ARCHIVIO ISTITUZIONALE
DELLA RICERCA

Alma Mater Studiorum Università di Bologna
Archivio istituzionale della ricerca

The multiple traveling salesman problem in presence of drone- and robot-supported packet stations

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

Published Version:
Kloster K., Moeini M., Vigo D., Wendt O. (2023). The multiple traveling salesman problem in presence of drone- and robot-supported packet stations. EUROPEAN JOURNAL OF OPERATIONAL RESEARCH, 305(2), 630-643 [10.1016/j.ejor.2022.06.004].

Availability:
This version is available at: https://hdl.handle.net/11585/899685 since: 2024-04-18

Published:

DOI: http://doi.org/10.1016/j.ejor.2022.06.004

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/).
When citing, please refer to the published version.

(Article begins on next page)

12 September 2026

The multiple traveling salesman problem in presence of drone- and robot-supported packet stations

Konstantin Kloster[a], Mahdi Moeini[a,*], Daniele Vigo[b], Oliver Wendt[a]

[a]*Chair of Business Information Systems and Operations Research (BISOR), Technische Universität Kaiserslautern, 67663 Kaiserslautern, Germany.*
[b]*Department of Electrical, Electronic and Information Engineering "G. Marconi," University of Bologna, 40136 Bologna, Italy.*

Abstract

In this paper, we introduce the *multiple Traveling Salesman Problem with Drone Stations* (mTSP-DS), which is an extension to the classical *multiple Traveling Salesman Problem* (mTSP). In the mTSP-DS, we have a depot, a set of trucks, and some packet stations that host a given number of autonomous vehicles (drones or robots). The trucks start their mission from the depot and can supply some packet stations, which can then launch and operate drones/robots to serve customers. The goal is to serve all customers either by truck or by drones/robots while minimizing the makespan. We formulate the mTSP-DS as a *mixed integer linear programming* (MILP) model to solve small instances. To address larger instances, we first introduce two variants of a decomposition-based *matheuristic*. Afterwards, we suggest a third approach that is based on populating a solution pool with several restarts of an *iterated local search metaheuristic*, which is followed by determining the best combination of tours using a *set-partitioning* model. To verify the performance of our algorithms, we conducted extensive computational experiments. According to the numerical results, we observe that the use of drone stations leads to considerable savings in delivery time compared to traditional mTSP solutions. Furthermore, we investigated the energy consumption of trucks and drones. Indeed, depending on the energy consumption coefficients of trucks and drones as well as on the distance covered by drones, the mTSP-DS can also achieve energy savings in comparison to mTSP solutions.

Keywords: Logistics, Vehicle routing, Drone deliveries, Mixed integer linear programming, Heuristics

1. Introduction

Drones, also known as *unmanned aerial vehicles*, have the potential to transform many industries and thus may also have a social impact (Floreano & Wood, 2015). Indeed, infrastructure, agriculture, transport, and security sector offer the best financial

*Corresponding author
Email address: mahdi.moeini@wiwi.uni-kl.de (Mahdi Moeini)

opportunities for the use of drones (see (Otto et al., 2018) and references therein). Within the transport area, drones could change the way that packages are delivered to the customers. In this context, a loaded drone, carrying a light and small- or medium-sized parcel, could fly to a customer, deliver the package, and then return to its starting point.

Due to technical limitations, drones, unlike trucks, can usually only transport one parcel at a time (Agatz et al., 2018). As a result, the total distance covered for deliveries by drones will be greater than for traditional deliveries by truck. Nevertheless, it is possible to reduce delivery costs by using drones, e.g., since the delivery drones operate autonomously, using drones can reduce costs by decreasing labor costs. Considering that acquisition costs for drones are also relatively low in comparison to trucks, drones allow for higher parallel operations, which in turn can lead to reduced delivery times. Another factor that has a positive effect on delivery times is that drones are not tied to a road network. Indeed, this enables drones to avoid congestion, especially in densely populated urban areas. Finally, drones have also benefits in terms of CO_2 emissions. According to Goodchild & Toy (2018), drones emit less CO_2 than trucks when customers are close to the depot or when only a few customers need to be supplied.

Although many technological advancements have been made in recent years, drone usage is still accompanied by shortcomings that need to be considered. For example, the size and the weight of parcels that can be transported by drones are limited. An alternative to drones, which circumvents some of these problems, are delivery robots such as those made known by, e.g., Starship Technologies (Figliozzi, 2020). In fact, robots move either on roads or sidewalks, reaching speeds of up to 30 km/h and 6 km/h, respectively, and are thus slower than trucks and drones (Moeini & Salewski, 2020; Schermer et al., 2020). However, robots offer some advantages, e.g., the use of sidewalks may reveal shortcuts that trucks cannot use. Moreover, compared to drones, robots can transport much heavier packages and, some types of robots can carry several packages at once. Furthermore, robots are more energy-efficient than drones and trucks. One problem of drones and robots alike is their relatively short range. However, this problem can be overcome, for example, by the use of *drone stations*.

A *drone station* is a facility that can be used to launch and operate drones (Kim & Moon, 2018). When a truck delivers parcels to a drone station, they are automatically processed and loaded into a drone. Furthermore, the batteries of returned drones could be replaced and charged in such stations. In practice, drone stations have already been tested by DHL (DHL, 2018). Even though the concept of drone stations can also be transferred to robots, for the sake of easing the readability of the paper, we will use the term drone station throughout this paper.

The aforementioned technical problems also have implications on the way that drones are approached from an operational perspective that we address in this paper. More precisely, the contributions of this paper are as follows:

- We introduce the *multiple Traveling Salesman Problem with Drone Stations* (mTSP-DS), a tour planning model that integrates drones and drone stations into the classical *multiple Traveling Salesman Problem* (mTSP). In addition, the mTSP-DS also incorporates elements from *parallel machine scheduling* (see, e.g., Cheng & Sin, 1990) and *facility location* problems (Cornuéjols et al., 1990).

In the next sections, we define the mTSP-DS and formulate it as a *mixed integer linear program* (MILP).

- The MILP model of the mTSP-DS can be solved by any standard MILP solver, e.g., *Gurobi Optimizer*. But, this is possible only on small instances because the runtime grows too fast due to the complexity of the mTSP-DS. To overcome this issue, we first introduce two variants of a decomposition-based *matheuristic*. Afterwards, we present a two-phase matheuristic that is based on populating a solution pool and using a *set-partitioning* model. Recently, a similar approach has proven successful for truck and trailer routing problems(Accorsi & Vigo, 2020), which share some similarities with the mTSP-DS.

- We report the results of our extensive computational experiments, which prove the efficiency of the introduced algorithms as well as the usefulness of drones and drone stations in last-mile delivery. According to the numerical results, we observe not only that the algorithms are able to provide high-quality solutions, but also we show that the use of drone stations and drones leads to considerable savings in delivery time compared to traditional mTSP solutions. Furthermore, we show that depending on the energy consumption coefficients of trucks and drones, as well as on the distance covered by drones, the mTSP-DS can also achieve energy savings in comparison to mTSP solutions.

The remainder of this paper is structured as follows. Section 2 is devoted to an overview of the existing literature regarding the use of drones in last-mile delivery from a tour planning perspective. In Section 3, we provide a precise and formal description of the mTSP-DS as well as its MILP formulation. Afterwards, we describe our algorithms in Section 4. We report the computational experiments and their numerical results in Section 5. Finally, some concluding remarks are drawn in Section 6.

2. Drones and drone stations in last-mile delivery

Because of the economic potential of drones for the logistics industry, researchers have in recent years started to examine drones from an operational perspective and to incorporate them into tour planning models. Detailed state-of-the-art reviews can be found, e.g., in (Khoufi et al., 2019; Macrina et al., 2020; Otto et al., 2018; Rojas Viloria et al., 2021; Schermer et al., 2019a,b). However, for the sake of completeness, we provide a short overview of the existing literature related to this paper.

Since the operational range of drones is limited and not all packages can be transported by drones, it is unlikely that drones completely replace delivery by trucks. Thus, strategies in which trucks are assisted by drones during delivery became prevalent.

The first work that follows such an approach is the *Flying Sidekick Traveling Salesman Problem* (FSTSP), where a truck works in conjunction with a single drone (Murray & Chu, 2015). It is an extension of the *Traveling Salesman Problem* (TSP), and the goal is to serve all customers either by truck or by drone. As in the TSP, the truck starts from the depot and has to make a round trip, which ends at the depot again. The drone, however, is either launched from the depot or carried by the truck. In the latter

case, the driver can load and launch the drone from a customer node. Once launched, the drone will fly to a customer, drop (deliver) a parcel, and reunite with the truck at another customer location. After servicing the drone, the driver can dispatch it to the next customer or transport it to the depot. Alternatively, the drone might as well return to the depot after a delivery, where the drone will be taken out of service. Murray & Chu (2015) presented a MILP model and a simple heuristic to solve the FSTSP.

As a similar problem, Agatz et al. (2018) proposed the *TSP with Drone* (TSP-D). Unlike the FSTSP, the TSP-D allows the drone to return to its origin after a delivery, where, in the meantime, the truck can wait at this node. Additionally, the TSP-D assumes that the truck may revisit nodes. Agatz et al. (2018) formulated the TSP-D as an *integer program* and suggested several variations of a *route-first, cluster-second* heuristic. As an extension, Marinelli et al. (2018) introduced *en-route operations* for the TSP-D. This means that starting and retrieving drones is not limited to nodes of the underlying graph, but can also be performed from points on the edges. Marinelli et al. (2018) proposed a heuristic based on the *greedy randomized adaptive search procedure* and numerically showed that en-route operations can reduce the overall delivery time.

The *Vehicle Routing Problem with Drones* (VRPD) is another extension to the FSTSP. The VRPD considers several trucks such that each truck can transport one or multiple drones (Wang et al., 2017). Several papers have studied the VRPD and its variants and suggested different solution methods for solving the VRPD (see, e.g., Kitjacharoenchai et al., 2020; Sacramento et al., 2019; Schermer, 2019; Schermer et al., 2018a,b, 2019b, and references therein). In particular, the *Vehicle Routing Problem with Drones and en Route Operations* (VRPDERO) is a variant of the VRPD that incorporates en route operations (Schermer et al., 2019a). Schermer et al. (2019a) provided a MILP formulation for the VRPDERO and introduced valid inequalities to enhance the performance of the MILP solver. In addition, Schermer et al. (2019a) proposed a heuristic based on *Variable Neighborhood Search* and *Tabu Search*.

In the problems mentioned so far, the truck has two functions: besides the traditional delivery of parcels, the truck can be seen as a moving depot, which increases the operational range of drones. In the *Two-echelon Routing Problem with Truck and Drones*, the truck gives up delivering packages and operates only as a mobile depot (Vu et al., 2021). In these problems, the truck visits a set of so-called truck nodes at which it might stop to launch drones, which will then fly to the customer nodes to deliver packages. Once all drones have returned from their deliveries, the truck continues its tour. The authors propose single and multi-trip drone variants of the problem and present MILP formulations as well as a GRASP approach to solve the problem. In the *Mothership and Drone Routing Problem* (MDRP), the truck is replaced by a so-called *mothership* (Poikonen & Golden, 2019). The mothership is a large aircraft that fulfills the function of the mobile depot. Compared to a truck, the mothership has the advantage of not being tied to the road network, i.e., a feature similar to drones. Therefore, the mothership can move in continuous Euclidean space and launch drones from anywhere. To solve the MDRP, Poikonen & Golden (2019) introduced an exact *branch-and-bound* algorithm and several heuristic methods.

If most customers are relatively far away from the depot, then using the FSTSP can be an appropriate approach, where the operation range of drones is extended by trucks. However, if most of the customers are within the range of the depot, it makes more

sense to supply the customers directly by drone from the depot. For this case, Murray & Chu (2015) suggested the *Parallel Drone Scheduling TSP* (PDSTSP). As in the FSTSP, the PDSTSP assumes that some given customers can only be served by the truck. Furthermore, in the PDSTSP, the truck is tasked to serve customers through a TSP tour. Simultaneously, a set of drones delivers parcels to customers that are located within a given range through *round trips*, i.e., starting from the depot, serving a customer, and returning to the depot. When a drone returns to the depot after a delivery, the drone can be loaded again to serve another customer. One particular characteristic of the PDSTSP is that the truck and the drones work independently of each other. The authors introduced a MILP formulation and a heuristic for the PDSTSP, where the heuristic uses the *longest processing time* (LPT) algorithm to schedule the drone deliveries. Nguyen et al. (2022) present the *min-cost Parallel Drone Scheduling VRP* as an extension of the PDSTSP in which several trucks and a cost objective are considered. In addition, the authors formulate the problem as a MILP and present a *ruin-and-recreate* algorithm.

The potential of the PDSTSP approach and its variants depends on the location of the depot. To cover other areas as well, additional depots could be built. However, the construction and maintenance of conventional depots can be expensive, especially in densely-populated urban areas. To circumvent this obstacle, Kim & Moon (2018) suggested the use of a *drone station* and introduced the *Traveling Salesman Problem with a Drone Station* (TSP-DS), which extends the PDSTSP to include an additional depot as drone station, where the drones are automatically loaded, launched, retrieved, and serviced. In the TSP-DS, the truck serves the customers through a TSP tour, and some customers might be served by drones that are located at the drone station. For this purpose, the truck can activate the drone station by visiting it during a TSP tour, and supplying the drone station with parcels. Once activated, the drone station uses drones to serve customers within the range of the drone station. The TSP-DS is formulated as a MILP and, if enough drones are provided, it is possible to decompose the problem into an independent TSP and a parallel machine scheduling problem (Kim & Moon, 2018).

Based on the TSP-DS, Schermer et al. (2019c) introduced the *Traveling Salesman Drone Station Location Problem* (TSDSLP) that incorporates several possible drone stations from which a subset might be activated by the truck. An optimal solution of the TSDSLP indicates which drone stations should be activated, assigns customers to a truck or drone delivery, and sequences the deliveries such that the mission time is minimized. Schermer et al. (2019c) presented a MILP formulation that can be used to solve small-sized problem instances. Experiments show that using the TSDSLP can significantly reduce delivery times compared to the classic TSP. In the next sections, we introduce and investigate a natural extension to the TSDSLP, where we consider more than one truck.

3. The multiple traveling salesman problem with drone stations

In this section, we introduce the *multiple Traveling Salesman Problem with Drone Stations* (mTSP-DS) that is a generalization of the TSDSLP and combines elements of the mTSP as well as of the *facility location* and the *parallel machine scheduling* problems. In Section 3.1, we describe the problem and, afterwards, we present a formal description as well as a MILP formulation of the mTSP-DS in Section 3.2.

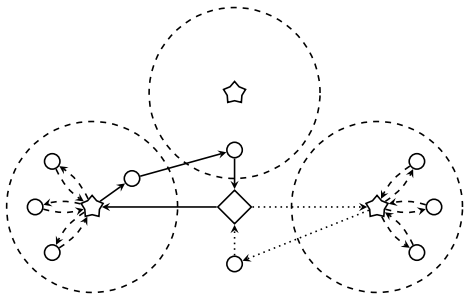


Figure 1 A sample mTSP-DS solution with 2 trucks (solid and dotted arrows) and 3 potential drone stations. The square, circle, and star shapes represent the depot, customers, and drone stations, respectively. The dashed circles around the drone stations show the operation range of drones, and the dashed arrows depict the drone deliveries.

3.1. Problem definition

Assume that a single depot, a set of customers, and a fleet of trucks are given. Moreover, we suppose that a set of drone stations, each of them hosting a certain number of drones (or robots), is available. In the mTSP-DS, each truck must make a tour that starts from the depot, visits some customers, and ends at the depot again. A truck can visit a number of available drone stations to hand over some parcels. In such a case, the visited drone stations are considered as *activated*, i.e., the activated drone stations can load and launch a set of drones located at the drone stations to serve customers, which are in operation range of drones, through round trips starting and ending at the same drone station. Moreover, each drone may serve multiple customers through different round trips. The goal of the mTSP-DS is to serve all customers either by trucks or by drones while minimizing the overall *makespan*. In this context, the makespan is defined as the *latest* arrival time of a truck at the depot or of a drone at a drone station.

To formulate the mTSP-DS, we make the following assumptions:

- Every customer must be visited exactly once, either by a truck or by a drone.
- A predefined number of drone stations can be activated. Drone stations may also not be visited more than once.
- Once a truck or drone has arrived at a customer location, we assume that the time required to handover the parcel is negligible.
- Each drone station operates the same number of drones of the same type and size.
- Drones may not exceed their range of operation during tours. On return, their battery and cargo will be replaced instantly.
- Our formulation takes no kind of costs into account. In particular, the operation of drone stations does not incur any costs.

Figure 1 provides a visualization of the mTSP-DS, where three drone stations are available but only two drone stations are activated by two trucks (one drone station per truck). Indeed, which drone stations to activate by which truck is one of the questions that is answered by solving the MILP formulation presented in the next section.

3.2. MILP formulation of the mTSP-DS

To formulate the mTSP-DS, assume that a complete graph $\mathcal{G} = (V, E)$ is given, where the set of nodes V consists of n customers $V_N = \{1, ..., n\}$, m drone stations $V_S = \{s_1, ..., s_m\}$, and the depot, which is represented by nodes 0 and $n+1$ ($0 \equiv n+1$). The depot is referred to as node 0 when it represents the start of a tour, and node $n + 1$ is used when it represents the end of a tour. The set of nodes is thus defined by $V = \{0\} \cup V_N \cup V_S \cup \{n+1\}$. For the sake of simplifying the notation, we additionally define the sets V_L and V_R as $V_L = V \setminus \{n + 1\}$ and $V_R = V \setminus \{0\}$. Moreover, the mathematical model of the mTSP-DS depends on the following parameters:

- $C \in \mathbb{Z}^+$ specifies the maximum number of drone stations that can be activated. It is clear that C should be less than or equal to the number of drone stations m.
- $D_N \in \mathbb{Z}^+$ sets the number of available drones per drone station. We define $D = \{1, ..., D_N\}$ as the set of all drones hosted in a drone station.
- $\alpha \in \mathbb{R}^+$ is a factor to indicate the drone velocity relative to the truck speed. More precisely, $\alpha > 1$ shows that the drones are faster than the trucks, while $\alpha < 1$ means that the drones are slower, and $\alpha = 1$ is used when drones and trucks have the same velocity. The case of $\alpha < 1$ typically refers to *autonomous ground vehicles* (AGVs) or robots, which move at a slower speed than conventional trucks. It is usually assumed that the trucks can move at a unit velocity of $v = 1$, and the drones' velocity is defined as $\overline{v} = \alpha \cdot v$ (see, e.g., Wang et al., 2017).
- The maximum distance that can be covered by the drones without recharging is determined by $\mathcal{E} \in \mathbb{R}^+$. This parameter indicates that the range of the drone stations can be derived as $\mathcal{E}_r = \mathcal{E}/2$.
- The total number of trucks is given by $K_N \in \mathbb{Z}^+$ and the set of trucks is defined by $K = \{1, ..., K_N\}$, $|K| = K_N$. If $K_N = 1$, the mTSP-DS becomes equivalent to the special case of the TSDSLP (Schermer et al., 2019c).

The distance that trucks need to traverse between nodes i and j is specified by d_{ij}, which is considered as the Euclidean distance between i and j (where $i, j \in V$). Analogously, $\overline{d}_{ij}$ is defined as the distance that drones need to cover between nodes i and j, for all $i, j \in V$. For trucks, the time required to travel from node i to node j is defined as $t_{ij} = d_{ij}$. Regarding drones, this time is defined as $\overline{t}_{ij} = \overline{d}_{ij}/\alpha$. Furthermore, we assume that the distances are symmetric.

Finally, we need the following decision variables to formulate the mTSP-DS:

$\tau \in \mathbb{R}_{\geq 0}$	:	A continuous variable that represents the makespan.
$\underset{\forall k \in K, i \in V_L, j \in V_R}{x_{ij}^k} \in \{0, 1\}$	:	Variables that are used to indicate whether truck k traverses edge (i,j).
$\underset{\forall d \in D, s \in V_S, j \in V_N}{y_{sj}^d} \in \{0, 1\}$	:	Variables that specify if customer j is served by drone d from drone station s.
$\underset{\forall s \in V_S}{z_s} \in \{0, 1\}$	:	Variables that state whether a drone station is activated.
$\underset{\forall k \in K, i \in V}{a_i^k} \in \mathbb{R}_{\geq 0}$	:	Continuous variables that indicate the time at which truck k arrives at node i.

The MILP formulation of the mTSP-DS is given by (1) - (13). In the following, we present the objective function as well as the constraints, and describe them step-by-step.

$$\min \quad \tau \tag{1}$$

$$\text{s.t.} \quad a_{n+1}^k \leq \tau : \forall k \in K, \tag{2}$$

$$a_s^k + \sum_{j \in V_N} 2 \cdot \bar{t}_{sj} \cdot y_{sj}^d \leq \tau : \forall k \in K, s \in V_s, d \in D, \tag{3}$$

$$\sum_{k \in K} \sum_{\substack{i \in V_L \\ i \neq j}} x_{ij}^k + \sum_{s \in V_s} \sum_{d \in D} y_{sj}^d = 1 : \forall j \in V_N, \tag{4}$$

$$\sum_{j \in V_N} x_{0j}^k = \sum_{i \in V_N} x_{i,n+1}^k = 1 : \forall k \in K, \tag{5}$$

$$\sum_{\substack{i \in V_L \\ i \neq h}} x_{ih}^k - \sum_{\substack{j \in V_R \\ h \neq j}} x_{hj}^k = 0 : \forall k \in K, h \in V_N \cup V_S, \tag{6}$$

$$\sum_{k \in K} \sum_{\substack{i \in V_L \\ i \neq s}} x_{is}^k \leq 1 : \forall s \in V_S, \tag{7}$$

$$\sum_{k \in K} \sum_{\substack{i \in V_L \\ i \neq s}} x_{is}^k = z_s : \forall s \in V_S, \tag{8}$$

$$\sum_{s \in V_S} z_s \leq C, \tag{9}$$

$$\sum_{d \in D} \sum_{j \in V_N} y_{sj}^d \leq n z_s : \forall s \in V_S, \tag{10}$$

$$2 \cdot \bar{d}_{sj} \cdot y_{sj}^d \leq \mathcal{E} : \forall s \in V_S, d \in D, j \in V_N, \tag{11}$$

$$M(x_{ij}^k - 1) + a_i^k + t_{ij} \leq a_j^k : \forall k \in K, i \in V_L, j \in V_R, i \neq j, \tag{12}$$

$$\sum_{i \in \mathcal{S}} \sum_{\substack{j \in \mathcal{S} \\ i \neq j}} x_{ij}^k \leq |\mathcal{S}| - 1 : \forall k \in K, \mathcal{S} \subset V, \{0, n+1\} \notin \mathcal{S}, |\mathcal{S}| > 1. \tag{13}$$

The objective function of the mTSP-DS is to minimize the makespan τ. Constraints (2) and (3) define τ through lower bounds. More precisely, constraints (2) set the arrival time of each truck as a lower bound for τ. This ensures that τ is at least as large as the last arrival of a truck at the depot. Further lower bounds are provided by the arrival times of the drones to their drone station. To calculate these bounds, constraints (3) take the activation times of the drone stations. The activation times are added to the times that the individual drones in the corresponding drone station need for deliveries. If customer j is served by drone d from drone station s, the time for this operation is determined by $2 \cdot \bar{t}_{sj}$, since the way back also has to be taken into account.

Constraints (4) ensure that each customer is served either by a truck or by a drone exactly once. The vehicle flow conditions are defined through constraints (5) - (7). More precisely, constraints (5) state that each vehicle has to start and end its tour at the

depot. In addition, flow-conservation constraints (6) enforce that, for all nodes except the depot, the number of incoming flows is equal to the number of outgoing flows. Finally, constraints (7) guarantee that each drone station is visited at most once.

We use constraints (8) and (9) to handle the activation of drone stations. Constraints (8) state that drone stations cannot be visited without activating them, and constraint (9) ensures that at most C drone stations are activated. Drone operations are controlled by constraints (10) and (11). Indeed, constraints (10) limit the number of drone operations, that can be performed by an activated drone station, to the number of customers. Alternatively, the actual number of customers within the range of the drone station could be set as a limit. Constraints (11) determine which customers are eligible for drone deliveries. For this purpose, $\mathcal{E}$ sets an upper bound on the distance traveled by a drone, i.e., from the drone station to a customer location and the way back, before that any recharge becomes required.

The time at which a truck arrives at a node j is set by constraints (12), where M is a sufficiently large number. More precisely, if the edge (i, j) is a part of truck k's tour, a_j^k is determined by the time that truck k needs to get to node i plus the time the truck needs to travel from i to j. Otherwise, i.e., if (i, j) is not traversed by truck k, inequality (12) becomes redundant. If all nodes are located at unique coordinates, constraints (12) are sufficient to eliminate subtours. In other cases, however, we additionally need constraints (13): These constraints ensure that the number of selected edges of each non-empty subset $\mathcal{S} \subset V$, not containing the depot, does not exceed $|\mathcal{S}| - 1$. The exponential number of constraints (13) require that, while solving the model (1) - (13) by a MILP solver, we implement constraints (13) as *lazy constraints* using a callback function. More precisely, each time the MILP solver finds a solution that satisfies all other constraints, all truck tours are checked for subtours. If a subtour is found, only the corresponding violated constraint is added to the model, and the model is solved again.

We can solve the MILP model (1)-(13) to obtain valid mTSP-DS solutions. For this purpose, any standard MILP solver, e.g., Gurobi Optimizer or IBM Cplex, can be used. Nevertheless, with the intention of enhancing the solver's performance, we can use *valid inequalities*, which are a way to shrink the solution space without excluding optimal solutions (Schermer et al., 2018b). Schermer et al. (2018b) have introduced valid inequality (14) for the VRPD; however, it can also be employed for the mTSP-DS. In fact, for each truck $k \in K$, the valid inequality (14) sets the travel time of the truck as a lower bound on the earliest arrival time of this truck at the depot. Despite the simplicity of these valid inequalities, the numerical results show that they are quite effective. Hence, we used them in our computational experiments.

$$\sum_{i \in V_L} \sum_{\substack{j \in V_R \\ i \neq j}} x_{ij}^k t_{ij} \leq a_{n+1}^k : \forall k \in K. \tag{14}$$

Nevertheless, even with these valid inequalities (14), the mTSP-DS remains a computationally complex problem, and we can only solve small instances using MILP solvers. This argument has been supported numerically through our computational experiments. Hence, in order to address large mTSP-DS instances, we designed *matheuristic* algorithms that we present in the next section.

4. Matheuristics for solving the mTSP-DS

In the literature, *matheuristics* are defined as algorithms that are results of interoperation between heuristics and mathematical programming techniques (Boschetti et al., 2009). Based on this definition, Archetti & Speranza (2014) identify three classes of matheuristics: *decomposition approaches*, *improvement heuristics*, and *branch-and-price/column generation-based approaches*. In the context of tour planning, decomposition approaches have been successfully applied to a variety of routing problems (Archetti & Speranza, 2014; Dell'Amico et al., 2020; Schermer et al., 2019b).

In this section, we present three matheuristic algorithms to solve the mTSP-DS. More precisely, in Section 4.1, we introduce two variants of a decomposition-based matheuristic. Afterwards, in Section 4.2, we present a two-phase algorithm inspired by Accorsi & Vigo (2020), which first populates a pool of solutions using a metaheuristic and then solves a set-partitioning formulation to determine the best combination of tours from the solutions in the pool.

4.1. A decomposition-based matheuristic

The main idea of the algorithms presented in this section is to decompose the mTSP-DS into smaller and easier-to-handle subproblems. Indeed, the mTSP-DS is composed of the following interconnected subproblems:

- *Drone Station Location Problem*: Which drone stations should be activated?
- *Allocation Problem*: Which customers should be served by which truck and which selected drone stations should be activated by which truck?
- *Sequencing Problem*: In what order should the customers of each allocation be served? When should the drone stations be activated?
- *Assignment Problem*: Which customers within the range of an activated drone station should be served by truck and which one(s) by drone(s)?
- *Scheduling Problem*: By which drone should the selected customers be served? This subproblem corresponds to the parallel machine scheduling problem.

Algorithm 1: mTSP-DS Matheuristic

```
1  init solution_pool
2  d_station_combos = get_all_d_station_combos(V_S, C)
3  foreach d_station_combo ∈ d_station_combos do
4      solution = get_mtsp_tours(depot, V_N, d_station_combo)
5      foreach d_station ∈ d_station_combo do
6          solution = solve_local_dasp(solution, d_station)
7      solution_pool.add(solution)
8  best_solution = get_best_solution(solution_pool)
9  best_solution = post_processing(best_solution)
10 return best_solution
```

These subproblems are addressed in Algorithm 1, where lines 2 and 3 represent the *drone station location* subproblem. In this algorithm, the problem is solved by checking all possible drone station combinations. Consequently, the function *get_all_d_station_combos* returns a list of $\binom{V_S}{C}$ possibilities. Afterwards, all of the subsequent subproblems are solved for all these possible combinations.

The *allocation* and the *sequencing* problems can be grouped under the term *routing problem*. In Algorithm 1, this routing problem is solved by the *get_mtsp_tours* function, which is in line 4. The function takes the depot, the set of customers V_N, and the current drone station combination as input, and returns a set of K_N vehicle tours. Any mTSP heuristic or a MILP solver can be used to implement this function. In our experiments, we used the Lin-Kernighan-Helsgaun mTSP solver (Helsgaun, 2017).

The key element of this algorithm, the *Drone Assignment and Scheduling Problem* (DASP), is primarily a combination of the *assignment* and the *scheduling* problem (for details, refer to Section 4.1.1). But depending on the implementation, it can also resolve parts of the routing problem. Since we solve the DASP for each drone station separately, we refer to this problem as the *local DASP*. Indeed, the function *solve_local_dasp* in line 6 gets a solution, which consists of standard mTSP tours in the first iteration of the loop in lines 5-6, and the current drone station as input. Afterwards, the function decides which of the customer(s) within the range of the drone station should be supplied by truck and which one(s) by drone. Then, the function returns an updated set of vehicle tours and the scheduled drone deliveries for the current drone station. We reuse this updated solution as input for the *solve_local_dasp* function in subsequent iterations of the loop in lines 5-6. In the forthcoming subsections, we will introduce two MILP formulations that can be used to solve the local DASP.

As soon as the local DASP has been solved for all locations in a drone station combination, the solution is stored in a *solution pool*. In line 8 of Algorithm 1, the solution with the minimal objective value is chosen from this pool. In some cases, truck tours that have been modified by the local DASP may be much shorter than before, which can lead to unbalanced tours. To rebalance them, and to possibly improve the solution quality, we use a *post_processing* procedure, which is in line 9 of Algorithm 1. During this *post_processing*, we use inter- and intra-route operations (Toth & Vigo, 2014) that we repeat until no further improvement can be achieved:

- *inter-route* operation: We remove a customer from the longest tour and insert it into the shortest one. The savings of all possible customer relocations are calculated, and then the best one is performed.
- *intra-route* operation: Each time a customer relocation is performed, 2-opt is applied to both involved tours, i.e., the subtour between two nodes of a tour is reverted, if it reduces the tour's length.

Since the mTSP-DS is an asymmetric problem, reversing a truck tour might improve the makespan. If this is the case, the corresponding tour is reversed to be saved as a better solution. Finally, the algorithm returns the post-processed solution.

4.1.1. Local DASP

Firstly, the goal of the local DASP is to decide which customers within the range of a single drone station should be served by trucks and which ones by drones. Secondly,

the local DASP schedules the drone deliveries. In this section, we present a MILP formulation to solve this problem in the best possible way, i.e., by mean of an exact solution of the MILP. In the following, we start by presenting additional concepts and notation that we are going to use in the description of the local DASP.

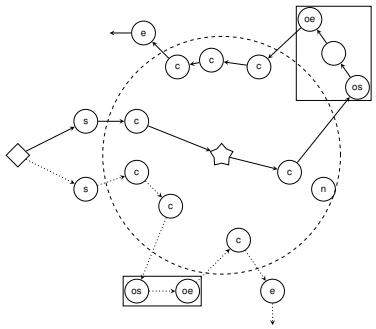


Figure 2 In the local DASP, we only consider customers in the immediate surrounding of the drone station (the star-shaped node). Trucks depart from start nodes (*s*) and end their tours at end nodes (*e*). Nodes labeled with the letter *c* can be served either by truck or by drone. The rectangular boxes represent outlier groups. The node labeled with the letter *n* is already assigned to another drone station and is thus not considered.

The *solve_local_dasp* function in line 6 of Algorithm 1 receives a solution and a drone station as argument. Moreover, the local DASP only considers trucks that visit at least one node within the range of the passed drone station and only nodes in the immediate surrounding of the drone station. In case of overlapping drone ranges, i.e., if a node can potentially be served from more than one drone station, the local DASP does not involve the nodes that are already assigned to drones in other drone stations. Figure 2 gives an illustration of the situation. In the figure, we observe segments of two truck tours that are in the vicinity of a drone station. Assume that the set of trucks that visit at least one node in the vicinity of the drone station is defined as $\overline{K} = \{1, ..., k\}$. In Figure 2, the two nodes that are labeled with an *s* lead the trucks into the range of the drone station. Therefore, in the following, we call these two nodes *start nodes*. We define the set of such start nodes as $V_{start} = \{start_1, ..., start_k\}$, $|V_{start}| = k$, i.e., each considered truck has a start node, which leads the truck into the range of the drone station. We note that even the depot can be a start node. In this case, several vehicles can start from the same node, but in the set V_{start}, we nevertheless reference them as individual start nodes. The nodes labeled with the letter *c* within the range of the drone station indicate the customers that can be served either by vehicles or by drones. In the local DASP, those customers constitute the set of customers V_N and the single drone station is referred to as node ds. The nodes labeled with an *e* represent the nodes at which the trucks finish their tours in the local DASP; hence, those nodes are called *end nodes*. We define $V_{end} = \{end_1, ..., end_k\}$, $|V_{end}| = k$ as the set of end nodes.

While a truck is driving through the perimeter of a drone station, the truck may serve customers outside the range of the drone station, and then enter the area within range again. In the following, these nodes out of range are referred to as *outliers*. Each

truck can have none, one, or several *groups* of outliers in its tour, and each *outlier group* consists of one or more outliers. In Figure 2, the two rectangular boxes represent two sample outlier groups. The first node of each outlier group is called *outlier-start* (os) node. The set of outlier-start nodes is denoted by $V_{OS} = \{os_1, ..., os_o\}$, where o is the number of outlier groups. Similarly, the set of the last nodes of all outlier groups, the set of *outlier-end* (oe) nodes, is defined as $V_{OE} = \{oe_1, ..., oe_o\}$. If an outlier group consists of only one node, this node is simultaneously an outlier-start and an outlier-end node. Now, consider the outlier-start and outlier-end node pairs of each outlier group; then, the set of outlier groups can be defined as $O = \{(os_1, oe_1), ..., (os_o, oe_o)\}$, where each pair (os_i, oe_i) can refer to at least one outlier node.

If an outlier group consists of more than two nodes, then the nodes enclosed by the outlier-start and outlier-end are not directly considered in the local DASP that we present. Hence, the model will not modify parts of the truck tours that correspond to outlier groups. However, we take into account the time that trucks need to traverse outlier groups. More precisely, $travel_cost(os, oe)$ represents the time required by a truck to travel from the outlier-start node os to the outlier-end node oe, including the nodes in between. Moreover, the nodes before start nodes and after end nodes are ignored too. We only consider the time that truck k needs to get from the depot to the start node ($cost_to_start_k$) and from the end node back to the depot ($cost_after_end_k$), again including the nodes in between.

We use the set $V_L = V_{start} \cup V_N \cup \{ds\} \cup V_{OE}$ as the nodes that can be the first nodes of an arc. Similarly, $V_R = V_N \cup \{ds\} \cup V_{OS} \cup V_{end}$ is defined as the set of nodes that can be the second nodes of an arc. This means that in the following MILP formulation, the trucks can only traverse the outlier groups in their original order of nodes. Finally, the set of all nodes is defined as $V = V_{start} \cup V_N \cup \{ds\} \cup V_{OS} \cup V_{OE} \cup V_{end}$.

In the MILP formulation of the local DASP, we use the following decision variables:

$$\overline{\tau} \in \mathbb{R}_{\geq 0} \quad : \quad \text{A continuous variable representing the makespan.}$$

$$\underset{\substack{\forall k \in K, i \in V_L, j \in V_R \\ \wedge \forall k \in K, (os,oe) \in O}}{x_{ij}^k} \in \{0,1\} \quad : \quad \text{Variables that indicate whether truck } k \text{ traverses arc } (i,j).$$

$$\underset{\forall d \in D, j \in V_N}{y_j^d} \in \{0,1\} \quad : \quad \text{Variables that specify if customer } j \text{ is served by drone } d \text{ from drone station } ds.$$

$$\underset{\forall k \in K, i \in V}{a_i^k} \in \mathbb{R}_{\geq 0} \quad : \quad \text{Continuous variables that indicate the time at which truck } k \text{ arrives at node } i.$$

$$\min \quad \overline{\tau} \tag{15}$$

$$\text{s.t.} \quad a_{end_k}^k \leq \overline{\tau} : \forall k \in K, \tag{16}$$

$$a_{ds}^k + \sum_{j \in V_N} 2 \cdot \bar{t}_{ds,j} \cdot y_j^d \leq \overline{\tau} : \forall k \in K, d \in D, \tag{17}$$

$$\sum_{k \in K} \sum_{\substack{i \in V_L \\ i \neq j}} x_{ij}^k + \sum_{d \in D} y_j^d = 1 : \forall j \in V_N, \tag{18}$$

$$\sum_{\substack{j \in V_N \cup ds \\ \cup V_{OS} \cup end_k}} x^k_{start_k,j} = \sum_{\substack{i \in V_N \cup ds \\ \cup V_{OE} \cup start_k}} x^k_{i,end_k} = 1 : \forall k \in K, \tag{19}$$

$$\sum_{k \in K} \sum_{\substack{i \in V_L \\ i \neq ds}} x^k_{i,ds} = \sum_{k \in K} \sum_{\substack{j \in V_R \\ j \neq ds}} x^k_{ds,j} = 1, \tag{20}$$

$$\sum_{k \in K} \sum_{\substack{i \in V_L \\ i \neq oe}} x^k_{i,os} = 1 : \forall (os, oe) \in O, \tag{21}$$

$$\sum_{k \in K} \sum_{\substack{j \in V_R \\ j \neq os}} x^k_{oe,j} = 1 : \forall (os, oe) \in O, \tag{22}$$

$$\sum_{i \in V_L} x^k_{i,os} - x^k_{os,oe} = 0 : \forall k \in K, (os, oe) \in O, \tag{23}$$

$$\sum_{\substack{i \in V_L \\ i \neq h}} x^k_{ih} - \sum_{\substack{j \in V_R \\ j \neq h}} x^k_{hj} = 0 : \forall k \in K, h \in V_N, \tag{24}$$

$$cost_to_start_k = a^k_{start_k} : \forall k \in K, \tag{25}$$

$$M(x^k_{ij} - 1) + a^k_i + t_{ij} \leq \begin{array}{r} a^k_j : \forall k \in K, i \in V_L, \\ j \in V_N \cup ds \cup V_{OS}, i \neq j, \end{array} \tag{26}$$

$$\begin{array}{r} M(x^k_{os,oe} - 1) + a^k_{os} + \\ traversal_cost(os, oe) \end{array} \leq a^k_{oe} : \forall k \in K, (os, oe) \in O, \tag{27}$$

$$\begin{array}{r} M(x^k_{i,end_k} - 1) + a^k_i + t_{i,end_k} + \\ cost_after_end_k \end{array} \leq a^k_{end_k} : \forall k \in K, i \in V_L, \tag{28}$$

$$\sum_{i \in \mathcal{S}} \sum_{\substack{j \in \mathcal{S} \\ i \neq j}} x^k_{ij} \leq \begin{array}{l} |\mathcal{S}| - 1 : \forall k \in K, |\mathcal{S}| > 1, \\ \mathcal{S} \subset (V \setminus (V_{start} \cup V_{end})). \end{array} \tag{29}$$

The MILP formulation for the local DASP is given by (15) - (29). The objective function and some of the constraints of this formulation are the same as they are in the mTSP-DS. However, some of the constraints are specific for the local DASP.

The objective of the local DASP consists in minimizing the makespan $\overline{\tau}$. The makespan $\overline{\tau}$ is defined through lower bounds provided by constraints (16) and (17). In constraints (16), lower bounds for $\overline{\tau}$ are set through the arrival time of each truck at its end node. As shown in constraints (17), the activation time of the drone station plus the delivery times of the drones result in additional lower bounds on $\overline{\tau}$.

Constraints (18) ensure that each customer within the range of the drone station is served only once, either by a drone or by a truck. Constraints (19) guarantee that each vehicle starts its tour at its start node and ends its tour at its end node. From its start node, a truck can visit either a customer, the drone station, the start of an outlier group, or its end node. Similarly, a truck can reach its end node from the customers, the drone station, the outlier-ends, or its start node.

We use constraints (20) to enforce that drone station ds has exactly one incoming

and one outgoing arc. Furthermore, each outlier-start node must have exactly one incoming arc, and each outlier-end node must have exactly one outgoing arc. These conditions are defined by constraints (21) and (22). Constraints (21) and (22) also allow that a truck travels from an outlier-end node to an outlier-start node of another group. Constraints (23) ensure that if a truck visits an outlier-start node, then the truck must also visit the outlier-end node of the same outlier group. For all customers that can be served by drone, the vehicle flow is preserved by constraints (24), meaning that each of those customer nodes must have the same amount of incoming and outgoing arcs.

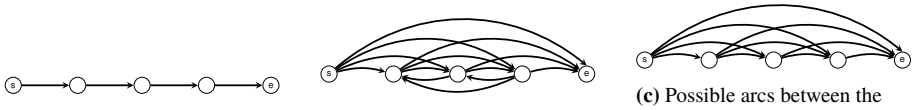
The earliest arrival of the trucks at the nodes is computed by constraints (25) - (28). More precisely, constraints (25) are used to determine the arrival time instants of trucks at their start nodes. Constraints (26) define the earliest arrival times at the customer nodes V_N, the drone station ds, and the outlier start nodes. For each node j in this set of nodes, the arrival time is set by the arrival time at the predecessor i plus the travel time from node i to j. Constraints (27) set the earliest arrival times at outlier-end nodes. For each outlier-end node oe, the earliest arrival time is calculated by the sum of the arrival time at the outlier-start node os belonging to the same outlier group and the $traversal_cost(os, oe)$. Finally, the arrival time of a truck at its end node is determined by constraints (28), where the arrival time at an end node comprises three components: the time it takes to reach the predecessor of the end node, the time needed to get from the predecessor to the end node, and the pre-computed constant *cost_after_end_k*. As in the MILP formulation of the mTSP-DS, subtours are also eliminated in this model through *lazy constraints* (see constraints (29)) if multiple nodes have identical coordinates. That is, whenever the MILP solver encounters an incumbent solution, all truck tours of this solution will be checked for subtours. If a solution contains a subtour, a corresponding subtour elimination constraint will be added to the model and it will be solved again.

To enhance the performance of the MILP solver, valid inequalities (14) can also be employed without adjustment. In addition, the original truck tours of the input solution are passed as an initial solution to the MILP solver. This is possible because the partial mTSP-DS solution (see Figure 2), which is passed to the local DASP, is also a valid local DASP solution. In fact, this implicitly creates an upper bound for the local DASP, i.e., this upper bound is equal to the longest truck tour of the input solution that visits at least one customer within the range of the drone station or the drone station itself.

4.1.2. Local DASP with limited truck rerouting

The time required to solve the MILP model of the local DASP, which was introduced in the previous subsection, increases quite fast as the number of customers within the range of drones increases. Therefore, this section outlines a modification to the local DASP that attempts to reduce the runtime for solving the model. The core idea is to limit the capabilities of the model to reroute truck tours. Hence, we call this approach the *local DASP with limited truck rerouting* (DASP-LTR). The main difference to the local DASP is that the truck tours, which lead through the delivery area of the drone station, are treated in isolation from each other. More precisely, in the local DASP-LTR, trucks are only allowed to visit nodes that they also visit in the input solution.

Another restriction of the model concerns the possible arcs between the nodes. To simplify the description of the situation, we use Figure 3. Assume that the arrows in Figure 3a represent a truck tour in the input solution. In the local DASP, each of the



(a) A sample truck tour passed to the local DASP.

(b) Possible arcs between the nodes in the local DASP.

(c) Possible arcs between the nodes in the local DASP-LTR.

Figure 3 Illustration of the arcs between the nodes in the local DASP versus DASP-LTR. In the local DASP-LTR, the truck cannot be rerouted because the nodes are only connected to their successor nodes.

customer nodes in the set V_N can be connected to all other nodes in this set, which is illustrated in Figure 3b. However, as we see in Figure 3c, the local DASP-LTR concerns a smaller set of possible connections. In fact, in the local DASP-LTR, each node in the set V_N can only be connected to its succeeding nodes in the corresponding input solution truck tour. As a result, there will always be only one valid truck tour, regardless of how many customers from the set V_N are served by a drone.

Restricting the possible connections between nodes reduces the number of variables in the model significantly. Consequently, the search space becomes smaller, and in comparison to the local DASP, we can solve the DASP-LTR in a shorter computation time. However, due to the imposed restrictions, the solutions obtained through using the DASP-LTR might have lower quality.

4.2. A two-phase matheuristic

The third approach, which we present in this paper, is a two-phase algorithm that starts by generating a solution pool using several restarts of a metaheuristic. Then, in a polishing phase, a set-partitioning model is solved to determine the best combination of tours from the solution pool. The basic idea of this approach and its effectiveness on several variations of the VRP was presented by Subramanian et al. (2013). More recently, Accorsi & Vigo (2020) presented a variant of this approach for the class of truck and trailer routing problems. Algorithm 2 provides an overview of the approach.

Phase 1 of the algorithm consists of two steps that are repeated Γ times. The first step is the construction of an initial solution. To this end, the algorithm selects C drone stations randomly. All customers who are in the range of one of the selected drone stations are assigned to be served by the drones in this drone station. Customers who are in the range of multiple drone stations are randomly assigned to one of them. The remaining customers and the selected drone stations are assigned to the different vehicles using the *sweep* algorithm (Gillett & Miller, 1974). At each iteration, the sweep algorithm starts the assignment from a randomly selected node. After that, the algorithm computes a TSP tour for each vehicle using the Lin-Kernighan-Helsgaun TSP solver (Helsgaun, 2000). However, any other mTSP algorithm could be used instead. Indeed, our preliminary experiments showed that the quality of the initial solutions does not have a big impact on the quality of the final solutions.

To complete the construction of the initial solution, the drone deliveries have to be scheduled for all selected drone stations. Moreover, in the further course of our matheuristic, drone deliveries must be rescheduled whenever the drone station assignments are changed. We preliminary tested *LPT* (Cheng & Sin, 1990), *MULTIFIT*

Algorithm 2: Two-phase matheuristic

```
   input: Γ, π_base, δ, φ, λ
 1 Ω = init_solution_pool()
 2 Ψ = init_drone_delivery_cache()
 3 S* = empty_solution()
 4 for r = 1 to Γ do                          // Phase 1; Γ: Number of restarts
 5 |   S = construct_initial_solution(Ψ)
 6 |   S = improve_solution(S, Ω, Ψ, π_base, δ, φ, λ)
 7 |   if cost(S) < cost(S*) then
 8 |   |   S* = S
 9 |   end
10 end
11 S* = polish(S*, Ω)                                          // Phase 2
12 return S*
```

(Coffman et al., 1978), *COMBINE* (Lee & Massey, 1988), and a *binary programming* model to schedule the drone deliveries, and decided to use the COMBINE algorithm since it provides a good trade-off between solution quality and runtime. As a combination of LPT and MULTIFIT, the COMBINE algorithm uses LPT to find a scheduling to feed, as a tighter initial upper bound, into the MULTIFIT algorithm. To enhance the runtime performance of our approach, we save the resulting drone delivery schedules in a global cache Ψ. Thus, we can reuse them if we encounter an already scheduled combination of a drone station and a certain set of customers during the search.

In the second step of phase 1, we improve the initial solution by applying a metaheuristic based on the *iterated local search* (ILS) framework (Lourenço et al., 2003) until the termination criterion of δ nonimproving iterations is met. Pseudocode for the *improve_solution* function in line 6 of Algorithm 2 is given in Algorithm 3.

The *rvnd_local_search* function in line 4 of Algorithm 3 improves the solution using *variable neighborhood descent with random neighborhood ordering* (RVND), which searches through a predefined set of local search neighborhoods in a randomized order (Penna et al., 2013). We use the following neighborhoods in the RVND:

- *Intra-route and inter-route relocate:* Removes a node from a truck tour and reinserts it into one of the truck tours.
- *Intra-route and inter-route swap:* Swaps a pair of nodes in the truck tours.
- *Multirange drone relocate:* Assigns a customer who is in range of multiple drone stations to a different drone station.
- *Drone-to-truck relocate:* Removes a customer node that is scheduled to be served by a drone and inserts it into a truck tour.
- *Truck-to-drone relocate:* Removes a customer node from a truck tour and assigns it to a drone station.

During the local search, we evaluate the neighborhoods, as well as the moves within each neighborhood, in a randomized order following a first-improvement strategy. Once an improving move is found and performed, we restart the search in the current neighborhood, and if no more improving moves can be found in the current neighborhood,

Algorithm 3: improve_solution

input: solution S, Ω, Ψ, π_{base},δ, ϕ, λ

1 $\pi = \pi_{base}$ // Initialize the sparsification factor π with the // value of the initial sparsification factor π_{base}
2 $i = 0$, $S' = S$
3 **while** *true* **do**
4 S = rvnd_local_search(S,Ω, Ψ, π)
5 **if** *cost(S)* < *cost(S′)* **then**
6 $i = 0$, $S' = S$, $\pi = \pi_{base}$
7 **end**
8 $i = i + 1$
9 **if** $i \geq \delta$ **then** // Terminate after δ nonimproving iterations
10 break
11 **end**
12 **if** $(i \mod (\phi \cdot \delta)) == 0$ **then**
13 $\pi = \lambda \cdot \pi$ // Increase π after $\phi \cdot \delta$ nonimproving iterations
14 **end**
15 $S = S'$
16 S = shake(S)
17 **end**
18 **return** S'

we add the solution to the solution pool. Furthermore, we repeat the local search until a complete RVND run is executed without improvement.

After applying a move that changes the customer assignments of a drone station (multirange drone relocate, drone-to-truck relocate, or truck-to-drone relocate), we check if we already encountered the resulting customer assignment for this drone station. If yes, we reuse the respective drone delivery schedule from the global cache. Otherwise, we will schedule the drone deliveries using the COMBINE algorithm and store the schedule in the global cache.

To further speed up computations, we use *granular neighborhoods* for the relocate, swap, and drone-to-truck relocate operators (Toth & Vigo, 2003). A move is thereby only evaluated if the current granular neighborhood contains at least one of its resulting arcs. The granular neighborhood is defined by the $\pi \cdot (|V_N| + 1)$ shortest truck arcs, where π is the sparsification factor, and V_N is the set of customer nodes.

Furthermore, we update the size of the granular neighborhood dynamically (Accorsi & Vigo, 2020; Goeke, 2019). Indeed, we start with the initial sparsification factor π_{base}, and use the parameter ϕ ($0 < \phi < 1$) to determine the number of nonimproving iterations, after which the size of the granular neighborhood is increased. More precisely, after each $\phi \cdot \delta$ nonimproving iterations, we increase the size of the granular neighborhood by multiplying π with the factor λ (lines 12-14 of Algorithm 3). If the local search is able to improve the current restart's best solution, we reset π to π_{base}.

After the local search, in line 16 of Algorithm 3, we apply the *shake* function to the current restart's best solution. The shake function randomly deletes 50% of the customer nodes in the solution and reinserts them into the position that minimizes the insertion

cost. During the reinsertion, we consider truck tours as well as drone deliveries. Finally, the *improve_solution* function terminates after δ nonimproving iterations and returns the best-found solution.

We repeat the creation of initial solutions and the improvement of those solutions Γ times. During this first phase of the two-phase algorithm, we not only populate the solution pool, but we also keep track of the best solution encountered so far. The solution pool and the best-found solution will then be used in the subsequent polishing phase (line 11 of Algorithm 2).

The goal of the polishing phase is to find the best combination of truck tours and drone deliveries from the solution pool by solving a set-partitioning MILP model. To do this, we first need to prepare the set of routes $\mathcal{R}$ and to determine the cost for each route. For this purpose, we generate a set of nodes $\mathcal{X}_r$ for each truck tour r of a solution in the solution pool. We first add all nodes, except the depot, visited in truck tour r to the set $\mathcal{X}_r$. If truck tour r contains one or more drone stations, we also add the nodes served by drones from those drone stations to the set $\mathcal{X}_r$. We then define the set of routes as $\mathcal{R} = \{\mathcal{X}_1, ..., \mathcal{X}_{K_N \cdot |\Omega|}\}$, where K_N is the number of trucks and $|\Omega|$ is the number of solutions in the solution pool.

If a route $r \in \mathcal{R}$ contains one or more drone stations, the cost c_r of this route is equal to the maximum of the truck tour travel time and the drone delivery times of the activated drone stations in the corresponding solution. Otherwise, c_r is equal to the truck tour travel time. Furthermore, we define a 0-1 matrix whose entries a_{ir}, for $i \in V_N \cup V_S$ and $r \in \mathcal{R}$ are defined as follows: $a_{ir} = 1$ if and only if node $i \in V_N \cup V_S$ is contained in the route $r \in \mathcal{R}$. For the *mTSP-DS set-partioning* (mTSP-DS-sp) model, we also introduce the binary decision variables x_r, which are used to indicate if route r is a part of the solution. Finally, we use the continuous decision variable τ_s to represent the makespan.

Using the presented notation, we formulate the mTSP-DS-sp as follows:

$$\min \quad \tau_s \tag{30}$$

$$s.t. \qquad c_r \cdot x_r \leq \tau_s : \quad \forall r \in \mathcal{R}, \tag{31}$$

$$\sum_{r \in \mathcal{R}} x_r = K_N, \tag{32}$$

$$\sum_{r \in \mathcal{R}} a_{ir} \cdot x_r = 1 : \quad \forall i \in V_N, \tag{33}$$

$$\sum_{r \in \mathcal{R}} a_{ir} \cdot x_r \leq 1 : \quad \forall i \in V_S, \tag{34}$$

$$\sum_{i \in V_S} \sum_{r \in \mathcal{R}} a_{ir} \cdot x_r \leq C, \tag{35}$$

$$x_r \in \{0, 1\} : \quad \forall r \in \mathcal{R}, \tag{36}$$

$$\tau_s \in \mathbb{R}. \tag{37}$$

The objective of the MILP model (30) - (37) is to minimize the makespan τ_s that is determined by constraints (31), which set τ_s to be equal to the cost of the longest

selected route. Constraint (32) ensures that exactly K_N routes are selected. We make this restriction to be consistent with the mTSP-DS MILP formulation (1) - (13), but it would also be possible to allow the set-partitioning model to select fewer than K routes. Constraints (33) guarantee that each customer node is visited in exactly one of the selected routes. Due to constraints (34), each drone station node can be visited in at most one of the selected routes. Finally, constraint (35) states that the number of drone station nodes in all selected routes is less than or equal to the maximum number of drone stations C. To enhance the performance of the MILP solver, we use the best solution found in the first phase of the two-phase algorithm as a start solution.

5. Computational experiments and their numerical results

We carried out extensive computational experiments to assess the performance of the algorithms that we presented for solving the mTSP-DS. In this section, we present the results of our computational experiments and their evaluations.

More precisely, in Section 5.1, we introduce the instances and the parameters that we used for our experiments. Afterwards, we report the results and analyze the performance of the algorithms in Section 5.2. Finally, in Section 5.3, we use the numerical results to investigate the effects of drones on energy consumption of a fleet of delivery vehicles.

5.1. Instance generation and experiment parameters

For the experiments, we adopted 25 scattered and 25 clustered instances from the well-known A, B, P (Augerat, 1995), E (Christofides & Eilon, 1969), and M (Christofides et al., 1979) instance sets for the *Capacitated Vehicle Routing Problem* (CVRP). Depending on the minimum feasible number of trucks required to solve the corresponding CVRP instance, we placed drone stations in those instances: 4 drone stations if the respective number of trucks is less than or equal to 6, and 6 drone stations otherwise. We positioned the drone stations in such a way that the number of customers that can be supplied by drones is maximized (Kloster et al., 2022). Consequently, the positioning of the drone stations depends on the operational drone range, which we set to $\mathcal{E}_r \in \{8, 12, 16\}$ in our experiments (see Agatz et al., 2018; Schermer et al., 2019c). Furthermore, we assume that all available drone stations can be activated, i.e., depending on the instance $C = 4$ or $C = 6$, and we assume that each drone station is equipped with $D_N \in \{1, 2, 3\}$ drones. For the drone velocity relative to the trucks, we tested the values $\alpha \in \{0.5, 1, 2\}$, where $\alpha = 0.5$ refers to the case of autonomous robots, as they are generally considered to be slower than conventional delivery trucks. These values are commonly accepted in the literature (see Agatz et al., 2018; Schermer et al., 2019a,b,c). Finally, tests were carried out with $K_N \in \{2, 3, 4\}$ trucks, and if the minimum feasible number of trucks required to solve the corresponding CVRP instance is greater than 4, we also tested the respective value. Considering all parameter variations, 5265 experiments were carried out for each solution approach.

We implemented the algorithms in Python, and solved the MILP models with Gurobi Optimizer 8.1.1 (Optimization, 2019). All experiments were carried out on Intel Xeon Gold 6126 CPUs with 32GB RAM.

A time limit of 20 minutes (per run) was imposed on Gurobi Optimizer in solving the MILP models of the mTSP-DS. The runtime of the *solve_local_dasp* function in

the decomposition-based matheuristic (see Algorithm 1) is mainly determined by the number of customers within the range of the respective drone stations. Therefore, we set a time limit based on the drone range $\mathcal{E}_r$ for this function. More precisely, the time limit was set to 1s for $\mathcal{E}_r = 8$, 3s for $\mathcal{E}_r = 12$, and 5s for $\mathcal{E}_r = 16$. We did not set any explicit time limit for the two-phase algorithm. Instead, the runtime of the algorithm is implicitly determined by its parameters. Based on some preliminary experiments, we decided to use the following parameters for the two-phase algorithm to have the best performance of the algorithm:

- number of restarts $\Gamma = 50$,
- number of nonimproving ILS iterations $\delta = 50$,
- initial sparsification factor $\pi_{base} = 1.25$,
- fraction of nonimproving iterations after the granular neighborhood size is increased $\phi = 0.2$,
- factor to increase the size of the granular neighborhood $\lambda = 2$.

5.2. Results

In this section, we examine the performance of Gurobi in solving the MILP formulation (1) - (13), with and without valid inequalities (14), the decomposition-based matheuristic (with both local DASP variants), and the two-phase algorithm.

Unfortunately, without valid inequalities (14), Gurobi could not manage to solve the instances within the time limit. For many experiments, the solver cannot find any feasible solution at all, and for the remaining experiments, the average MIP Gap is 90.8%. Table 1 shows that introducing valid inequalities (14) improves the performance of the solver significantly and lowers the average MIP Gap to 44.4%. Some experiments with smaller instances ($n \leq 50$) were even solved optimally. In total, the solver was able to obtain 90 optimal solutions in the experiments with scattered instances and 3 optimal solutions in the experiments with clustered instances. For larger instances, however, the quality of the solutions remains poor, and in many cases still no feasible solution was found (indicated by "-" in Table 1), especially for experiments with $K_N > 4$.

Following the evaluation of the MILP solver, we tested the performance of the matheuristic algorithms. In Table 1, we present the average runtimes of each solution method. Averaged across all instances and parameters, the runtime of the DASP and DASP-LTR is 12.9 and 2.0 seconds, respectively. Furthermore, with an average runtime of 126.6 seconds, the two-phase algorithm is slower than both variants of the decomposition-based matheuristics, whereby the runtime of the second phase of the two-phase approach accounts for 57.4 seconds of the total runtime on average. Especially for larger instances and experiments with many trucks, the runtime of the second phase increases quite quickly. For such experiments, it might thus make sense to limit the size of the solution pool $|\Omega|$. However, the second phase of the two-phase approach improved the best solution found during the first phase in 28.5% of the experiments. In doing so, the second phase improved the objective value by 3.8% on average.

To analyze the solution quality of the proposed methods, we use the following metric

Table 1 Average runtime (in seconds) and savings Δ_{mTSP}. For Gurobi with valid inequalities (14), we additionally present the MIP Gap and the number found of optimal solutions. We also provide additional information for the two-phase approach: size of the solution pool ($|\Omega|$), runtime of the second phase (t_2), percent of experiments in which the second phase improved the solution ($\%imp_2$), and the percentage of drone-eligible customers that are served by drones (%y).

	Gurobi with (14)				DASP		DASP-LTR		Two-Phase					
Instance	Gap	#opt	t	Δ_{mTSP}	t	Δ_{mTSP}	t	Δ_{mTSP}	t	Δ_{mTSP}	$\|\Omega\|$	t_2	$\%imp_2$	%y
A-n34-k5	31.3%	0	1200.0	0.5%	5.0	1.2%	0.5	0.7%	62.6	7.5%	24212.5	21.8	6.5%	85.5%
A-n36-k5	46.8%	0	1200.0	-0.5%	8.8	0.7%	0.6	2.0%	52.2	3.3%	20469.2	18.5	3.7%	70.0%
A-n45-k7	44.2%	0	1200.0	-4.8%	13.2	3.3%	1.0	1.6%	110.4	8.6%	36045.5	46.8	15.7%	82.5%
A-n46-k7	43.5%	0	1200.0	-4.6%	11.3	6.0%	0.7	4.2%	79.7	8.8%	26805.6	30.6	26.9%	78.6%
A-n48-k7	48.6%	0	1200.0	-15.1%	11.2	4.3%	0.9	3.7%	101.1	9.4%	33200.2	40.4	21.3%	78.9%
A-n55-k9	-	0	1200.0	-	10.5	5.7%	0.8	6.1%	96.8	11.9%	27219.9	41.8	19.4%	77.8%
A-n60-k9	-	0	1200.0	-	13.7	6.7%	1.1	3.9%	101.2	9.8%	28561.3	41.0	16.7%	71.7%
A-n62-k8	-	0	1200.0	-	15.6	3.5%	1.3	3.7%	158.9	8.3%	41039.5	76.5	28.7%	77.7%
A-n69-k9	-	0	1200.0	-	16.2	7.8%	1.2	5.3%	153.5	13.5%	37902.9	71.1	26.9%	79.4%
A-n80-k10	-	0	1200.0	-	18.4	4.7%	1.7	4.4%	178.0	8.8%	43323.0	72.8	19.4%	75.8%
E-n22-k4	20.4%	17	1015.0	15.7%	3.7	8.4%	0.3	2.8%	33.2	16.5%	14623.1	9.4	22.2%	73.8%
E-n23-k3	12.5%	17	1005.6	10.3%	1.0	5.3%	0.3	4.1%	35.1	11.0%	13861.5	9.4	4.9%	86.6%
E-n30-k3	41.5%	0	1200.0	9.0%	6.6	7.0%	0.5	6.2%	40.8	11.1%	17162.4	12.5	16.0%	64.3%
E-n33-k4	36.4%	0	1200.0	6.8%	8.4	6.1%	0.6	4.2%	72.1	8.8%	27914.9	27.5	17.3%	83.4%
E-n51-k5	36.6%	0	1200.0	-0.6%	12.1	8.7%	0.8	4.2%	90.4	18.3%	31547.5	37.6	46.3%	70.4%
E-n76-k7	42.3%	0	1200.0	-0.5%	18.9	12.9%	1.7	8.4%	170.1	23.4%	42801.3	76.0	59.3%	69.0%
E-n101-k14	-	0	1200.0	-	20.0	11.5%	2.8	5.3%	362.8	20.5%	53812.2	203.5	46.3%	63.1%
M-n151-k12	-	0	1200.0	-	21.4	11.9%	6.4	0.9%	688.0	19.2%	83445.5	328.5	54.6%	56.4%
M-n200-k16	-	0	1200.0	-	24.8	10.5%	10.7	3.6%	1042.8	17.3%	77116.8	537.1	45.4%	53.4%
P-n16-k8	13.3%	31	935.2	18.7%	5.4	11.2%	0.3	7.5%	43.2	18.7%	15310.5	14.9	20.4%	75.7%
P-n19-k2	19.8%	15	1029.7	18.5%	4.2	15.4%	0.3	11.2%	26.9	18.6%	12262.7	7.4	25.9%	71.5%
P-n40-k5	24.0%	7	1148.8	13.4%	10.1	11.1%	0.6	4.7%	67.0	18.7%	26258.5	25.7	49.1%	70.0%
P-n50-k7	34.4%	3	1193.1	11.1%	15.1	9.9%	0.8	3.7%	101.1	18.0%	31872.7	44.8	44.4%	70.7%
P-n60-k10	38.1$	0	1200.0	10.2%	16.9	13.1%	1.1	8.1%	146.2	20.5%	38294.4	73.6	45.4%	67.3%
P-n65-k10	37.4%	0	1200.0	8.7%	18.4	11.9%	1.3	3.4%	179.7	21.9%	44426.5	92.7	50.0%	68.2%
Mean Scattered	33.9%		1169.0	5.2%	12.8	7.9%	1.6	4.5%	174.4	14.1%	34864.6	81.9	29.9%	72.7%
B-n31-k5	35.4%	3	1189.3	4.0%	8.4	1.7%	0.9	1.4%	24.5	3.0%	11613.1	7.1	1.9%	82.0%
B-n34-k5	30.7%	0	1200.0	9.7%	8.5	7.0%	1.0	6.7%	39.9	13.1%	13830.3	11.9	45.4%	76.0%
B-n35-k5	45.5%	0	1200.0	8.8%	9.2	7.8%	0.6	7.0%	34.4	9.7%	12122.7	10.4	18.5%	83.3%
B-n38-k6	55.1%	0	1200.0	5.2%	8.2	6.7%	0.9	5.7%	40.3	7.3%	14734.3	13.5	14.8%	68.6%
B-n39-k5	64.2%	0	1200.0	10.2%	9.7	10.0%	1.2	9.1%	44.8	11.5%	17322.8	16.7	15.7%	71.0%
B-n41-k6	64.2%	0	1200.0	4.8%	6.5	4.7%	2.0	6.3%	62.1	9.8%	21111.5	22.6	25.9%	70.9%
B-n43-k6	56.8%	0	1200.0	-1.5%	8.5	3.4%	1.2	2.7%	65.5	8.5%	22197.4	25.0	29.6%	75.1%
B-n44-k7	70.5%	0	1200.0	7.2%	13.4	5.6%	1.4	6.4%	53.3	10.2%	16329.9	22.3	21.3%	78.9%
B-n45-k5	49.6%	0	1200.0	0.7%	10.9	1.5%	0.9	1.8%	53.1	6.4%	17999.7	18.9	6.5%	71.4%
B-n45-k6	44.9%	0	1200.0	7.2%	9.8	9.3%	1.4	8.1%	70.4	13.6%	23887.9	28.3	46.3%	65.0%
B-n50-k7	53.3%	0	1200.0	7.0%	12.9	13.3%	1.6	11.7%	57.3	17.0%	17455.1	20.7	39.8%	67.9%
B-n50-k8	64.1%	0	1200.0	-0.9%	10.9	6.4%	1.0	5.7%	63.5	7.3%	19049.8	24.0	4.6%	80.5%
B-n51-k7	46.0%	0	1200.0	0.4%	12.2	9.5%	3.0	9.2%	53.6	13.9%	14886.2	17.8	25.9%	72.5%
B-n52-k7	73.0%	0	1200.0	1.0%	14.2	7.8%	2.4	8.9%	66.8	14.1%	19163.5	23.9	39.8%	72.2%
B-n56-k7	77.2%	0	1200.0	0.3%	15.4	9.0%	2.3	7.3%	70.8	11.9%	22242.7	26.8	21.3%	65.5%
B-n57-k7	-	0	1200.0	-	13.1	5.4%	3.7	3.1%	73.2	8.0%	20483.0	27.1	24.1%	74.5%
B-n57-k9	-	0	1200.0	-	15.6	8.7%	2.3	8.8%	74.0	11.2%	20191.0	30.2	31.5%	77.5%
B-n63-k10	-	0	1200.0	-	14.4	5.5%	2.0	3.3%	99.1	10.0%	25115.9	40.6	32.4%	75.5%
B-n64-k9	-	0	1200.0	-	15.1	9.9%	2.9	6.6%	61.5	12.6%	16633.4	21.8	23.1%	69.4%
B-n66-k9	-	0	1200.0	-	14.3	3.5%	3.0	3.8%	110.5	8.2%	27936.4	42.6	21.3%	78.1%
B-n67-k10	-	0	1200.0	-	16.1	4.9%	1.3	4.2%	81.1	8.3%	20959.5	31.4	13.9%	77.8%
B-n68-k9	-	0	1200.0	-	18.4	4.6%	3.2	2.1%	97.6	9.1%	23867.0	39.4	42.6%	66.9%
B-n78-k10	-	0	1200.0	-	18.8	7.1%	2.7	4.2%	132.2	10.7%	30806.6	55.3	35.2%	64.5%
M-n101-k10	-	0	1200.0	-	19.4	6.8%	4.0	5.4%	165.2	14.0%	31401.7	66.4	47.2%	62.3%
M-n121-k7	-	0	1200.0	-	21.0	12.9%	11.4	11.9%	336.6	15.9%	37507.0	207.4	49.1%	68.9%
Mean Clustered	55.1%		1199.6	4.3%	13.0	6.9%	2.3	6.1%	81.3	10.6%	20753.9	34.1	27.1%	72.6%
Mean Overall	44.4%		1184.7	4.8%	12.9	7.4%	2.0	5.3%	126.6	12.3%	27628.4	57.4	28.5%	72.7%

(Schermer et al., 2019b):

$$\Delta_{mTSP} = 100\% - \frac{\tau}{\tau^*_{mTSP}}, \tag{38}$$

where τ represents the objective value obtained either by the solver or by an algorithm in solving the mTSP-DS, and the baseline is τ^*_{mTSP}, which is the length of the mTSP solution for a given instance obtained by the Lin-Kernighan-Helsgaun mTSP solver,

i.e., in absence of drones and drone stations. Using this metric, negative values of Δ_{mTSP} show a deterioration compared to the corresponding mTSP solution.

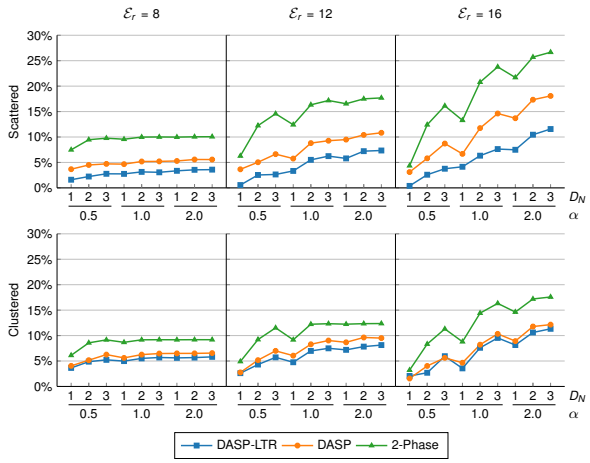
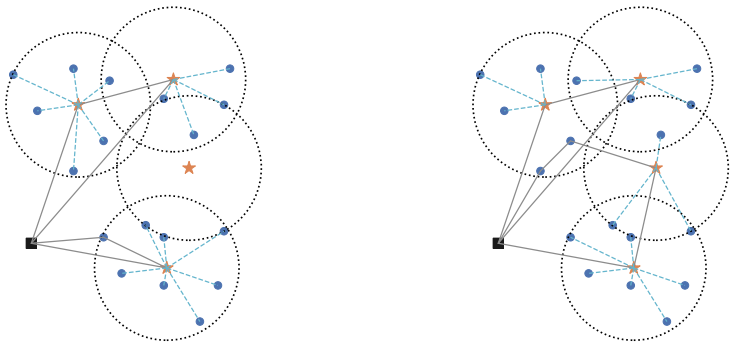


Figure 4 Average Δ_{mTSP} grouped by instance type, $\mathcal{E}_r$, α, and D_N.

According to the results, presented in Table 1, the two-phase algorithm outperforms both variants of the decomposition-based matheuristics in terms of the savings in comparison to mTSP solutions, and the DASP performs better than the DASP-LTR. The two-phase algorithm achieved an average Δ_{mTSP} of 12.3% and was able to find 76 of the 93 known optimal solutions. With an average Δ_{mTSP} of 7.4%, the DASP found 9 known optimal solutions, and finally, the DASP-LTR achieved an average Δ_{mTSP} of 5.3% and found two of the known optimal solutions. Interestingly, the difference between the DASP and the local DASP-LTR in terms of solution quality is negligible for many experiments with clustered instances. This is also illustrated in Figure 4, which shows the average makespan savings Δ_{mTSP} of the heuristic approaches grouped by the instance type, $\mathcal{E}_r$, α, and D_N.

In experiments with $\mathcal{E}_r = 8$, the savings from additional or faster drones are small or nonexistent, suggesting that most of the customers within the range of a drone station that have an impact on the makespan can already be supplied with comparatively few or slow drones. Indeed, in experiments with $\mathcal{E}_r = 8$, 79.1% of the drone-eligible customers are supplied by drones. More generally speaking, Table 1 shows that the percentage of drone-eligible customers that are served by drones is relatively high. Nevertheless, it is not always useful to activate all available drone stations, particularly when investigating small problem instances. In fact, in 33 of the 93 known optimal solutions obtained by Gurobi Optimizer, fewer than C drone stations were activated. One of these solutions is visualized in Figure 5a. Through the set-partitioning model, the two-phase algorithm also has the ability to activate fewer than C drone stations. While this did not happen in a single case, some of the solutions obtained by the two-phase algorithm are nevertheless optimal, even if the number of activated drone stations differs in the corresponding Gurobi solution. In Figure 5b, we show an optimal solution of the same instance shown in Figure 5a, in which all drone stations are activated.

Overall, all the algorithms that we presented are able to achieve savings compared to mTSP solutions. In particular, the two-phase algorithm consistently delivered the best results. However, especially for the larger instances, the good solution quality of the two-phase algorithm goes hand-in-hand with a comparatively longer runtime.



(a) Method: Gurobi with (14), t = 495.5 **(b)** Method: Two-Phase, t = 14.7

Figure 5 Two optimal mTSP-DS solutions for the instance P-n19-k2 with the parameters D_N=3, $C = 4$, $\alpha = 1$, $\mathcal{E}_r = 12$, $K_N = 2$. The objective value of the solutions is 76.6 and they achieve savings of $\Delta_{mTSP} = 32.0\%$. The square, circle, and star shapes represent the depot, customers, and drone stations, respectively. The dotted circles around the drone stations show the operation range of drones, the dashed lines depict the drone deliveries, and the solid lines represent the truck tours.

5.3. Energy consumption

The previous sections have shown that the use of drones can reduce the overall delivery time. However, since drones can only transport one package at a time in the mTSP-DS, it might happen that drones require to travel a longer distance than trucks. In this section, we use the data generated in our experiments, and we compare the results of the two-phase algorithm (see Section 4.2) with mTSP routes to analyze the effects of drone usage in terms of *energy consumption*.

In the following, we indicate the sum of the distances covered by trucks in a mTSP solution by d_{mTSP}. Similarly, we refer to the sums of distances covered by trucks and by drones in a heuristic solution as d_t and d_d, respectively. We also define the average energy consumption of trucks and drones per kilometer by e_t and e_d, and we assume that the energy consumption of drone stations is negligible. Using this notation, we utilize the following metric to compute the energy savings:

$$\Delta_E = 100\% - \frac{d_t \cdot e_t + d_d \cdot e_d}{d_{mTSP} \cdot e_t}. \tag{39}$$

In the literature, we find considerably different values for energy consumption of vehicles (see, e.g., Zhang et al., 2021, and references therein). In this paper, we follow the approach of Goodchild & Toy (2018) and consider different values for e_t and e_d to cover a range of different scenarios. For e_t, we use values from 100 to 1000 Wh/km,

increasing in steps of 100, and for e_d, we utilize values from 10 to 100 Wh/km, increasing in steps of 10.

Figure 6 visualizes the average energy savings of the solutions obtained by the two-phase algorithm for all ratios of e_t and e_d. In our experiments, savings can be achieved if $e_t/e_d \geq 6.7$ for scattered instances and if $e_t/e_d \geq 7.9$ for clustered instances.

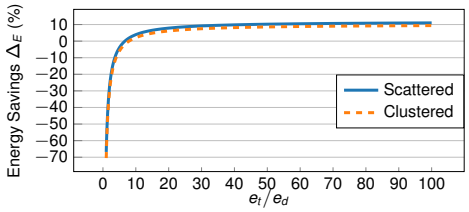


Figure 6 Average energy savings Δ_E for different truck/drone energy consumption ratios.

In addition, in Figure 7, we display the average energy consumption depending on the drone range and velocity for two different settings, where we assume that drones consume 50, 75, and 100 Wh/km if their relative velocity α is 0.5, 1, and 2, respectively. Furthermore, we assume that if $\alpha = 0.5$, energy-efficient robots that consume 25 Wh/km could be used instead of drones[1]. Figure 7a represents the use of conventional trucks[2], where e_t is set to 1016 Wh/km. In this case, the solutions obtained by the two-phase algorithm always achieve energy savings in comparison to the mTSP routes. In experiments with $\mathcal{E}_r$ = 8, robots achieve higher energy savings than drones. At higher drone ranges, however, faster drones can serve more customers. Figure 7a shows that this increases the energy savings, but it also increases the distance traveled by drones significantly.

If we replace the conventional trucks with electric ones[3], and assume that they consume 215 Wh/km, the long distances covered by drones become an issue. Figure 7b shows that in this second setting, the average energy savings are negative if drones are used, meaning that the mTSP solutions require less energy. In such cases, the energy consumption of the drones and robots is higher than the energy savings obtained by shorter truck tours.

As a final note, we know that the two objectives, minimization of the makespan and minimization of energy consumption, are not necessarily complementary to each other. Therefore, if the main goal is to minimize energy consumption, the objective function of the mTSP-DS should be adjusted accordingly.

6. Conclusion

In this paper, we presented the *multiple Traveling Salesman Problem with Drone Stations* (mTSP-DS). In the mTSP-DS, for a given depot, a set of trucks, a set of

[1]Figliozzi (2020) estimates that a *Starship* delivery robot consumes 24.7 Wh/km.
[2]Estimated energy consumption of a *RAM ProMaster 2500* truck (Figliozzi, 2020).
[3]More precisely, we assume the use of a *Volkswagen e-Crafter* fleet (Volkswagen, 2018).

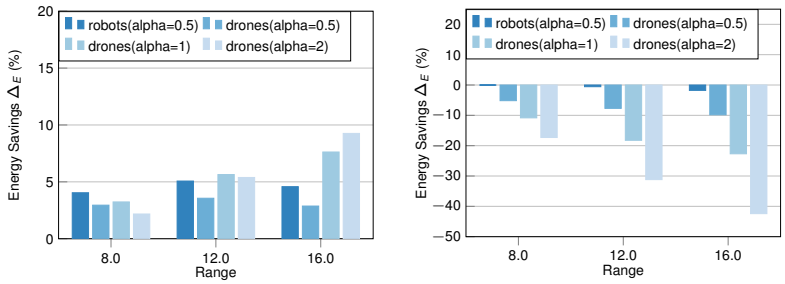


Figure 7 Average energy savings Δ_E grouped by drone range $\mathcal{E}_r$ and relative drone velocity α. In both figures, we suppose that robots consume 25 Wh/km, and drones with α equal to 0.5, 1.0, and 2.0 consume 50, 75, and 100 Wh/km, respectively. In (a), we consider the use of conventional trucks, which consume 1016 Wh/km. In (b), we assume that electric trucks, which consume 215 Wh/km, are used.

customers, and a set of drone stations hosting a certain number of autonomous drones, the goal consists in serving all customers either by truck or by drone in the shortest possible time. For this purpose, trucks start their tour from the depot, serve the customers, and can visit drone stations to activate them and supply them with parcels. Then, an activated drone station can launch and operate drones autonomously. Indeed, through parallel serving operations conducted by drones and trucks, we can reduce the total mission time. Compared to previous approaches, in which the drones supply customers directly from the depot, drone stations allow a more targeted use of drone deliveries, especially in densely populated urban areas, where it is not practicable to build additional depots.

We formulated the mTSP-DS as a *mixed integer linear programming* (MILP) model, which can be solved by any standard MILP solver. However, this is only possible for small instances. To address larger instances, the use of a solver is not reasonable anymore because of the excessively long runtime. To overcome this issue and to be able to solve larger problem instances, we designed three algorithms.

The first algorithm is called local *drone assignment and scheduling problem* (DASP), which decomposes the problem into easier subproblems. The second method is the local *DASP with limited truck rerouting* (DASP-LTR), which restricts the rerouting of trucks within the range of a drone station. We formulated both variants of the local DASP as a MILP and embedded them in an iterative heuristic. The third algorithm is a two-phase approach, where after generation of a solution pool, we apply a polishing phase through which we solve a set-partitioning model to find the best combination of tours.

The results of our extensive computational experiments show that the two-phase algorithm is superior to the DASP approaches in terms of solution quality. However, the runtime of both DASP approaches is considerably shorter. Furthermore, our experiments have shown that, depending on the parameters, the mTSP-DS can substantially reduce the makespan in comparison to traditional mTSP solutions.

Nevertheless, our numerical investigations confirm that a reduction in delivery time

is not necessarily accompanied by reduced energy consumption. Indeed, whether the mTSP-DS reduces energy consumption depends strongly on the consumption coefficients of the utilized trucks/drones and the distance covered by drones.

Future research might investigate the trade-off relationships of different objective functions for the mTSP-DS, e.g., the minimization of energy consumption, cost, and makespan in a multi-objective optimization context. From an algorithmic perspective, design of efficient exact methods, which might also use the algorithms that we presented in this paper, can be an promising future research direction to follow. Finally, since examining all possible combinations in the DASP is quite time-consuming; hence, a future work might investigate how to assess the potential of drone station activation to reduce the amount of examined combinations.

References

Accorsi, L., & Vigo, D. (2020). A hybrid metaheuristic for single truck and trailer routing problems. *Transportation Science*, *54*, 1351–1371.

Agatz, N., Bouman, P., & Schmidt, M. (2018). Optimization approaches for the traveling salesman problem with drone. *Transportation Science*, *52*, 965–981.

Archetti, C., & Speranza, M. G. (2014). A survey on matheuristics for routing problems. *EURO Journal on Computational Optimization*, *2*, 223–246.

Augerat, P. (1995). *Approche polyèdrale du problème de tournées de véhicules*. Ph.D. thesis Institut National Polytechnique de Grenoble - INPG.

Boschetti, M. A., Maniezzo, V., Roffilli, M., & Bolufé Röhler, A. (2009). Matheuristics: optimization, simulation and control. In M. J. Blesa, C. Blum, L. Di Gaspero, A. Roli, M. Sampels, & A. Schaerf (Eds.), *Hybrid Metaheuristics* (pp. 171–177). Springer.

Cheng, T. C. E., & Sin, C. C. S. (1990). A state-of-the-art review of parallel-machine scheduling research. *European Journal of Operational Research*, *47*, 271–292.

Christofides, N., & Eilon, S. (1969). An Algorithm for the Vehicle-dispatching Problem. *Journal of the Operational Research Society*, *20*, 309–318.

Christofides, N., Mingozzi, A., & Toth, P. (1979). The vehicle routing problem. In N. Christofides, A. Mingozzi, P. Toth, & C. Sandi (Eds.), *Combinatorial Optimization* (pp. 315–338). Wiley.

Coffman, E. G., Jr., Garey, M. R., & Johnson, D. S. (1978). An application of bin-packing to multiprocessor scheduling. *SIAM Journal on Computing*, *7*, 1–17. doi:10.1137/0207001.

Cornuéjols, G., Nemhauser, G., & Wolsey, L. (1990). The uncapicitated facility location problem. In P. B. Mirchandani, & R. L. Francis (Eds.), *Discrete Location Theory* (pp. 119–171). John Wiley and Sons.

Dell'Amico, M., Montemanni, R., & Novellani, S. (2020). Matheuristic algorithms for the parallel drone scheduling traveling salesman problem. *Annals of Operations Research*, *289*, 211–226.

DHL (2018). Dhl's parcelcopter: changing shipping forever. URL: `https://discover.dhl.com/business/business-ethics/parcelcopter-drone-technology` accessed 4 May 2020.

Figliozzi, M. A. (2020). Carbon emissions reductions in last mile and grocery deliveries utilizing air and ground autonomous vehicles. *Transportation Research Part D: Transport and Environment*, *85*, 102443.

Floreano, D., & Wood, R. J. (2015). Science, technology and the future of small autonomous drones. *Nature*, *521*, 460–466.

Gillett, B. E., & Miller, L. R. (1974). A heuristic algorithm for the vehicle-dispatch problem. *Operations research*, *22*, 340–349.

Goeke, D. (2019). Granular tabu search for the pickup and delivery problem with time windows and electric vehicles. *European Journal of Operational Research*, *278*, 821–836.

Goodchild, A., & Toy, J. (2018). Delivery by drone: an evaluation of unmanned aerial vehicle technology in reducing co2 emissions in the delivery service industry. *Transportation Research Part D: Transport and Environment*, *61*, 58–67.

Helsgaun, K. (2000). An effective implementation of the lin–kernighan traveling salesman heuristic. *European Journal of Operational Research*, *126*, 106–130.

Helsgaun, K. (2017). *An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems*. Technical Report Roskilde Universitet.

Khoufi, I., Laouiti, A., & Adjih, C. (2019). A survey of recent extended variants of the traveling salesman and vehicle routing problems for unmanned aerial vehicles. *Drones*, *3*, 66.

Kim, S., & Moon, I. (2018). Traveling salesman problem with a drone station. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *49*, 42–52.

Kitjacharoenchai, P., Min, B.-C., & Lee, S. (2020). Two echelon vehicle routing problem with drones in last mile delivery. *International Journal of Production Economics*, *225*, 107598.

Kloster, K., Moeini, M., Vigo, D., & Wendt, O. (2022). Instances for the multiple traveling salesman problem with drone stations (mtsp-ds). doi:`https://doi.org/10.5281/zenodo.6380739`.

Lee, C.-Y., & Massey, J. D. (1988). Multiprocessor scheduling: combining lpt and multifit. *Discrete Applied Mathematics*, *20*, 233–242.

Lourenço, H. R., Martin, O. C., & Stützle, T. (2003). Iterated local search. In F. Glover, & G. A. Kochenberger (Eds.), *Handbook of Metaheuristics* (pp. 320–353). Springer. doi:https://doi.org/10.1007/0-306-48056-5_11.

Macrina, G., Pugliese, L. D. P., Guerriero, F., & Laporte, G. (2020). Drone-aided routing: a literature review. *Transportation Research Part C: Emerging Technologies*, *120*, 102762. doi:https://doi.org/10.1016/j.trc.2020.102762.

Marinelli, M., Caggiani, L., Ottomanelli, M., & Dell'Orco, M. (2018). En route truck–drone parcel delivery for optimal vehicle routing strategies. *IET Intelligent Transport Systems*, *12*, 253–261.

Moeini, M., & Salewski, H. (2020). A genetic algorithm for solving the truck-drone-atv routing problem. In H. A. Le Thi, H. M. Le, & T. Pham Dinh (Eds.), *Optimization of Complex Systems: Theory, Models, Algorithms and Applications* (pp. 1023–1032). Springer.

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

Nguyen, M. A., Dang, G. T.-H., Hà, M. H., & Pham, M.-T. (2022). The min-cost parallel drone scheduling vehicle routing problem. *European Journal of Operational Research*, *299*, 910–930. doi:10.1016/j.ejor.2021.07.008.

Optimization, G. (2019). Gurobi Optimizer Reference Manual.

Otto, A., Agatz, N., Campbell, J., Golden, B., & Pesch, E. (2018). Optimization approaches for civil applications of unmanned aerial vehicles (uavs) or aerial drones: a survey. *Networks*, *72*, 411–458.

Penna, P. H. V., Subramanian, A., & Ochi, L. S. (2013). An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics*, *19*, 201–232.

Poikonen, S., & Golden, B. (2019). The mothership and drone routing problem. *INFORMS Journal on Computing*, *32*, 249–262.

Rojas Viloria, D., Solano-Charris, E. L., Muñoz-Villamizar, A., & Montoya-Torres, J. R. (2021). Unmanned aerial vehicles/drones in vehicle routing problems: a literature review. *International Transactions in Operational Research*, *28*, 1626–1657.

Sacramento, D., Pisinger, D., & Ropke, S. (2019). An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones. *Transportation Research Part C: Emerging Technologies*, *102*, 289–315. doi:https://doi.org/10.1016/j.trc.2019.02.018.

Schermer, D. (2019). Integration of drones in last-mile delivery: the vehicle routing problem with drones. In *Operations Research Proceedings (GOR (Gesellschaft für Operations Research e.V.))* (pp. 17–22). Springer.

Schermer, D., Moeini, M., & Wendt, O. (2018a). Algorithms for solving the vehicle routing problem with drones. In N. T. Nguyen, D. H. Hoang, T.-P. Hong, H. Pham, & B. Trawiński (Eds.), *Intelligent Information and Database Systems* (pp. 352–361). Springer.

Schermer, D., Moeini, M., & Wendt, O. (2018b). *A variable neighborhood search algorithm for solving the vehicle routing problem with drones*. Technical Report Technische Universität Kaiserslautern.

Schermer, D., Moeini, M., & Wendt, O. (2019a). A hybrid vns/tabu search algorithm for solving the vehicle routing problem with drones and en route operations. *Computers & Operations Research*, *109*, 134–158.

Schermer, D., Moeini, M., & Wendt, O. (2019b). A matheuristic for the vehicle routing problem with drones and its variants. *Transportation Research Part C: Emerging Technologies*, *106*, 166–204.

Schermer, D., Moeini, M., & Wendt, O. (2019c). The traveling salesman drone station location problem. In H. A. Le Thi, H. M. Le, & T. Pham Dinh (Eds.), *Optimization of Complex Systems: Theory, Models, Algorithms and Applications* (pp. 1129–1138). Springer.

Schermer, D., Moeini, M., & Wendt, O. (2020). The drone-assisted traveling salesman problem with robot stations. In *Proceedings of the 53rd Hawaii International Conference on System Sciences* (pp. 1308–1317).

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

Toth, P., & Vigo, D. (2003). The granular tabu search and its application to the vehicle-routing problem. *Informs Journal on computing*, *15*, 333–346.

Toth, P., & Vigo, D. (2014). *Vehicle routing: problems, methods, and applications*. SIAM.

Volkswagen (2018). Volkswagen Commercial Vehicles electrifies the Crafter: e-Crafter zero-emission van goes online. URL: `https://www.vwpress.co.uk/en-gb/releases/3461` accessed 19 May 2021.

Vu, L., Vu, D. M., Hà, M. H., & Nguyen, V.-P. (2021). The two-echelon routing problem with truck and drones. *International Transactions in Operational Research*, . doi:`10.1111/itor.13052`.

Wang, X., Poikonen, S., & Golden, B. (2017). The vehicle routing problem with drones: several worst-case results. *Optimization Letters*, *11*, 679–697.

Zhang, J., Campbell, J. F., Sweeney II, D. C., & Hupman, A. C. (2021). Energy consumption models for delivery drones: a comparison and assessment. *Transportation Research Part D: Transport and Environment*, *90*, 102668. doi:`https://doi.org/10.1016/j.trd.2020.102668`.