of the drone range on the performance of the system. We run a set of experiments for different instance sizes, i.e., $n = 20$, 50, and 100, and for various drone ranges. Figure 13 shows the savings of the combined truck-and-drone system over the truck-only TSP and the number of locations served by the drone for different ranges.

As expected, we see that the savings increase with the drone range. Figure 13(a) also shows that the smaller instances are more sensitive to a decrease of the drone range than the larger instances. One potential reason for this is that the average distance between customer locations is smaller in the larger instances, which means the drone requires a smaller range to serve customers. Another possible reason is that the higher number of feasible truck and drone tours in the larger instances may make them more robust with respect to the drone range.

Figure 13(b) shows that the number of drone nodes increases with the drone range up to a certain point, after which it decreases slightly to a stable level. This observation is associated with the fact that if the drone

Figure 13. Savings and Drone Nodes for Different Drone Flight Ranges, Uniform, $\alpha = 2$, Averaged Over 10 Instances
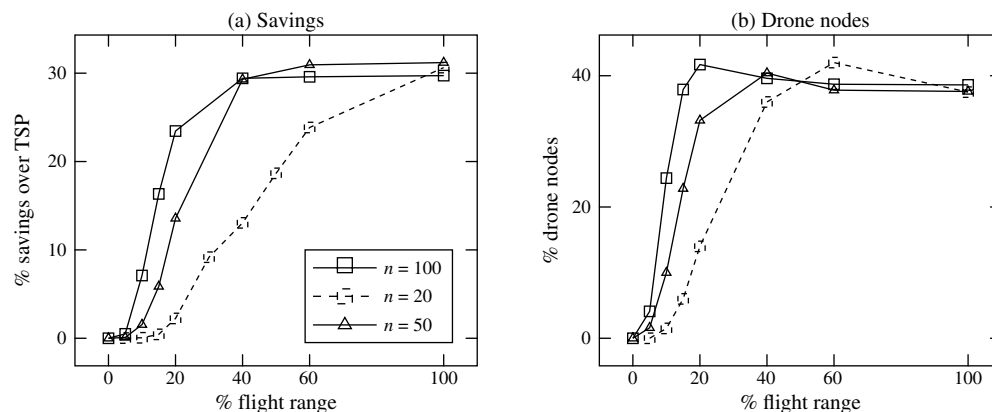
range is limited, it may be better to schedule many but shorter drone flights, while at higher drone ranges, it may be better to schedule fewer but longer drone flights.

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 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 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. One promising approach is to combine the operation-based integer programming formulation developed in this paper with a branch-and-cut-and-price approach.

A natural extension of the problem is to consider a setting with multiple trucks and multiple drones as done, for example, in the theoretical worst case analysis of Wang, Poikonen, and Golden (2017). 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 take place only while the truck is parked at certain locations, then it can be easily incorporated into 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

Applegate DL, Bixby RE, Chvátal V, Cook WJ (2001) TSP cuts which do not conform to the template paradigm. Jünger M, Naddef D, eds. *Computational Combinatorial Optimization*, Lecture Notes Comput. Sci., Vol. 2241 (Springer, Berlin Heidelberg), 261–303.

Applegate DL, Bixby RE, Chvátal V, Cook WJ (2011) *The Traveling Salesman Problem: A Computational Study* (Princeton University Press, Princeton, NJ).

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

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

Belenguer JM, Benavent E, Martínez A, Prins C, Prodhon C, Villegas JG (2016) A branch-and-cut algorithm for the single truck and trailer routing problem with satellite depots. *Transportation Sci.* 50(2):735–749.

Bouman P (2018) pcbouman-eur/drones-TSP: Traveling salesman with drone (Java code) v1.0.0. https://doi.org/10.5281/zenodo.1204697.

Bouman P, Agatz N, Schmidt M (2018) Instances for the TSP with drone (and some solutions). https://doi.org/10.5281/zenodo.1204676.

Christofides N (1976) Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, Defense Technical Information Center, Fort Belvoir, VA.

Current JR, Schilling DA (1989) The covering salesman problem. *Transportation Sci.* 23(3):208–213.

Drexl M (2011) 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.

Drexl M (2012) Synchronization in vehicle routing–A survey of VRPs with multiple synchronization constraints. *Transportation Sci.* 46(3):297–316.

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

Evers L, Dollevoet T, Barros AI, Monsuur H (2014) Robust UAV mission planning. *Ann. Oper. Res.* 222(1):293–315.

Garey MR, Johnson DS (1979) *Computers and Intractability: A Guide to NP-Completeness* (WH Freeman, New York).

Golden B, Naji-Azimi Z, Raghavan S, Salari M, Toth P (2012) The generalized covering salesman problem. *INFORMS J. Comput.* 24(4):534–553.

Gulczynski DJ, Heath JW, Price CC (2006) The close enough traveling salesman problem: A discussion of several heuristics. Alt FB, Fu MC, Golden BL, eds. *Perspectives in Operations Research*, Oper. Res./Comput. Sci. Interfaces Series, Vol. 36 (Springer, Boston), 271–283.

Hern A (2014) DHL launches first commercial drone "Parcelcopter" delivery service. *Guardian* (September 24), https://www.theguardian.com/technology/2014/sep/25/german-dhl-launches-first-commercial-drone-delivery-service.

Karp RM (1972) Reducibility among combinatorial problems. Miller RE, Thatcher JW, Bohlinger JD, eds. *Complexity of Computer Computations*, IBM Res. Sympos. Series (Springer, Boston), 85–103.

Kim MH, Baik H, Lee S (2014) Response threshold model based UAV search planning and task allocation. *J. Intelligent Robotic Systems* 75(3–4):625–640.

Kruskal JB (1956) On the shortest spanning subtree of a graph and the traveling salesman problem. *Proc. Amer. Math. Soc.* 7(1):48–50.

Laporte G (1992) The traveling salesman problem: An overview of exact and approximate algorithms. *Eur. J. Oper. Res.* 59(2):231–247.

Lin SW, Vincent FY, Chou SY (2009) Solving the truck and trailer routing problem based on a simulated annealing heuristic. *Comput. Oper. Res.* 36(5):1683–1692.

Murray CC, Chu AG (2015) The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Res. Part C: Emerging Tech.* 54:86–109.

Nicas J, Bensinger G (2015) Delivery drones hit bumps on path to doorstep. *Wall Street Journal* (March 20), https://www.wsj.com/articles/technical-hurdles-delay-drone-deliveries-1426867441.

Popper B (2013) UPS researching delivery drones that could compete with Amazon's Prime Air. *Verge* (December 3), https://www.theverge.com/2013/12/3/5169878/ups-is-researching-its-own-delivery-drones-to-compete-with-amazons.

Rosenkrantz DJ, Stearns RE, Lewis PM II (1977) An analysis of several heuristics for the traveling salesman problem. *SIAM J. Comput.* 6(3):563–581.

Scheuerer S (2006) A tabu search heuristic for the truck and trailer routing problem. *Comput. Oper. Res.* 33(4):894–909.

Shamos MI (1978) Computational geometry. Unpublished doctoral thesis, Yale University, New Haven, CT.

Shuttleworth R, Golden BL, Smith S, Wasil E (2008) Advances in meter reading: Heuristic solution of the close enough traveling salesman problem over a street network. Golden B, Raghavan S, Wasil E, eds. *The Vehicle Routing Problem: Latest Advances and New Challenges*, Oper. Res./Comput. Sci. Interfaces, Vol. 43 (Springer, Boston), 487–501.

Valavanis KP, Vachtsevanos GJ (2015) UAV applications: Introduction. Valavanis K, Vachtsevanos GJ, eds. *Handbook of Unmanned Aerial Vehicles* (Springer, Dordrecht, Netherlands), 2639–2641.

Villegas JG, Prins C, Prodhon C, Medaglia AL, Velasco N (2011) A grasp with evolutionary path relinking for the truck and trailer routing problem. *Comput. Oper. Res.* 38(9):1319–1334.

Wang X, Poikonen S, Golden B (2017) The vehicle routing problem with drones: Several worst-case results. *Optim. Lett.* 11(4):679–697.

Wohlsen M (2014) The next big thing you missed: Amazon's delivery drones could work. They just need trucks. *Wired* (June 10), https://www.wired.com/2014/06/the-next-big-thing-you-missed-delivery-drones-launched-from-trucks-are-the-future-of-shipping/.