European Journal of Operational Research 300 (2022) 571–589

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Production, Manufacturing, Transportation and Logistics

The parallel drone scheduling problem with multiple drones and vehicles

Raïssa G. Mbiadou Saleu[a,*], Laurent Deroussi[a], Dominique Feillet[b], Nathalie Grangeon[a], Alain Quilliot[a]

[a] *Université Clermont Auvergne, LIMOS UMR CNRS 6158, 1 Rue de la Chebarde, Aubière Cedex 63178, France*
[b] *Mines Saint-Etienne, Univ. Clermont Auvergne, LIMOS UMR CNRS 6158, Centre CMP, Gardanne F-13541, France*

A R T I C L E I N F O

Article history:
Received 2 November 2020
Accepted 10 August 2021
Available online 20 September 2021

Keywords:
Routing
Drone delivery
City logistics
m-TSP
ILS
Dynamic programming

A B S T R A C T

Delivery of goods into urban areas constitutes an important issue for logistics service providers. One of the most talked-about developments in recent years has been the potential use of unmanned aerial vehicles, or drones, for transporting packages, food, medicine, and other goods. Delivery by drones offers new possibilities, but also induces new challenging routing problems. In this paper, we address and extend the so-called Parallel Drone Scheduling Traveling Salesman Problem. Basically, in this problem, deliveries are split between a vehicle and one or several drones. The vehicle performs a classical delivery tour from the depot, while the drones are constrained to perform back and forth trips. The objective is to minimize the completion time. We extend the problem by considering several vehicles. We call it *Parallel Drone Scheduling Multiple Traveling Salesman Problem*. We propose a hybrid metaheuristic for its solution. The procedure starts by building a giant tour visiting all customers. Then, the giant tour is split in order to determine a set of vehicles tours (each vehicle tour following the order defined by the giant tour) and a set of customers assigned to drones. Thirdly, an improvement step move customers between vehicles or between vehicles and drones. We also propose a Mixed Integer Linear Programming formulation and a simple branch-and-cut approach. The proposed approach is validated via an experimental campaign on instances taken from the CVRPLIB[1] library. Computational experiments comparing several variants of the hybrid metaheuristic give some insights on this drone delivery system.

1. Introduction

The E-commerce boom and new urban restrictions on truck traffic have led to innovative models of parcel distribution. In December 2013, the chief executive officer of the largest online retailer Amazon Jeff Bezos shared with the world his vision of using flying robots to deliver products. «*I know this looks like science fiction. It's not*» he assured of the service called *Amazon Prime Air* (Amazon, 2013): a drone designed to deliver packages in just 30 minutes (Fig. 1). Using unmanned aerial vehicles (UAVs) for door-to-door deliveries seemed like a laughable pipe dream when Amazon tossed out that fanciful idea. Even though, some people didn't find the concept so far-fetched, many other companies and logistics providers started getting interested in experimenting with drone delivery. In August 2014, Google revealed its drone delivery project called *wing* (XCompany, 2014), with experiments run in Australia (Fig. 2). In September 2014, DHL Deutsche Post, Germanys privatized postal system, launched its *parcelcopter*, a helicopter-style drone which could deliver medications and urgently needed goods to the remote North Sea island of Juist (DHL, 2014) (Fig. 3). In 2016, the American courier company UPS has launched tests on the use of drones to deliver parcels to remote or hard-to-reach areas. In February 2017, they successfully tested a drone delivery method in which a drone (called HorseFly) is launched from the roof of a delivery truck (ABCNews, 2017) (Fig. 4). Many other companies like Alibaba (Kelion, 2015), Autralia Post (Francis, 2016) or DPD Group (Dpdgroup, 2014) have launched similar projects. Drones have also been deployed for humanitarian logistics (Zipline, 2016).

Different types of integration of drones into last-mile delivery have been investigated by companies and academics. Drones can be used alone or in combination with other means of delivery such as traditional trucks. In the combined operations of aerial drones and trucks, synchronization may or may not be required:

* Corresponding author.
E-mail addresses: raissa.saleu@isima.fr (R.G. Mbiadou Saleu), laurent.deroussi@uca.fr (L. Deroussi), feillet@emse.fr (D. Feillet), nathalie.grangeon@uca.fr (N. Grangeon), alain.quilliot@isima.fr (A. Quilliot).

[1] Capacited Vehicle Routing Problem LIBrary

https://doi.org/10.1016/j.ejor.2021.08.014
0377-2217/© 2021 Elsevier B.V. All rights reserved.

Fig. 1. Amazon Prime Air.

Fig. 2. Google Wing.

Fig. 3. DHL Parcelcopter.

Fig. 4. UPS HorseFly.

- Synchronization occurs when UAVs travel on trucks that not only perform their delivery tours but also serve as a mobile depot for UAVs. In this process, a UAV can be loaded with a package from a truck, deliver the package and return to a truck (potentially the same) for a next delivery. In the meanwhile, trucks continue their tours.
- Synchronization is not required when UAVs and vehicles perform independent tasks. Typically, both execute independent deliveries from a central distribution center.

Most studies have been carried out with the prospect of making drones and trucks work in tandem, drones taking off from the trucks and next landing back while those trucks are themselves involved into delivery tasks. This approach is likely to prevail when distribution centers are far from customers. But, as pointed out in Mathew, Smith, & Waslander (2015), if distribution centers are small, deeply embedded in a urban area and close to customers, direct deliveries with drones are possible. This process is simpler to implement by the logistics provider since it will not require any synchronization protocol. The provider will just have to split tasks between trucks and drones. In addition, having drones and trucks working in parallel will also ensure more robustness to the global delivery process.

In this paper, we are interested in the latter, that is, deliveries performed by drones and vehicles (trucks) in parallel, without synchronization. Vehicles perform classical parcel delivery tours; drones repeat single-customer back-and-forth trips. We propose a hybrid metaheuristic combining Iterated Local Search and Dynamic Programming. This heuristic is inspired from a heuristic that we developed previously for a simplified problem, where the fleet of trucks was limited to a single vehicle (Mbiadou Saleu, Deroussi, Feillet, Grangeon, & Quilliot, 2018). Note that generalizing the heuristic necessitates significant changes, as will be seen in Section 4.

The paper is structured as follows. In Section 2, we discuss upon the most relevant related literature. Section 3 defines our problem formally and provides a Mixed Integer Linear Programming (MILP) formulation. We propose a hybrid metaheuristic to solve the problem in Section 4. Section 5 presents experimental results. Section 6 concludes the paper.

2. Related work

The problem of parcel delivery with drones has received increasing attention these last years. Proposals for drone delivery vary widely, with drones being used independently or in conjunction with deliveries by trucks. This review focuses on routing problems involving truck-drone combination for parcel delivery. However, we note that works on drone-only delivery systems can be found in Dorling, Heinrichs, Messier, & Magierowski (2017), San, Lee, & Chang (2016), Sundar & Rathinam (2014), Choi & Schonfeld (2017), Troudi, Addouche, Dellagi, & Mhamedi (2018), Cheng, Adulyasak, & Rousseau (2018), Song, Park, & Kim (2018). Most of these studies assume that a drone can lift multiple packages within its maximum payload and serve customers in a service area of given radius. The main constraints are on weight battery energy and customers time windows. Additionally, apart from package delivery applications, many works investigate the potential of UAVs in other non-military applications like disaster management, rescue operations, agriculture or healthcare. A survey about the use of UAVs for civil applications is provided by Otto, Agatz, Campbell, Golden, & Pesch (2018).

The problem of combining a drone with a traditional delivery truck for parcel delivery was first formally defined by Murray & Chu (2015). They introduced two different problems: the *Flying Sidekick Traveling Salesman Problem* (FSTSP), where deliveries are

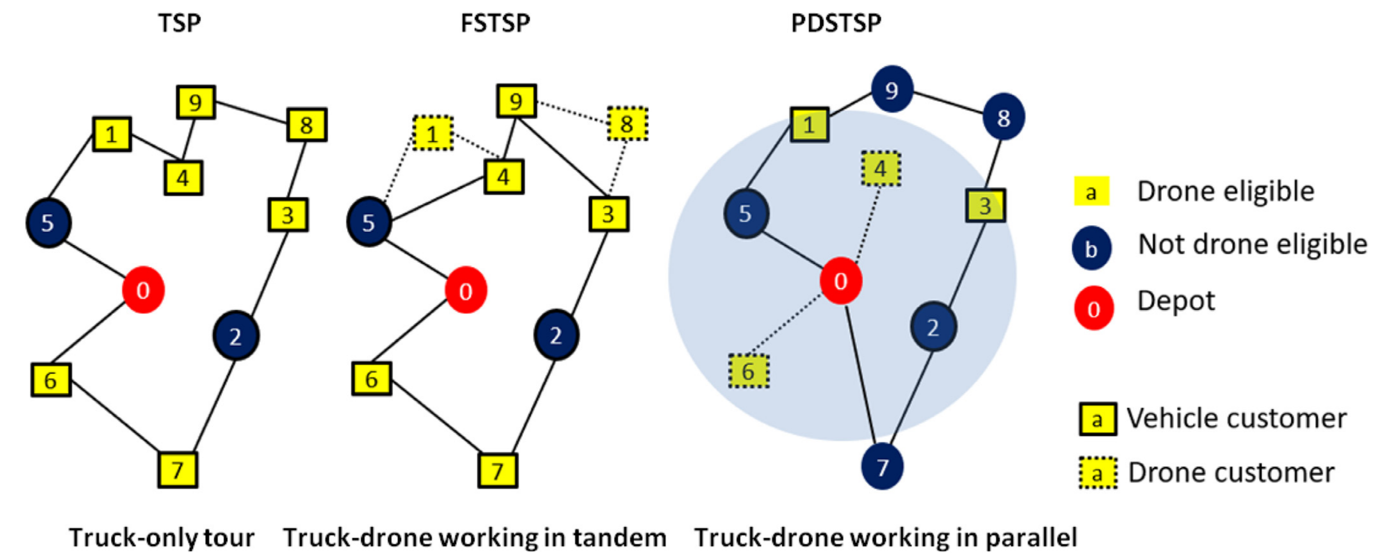


Fig. 5. Illustration of FSTSP and PDSTSP vs truck-only delivery.

performed by a single vehicle and a single drone working in tandem (the drone travels on the truck and can temporarily fly to make a delivery on its own) and the *Parallel Drone Scheduling Traveling Salesman Problem* (PDSTSP), where a vehicle and a fleet of drones operate separately from the depot. Fig. 5 illustrates a standard delivery system with a truck and the two proposed delivery systems with drones. MILP formulations and greedy construction heuristics for both problems were provided.

Since then, many studies have explored different variants of truck-drone combination for parcel delivery. Considering works related to the FSTSP, the single-truck single-drone variant (also called *TSP with drone*, TSP-D) is addressed in many papers. Agatz, Bouman, & Schmidt (2018) provided a MILP formulation and a *route first - cluster second* heuristic. Ponza (2016), Daknama & Kraus (2017), Marinelli, Caggiani, Ottomanelli, & Dell'Orco (2017), Ha, Deville, Pham, & Hà (2018), Yurek & Ozmutlu (2018), de Freitas & Penna (2020), and Schermer, Moeini, & Wendt (2018) proposed neighborhood-search based heuristics. Bouman, Agatz, & Schmidt (2018) provide exact solution approaches based on dynamic programming.

Other works consider a single-truck and multiple UAVs. Problems in this class assume that from each truck stop one or several drones can be launched. Chang & Lee (2018) and Mourelo Ferrandez, Harbison, Webwer, Sturges, & Rich (2016) proposed clustering based heuristics to find truck stops from where drones are launched. Tu, Dat, & Dung (2018) called this problem TSP-mD and they proposed an adaptive large neighborhood search (ALNS) heuristic. In Murray & Raj (2020), the same problem is called mFSTSP and the authors provided a MILP formulation and a three-phases heuristic solution approach.

The case involving both multiple trucks and multiple UAVs is considered in Kitjacharoenchai et al. (2019) where autonomous drones are allowed to fly from a delivery truck, make deliveries, and fly to any available truck nearby. A MILP formulation and an algorithm based on insertion heuristics are provided. Wang, Poikonen, & Golden (2017) studied the so-called Vehicle Routing Problem with Drones (VRPD) in which a drone launched from a truck must be picked up by the same truck at the same or at a different location. A worst-case analysis on the benefits achieved using drones is provided. This line of research is pursued by the same authors in Poikonen, Wang, & Golden (2017). Pugliese & Guerriero (2017) extended the VRPD by considering time window constraints. The problem is modeled as a variant of the vehicle routing problem with time windows (VRPTW) where each vehicle is equipped with drones (VDRPTW). A MILP formulation is proposed.

Literature related to the PDSTSP is more limited. In our previous work (Mbiadou Saleu et al., 2018), we proposed an iterative two-step heuristic, composed of: a coding step that transforms a solution into a customer sequence, and a decoding step that decomposes the customer sequence into a tour for the vehicle and series of trips for the drone(s). Decoding was expressed as a bicriteria shortest path problem and carried out by dynamic programming. Dell'Amico, Montemanni, & Novellani (2020) presented a set of matheuristic methods. Kim & Moon (2018) considered a variant of the PDSTSP in which UAVs can not only be deployed from the depot but also from dedicated drone stations.

Ham (2018) studied a multi-truck multi-depot variant of the PDSTSP considering two different types of drone tasks: drop-off and pickup. After a drone finishes a delivery task, it can either fly back to the depot to deliver the next parcels or fly directly to another customer to pick up a returned parcel. The problem is solved with a constraint programming approach.

Ulmer & Thomas (2018) studied a dynamic version of the PDSTSP. They called their problem Same-Day Delivery Routing Problem with Heterogeneous Fleets (SDDPHF). The SDDPHF considers two fleets of vehicles and drones that deliver goods from the depot to customers who dynamically request services. When a new customer request arrives, a dispatcher has to decide whether or not to accept the customer for the same-day delivery and determine the according assignment and routing decisions. The SDDPHF takes into account a time window for each request, the loading time for both vehicles and drones, the time to drop off a package from a vehicle or a drone as well as the recharging or battery swap time for drones. The objective is to maximize the expected number of customers served the same day. To solve the problem, the authors proposed an adaptive dynamic programming approach called parametric policy function approximation (PFA).

In another class of truck-drone combination problems, all deliveries are assigned to UAVs: trucks are only used to carry the UAVs. Mathew et al. (2015) called this problem *Heterogeneous Delivery Problem* (HDP). They proved that the HDP is NP-hard and proposed an efficient reduction to the Generalized Traveling Salesman Problem. In Poikonen & Golden (2020), Poikonen introduced the *multi-visit drone routing problem* (MVDRP) which considers a tandem between a truck and a drone. The drone can take off from the truck with one or more packages to deliver to customers. The drone may

Table 1
UAVs and vehicles performing independent tasks.

Reference	Problem	#V	#D	Drone capacity	Solution methods
Murray & Chu (2015)	PDSTSP	1	m	1	MILP, heuristics
Mbiadou Saleu et al. (2018)	PDSTSP	1	m	1	MILP, iterative two-step heuristic
Dell'Amico et al. (2020)	PDSTSP	1	m	1	MILP, matheuristic
Ulmer & Thomas (2018)	SDDPHF	k	m	1	Adaptive dynamic programming
Ham (2018)	PDSTSP drop&pickup	k	m	≥ 1	Constraint programming
This work	**PDSMTSP**	$\boldsymbol{k}$	$\boldsymbol{m}$	**1**	**MILP, hybrid metaheuristic**

return to the truck to swap/recharge batteries, pick up a new set of packages, and launch again to customer locations. The goal of the MVDRP is to minimize completion time. To solve the problem, the author proposed a flexible heuristic. The solution method is extended to a truck and multi-drones model, named k-MVDRP. In Bin Othman, Shurbevski, & Nagamochi (2017), Othman et al. introduced two settings of the problem. In the first setting, while the drone is making a delivery, the truck is not allowed to wait at the last rendez-vous point nor to intercept the drone at any rendez-vous point more than once. Conversely, in the second setting, the truck is allowed to wait for the drone at the last rendez-vous point or to re-visit a rendez-vous point multiple times. They modeled this problem as a problem of finding a special type of path in a graph with a special structure, and proposed a polynomial-time approximation algorithm for each of the problem settings. Luo, Liu, & Shi (2017) studied the two-echelon cooperated ground vehicle and its carried UAV routing problem (2E-GU-RP) which considers a set of targets, each of whom must be served exactly once by the UAV. They provided an integer programming model and two heuristics.

Dayarian, Savelsbergh, & Clarke (2020) introduced another original truck-drone combination problem where delivery vehicles are regularly resupplied by drones. Resupply can take place whenever a delivery vehicle is stationary and a drone can land on the vehicle's roof. The problem is named Vehicle Routing Problem with Drone Resupply (VRPDR). The authors presented several policy function approximations to analyze different routing and assignment strategies.

Table 1 reports the main papers dealing with UAVs and vehicles performing independent tasks. Columns #V and #D indicate the number of vehicles and drones, respectively.

The table clearly shows that when the fleet is limited to a single truck, authors have focused on the PDSTSP. Furthermore, it shows that the natural extension of this problem to a fleet composed of multiple trucks have not been investigated so far. In this paper, we fill this gap by investigating this extension.

3. The Parallel Drone Scheduling Multiple Traveling Salesman Problem

In this section, we introduce the Parallel Drone Scheduling Multiple Traveling Salesman Problem (PDSMTSP), provide a MILP formulation and describe a simple branch-and-cut solution method.

3.1. Problem statement

Let $G = (N \cup \{0\}, A)$ be a complete directed graph where $N = \{1, \ldots, n\}$ is the customer set and 0 is the depot. A fleet of K vehicles and a fleet of M drones are available. Each vehicle delivers customers with a single tour starting from the depot, visiting a subset of customers and returning back to the depot. Drones are constrained to perform back and forth trips between the depot and the customers, delivering a single customer in each trip. Some customers are not eligible for drone delivery, because of practical constraints such as the limited payload or the flying range of drones. The subset of customers that can be served by a drone is denoted N_d. These customers are consistently called *drone-eligible* in the rest of the paper. Fig. 6 depicts a solution of the PDSMTSP on an instance with 10 customers, among which 5 are drone-eligible.

A travel time t_{ij} is incurred when a vehicle goes through an arc $(i, j) \in A$. A travel time $\widehat{t_i}$ is incurred when a drone serves a customer $i \in N_d$ ($\widehat{t_i}$ includes take-off, back and forth flight trips, service and landing times). Assuming that both the vehicles and the drones can start from the depot at time 0, the objective of the PDSMTSP is to minimize the delivery completion time, *i.e.*, the time at which all the vehicles and all drones are back to the depot, with the service of all customers carried out.

Clearly, the choice of completion time for the objective function is motivated by previous works on the subject and is somehow academic. In practice, service quality or other types of costs are likely to be considered, among which those related to: vehicle usage, energy, human resources, maintenance, logistics operations... Even so, completion time nicely reflects the objective of limiting as much as possible the time devoted to distribution and could equivalently represent a constraint on the time available for deliveries.

3.2. MILP formulation

In this section, we express the PDSMTSP with a MILP formulation. We introduce the following decision variables:

- $z_i = 1$ if customer i is visited by a vehicle, 0 if it is visited by a drone ($i \in N$);
- $x_{ijk} = 1$ if arc (i, j) belongs to vehicle tour k, 0 otherwise ($(i, j) \in A,\ 1 \leq k \leq K$);
- $w_{ij} = 1$ if arc (i, j) belongs to a vehicle tour, 0 otherwise ($(i, j) \in A$);
- $y_{im} = 1$ if customer i is assigned to drone m, 0 otherwise ($i \in N_d, 1 \leq m \leq M$);
- $T \geq 0$ indicating the completion time.

The model is:

$$\text{minimize } T \tag{1}$$

subject to

$$T \geq \sum_{(i,j)\in A} t_{ij} x_{ijk} \quad (1 \leq k \leq K) \tag{2}$$

$$T \geq \sum_{i\in N_d} \widehat{t_i} y_{im} \quad (1 \leq m \leq M) \tag{3}$$

$$z_i = 1 \quad (i \in N \setminus N_d) \tag{4}$$

$$\sum_{1\leq m\leq M} y_{im} = 1 - z_i \quad (i \in N_d) \tag{5}$$

$$w_{ij} = \sum_{1\leq k\leq K} x_{ijk} \quad ((i, j) \in A) \tag{6}$$

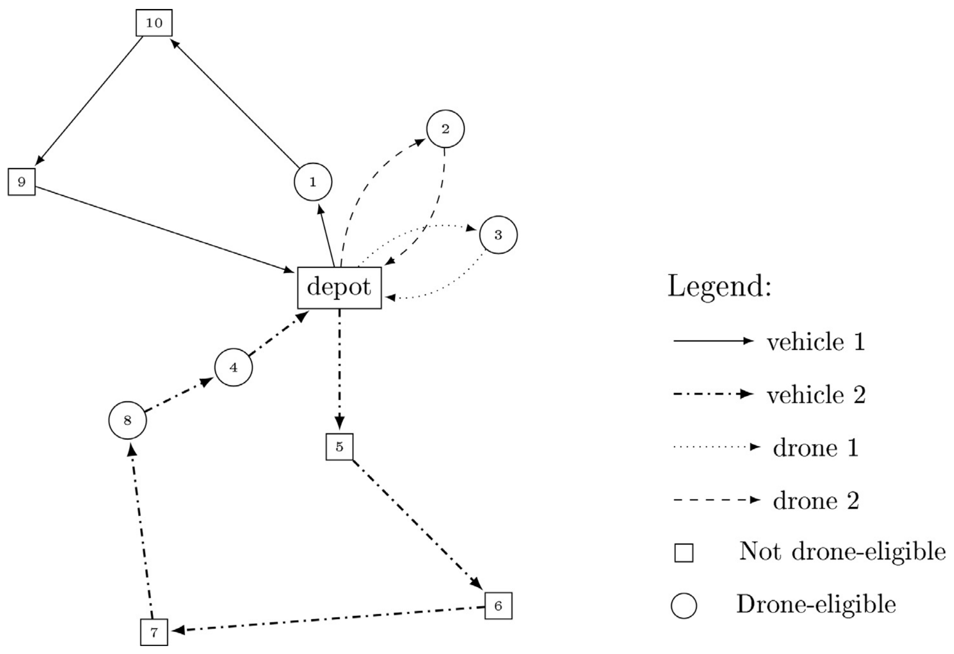


Fig. 6. A set of 10 customers served by 2 vehicles and 2 drones.

$$\sum_{(i,j)\in A} w_{ij} = z_i \quad (i \in N) \tag{7}$$

$$\sum_{(0,j)\in A} x_{0jk} \le 1 \quad (1 \le k \le K) \tag{8}$$

$$\sum_{(i,j)\in A} x_{ijk} = \sum_{(j,i)\in A} x_{jik} \quad (i \in N,\ 1 \le k \le K) \tag{9}$$

$$\sum_{j\in S} \sum_{l\in N\cup\{0\}\setminus S} w_{jl} \ge z_i \quad (S \subseteq N, S \ne \emptyset, i \in S) \tag{10}$$

$$z_i \in \{0,1\} \quad (i \in N) \tag{11}$$

$$x_{ijk} \in \{0,1\} \quad ((i,j) \in A,\ 1 \le k \le K) \tag{12}$$

$$w_{ij} \in \{0,1\} \quad ((i,j) \in A) \tag{13}$$

$$y_{im} \in \{0,1\} \quad (i \in N_d, 1 \le m \le M) \tag{14}$$

$$T \ge 0 \tag{15}$$

The objective (1) is to minimize completion time T. Constraints (2) and (3) give lower bounds on T expressed by completion times of each vehicle and drone. Constraints (4) ensure that customers that are not drone-eligible are served by a vehicle. Constraints (5) guarantee that every drone-eligible customer is also served, either by a vehicle or a drone. Constraints (6) and (7) express that if a customer is served by a vehicle, it is assigned to a vehicle tour. Constraints (8) stipulate that each vehicle leaves the depot at most once. Constraints (9) ensure flow conservation for vehicle tours. Subtour Elimination Constraints (SEC) are provided by constraints (10). These constraints ensure that given a non empty subset of customers $S \subseteq N$, if at least one customer in S is visited by a vehicle, there exists at least one outgoing arc from S. Finally, constraints (11) to (15) define decision variables.

3.3. Branch-and-cut procedure

The model described in the previous section could be used to solve exactly the problem with a MILP solver, but, because of the weak LP relaxation quality and the exponential size of subtour elimination constraints (9), it may only allow solving very small-size instances. In order to address slightly larger instances, we implemented a simple branch-and-cut algorithm in which subtour elimination constraints (SECs) are dynamically added to the formulation. When, at any node of the branch-and-bound tree, the flow supported by the linear relaxation contains a subtour, the corresponding SEC is added to the formulation.

The branch-and-cut is implemented with CPLEX and exploits the possibilities offered by this solver. We respectively use *Lazy Constraint Callbacks* and *User Cut Callbacks* to add SECs when incumbent solutions are integer or fractional. We implemented two separation algorithms to cover these two cases.

The separation of SECs for integer solutions amounts to detecting a circuit in a graph. This is achieved by a simple search algorithm on the graph defined by arcs (i,j) with $w_{ij} > 0$. Any circuit not connected to the depot gives a new SEC.

When the incumbent solution is fractional, we build an auxiliary graph $G_{aux} = (V_{aux}, A_{aux})$ such that $V_{aux} = \{0\} \cup \{i \in N : z_i > 0\}$ and $A_{aux} = \{(i,j) \in A : w_{ij} > 0\}$. With each arc $(i,j) \in A_{aux}$, we associate a capacity $c_{ij} = w_{ij}$. Then, for each customer $i \in V_{aux}$, we apply Ford and Fulkerson algorithm (Ford Jr & Fulkerson, 2015) to compute the maximal flow between the depot and i. Given $i \in V_{aux}$, capacity constraints ensure that the maximal flow cannot exceed z_i. If it is less than z_i, the customers not reached during the search for an augmenting chain form a set S from which a new SEC is generated.

4. Hybrid metaheuristic

In this section we describe the solution approach that we proposed. As already explained, it extends a procedure developed in Mbiadou Saleu et al. (2018) for the case with a single vehicle (PDSTSP). We quickly recall this procedure below.

The procedure starts by building a TSP tour τ (also called giant tour) visiting all the customers. This tour is then decomposed into a subsequence $\tau_{vehicle}$ and a complementary subset π_{drones}, which respectively give a tour for the vehicle and a set of customers assigned to the drones. This decomposition is performed by dynamic programming, with a labeling mechanism. Vehicle tour $\tau_{vehicle}$ is then reoptimized with the Lin-Kernighan heuristic (Helsgaun, 2000). The assignment of customers from π_{drones} to individual drones is solved as a Parallel Machine Scheduling (PMS) problem with a greedy heuristic. The next step is to reconstruct a new giant tour that will be used for the subsequent iteration. This is carried out by inserting in the vehicle tour all the customers served by drones, with a randomized best insertion strategy.

The solution method developed for the PDSMTSP basically follows the same idea, but with some important changes. Decomposing the giant tour τ becomes much harder because a tour is expected for each one of the K vehicles. To preserve acceptable computing times, we heavily restrict the possibilities in the decomposition. Roughly speaking, vehicle tours will correspond to successive subsequence of τ: having two subsequences that overlap is not allowed. For that reason, the quality of the decomposition cannot be entirely ensured and we introduce some quick local search operators to intensify the search. The heuristic is coined hybrid metaheuristic to reflect the fact that it combines components from different paradigms (a global framework similar to an iterated local search, dynamic programming to decompose the giant tour, a greedy algorithm to optimize drone operations...) as will be detailed hereafter.

In Section 4.1, we describe in more details the general scheme of the heuristic. Section 4.2 presents the different local search moves implemented. The decoding procedure, decomposing the giant tour, is detailed in Section 4.3.

4.1. General scheme of the hybrid metaheuristic

We adopt the following notation. A solution S is represented as a vector of K customer sequences and M sets $(\tau_1, \tau_2, \ldots, \tau_K, \pi_1, \ldots, \pi_M)$, where $\tau_1, \tau_2, \ldots, \tau_K$ indicate the visit order of the customers for the K vehicles and π_1 to π_M the sets of assigned customers for the M drones. We denote $c^1_k(S)$ the completion time for vehicle k in solution S and $c^2_m(S)$ the completion time for drone m. Finally, $c(S) = \max(c^1_1(S), \ldots, c^1_K(S), c^2_1(S), \ldots, c^2_M(S))$ is the solution cost. To illustrate the notation, let us consider the solution depicted in Fig. 6: the solution vector is $((1, 10, 9), (5, 6, 7, 8, 4), (3), (2))$.

The general scheme of the heuristic is summarized in Algorithm 1 and is explained hereafter.

Algorithm 1 General scheme of the hybrid metaheuristic.

```
1:  $\tau \leftarrow solveTSP()$
2:  $bestSol \leftarrow (\tau_1^{rec} = \tau, \tau_2^{rec} = \emptyset, \ldots, \tau_K^{rec} = \emptyset, \pi_1^{rec} = \emptyset, \ldots, \pi_M^{rec} = \emptyset)$
3:  while the computing time limit is not reached do
4:      $(\tau_1, \tau_2, \ldots, \tau_K, \pi_{drones}) \leftarrow split(\tau)$
5:      $(\tau_1^{opt}, \tau_2^{opt}, \ldots, \tau_K^{opt}) \leftarrow reoptimizeTSP(\tau_1, \tau_2, \ldots, \tau_K)$
6:      $(\pi_1^{opt}, \ldots, \pi_M^{opt}) \leftarrow optimizePMS(\pi_{drones})$
7:      $(\tau_1^{imp}, \tau_2^{imp}, \ldots, \tau_K^{imp}, \pi_1^{imp}, \ldots, \pi_M^{imp}) \leftarrow improveSol(\tau_1^{opt}, \tau_2^{opt}, \ldots, \tau_K^{opt}, \pi_1^{opt}, \ldots, \pi_M^{opt})$
8:      if solution $(\tau_1^{imp}, \tau_2^{imp}, \ldots, \tau_K^{imp}, \pi_1^{imp}, \ldots, \pi_M^{imp})$ is better than $bestSol$ then
9:          $bestSol \leftarrow (\tau_1^{imp}, \tau_2^{imp}, \ldots, \tau_K^{imp}, \pi_1^{imp}, \ldots, \pi_M^{imp})$
10:     end if
11:     $\tau \leftarrow buildGiantTour(bestSol)$
12: end while
```

Initialization (Lines 1 and 2). We initialize the algorithm with a giant TSP tour τ where a vehicle is visiting the depot and all the customers. This tour is obtained with a nearest-neighbor construction procedure. It provides a starting solution $(\tau_1 = \tau, \tau_2 = \emptyset, \ldots, \tau_K = \emptyset, \pi_1 = \emptyset, \ldots, \pi_M = \emptyset)$, where all the customers are visited by the first vehicle in the order of sequence τ. No customers are assigned to other vehicles nor to the drones.

Decoding (Line 4). Procedure *split* decomposes sequence τ in K complementary subsequences: $\tau_1, \tau_2, \ldots, \tau_K$ for customers assigned to the K vehicles and a set π_{drones} for customers assigned to the fleet of drones. This complex procedure is detailed in Subsection 4.3.

Route reoptimization (Line 5). Every vehicle route τ_1 to τ_K is reoptimized with the TSP Lin-Kernighan heuristic, using Helsgaun's implementation (Helsgaun, 2000).

Drone assignement (Line 6). The output of the decoding procedure does not provide a detailed planning for the drones; it only indicates which customers will be served by drones. The assignment of customers to individual drones is a PMS problem and is solved with the well-known longest processing time heuristic (Pinedo & Hadavi, 1992).

Local search (Line 7). Procedure *improveSol* performs local search moves to improve the current solution. These moves and the local search strategy are described in Section 4.2.

Update of the current solution (Lines 8 and 10).] The new solution $(\tau_1^{imp}, \tau_2^{imp}, \ldots, \tau_K^{imp}, \pi_1^{imp}, \ldots, \pi_M^{imp})$ is compared with *bestSol* and the latter is updated if needed. A solution S is considered to be better than another solution S' if one of the two following conditions hold:

- $c(S) < c(S')$
- $c(S) = c(S')$ and $\sum_{k=1}^{K} c^1_k(S) + \sum_{m=1}^{M} c^2_m(S) < \sum_{k=1}^{K} c^1_k(S') + \sum_{m=1}^{M} c^2_m(S')$

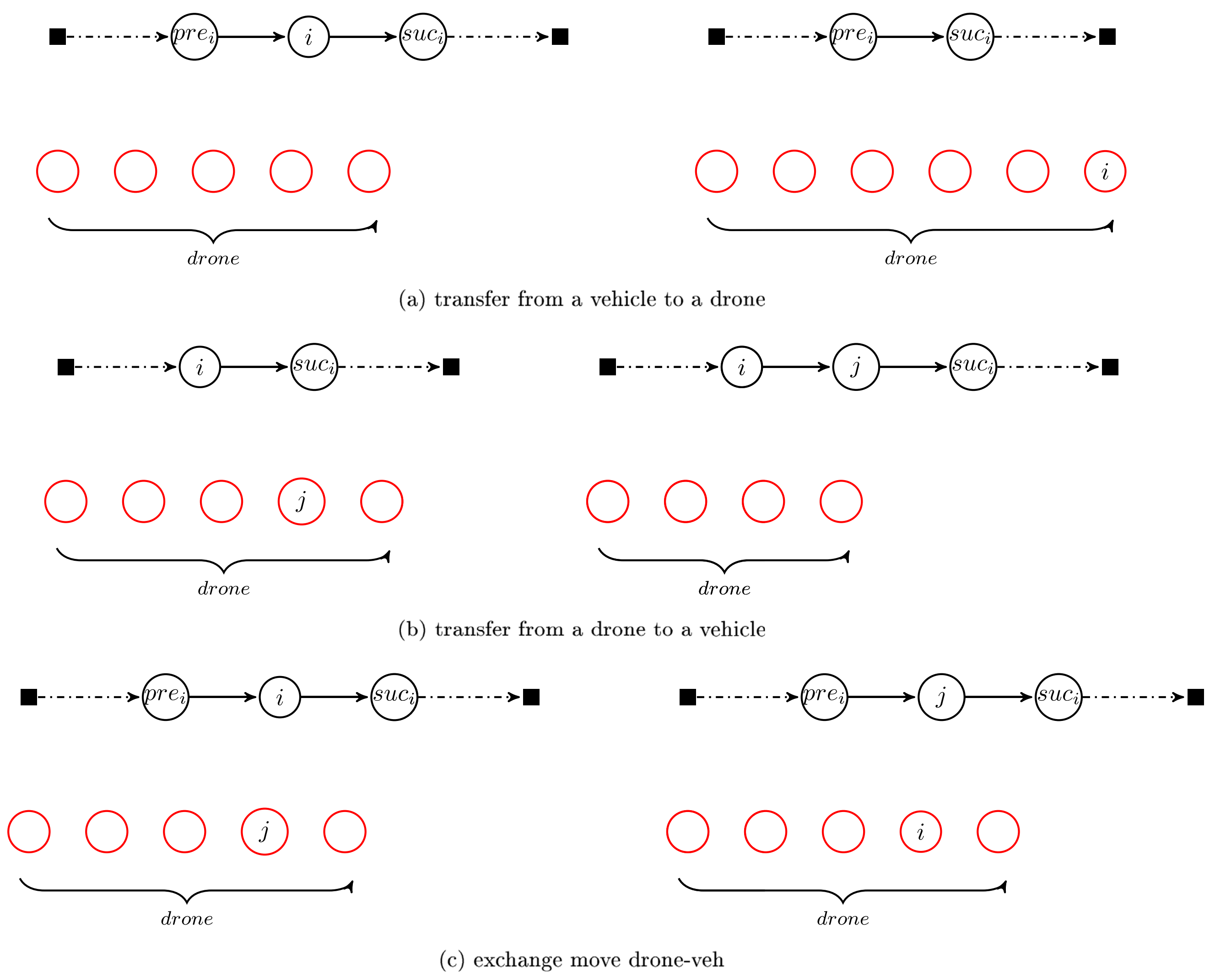
This way, when two solutions have the same objective value, the one that minimizes the sum of completion times is preferred.

Construction of the new giant tour (Line 11). The tours of the best solution are first concatenated in a random order. Then, customers assigned to drones are randomly inserted. The resulting giant tour is optimized with 2-opt.

Note that this algorithm can be interpreted as an Iterated Local Search (Lourenço, Martin, & Stützle, 2003) (ILS). Each iteration of the algorithm ends with a local optimum. This local optimum is perturbed by randomly constructing a new giant tour and decoding this giant tour. The new solution is then improved by local search until a new local optimum is found. The main focus, and originality, of our algorithm is the way the perturbed solution is obtained. The objective at this step is to both avoid restarting from a random solution and being attracted to the same local optimum. The giant tour construction procedure ensures that a large quantity of information from the best solution is preserved. The projection to the space of permutations enables large moves in the solution space, and thus limits the risk that the same local optima are repeatedly explored.

The drawback of this approach is the time spent when decoding solutions. Different mechanisms have been implemented to limit this time. However, this step remains the bottleneck of the algorithm. Seeing the complexity of each iteration, we decided to accept new solutions according to a better-walk mechanism, that is, only improving solutions are accepted.

The three main novelties of this algorithm compared to the one developed in our previous work (Mbiadou Saleu et al., 2018) are:



pre_i
i
suc_i
drone
(a) transfer from a vehicle to a drone
j
(b) transfer from a drone to a vehicle
(c) exchange move drone-veh

Fig. 7. Local search operators between a vehicle and a drone (left-hand side: before the move; right-hand side: after the move).

- The introduction of ad hoc local search moves to complement the decomposition procedure and converge towards better solutions.
- The modification of the split procedure to handle multiple vehicles, with new label definition, extension rules, lower bounds and upper bounds.
- The introduction of the Iterated Local Search framework to improve the exploration of the search space.

4.2. Local search

Figs. 7 and 8 describe the different types of local search moves that we implemented, respectively between a vehicle and a drone, or between two vehicles:

- *Tranfer move* (Fig. 7(a), Fig. 7(b)): a drone-eligible customer is transferred from a vehicle to a drone or from a drone to a vehicle.
- *Exchange move drone-veh* (Fig. 7(c)): a drone-eligible customer in a vehicle tour and a drone customer are exchanged.
- *Relocate move* (F Fig. 8(a)): a customer located in a vehicle route is moved to another vehicle route.
- *Exchange move veh-veh* (Fig. 8(b)): two customers of two different vehicle routes are interchanged.
- *Cross move* (Fig. 8(c)): two sequences of customers are exchanged by crossing two edges in two different routes.

The moves are applied until a local optimum is obtained. If the maximal completion time is due to a vehicle, the above order is followed when applying moves. If it is by a drone, the search is limited to the transfer and exchange moves, in this order. A first improvement strategy is employed: as soon as an improving move is found, the solution is updated and the neighborhoods are explored again from the beginning.

In all cases, the search is restricted to neighbor solutions that involve the vehicle or drone with the maximal completion time. Indeed, no improvement is possible otherwise. When a customer is moved from this vehicle/drone, it is first tentatively moved to the vehicles/drones with minimum completion time.

The transfer neighborhood has a size $O(n \times M)$ when the transfer is from a vehicle to a drone, $O(n \times |N_d|)$ otherwise. The size for the exchange move between a drone and a vehicle is also $O(n \times |N_d|)$. Other neighborhoods have a size $O(n^2)$. Every solution in every neighborhood can be evaluated in constant time $O(1)$.

4.3. Decoding procedure split(τ)

In this section, we explain how K vehicle tours and a set of customers assigned to drones are extracted from a sequence τ with

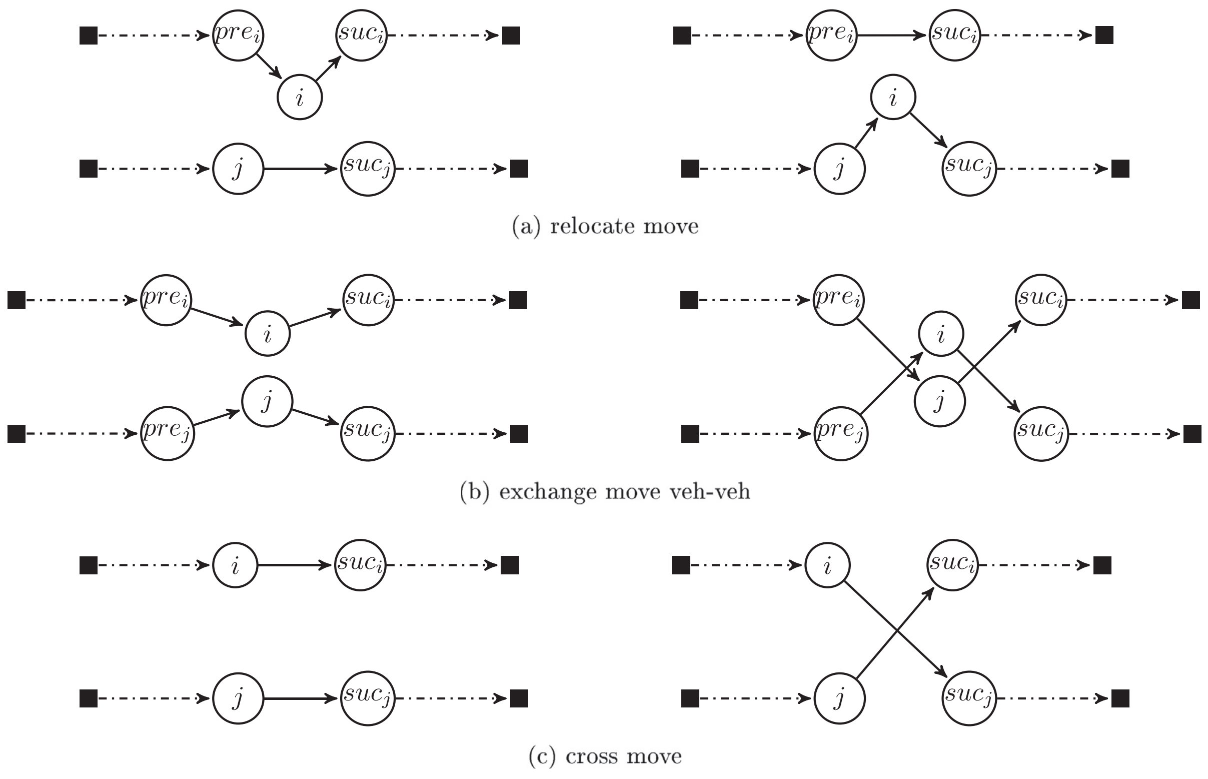


Fig. 8. Local search operators between two vehicles (left-hand side: before the move; right-hand side: after the move).

the *split* procedure. We introduce the following notation. $i <_\tau j$ indicates that i precedes j in τ; $j = pred_\tau(i)$ means that j is the direct predecessor of i in τ; equivalently, $j = succ_\tau(i)$ means that j is the direct successor of i. Given i and j with $i <_\tau j$, $time_\tau(i, j)$ is the path length (in terms of the sum of travel times) from i to j by following arcs in τ; to simplify further notation, $time_\tau(i, i)$ is also introduced and set to 0.

Procedure $split(\tau)$ decomposes sequence τ by assigning each customer to one of the vehicles (sequences $\tau_1, \dots \tau_K$) or to the set of drone customers (set π_{drones}). It respects the following constraints: if $i <_\tau j$ and $i, j \in \tau_k$ then $i <_{\tau_k} j$; if $i <_\tau j$, $i \in \tau_k$ and $j \in \tau_l$ with $k \neq l$, then $k < l$. Intuitively, these constraints mean that the decoding procedure cut sequence τ in K pieces: each piece contains the customers assigned to a vehicle plus some other customers assigned to drones.

We perform this decomposition by introducing an acyclic directed graph and solving a multi-criteria shortest path problem Climaco & Martins (1982) by dynamic programming.

Definition of the acyclic graph

We introduce $G^\tau = (V^\tau, A^\tau)$ an acyclic graph defined as follows. $V^\tau = \{0, 1, 2, \dots, n, n+1\}$ represents the set of customers completed by two copies of the depot: the original depot 0 and its copy $n+1$. In the multi-creteria shortest path problem, 0 will be the origin and $n+1$ the destination. Given i and j in V^τ, with $i \neq j$, arc (i, j) exists in A^τ if and only if the two following conditions are both satisfied:

- $i <_\tau j$
- all the customers between i and j in τ are drone-eligible: $i <_\tau \mu <_\tau j \Rightarrow \mu \in N_d$

With every arc $(i, j) \in A^\tau$, we associate a cost vector (c^1_{ij}, c^2_{ij}). The first cost component c^1_{ij} represents the cost incurred if a vehicle travels directly from i to j: $c^1_{ij} = t_{ij}$. The second cost component c^2_{ij} represents the corresponding cost induced for the drones. If the vehicle travels directly from i to j, all customers μ in-between (which by definition of A^τ are all drone-eligible) are assigned to the drones: $c^2_{ij} = \frac{1}{M} \times \sum_{\{\mu \in V^\tau :\ i <_\tau \mu <_\tau j\}} \widehat{t}_\mu$. This value does not exactly give the contribution of these customers to the completion time of the drones, which could only be obtained by solving a PMS problem, but provides an optimistic (lower-bound) value and a good approximation.

Fig. 9 shows an illustrative example for an instance with 5 customers. Customers 2 and 4 are not drone-eligible. The vehicle travel cost matrix is reported in Table 2(a) in which we add the copy of the depot (node 6). Table 2(b) presents the drone-trip costs for drone-eligible customers. We assume $\tau = (1, 2, 3, 4, 5)$ and a single drone.

Dynamic programming scheme

Paths starting from node 0 in graph G^τ can be matched with partial decompositions of sequence τ. The information about this decomposition is captured in a label associated with the path. A label is defined with the following attributes:

- n^{veh} is the index of the current vehicle; the tours (sequences τ_k) of vehicles with index $k < n^{veh}$ are known; the tours of vehicles with index $k > n^{veh}$ are still empty;
- c^{preVeh} is the cost of the longest of the tours that are already known ;
- c^{curVeh} is the cost of the current vehicle tour;
- c^{drone} is the approximated drone cost;
- *pred* is the label from which this label was obtained.

We note $L = (n^{veh}, c^{preVeh}, c^{curVeh}, c^{drone}, pred)$. Equivalently, the atributes of a label L are denoted $n^{veh}(L)$, $c^{preVeh}(L)$ and so on.

The dynamic programming procedure is described in Algorithm 2. The principle is to progressively associate a list of labels $\mathcal{L}(i)$ with each node i of G^τ. At the initialization, $\mathcal{L}(i)$ is set to $\emptyset$ for every node i (Line). The procedure then starts by assigning label $(1, 0, 0, 0, \emptyset)$ to $\mathcal{L}(0)$ (Line 2).

Algorithm 2 Procedure *split*(τ).

```
1: $\mathcal{L}(i) \leftarrow \emptyset$ for $0 \le i \le n+1$
2: $\mathcal{L}(0) \leftarrow \{(1, 0, 0, 0, \emptyset)\}$
3: $i \leftarrow 0$
4: while $i <_\tau n+1$ do
5:     increment $i$
6:     for all $j$ subject to $(j, i) \in A^\tau$ do
7:         for all $L \in \mathcal{L}(j)$ do
8:             $L' \leftarrow$ extend $L$ by adding arc $(j, i)$ to the current tour
9:             if $L'$ is not pruned by bounding mechanisms then
10:                insert with dominance $L'$ in $\mathcal{L}(i)$
11:            end if
12:            if extending $L$ by starting a new vehicle tour should be considered then
13:                $L' \leftarrow$ extend $L$ by adding arc $(j, 0)$ to the current tour and starting a new tour with arc $(0, i)$
14:                if $L'$ not pruned by bounding mechanisms then
15:                    insert with dominance $L'$ in $\mathcal{L}(i)$
16:                end if
17:            end if
18:        end for
19:    end for
20: end while
21: $L^* \leftarrow$ best label in $\mathcal{L}(n+1)$
22: derive $\tau_1, \tau_2, \ldots \tau_K$ by backtracking from $L^*$
23: $\pi_{drones} \leftarrow$ nodes not in $\tau_1 \cup \tau_2 \cup \ldots \cup \tau_K$
```

The procedure goes through the graph from the origin depot node 0 to the destination depot node $n+1$ by following the order defined in sequence τ. Function *increment* on Line 5 is introduced for this purpose. The list of labels $\mathcal{L}(i)$ of a given node i is obtained by extending labels in the list of labels of its predecessor nodes (Lines 6–19). For any predecessor node j and any label L of $\mathcal{L}(j)$, two possible extensions may occur:

1. A first extension is to add arc (j, i) to the current vehicle tour. The new generated label L' is defined by (Line 8):
$$\begin{cases} n^{veh}(L') = n^{veh}(L) \\ c^{preVeh}(L') = c^{preVeh}(L) \\ c^{curVeh}(L') = c^{curVeh}(L) + c^1_{ji} \\ c^{drone}(L') = c^{drone}(L) + c^2_{ji} \\ pred(L') = L \end{cases}$$
2. A second extension consists in closing the tour of the current vehicle and starting a new vehicle tour with an available vehicle. This extension is not allowed when: $j = 0$, $i = n+1$ or $n^{veh}(L) = K$; furthermore, the extension is not considered when pursuing with the current vehicle does not increase the current global cost: indeed, it would then be suboptimal to start the tour of a new vehicle (Line 12). When the extension is carried out, the new label L' is given by (Line 13):
$$\begin{cases} n^{veh}(L') = n^{veh}(L) + 1 \\ c^{preVeh}(L') = \max(c^{preVeh}(L), c^{curVeh}(L) + t_{j,0}) \\ c^{curVeh}(L') = t_{0,i} \\ c^{drone}(L') = c^{drone}(L) + c^2_{ji} \\ pred(L') = L \end{cases}$$

In order to limit the number of labels generated in the lists as the procedure advances in the graph, some bounding mechanisms (Lines 9 and 14) and a dominance rule (Lines 10 and 15) are introduced. These two components are detailed below.

At the end, the vehicle tours are retrieved through a backtracking mechanism based on attribute *pred* and starting with the best label found in $\mathcal{L}(n+1)$, *i.e.*, the label that minimizes $\max(c^{preVeh}(L), c^{curVeh}(L), c^{drone}(L))$. Nodes that are not in the vehicle tours are assigned to the drones.

Insertion with dominance

When a label has to be added to a list of labels, dominance tests are tried. These tests are done with all the labels in the list, both to check whether the new label should be discarded or if an existing label should be removed. The dominance rule is very simple. Given a node $i \in \tau$ and two labels L_a and L_b in $\mathcal{L}(i)$, L_a dominates L_b if the following inequalities are all satisfied:
$$\begin{cases} n^{veh}_a \le n^{veh}_b \\ \max\left(c^{preVeh}(L_a), c^{curVeh}(L_a) + t_{i,0}, c^{drone}(L_a)\right) \\ \le \max\left(c^{preVeh}(L_b), c^{curVeh}(L_b) + t_{i,0}, c^{drone}(L_b)\right) \\ c^{curVeh}(L_a) \le c^{curVeh}(L_b) \\ c^{drone}(L_a) \le c^{drone}(L_b) \end{cases}$$

Upper bound generation

The bounding mechanisms rely on three upper bounds that are computed before starting the dynamic programming algorithm, for a given sequence τ. A first upper bound UB_1 is given by the value of the best solution found so far by the hybrid algorithm. The other bounds rely on Algorithm 3. This algorithm computes the minimal cost $\chi(i, k)$ that can be achieved to serve all customers up to i in the order of sequence τ using k vehicles ($i \in \tau$, $1 \le k \le K$). Compared to Algorithm 2 the possibility to serve customers with drones is ignored. Algorithm 3 applies the following recursion:
$$\begin{cases} \chi(i, 1) = time_\tau(0, i) + t_{i,0} \quad (i \in \tau), \\ \chi(0, k) = 0 \quad (2 \le k \le K), \\ \chi(i, k) = \min_{j <_\tau i}\Big(\max\Big(\chi(j, k-1), t_{0,succ_\tau(j)} \\ +time_\tau(succ_\tau(j), i) + t_{i,0}\Big)\Big) \quad (i \in \tau : i \ne 0, 2 \le k \le K). \end{cases} \tag{16}$$

The recursion introduces vertex j at which the last route starts, computes the associated completion time and states that $\chi(i, k)$ is given by the best splitting point j.

Algorithm 3 Procedure *decompose*(τ).

```
1: $\chi(i, 1) \leftarrow time_\tau(0, i) + t_{i,0}, \ \forall i \in \tau$
2: $\chi(0, k) \leftarrow 0, \ \forall k = 2.K$
3: for $k = 2.K$ do
4:     $i \leftarrow 0$
5:     while $i \ne n+1$ do
6:         $i \leftarrow succ_\tau(i)$
7:         $\chi(i, k) \leftarrow \min_{j <_\tau i}\left(\max\left(\chi(j, k-1), t_{0,succ_\tau(j)} + time_\tau(succ_\tau(j), i) + t_{i,0}\right)\right)$
8:     end while
9: end for
10: return $\chi$
```

Upper bounds UB_2 and UB_3 try to circumvent the extra difficulty of having multiple vehicles. They both aggregate the K vehicles into a single one. To do so, we introduce what we call the vehicle acceleration coefficient $\alpha = \alpha(K)$, which approximates the speedup needed for a single vehicle to execute the same set of deliveries than a fleet of K vehicles in the same amount of time. The two bounds differ in the value of this coefficient. Then, a heuristic *split* procedure is used to determine the customers assigned to drones and those assigned to the speedy vehicle. Algorithm 3 is finally used to determine the K vehicle tours for the latter.

More precisely, the second upper bound UB_2 works as follows:

1. Determine the *acceleration coefficient* $\alpha = \frac{\chi(n+1,1)}{\chi(n+1,K)}$; this coefficient approximates the gains obtained when K vehicles are used instead of 1;

$$\begin{tikzcd}
0 \arrow[r, "{8,0}"] \arrow[rr, bend left, "{9,16}"] & 1 \arrow[r, "{10,0}"] & 2 \arrow[r, "{13,0}"] \arrow[rr, bend left, "{8,12}"] & 3 \arrow[r, "{11,0}"] & 4 \arrow[r, "{5,0}"] \arrow[rr, bend left, "{6,20}"] & 5 \arrow[r, "{8,0}"] & 6
\end{tikzcd}$$

Fig. 9. Graph G^τ for the instance of Tables 2(a) and 2(b), with $\tau = (1, 2, 3, 4, 5)$.

$$\begin{tikzcd}
 & 0 \arrow[r, "{8,0}"] \arrow[rr, bend left, "{9,16}"] & 1 \arrow[r, "{10,0}"] & 2 \arrow[r, "{13,0}"] \arrow[rr, bend left, "{8,12}"] & 3 \arrow[r, "{11,0}"] & 4 \arrow[r, "{5,0}"] \arrow[rr, bend left, "{6,20}"] & 5 \arrow[r, "{8,0}"] & 6 \\
SP(i): & 51 & 43 & 33 & 24 & 13 & 8 & 0
\end{tikzcd}$$

Fig. 10. Values $SP(i)$ for the graph of Fig. 9 ($M = 1$).

Table 2
Illustrative instance.

	0	1	2	3	4	5	6
	(a) Vehicle cost matrix t_{ij}						
0	0	8	9	11	6	8	0
1		0	10	7	10	12	8
2			0	13	8	6	8
3				0	11	7	11
4					0	5	6
5						0	8
6							0
	(b) Drone cost vector $\hat{t}_i$						
Customer	1	3	5				
Drone cost	16	12	20				

2. Apply procedure $split(\tau)$ assuming a single vehicle, changing costs c^1_{ij} to $\frac{t_{ij}}{\alpha}$ to account for the acceleration, deactivating the bounding mechanisms and keeping only the best label at each node;
3. Consider the subsequence τ' obtained with this procedure and re-apply Algorithm 3 to decompose it into K vehicles tours $\tau_1, \tau_2, \ldots, \tau_K$ (these tours can be obtained backwardly from $\chi(n+1, K)$);
4. Solve the PMS problem with the longest processing time heuristic for the remaining customers;
5. Compute the completion time and set UB_2 to this value.

The third upper bound UB_3 is obtained as UB_2 but changing the acceleration coefficient α to K. UB is set as the best (minimal) upper bound among these three bounds: $UB = \min(UB_1, UB_2, UB_3)$.

Lower bound generation

We introduce a lower bound $LB^{tot}(i, L)$ on the minimal completion time that can be achieved when extending a label L associated with a customer node i. $LB^{tot}(i, L)$ requires to precompute value $SP(i)$ of the shortest path in G^τ between i and $n+1$, with modified arc costs set to $c^1_{ij} + M \times c^2_{ij}$. Values $SP(i)$ are computed for all nodes i with a single backward exploration of the acyclic (topologically-ordered) graph G^τ. They indicate the minimal total distance that will be traveled by the vehicles and drones between i and $n+1$ in the solution space defined by the $split(\tau)$ procedure. Fig. 10 reports $SP(i)$ for the example introduced with Fig. 9.

Then, given $i \in \tau$ and $L \in \mathcal{L}(i)$, the total distance traveled by vehicles $n^{veh}(L)$ to K and by the M drones is at least $c^{curVeh}(L) + M \times c^{drone}(L) + SP(i)$. Hence, at least one of those vehicles or drones cannot complete its duty before time $LB^{tot}(i, L)$ with:

$$LB^{tot}(i, L) = \frac{c^{curVeh}(L) + M \times c^{drone}(L) + SP(i)}{K - n^{veh}(L) + 1 + M}$$

Bounding mechanism

A label $L \in \mathcal{L}(i)$ is pruned if one the following rules applies:

- R1: $\max\left(c^{preVeh}(L), c^{curVeh}(L) + t_{i,0}, c^{drone}(L)\right) \geq UB$;
- R2: $LB^{tot}(i, L) \geq UB$.

Rule R1 identifies labels whose completion time is already at least UB. Rule R2 applies lower bound $LB^{tot}(i, L)$.

Fig. 11 illustrates the $split(\tau)$ subroutine on the graph presented in Fig. 9, with $K = 2$ and $M = 1$. List $\mathcal{L}(i)$ is reported under every node i. In Fig. 11(a), the bounding mechanisms are not applied. Labels can only be removed by dominance. Dominated labels are crossed out. The best label is indicated in red with a '*' symbol beside in list $\mathcal{L}(6)$, as well as labels from which it inherits (and that allow to reconstruct the different vehicle tours). In Fig. 11(b), the bounding mechanisms are reinserted. Applied on this sequence, the upper bound generation algorithms provide an upper bound $UB = 27$. The bounding rules enabling to delete a label are reported at the right of the strikethrough label. In this figure, $\mathcal{L}(6) = \emptyset$ because the upper bound cannot be improved. In both cases, the vehicle tours obtained with the procedure are: $\tau_1 = (0, 1, 2, 0)$ and $\tau_2 = (0, 4, 5, 0)$. Customer 3 is assigned to the drone. The cost of the solution is 27.

5. Experiments and results

In this section, the effectiveness of the proposed solution method is examined. The environment used for the computational work is Intel core(TM) i5-6200U CPU @ 2.30 gigahertz 2.4 gigahertz; 8 gigabyte RAM; Windows 10; 64 bits. Solution methods are implemented in C++ language.

The MILP of Section 3.2 is solved by using IBM Concert Technology and CPLEX 12.6 on an Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60 gigahertz with 2x8 cores 25M cache and 62.5 gigabyte of RAM.

5.1. Problem instances

Not being aware of any instances in the literature for the PDSMTSP, we have selected 20 instances from the CVRPLIB (a repository of instances for the Capacited Vehicle Routing Problem). Instances have been chosen among those with the depot near the barycenter of all customers. Instance size varies between 50 and 199 customers. The 20 instances are:

- 5 instances from set CMT (*Christofides, Mingozzi and Toth*, 1979): CMT1, CMT2, CMT3, CM4, CMT5
- 3 instances from set E (*Christofides and Eilon*, 1969): E-n51-k5, E-n76-k8, E-n101-k8
- 2 instances from set M (*Christofides, Mingozzi and Toth*, 1979): M-n151-k12, M-n200-k16
- 7 instances from set P (*Augerat, 1995*): P-n51-k10, P-n55-k7, P-n60-k10, P-n65-k10, P-n70-k10, P-n76-k5, P-n101-k4
- 3 instances from *Uchoa* et al. (2014) benchmark: X-n110-k13, X-n115-k10, X-n139-k10

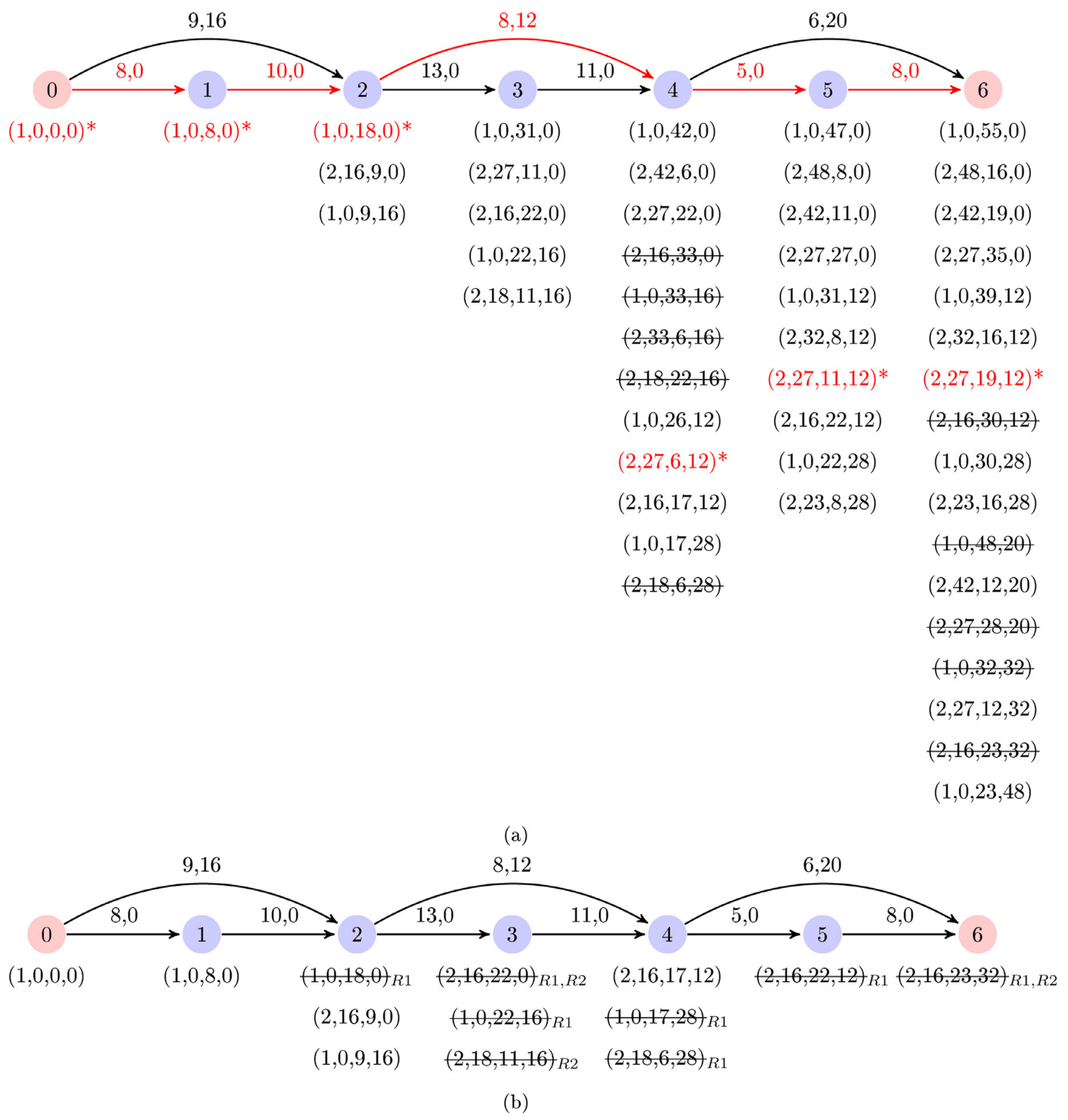


Fig. 11. Illustration of the *split* procedure on the graph of Fig. 9, without or with the bounding mechanisms ($K = 2$, $M = 1$). The last element of the definition of a label (*pred*) has not been represented for the sake of simplicity.

The number of customers in instances CMT1 to CMT5 is 50, 75, 100, 150 and 199, respectively. In other instances, the number of nodes (customers and depot) is indicated in the name, after letter ‘n’. To adapt these instances to our problem, we modified the fleet size and introduced drones. Precisely, we arbitrarily set $K = \lceil \frac{\text{fleet size}}{2} \rceil$ and $M = \lfloor \frac{\text{fleet size}}{2} \rfloor$. These values are reported in result tables. For all instances, we assume that all customers are drone eligible. We use the Manhattan distance for the vehicles and the Euclidean distance for drones. This is made possible because customers are represented by their coordinates x and y in all instances.

5.2. Limitation of the number of labels in procedure *split*(τ)

Seeing that the number of generated labels in procedure *split*(τ) can grow exponentially according to instance sizes, we proposed a method to limit this number. This method complements bounding mechanisms, previously presented, that might not be enough. The method is based on a threshold $\mathcal{T}_{max}$ that will limit the number of labels at a node $x \in \tau$ to at most $K \times \mathcal{T}_{max}$. To do so, we first group labels according to their value $n^{veh}(L)$. Then, in each group, we sort labels in the increasing order of value $c^{preVeh}(L) + c^{curVeh}(L) + c^{drone}(L)$ and only keep the $\mathcal{T}_{max}$ first labels.

To analyze the efficiency of this method, we conducted a set of experiments by varying parameter $\mathcal{T}_{max}$. We only considered the five CMT instances, which represents a sample of instances with different sizes. For each instance, we generated 25 sequence τ with a randomized version of the nearest neighbor heuristic. Then, for each sequence, we applied procedure *split*(τ) with different values of $\mathcal{T}_{max}$. Table 3 presents aggregated results. In the table, we report the average (Avg), maximum (Max) and standard deviation (SD) of the number of labels generated in the procedure, the relative error related to the difference of completion time value without and with limitation of labels and the computing time.

As expected, results show that when $\mathcal{T}_{max}$ increases, the number of labels generated in procedure *split*(τ) and the comput-

Table 3
Impact of parameter $\mathcal{T}_{max}$ on a set of five representative instances.

Instance	$\mathcal{T}_{max}$	#LabGen			Error (%)			CPU (s)		
		Avg	Max	SD	Avg	Max	SD	Avg	Max	SD
CMT1	10	32,592.76	41065	3221.32	11.96	22.61	6.32	0.05	0.06	0.004
(50,3,2)	20	58,203.16	71,635	5515.62	10.44	27.54	6.79	0.06	0.07	0.01
	30	77,896.36	96,438	7897.76	9.22	27.54	6.32	0.09	0.12	0.02
	40	93,623.92	116,711	10,518.46	7.86	19.40	6.06	0.08	0.10	0.013
	50	105,935.48	134,491	12,916.17	6.99	19.40	5.98	0.09	0.13	0.019
	60	116,271.88	148,682	14,477.09	5.72	19.40	5.62	0.13	0.20	0.03
	70	125,979.28	161,281	15,664.12	4.88	19.40	5.37	0.12	0.16	0.02
	80	135,242.32	171,434	16,660.01	4.71	16.69	5.11	0.13	0.20	0.03
CMT2	10	119,888.48	145,406	12,242.51	18.66	29.76	5.06	0.12	0.16	0.01
(75,5,5)	20	224,724.88	259,763	22,649.44	13.41	25.28	4.93	0.17	0.22	0.02
	30	317,329.20	36,0712	29,324.43	11.03	24.42	4.90	0.28	0.38	0.05
	40	398,036.44	442,525	34,354.79	9.48	20.93	4.50	0.32	0.46	0.06
	50	467,528.88	525,287	40,268.87	8.84	21.40	5.02	0.41	0.59	0.09
	60	529,162.64	60,5701	46,719.84	6.96	19.75	4.04	0.66	0.99	0.15
	70	586,593.72	676,084	51,625.91	6.17	17.44	3.70	0.58	0.90	0.13
	80	639,260.80	733,429	55,147.65	5.67	17.44	3.71	0.68	1.15	0.16
CMT3	10	170,724.80	217,266	16,096.19	16.43	25.62	4.83	0.19	0.25	0.02
(100,4,4)	20	314,293	393,205	32,492.02	14.18	21.49	4.37	0.24	0.32	0.03
	30	448,786.28	549,272	44,498.56	12.80	22.03	4.82	0.42	0.71	0.09
	40	567,207.48	690,704	58,322.19	11.98	19.17	4.43	0.49	0.71	0.10
	50	677,780.20	824,038	70,712.04	11.45	17.95	4.55	0.65	0.96	0.13
	60	779,789.40	943,959	81,315.07	10.21	17.95	4.52	1.02	1.54	0.26
	70	867,494.60	1,047,865	91,286.82	9.77	16.95	4.10	0.98	1.52	0.24
	80	947,767.12	1,142,010	102,206.65	9.17	16.95	3.98	1.21	2.08	0.31
CMT4	10	547,245.72	609,538	39,888.51	20.98	29.52	4.36	0.55	0.67	0.04
(150,6,6)	20	1,045,624.48	1,172,400	74,623.51	19.64	27.11	4.42	0.77	0.88	0.06
	30	1,514,593.56	1,737,610	115,363.92	18.33	25.71	4.10	1.28	1.60	0.17
	40	1,947,959.24	2,246,024	149,516.15	17.14	26.04	4.84	1.57	1.96	0.18
	50	2,359,557.20	2,750,357	177,303.68	16.40	23.81	4.45	2.10	2.67	0.25
	60	2,730,814.68	3,206,762	209,068.25	15.36	23.81	4.41	2.63	3.30	0.32
	70	3,079,424.64	3,607,688	233,764.46	15.12	22.97	4.17	3.28	4.15	0.49
	80	3396502.48	3996079	258893.28	13.83	20.95	3.94	3.73	4.74	0.47
CMT5	10	1419540.92	1643841	108970.19	22.94	33.33	3.94	1.58	2.13	0.20
(199,9,8)	20	2711808.24	3096235	195440.91	21.49	28.73	2.84	2.37	3.63	0.37
	30	3929397.12	4468400	284471.40	20.36	27.59	2.73	3.22	4.36	0.47
	40	5063679.84	5739811	348592.79	19.70	26.52	2.43	4.37	6.08	0.67
	50	6142347.12	6936879	427684.42	18.71	27.31	2.53	5.80	8.99	1.16
	60	7177526.08	8181177	486045.38	17.96	24.35	2.04	7.20	10.71	1.36
	70	8146948.64	9243250	567250.70	16.91	24.14	2.94	9.24	15.42	2.08
	80	9029270.76	10223006	635457.72	15.66	23.80	3.18	10.57	16.40	2.21

ing time increase while the relative error decreases. In view of these results, we considered that the best trade-off between solution quality (low error rate) and execution time was obtained with $\mathcal{T}_{max} = 60$. For the remainder of the experiments, when it is indicated that labels are limited, we keep this value.

5.3. Solution with the branch-and-cut algorithm

In this section, we report the results obtained when solving the PDSMTSP with the branch-and-cut algorithm presented in Section 3.2. We limit these experiments to the smallest instances ($n \leq 100$). The time limit is set to 3 hours (10,800 seconds). Results are detailed in Table 4.

The table is organized as follows. Gap1 is the percentage gap between the value of the linear relaxation z_{lr} at the root node (the integrity constraints are relaxed but the SECs are applied) and the cost of the best integer solution z_{ub} (Gap1 $= 100 \times \frac{z_{ub}-z_{lr}}{z_{ub}}$). Gap2 is the percentage gap between the best lower bound z_{lb} and z_{ub} (Gap2 $= 100 \times \frac{z_{ub}-z_{lb}}{z_{ub}}$). #Nodes gives the number of nodes explored in the branch-and-cut tree (a zero indicates that the process ended at the root node. #LC and #UC report the number of SECs added with LAZYCONSTRAINTCALLBACK (used to search for constraints violated by an integer solution) and USERCUTCALLBACK (used to search for constraints violated by a fractional solution), respectively. CPU is the computing time in seconds. #Veh and #Dro provide the number of vehicles and drones used in the best integer solution. #D.C. is the number of customers assigned to the drones in this solution. C.T. is its completion time (z_{ub}).

In Table 4, we can see that the branch-and-cut algorithm largely fails to solve the PDSMTSP in 3 hours. No optimal solutions are found (or, at least, no proven optimal). Furthermore, after 3 hours, and even for the smaller instances, gaps remain very large.

5.4. Hybrid metaheuristic and variants

In this section, we finally evaluate the hybrid metaheuristic, called **HM** hereafter. In order to investigate how results are impacted by the different components in **HM**, we introduce several variants and compare our results to those obtained with these variants:

HMb. The reconstruction of the giant tour is modified (Line 11 of Algorithm 1). We consider a parameter X. The idea of this new giant tour reconstruction is to concatenate the first X vehicle tours of the best solution in a random order and then insert the customers assigned to the drones and the customers of the remaining $K - X$ vehicles via best insertion. Initially the value of X is set to $X = K$. This value changes and is updated as follows: if the current solution is better than bestSol (Line 8 of Algorithm 1), the value of X is set to K (we intensify the search around the new best solution) otherwise the value of X is set to max $(0, X - 1)$ (we decrement X if $X > 0$ to diversify).

Table 4
Exact solution of the PDSMTSP with the branch-and-cut algorithm.

Instance	Branch-and-cut details						Solution details			
	Gap1 (%)	Gap2 (%)	#Nodes	#LC	#UC	CPU (s)	#Veh.	#Dro.	#D.C.	C.T.
CMT1 (50,3,2)	22.79	22.41	20100	51	23726	10,668.5	3	2	11	188
CMT2 (75,5,5)	97.20	97.20	957	49	20101	10,653.8	0	1	75	3630.86
CMT3 (100,4,4)	96.45	96.45	732	44	14515	10,650.8	1	4	94	4537.11
E-n51-k5 (50,3,2)	22.79	22.41	20210	50	23302	10,668.8	3	2	11	188
E-n76-k8 (75,4,4)	95.73	95.73	2954	38	13712	10,606	0	4	75	2975.51
E-n101-k8 (100,4,4)	96.45	96.45	732	44	14515	10,650.4	1	4	94	4537.11
P-n51-k10 (50,5,5)	64.74	64.63	2896	47	26372	10,616.4	4	5	13	230
P-n55-k7 (54,4,3)	67.24	67.05	5834	94	17239	10,651.9	3	3	11	308
P-n60-k10 (59,5,5)	65.87	65.73	5842	48	23596	10,560.4	5	5	12	246
P-n65-k10 (64,5,5)	83.69	83.68	2782	76	19446	10,615.6	5	5	14	580
P-n70-k10 (69,5,5)	96.86	96.86	2570	44	18956	10,644.1	1	5	68	3166.25
P-n76-k5 (75,3,2)	35.26	35.23	2776	70	15555	10,620.2	3	2	11	280
P-n101-k4 (100,2,2)	93.20	93.19	991	39	14748	10,581.5	1	2	99	4725.47

Table 5
Solution values.

Instance	LB	Method									
		HM	HMb	MS	HM(LL)	HMb(LL)	MS(LL)	HM(UB)	HMb(UB)	MS(UB)	B&C
CMT1 (50,3,2)	145.86	168	168	188	**166**	168	196	174	174	204	188
CMT2 (75,5,5)	101.54	**130.23**	133.60	148	132	133.41	152	140	140	152	3630.86
CMT3 (100,4,4)	160.86	**184**	186	208	187.04	186.17	204	195.42	197.24	216	4537.11
CMT4 (150,6,6)	115.62	**160.38**	150	184	162	162	180	166	164	192	
CMT5 (199,9,8)	72.65	**138**	139.29	152	140	**138**	154	142.04	140	152	
E-n51-k5 (50,3,2)	145.86	**168**	**168**	182	**168**	**168**	180	168.86	174	196	188
E-n76-k8 (75,4,4)	126.95	**154**	**154**	168	156	156	182	161.86	174	196	2975.51
E-n101-k8 (100,4,4)	160.86	186	**184**	208	188	190.17	224	196	196	216	4537.11
M-n151-k12 (150,6,6)	116.23	**154**	158.96	186	164	162	182	168	169.88	182	
M-n200-k16 (199,8,8)	80.69	**144**	148	162	148	146	156	152	152	168	
P-n51-k10 (50,5,5)	81.34	**111.07**	114	118	112.69	114	122	118	118	133.25	230
P-n55-k7 (54,4,3)	101.46	128	128	138	128	**126**	142	130	132	148	308
P-n60-k10 (59,5,5)	84.30	**114**	116	124	114.86	116	124	122	120	124	246
P-n65-k10 (64,5,5)	94.62	**126**	**126**	138	128	**126**	142	134	131.36	154	580
P-n70-k10 (69,5,5)	99.23	129.29	**128**	138	136	132	146	138	136.56	158	3166.25
P-n76-k5 (75,3,2)	181.34	202	**200**	214	202	202	243.44	210	210	258	280
P-n101-k4 (100,2,2)	321.74	342.69	**342**	396	346	348	388	353.26	354	422	4725.47
X-n110-k13 (109,7,6)	1189.78	**1864**	1898	2080	1898	1898	2044	1926	1960	1970	
X-n115-k10 (114,5,5)	1676.05	**2258**	2300	2658	2262	2274	2504	2316	2332	2862	
X-n139-k10 (138,5,5)	1582.46	2928.64	2740	3144	2534	**2492**	2696	2594	2550	3022	

- **MS.** The giant tour (Line 11 of Algorithm 1) is generated with a randomized nearest-neighbor heuristic. In our implementation, one of the three nearest neighbors is randomly chosen. This randomization is also introduced at the initialization of the algorithm (Line 1 of Algorithm 1). Compared to **HM**, the learning component is lost, and the iterative mechanism is that of a standard multi-start instead of an ILS (see Algorithm 4).
- **HM(LL), HMb(LL), MS(LL).** The three heuristics **HM**, **HMb** and **MS**, are declined in a second version. In these new versions, labels are limited using the technique presented in Section 5.2. Every elementary iteration of the algorithm should then be less efficient but faster.
- **HM(UB), HMb(UB), MS(UB).** The three heuristics **HM**, **HMb** and **MS**, are declined in a third version. In these new versions, the decoding procedure is restricted to the computation of the upper bounds: both the lower bound and the labeling algorithm are deactivated. Every elementary iteration of the algorithm should then be even less efficient but faster.

Algorithm 4 Multi-start heuristic **MS**.

```
1: $bestSol \leftarrow (\tau_1^{rec} = \emptyset, \tau_2^{rec} = \emptyset, \dots, \tau_K^{rec} = \emptyset, \pi_1^{rec} = \emptyset, \dots, \pi_M^{rec} = \emptyset)$
2: while $explorationTime \leq timeLimit$ do
3:     $\tau \leftarrow solveTSP()$
4:     $(\tau_1, \tau_2, \dots, \tau_K, \pi_{drones}) \leftarrow split(\tau)$
5:     $(\tau_1^{opt}, \tau_2^{opt}, \dots, \tau_K^{opt}) \leftarrow reoptimizeTSP(\tau_1, \tau_2, \dots, \tau_K)$
6:     $(\pi_1^{opt}, \dots, \pi_M^{opt}) \leftarrow optimizePMS(\pi_{drones})$
7:     $(\tau_1^{imp}, \tau_2^{imp}, \dots, \tau_K^{imp}, \pi_1^{imp}, \dots, \pi_M^{imp}) \leftarrow improveSol(\tau_1^{opt}, \tau_2^{opt}, \dots, \tau_K^{opt}, \pi_1^{opt}, \dots, \pi_M^{opt})$
8:     if solution $(\tau_1^{imp}, \tau_2^{imp}, \dots, \tau_K^{imp}, \pi_1^{imp}, \dots, \pi_M^{imp})$ is better than $bestSol$ then
9:         $bestSol \leftarrow (\tau_1^{imp}, \tau_2^{imp}, \dots, \tau_K^{imp}, \pi_1^{imp}, \dots, \pi_M^{imp})$
10:    end if
11: end while
```

Table 5 presents the completion times (C.T.) obtained with the 9 heuristics, *i.e.*, **HM** and the 8 variants, for the 20 instances and the branch-and-cut upper bounds found with a time limit of 3 hours. An additionnal column (LB) shows the values of the lower bound obtained at the end of the branch-and-cut. In the 9 heuristics, the computing time limit is set to 1000 seconds. Best solutions are highlighted in bold. An integer value for the completion time usually indicates that a vehicle is the critical resource. This is due to the use of the Manhattan distance for vehicles, with integer coordinates for customers. When the value of completion time is a real number, the critical resource is a drone.

Table 6
Decoding with procedure $split(\tau)$ (complete decoding).

Instance		Execution details							Solution details	
		#Split	#Lb	Del(%)	gD(%)	τ_{Split}	τ_{Opt}	τ_{LS}	#D.C.	C.T.
CMT1 (50,3,2)	**HM**	1920	169688	96.27	2.16	290.45	606.22	18.14	8	168
	HMb	1755	203593	96.26	2.03	340.73	566.92	19.48	8	168
	MS	2548	881	97.69	0.001	70.51	852.54	14.81	10	188
CMT2 (75,5,5)	**HM**	527	865676	97.78	3.89	651.69	279.33	8.85	11	130.23
	HMb	453	1009351	97.88	3.68	723.42	242.75	6.23	13	133.60
	MS	1375	401662	98.72	2.62	323.19	626.44	6.30	16	148
CMT3 (100,4,4)	**HM**	195	2501797	98.20	2.49	881.77	105.99	2.04	17	184
	HMb	135	3089482	98.22	2.55	930.39	72.42	1.72	15	186
	MS	1696	22831	99.07	0.01	156.16	772.92	8.07	16	208
CMT4 (150,6,6)	**HM**	51	9172075	98.97	3.03	955.19	42.57	1.00	17	160.38
	HMb	40	9611465	98.97	3.72	966.96	32.58	0.73	18	150
	MS	234	3990575	99.31	0.05	848.35	138.38	1.32	24	184
CMT5 (199,9,8)	**HM**	33	17588386	99.29	4.60	959.96	44.89	0.78	19	138
	HMb	23	19889825	99.29	5.77	987.09	28.56	0.57	14	139.29
	MS	25	15430506	99.45	0.28	978.73	27.86	0.37	27	152
E-n51-k5 (50,3,2)	**HM**	1927	167681	96.32	2.08	277.97	616.89	19.09	9	168
	HMb	1709	204593	96.27	2.02	348.56	560.89	19.15	9	168
	MS	2864	1109	97.98	0.001	76.31	846.46	9.44	8	182
E-n76-k8 (75,4,4)	**HM**	555	876503	97.61	2.82	713.09	247.92	6.01	13	154
	HMb	454	1020952	97.69	2.56	766.15	201.69	5.90	12	154
	MS	1983	174313	98.56	3.87	215.29	719.07	4.89	14	168
E-n101-k8 (100,4,4)	**HM**	162	2706778	98.17	2.56	897.93	87.58	1.81	16	186
	HMb	145	2890459	98.21	2.43	904.11	82.45	2.01	15	184
	MS	1758	25273	99.06	0.004	165.52	760.12	8.55	16	208
M-n151-k12 (150,6,6)	**HM**	45	9457220	98.95	3.25	958.30	37.46	0.80	25	154
	HMb	41	10503649	98.96	2.67	965.91	34.80	0.65	17	158.96
	MS	193	4395453	99.29	0.34	844.65	145.07	1.34	24	186
M-n200-k16 (199,8,8)	**HM**	25	18639347	99.26	3.97	983.16	30.20	0.62	20	144
	HMb	16	22132150	99.28	4.61	1010.45	20.36	0.42	19	148
	MS	24	16384011	99.45	1.01	991.39	22.97	0.41	25	162
P-n51-k10 (50,5,5)	**HM**	1221	223455	96.83	6.16	317.52	607.45	12.10	11	111.07
	HMb	1095	254528	96.94	5.98	352.82	579.69	11.86	10	114
	MS	1825	118732	98.16	0.02	154.68	790.60	7.95	11	118
P-n55-k10 (54,5,5)	**HM**	1624	183464	96.99	3.68	232.50	672.05	15.44	7	128
	HMb	1459	229487	96.99	3.55	293.84	621.46	16.16	7	128
	MS	2168	3839	98.60	0.005	68.51	862.47	10.54	8	138
P-n60-k10 (59,5,5)	**HM**	973	361918	97.23	5.18	424.83	509.61	10.46	11	114
	HMb	932	403321	97.39	4.99	457.95	481.91	10.16	11	116
	MS	1442	160015	98.42	0.01	174.40	768.30	9.02	14	124
P-n65-k10 (64,5,5)	**HM**	783	504060	97.43	5.09	509.63	399.71	11.12	11	126
	HMb	709	582851	97.51	4.81	579.55	371.07	8.39	12	126
	MS	1554	276576	98.44	0.01	282.48	661.43	8.03	14	138
P-n70-k10 (69,5,5)	**HM**	646	694435	97.61	4.21	626.20	327.40	7.47	11	129.29
	HMb	569	778133	97.69	3.98	666.11	295.18	7.12	13	128
	MS	1439	310237	98.59	0.01	293.67	655.65	5.33	15	138
P-n76-k5 (75,3,2)	**HM**	883	696083	97.37	1.20	628.64	309.95	9.73	11	202
	HMb	657	842403	97.35	1.11	707.73	246.43	9.25	10	200
	MS	2747	1367	96.81	0.40	123.94	781.10	12.49	11	214
P-n101-k4 (100,2,2)	**HM**	149	2403670	97.37	0.69	923.65	68.45	1.22	17	342.69
	HMb	96	2757296	97.29	0.59	945.61	50.45	1.07	17	342
	MS	1818	5678	97.45	0.002	154.54	767.80	9.78	20	396
X-n110-k13 (109,7,6)	**HM**	71	6129616	98.48	5.68	945.97	52.87	1.42	16	1864
	HMb	56	6810276	98.54	6.80	952.17	42.57	1.49	12	1898
	MS	254	3235309	99.09	0.05	831.01	156.30	1.31	17	2080
X-n115-k10 (114,5,5)	**HM**	26	8970734	98.37	3.39	986.31	14.70	0.45	15	2258
	HMb	24	8953906	98.44	4.31	1009.97	13.59	0.61	17	2300
	MS	295	2263434	99.04	0.04	854.58	131.20	1.52	20	2658
X-n139-k10 (138,5,5)	**HM**	1	77374287	98.56	1.18	3056.25	0.70	0.08	23	2928.64
	HMb	1	82009638	98.55	5.85	3726.85	0.68	0.13	23	2740
	MS	1	47270494	98.07	2.18	1798.23	0.56	0.05	24	3144

Tables 6 to 8 give more details on the execution of the heuristics. In Table 6 details are provided for methods **HM**, **HMb** and **MS**. Table 7 is about variants with limited labels in procedure $split(\tau)$: **HM(LL)**, **HMb(LL)** and **MS(LL)**. Table 8 considers variants with upper bounds only: **HM(UB)**, **HMb(UB)** and **MS(UB)**.

In Tables 6 and 7, the following information is reported. Column #Split reports the average number of calls to procedure $split(\tau)$. #Lb gives the average number of labels generated in procedure $split(\tau)$ (summed over all nodes and including labels eventually deleted by the different labels pruning mechanisms). Del is the percentage of labels deleted with the label pruning mechanisms. Column gD gives the average gap between the drone completion time found by procedure $split(\tau)$ (assuming a single drone with a speed multiplied by M) and the completion time obtained after having applied the LPT heuristic. τ_{Split}, τ_{Opt} and τ_{LS} indicate the computing time spent in procedure $split(\tau)$, reoptimization ($reoptimizeTSP(\tau_1, \tau_2, \ldots, \tau_K)$ and $optimizePMS(\pi_{drones})$) and local search ($improveSol(\tau_1^{opt}, \tau_2^{opt}, \ldots, \tau_K^{opt}, \pi_1^{opt}, \ldots, \pi_M^{opt})$), respectively.

Table 7
Decoding with limited procedure $split(\tau)$ (label elimination).

Instance		Execution details							Solution details	
		#Split	#Lb	Del(%)	gD(%)	τ_{Split}	τ_{Opt}	τ_{LS}	#D.C.	C.T.
CMT1 (50,3,2)	**HM(LL)**	2231	102388	94.05	2.90	164.58	706.18	32.57	8	166
	HMb(LL)	1853	107130	93.57	2.88	158.58	653.23	43.40	7	168
	MS(LL)	2696	3114	98.39	0.001	72.44	853.68	12.66	7	196
CMT2 (75,5,5)	**HM(LL)**	1071	419731	96.24	3.57	338.56	565.05	26.75	14	132
	HMb(LL)	902	428728	96.45	3.34	310.76	558.85	33.29	13	133.41
	MS(LL)	1284	241493	97.87	0.007	192.13	760.15	5.93	15	152
CMT3 (100,4,4)	**HM(LL)**	920	630959	93.05	4.23	407.14	496.82	27.10	18	187.04
	HMb(LL)	883	647251	93.08	3.59	410.27	457.01	26.71	17	186.17
	MS(LL)	1692	26008	97.89	0.01	135.04	794.69	8.17	17	204
CMT4 (150,6,6)	**HM(LL)**	401	2209683	96.24	3.59	616.58	322.68	16.54	18	162
	HMb(LL)	405	2238685	96.87	3.03	594.03	314.67	17.48	20	162
	MS(LL)	778	817197	98.24	0.01	376.63	574.26	4.85	28	180
CMT5 (199,9,8)	**HM(LL)**	180	5645118	98.02	4.21	743.80	223.56	7.97	23	140
	HMb(LL)	188	5711948	98.4297	3.25	720.29	229.33	8.68	21	138
	MS(LL)	304	3507043	98.82	0.09	647.20	332.00	1.70	35	154
E-n51-k5 (50,3,2)	**HM(LL)**	2242	100780	94.29	3.12	160.78	705.68	33.02	7	168
	HMb(LL)	2141	108046	93.49	2.77	172.09	649.58	34.40	8	168
	MS(LL)	2323	2001	97.86	0.50	67.30	868.55	5.66	10	180
E-n76-k8(75,4,4)	**HM(LL)**	1313	345500	94.57	3.14	319.37	564.96	34.84	15	156
	HMb(LL)	1250	349315	94.66	2.96	317.68	523.16	33.00	16	156
	MS(LL)	1573	130334	98.21	5.33	146.87	791.73	7.77	20	182
E-n101-k8	**HM(LL)**	921	629648	93.06	4.15	405.20	497.80	26.59	18	188
(100,4,4)	**HMb(LL)**	826	654237	93.05	3.87	415.56	453.82	25.66	18	190.17
	MS(LL)	1490	247410	98.53	0.004	211.90	725.76	6.92	17	224
M-n151-k12	**HM(LL)**	401	2228103	96.43	3.46	613.81	323.70	18.07	23	164
(150,6,6)	**HMb(LL)**	410	2258814	96.84	2.63	600.45	310.26	17.16	20	162
	MS(LL)	849	888681	98.09	0.01	368.20	583.89	5.08	26	182
M-n200-k16	**HM(LL)**	202	5140018	97.76	3.75	724.95	237.31	10.03	24	148
(199,8,8)	**HMb(LL)**	208	5258553	98.30	3.06	705.40	236.05	9.69	20	146
	MS(LL)	434	2347860	98.69	1.37	544.48	421.49	2.76	31	156
P-n51-k10 (50,5,5)	**HM(LL)**	1495	160311	95.84	5.92	206.31	694.00	23.35	10	112.69
	HMb(LL)	1425	171177	95.99	5.72	212.75	648.58	21.99	10	114
	MS(LL)	1826	112113	97.75	1.50	135.88	812.67	6.47	12	122
P-n55-k10 (54,5,5)	**HM(LL)**	1747	139775	95.84	4.75	168.45	720.44	25.52	8	128
	HMb(LL)	1690	150711	95.52	4.39	178.89	667.40	25.66	8	126
	MS(LL)	2389	14359	98.48	0.01	78.90	852.62	10.24	11	142
P-n60-k10 (59,5,5)	**HM(LL)**	1311	235830	96.25	5.19	253.27	648.90	23.03	12	114.86
	HMb(LL)	1303	250511	96.37	4.79	258.56	605.59	21.65	10	116
	MS(LL)	1787	116243	98.14	0.01	132.04	809.75	8.28	12	124
P-n65-k10 (64,5,5)	**HM(LL)**	1234	299012	96.11	4.81	296.03	606.55	24.45	11	128
	HMb(LL)	1188	306392	96.45	4.56	285.63	583.53	22.02	9	126
	MS(LL)	1594	189110	97.84	0.63	175.28	769.47	7.49	15	142
P-n70-k10 (69,5,5)	**HM(LL)**	1179	354052	95.99	3.81	326.20	574.87	26.53	12	136
	HMb(LL)	1058	361613	96.42	3.65	313.53	562.61	23.10	10	132
	MS(LL)	1603	208460	97.86	0.16	188.34	754.99	7.18	15	146
P-n76-k5 (75,3,2)	**HM(LL)**	1655	262675	94.42	2.09	259.27	613.38	31.95	11	202
	HMb(LL)	1440	275681	93.89	1.81	276.87	569.41	28.21	10	202
	MS(LL)	2236	1101	96.51	0.002	90.35	839.46	4.58	9	243.44
P-n101-k4	**HM(LL)**	1179	439302	93.36	0.78	292.53	604.22	18.60	18	346
(100,2,2)	**HMb(LL)**	1138	463067	93.09	0.78	299.16	559.76	19.74	16	348
	MS(LL)	1276	1366	96.43	0.75	64.43	882.19	6.85	19	388
X-n110-k13	**HM(LL)**	508	1502768	96.31	7.16	583.05	350.33	21.81	19	1898
(109,7,6)	**HMb(LL)**	510	1491711	96.47	6.06	577.05	328.85	21.97	15	1898
	MS(LL)	933	734380	98.14	0.01	342.45	612.55	4.78	18	2044
X-n115-k10	**HM(LL)**	616	1207908	93.83	5.92	592.19	327.68	29.34	17	2262
(114,5,5)	**HMb(LL)**	653	1204199	94.22	4.02	568.99	310.53	31.50	18	2274
	MS(LL)	1376	284800	97.98	0.006	251.63	683.53	6.86	17	2504
X-n139-k10	**HM(LL)**	385	2069330	92.49	7.73	690.26	207.10	34.70	26	2534
(138,5,5)	**HMb(LL)**	296	1986166	92.49	7.61	670.71	218.65	40.42	19	2492
	MS(LL)	1409	48399	98.81	0.006	208.41	719.99	7.14	25	2696

#D.C. is the number of customers assigned to the drones. C.T. gives the completion time.

In Table 8, column #Iter represents the number of iterations of the main loop, τ_{UB} is the time spent to compute the upper bound. Other columns are labeled as in tables 6 and 7.

In these tables, we can see that the two giant tour construction methods show very similar behaviors (first line and second line of tables 6 to 8 for each instance). Both solution values and the set of execution parameters shown in the tables are very similar. On the contrary, the multi-start variant (third line) is not able to find solutions that are as good as those of these two methods. Not a single best solution is found with this approach. It demonstrates the gains provided by the learning technique introduced in the ILS. On the 13 instances for which the branch-and-cut was able to find a feasible solution (upper bound), we see that most solutions are very far from those obtained with the heuristics. In a few case, the branch-and-cut finds solutions comparable with the worst heuristics. The lower bounds obtained with the branch-and-cut are not

Table 8
Decoding with upper bounds only.

Instance		Execution details					Solution details	
		#Iter	gD(%)	τ_{UB}	τ_{Opt}	τ_{LS}	#D.C.	C.T.
CMT1 (50,3,2)	**HM(UB)**	2299	1.80	58.74	784.98	51.70	10	174
	HMb(UB)	2275	1.72	51.63	830.11	37.41	10	174
	MS(UB)	2228	0.003	54.62	891.56	10.13	9	204
CMT2 (75,5,5)	**HM(UB)**	1318	2.85	67.49	800.17	42.48	14	140
	HMb(UB)	1371	2.81	64.33	827.78	33.03	16	140
	MS(UB)	1367	0.01	70.08	885.80	6.89	15	152
CMT3 (100,4,4)	**HM(UB)**	1068	2.17	86.60	780.01	38.91	17	195.42
	HMb(UB)	1307	2.18	88.23	787.96	34.99	17	197.24
	MS(UB)	1621	0.005	114.62	824.06	7.81	21	216
CMT4 (150,6,6)	**HM(UB)**	710	2.52	166.51	696.51	46.37	25	166
	HMb(UB)	718	2.51	172.37	713.95	34.37	22	164
	MS(UB)	992	0.02	211.78	728.63	5.40	27	192
CMT5 (199,9,8)	**HM(UB)**	430	3.19	297.64	611.46	24.33	26	142.04
	HMb(UB)	424	3.05	310.84	613.63	16.58	27	140
	MS(UB)	615	0.02	378.96	576.42	3.56	34	152
E-n51-k5 (50,3,2)	**HM(UB)**	2250	1.80	54.27	813.95	39.84	10	168.82
	HMb(UB)	2153	1.68	50.34	833.54	37.28	9	174
	MS(UB)	3177	0.17	79.70	835.66	13.94	13	196
E-n76-k8(75,4,4)	**HM(UB)**	1604	2.23	66.60	807.95	37.60	14	161.86
	HMb(UB)	1382	2.21	61.48	829.50	34.21	14	162
	MS(UB)	2077	0.005	95.73	828.64	9.55	16	186
E-n101-k8 (100,4,4)	**HM(UB)**	1317	2.20	90.14	782.18	36.48	19	196
	HMb(UB)	1211	2.13	86.73	797.36	33.91	17	196
	MS(UB)	1604	0.01	121.34	811.58	7.8	20	216
M-n151-k12 (150,6,6)	**HM(UB)**	698	2.66	174.18	710.21	34.59	26	168
	HMb(UB)	671	2.53	172.67	731.71	25.73	25	169.88
	MS(UB)	861	0.01	204.83	746.78	4.27	27	182
M-n200-k16 (199,8,8)	**HM(UB)**	423	2.85	289.41	625.17	20.70	29	152
	HMb(UB)	407	2.88	289.58	626.62	21.08	29	152
	MS(UB)	545	0.77	307.89	651.94	3.68	30	168
P-n51-k10 (50,5,5)	**HM(UB)**	1342	5.04	41.89	862.75	27.59	14	118
	HMb(UB)	1195	4.91	45.17	826.71	55.80	13	118
	MS(UB)	1882	0.43	56.55	889.16	7.15	14	133.25
P-n55-k10 (54,5,5)	**HM(UB)**	1262	2.87	52.25	831.88	35.11	10	130
	HMb(UB)	1678	2.91	51.91	795.34	67.06	7	132
	MS(UB)	2413	0.002	74.88	853.80	11.24	11	148
P-n60-k10 (59,5,5)	**HM(UB)**	1381	4.42	51.95	841.48	30.17	13	122
	HMb(UB)	1564	4.39	56.32	822.68	36.12	14	120
	MS(UB)	1722	0.02	63.50	879.45	7.93	12	124
P-n65-k10 (64,5,5)	**HM(UB)**	1369	4.09	56.85	817.76	39.89	12	134
	HMb(UB)	1489	3.96	59.04	815.87	39.92	15	131.36
	MS(UB)	1510	0.007	63.82	883.64	6.59	17	154
P-n70-k10 (69,5,5)	**HM(UB)**	1314	3.12	62.40	792.90	53.37	15	138
	HMb(UB)	1414	3.12	62.92	809.69	41.53	13	136.56
	MS(UB)	1358	0.004	65.22	884.86	6.18	20	158
P-n76-k5 (75,3,2)	**HM(UB)**	1782	1.09	74.51	749.87	54.73	11	210
	HMb(UB)	1808	1.07	72.22	778.88	47.46	14	210
	MS(UB)	1547	1.23	73.03	868.26	7.40	13	258
P-n101-k4 (100,2,2)	**HM(UB)**	1471	0.65	75.15	775.19	34.40	19	353.26
	HMb(UB)	1491	0.62	71.99	791.42	34.22	17	354
	MS(UB)	2369	1.39	119.92	784.19	11.01	20	422
X-n110-k13 (109,7,6)	**HM(UB)**	922	5.35	116.61	763.39	38.94	17	1926
	HMb(UB)	751	5.29	111.39	783.22	35.63	18	1960
	MS(UB)	963	0.02	138.06	816.56	4.69	18	1970
X-n115-k10 (114,5,5)	**HM(UB)**	1005	2.79	116.02	730.93	55.49	20	2316
	HMb(UB)	961	2.68	116.86	752.25	47.41	19	2332
	MS(UB)	1121	0.67	136.63	808.88	5.44	24	2862
X-n139-k10 (138,5,5)	**HM(UB)**	1029	3.03	152.88	696.88	54.09	25	2594
	HMb(UB)	806	2.96	152.67	711.63	51.821	22	2550
	MS(UB)	1144	0.1	208.01	731.57	5.71	29	3022

very helpful neither. They however confirm on some instances that the best heuristics find solutions that are at worst of reasonable quality.

Eliminating labels in the decoding procedure does not have a clear impact on solution values. In most cases results are very similar, even if most best values are found with the unrestricted $split(\tau)$ procedure. It is however interesting to see that eliminating labels provide some robustness. For some difficult instances, like X-n139-k10, it avoid getting stuck in a difficult decoding. On our set of instances, methods with label elimination always reach at least 25 iterations in the imparted 1000 seconds. On the contrary, when the number of labels is not limited in procedure $split(\tau)$, several cases with very few iterations, sometimes a single, can be observed.

Completely avoiding exploration is however not enough. When decoding is only based on upper bounds, solution quality significantly decreases. Again, no best values are found with this setting.

Table 9
HM with different parameters (depot at the left corner, 0% of drone eligible customers).

Instance	Depot,%DE	Execution details							Solution details	
		#Split	#Lb	Del(%)	gD(%)	τ_{Split}	τ_{Opt}	τ_{LS}	#D.C.	C.T.
CMT1 (50,3,2)	**center,100%**	1920	169688	96.27	2.16	290.45	606.22	18.14	8	168
	corner,100%	2414	71763	96.06	3.55	132.83	639.94	41.95	5	232
	center,0%	2676	215	19.30	0	78.72	691.69	35.78	0	212
CMT2 (75,5,5)	**center,100%**	527	865676	97.78	3.89	651.69	279.33	8.85	11	130.23
	corner,100%	937	505802	97.50	12.45	441.98	423.35	31.68	6	218
	center,0%	1571	349	20.43	0	91.14	702.71	43.78	0	168
CMT3 (100,4,4)	**center,100%**	195	2501797	98.20	2.49	881.77	105.99	2.04	17	184
	corner,100%	179	1529291	98.10	5.10	744.16	200.81	11.46	10	258.27
	center,0%	1378	544	16.19	0	122.06	660.72	27.57	0	232
CMT4 (150,6,6)	**center,100%**	51	9172075	98.97	3.03	955.19	42.57	1.00	17	160.38
	corner,100%	134	5410515	98.91	10.13	859.57	114.24	6.74	12	250
	center,0%	796	827	14.85	0	233.37	581.12	22.48	0	178
CMT5 (199,9,8)	**center,100%**	33	17588386	99.29	4.60	959.96	44.89	0.78	19	138
	corner,100%	47	13699887	99.25	20.79	925.43	63.14	5.55	16	234
	center,0%	415	1215	14.53	0	363.56	518.59	16.91	0	156
E-n51-k5 (50,3,2)	**center,100%**	1927	167681	96.32	2.08	277.97	616.89	19.09	9	168
	corner,100%	2609	68470	96.13	3.56	131.11	646.93	39.44	4	228
	center,0%	2588	196	18.31	0	76.33	706.57	31.60	0	202
E-n76-k8 (75,4,4)	**center,100%**	555	876503	97.61	2.82	713.09	247.92	6.01	13	154
	corner,100%	1200	465044	97.53	7.26	443.33	410.24	25.72	6	230
	center,0%	1857	321	16.82	0	92.14	726.62	17.66	0	192
E-n101-k8	**center,100%**	162	2706778	98.17	2.56	897.93	87.58	1.81	16	186
(100,4,4)	**corner,100%**	1242	366864	98.41	2.99	197.77	622.66	35.81	3	266
	center,0%	1581	632	13.36	0	115.76	697.33	21.04	0	234
M-n151-k12	**center,100%**	45	9457220	98.95	3.25	958.30	37.46	0.80	25	154
(150,6,6)	**corner,100%**	118	5424366	98.90	10.06	868.84	110.75	4.84	11	250
	center,0%	681	828	15.36	0	192.97	535.44	154.47	0	182
M-n200-k16	**center,100%**	25	18639347	99.26	3.97	983.16	30.20	0.62	20	144
(199,8,8)	**corner,100%**	52	12862411	99.22	15.07	916.17	73.03	4.55	16	238
	center,0%	382	1284	13.80	0	296.87	460.01	153.16	0	162
P-n51-k10 (50,5,5)	**center,100%**	1221	223455	96.83	6.16	317.52	607.45	12.10	11	111.07
	corner,100%	1528	147259	95.67	17.45	265.27	551.97	53.83	7	194
	center,0%	2087	193	22.98	0	66.18	765.55	31.43	0	140
P-n55-k10 (54,5,5)	**center,100%**	1624	183464	96.99	3.68	232.50	672.05	15.44	7	128
	corner,100%	2133	104852	96.78	8.66	152.52	644.42	33.98	4	200
	center,0%	2166	210	22.14	0	71.32	756.38	22.05	0	152
P-n60-k10 (59,5,5)	**center,100%**	973	361918	97.23	5.18	424.83	509.61	10.46	11	114
	corner,100%	1554	183807	96.99	16.66	218.55	586.51	52.13	7	198
	center,0%	1748	268	21.61	0	72.08	763.35	23.55	0	144
P-n65-k10 (64,5,5)	**center,100%**	783	504060	97.43	5.09	509.63	399.71	11.12	11	126
	corner,100%	1323	278130	97.04	13.80	322.02	513.30	37.16	6	206
	center,0%	1434	299	21.52	0	79.64	744.78	28.78	0	160
P-n70-k10 (69,5,5)	**center,100%**	646	694435	97.61	4.21	626.20	327.40	7.47	11	129.29
	corner,100%	1157	371786	97.26	14.75	391.59	451.33	41.45	7	218
	center,0%	1327	316	20.34	0	85.44	733.31	27.77	0	160
P-n76-k5 (75,3,2)	**center,100%**	883	696083	97.37	1.20	628.64	309.95	9.73	11	202
	corner,100%	1841	291512	97.49	3.33	285.25	515.62	29.24	4	260
	center,0%	1640	395	15.39	0	74.30	767.52	12.25	0	258
P-n101-k4	**center,100%**	149	2403670	97.37	0.69	923.65	68.45	1.22	17	342.69
(100,2,2)	**corner,100%**	462	1283694	97.37	1.61	689.36	252.76	5.78	9	382
	center,0%	1011	757	6.67	0	55.75	822.14	5.30	0	420
X-n110-k13	**center,100%**	71	6129616	98.48	5.68	945.97	52.87	1.42	16	1864
(109,7,6)	**corner,100%**	145	3875320	98.34	21.02	888.09	81.53	15.19	10	3296
	center,0%	1039	663	18.13	0	155.93	671.96	33.96	0	2066
X-n115-k10	**center,100%**	26	8970734	98.37	3.39	986.31	14.70	0.45	15	2258
(114,5,5)	**corner,100%**	129	4273478	98.24	10.82	919.12	60.80	6.91	8	3624
	center,0%	1260	817	15.44	0	162.89	635.10	35.77	0	2636
X-n139-k10	**center,100%**	1	77374287	98.56	1.18	3056.25	0.70	0.08	23	2928.64
(138,5,5)	**corner,100%**	69	8206711	98.64	5.99	952.10	37.62	3.17	15	3750
	center,0%	1030	1677	10.61	0	210.46	594.70	34.21	0	2844

Going into more details, with Table 6 we can see that the number of calls to procedure $split(\tau)$ decreases when the size of the instance increases. This was expected because the larger the size of the instance, the larger the number of labels generated in the decoding procedure and the longer the computing time of procedure $split(\tau)$. Table 6 also shows that dominance and bounding procedures are essential. They enable to prune more than 95% of the labels in most cases. Unfortunately, for some instances, up to millions of labels remain. For this reason, on large-sized instances ($n > 60$), about 70 to 90% of the computing time is spent in the decoding procedure, 10 to 30% in the optimization step and 1 to 4% only in the local search. On the contrary, when instances are smaller ($n \leq 60$), the heuristics spend more time in the optimization step (50 to 80% of the computing time). Finally, for almost all instances, gD has a value of less than 5% except for a few cases with **HM** and **HMb**. It shows that aggregating drones in the decoding procedure has a limited impact.

From Table 7, we can make the following additional observations. For instances of size $n < 100$, the time spent in the decoding procedure is less than 35% of the running time. On the other hand, it takes about 40% to 70% of the running time for the decoding procedure to be completed on larger instances ($n \geq 100$). It shows that eliminating labels in the decoding procedure surely accelerates the procedure. When the larger part of the computing time was initially spent in procedure $split(\tau)$, it enables a large increase in the number of iterations. Having an accelerated decoding with labels limitation leads to a lost of efficiency of this decoding and an increase in the time spent in the optimization step.

Finally Table 8 shows that, for the 3 variants, approximately 80% of the execution time is spent in the optimization step. The computation of the upper bound is very fast (less than 30% of the execution time in most cases). Most of the running time is spent in the optimization step. This was expected due to the simplicity of the upper bound computation scheme which may lead to solution of low quality.

From Tables 6, 7 and 8, another important observation to be made is the few time spent in local search. This shows that decoding and optimization components together are quite good enough to lead to solutions of good quality. However, local search still helps improving the solution.

5.5. Additional experiments

In order to go deeper into the evaluation of the methods and the understanding of the results, we carried out two additional experiments. First, we ran **HM** on all instances changing the position of the depot (now located at the bottom left corner) in order to see how our heuristic behaves when the depot is on the outskirts. Second, to discuss the benefit of adding drones in delivery operations, we ran **HM** on all instances considering that all deliveries are made only by vehicles (0% of drone eligible customers). Table 9 presents these results, with reported information similar to that of previous tables.

When analyzing Table 9, several observations can be made. When the depot is located on the outskirts, the completion time increases and the number of customers assigned to the drones decreases as expected. We also notice that the number of iterations of the algorithm increases as well as the time spent at each iteration on reoptimization and on local search. The latter can certainly be explained by the fact that, with less customers served by drones, vehicle tours are longer. The decrease of the time spent in the *split* procedure, with a significant decrease in the number of generated labels, is more difficult to interpret. Probably, returns to the depot are more impacting on label values, which helps dominance.

Considering the case of only using trucks, we generally observe a decrease of completion time ranging from 20% to 25%, except for instance X-n139-k10 which exhibits a particular behavior (a better completion time). This is however easily explained by the fact that *HM* is not effective on this instance as already discussed. Compared to the best heuristic (with value 2492), the result is consistent.

For the two rounds on experiments, it is however difficult to comment about the absolute completion times computed by our algorithms. Surely, it would be interesting to pursue these experiments with real instances, in order to better interpret those values in terms of added-value of drone delivery for logistics providers.

6. Conclusion

The idea of parcel delivery by drones is gradually becoming a reality all over the world. This paper investigated an extension of the PDSTSP where several vehicles and drones are combined for parcel delivery. We coined this problem as PDSMTSP.

We proposed a hybrid metaheuristic adapted from the iterative two-step heuristic descibed in Mbiadou Saleu et al. (2018) for the PDSTSP. Firstly, a giant tour visiting all customers is built. A second step uses dynamic programming for efficiently partitioning the customers of the giant tour between the set of vehicles and the fleet of drones with the restriction that each vehicle route follows the order defined by the giant tour. Introducing several vehicles has a huge computational impact on this step. To circumvent this difficulty, we introduced several upper bounding and lower bounding techniques. In addition, the restriction imposed in the order of vehicle routes can be very detrimental on effectiveness. In a thrid step, we apply local search to relax this constraint and converge towards better solutions. The general scheme of the hybrid metaheuristic can be interpreted as an Iterated Local Search.

Experiments conducted on instances taken from CVRPLIB allowed us to assess the performance of the proposed heuristic. They demonstrated the importance of the decoding procedure, the bounding techniques in this procedure and the learning mechanism of ILS in the heuristic. Unfortunately, this work being the first paper on the PDSMTSP, no comparison with the literature was possible. We implemented a simple branch-and-cut algorithm in an attempt of obtaining a comparison basis, but, unfortunately, no competitive results could be obtained with this branch-and-cut.

A perspective of this work is to develop an exact solution framework. Branch-and-price could be a promising approach. The partitioning between vehicles and drones would easily be managed, as it would only affect the master problem, but the completion time objective would certainly be more challenging. Another perspective is to better control the combinatorial explosion in procedure $split(\tau)$. Better compromises between decoding quality and number of labels generated can certainly be achieved.

Among possible prospects, we are also thinking about casting our model or the *split* procedure into the Constraint Programming (CP) framework and handle it through the use of CP tools. Still, we must take care of the fact that the model we have been studying here is weakly constrained: if we remove constraints related to drone eligibility, we are mainly dealing with partition and permutation constraints, that means cumulative constraints which are difficult to propagate.

On another hand, another perspective of this work is to investigate more realistic models, taking account of customer time windows, real-time traffic or drone recharging for example. Then, these additional constraints might be difficult to integrate in our approach while they would easily be added to a CP model and help propagation.

Finally, another interesting perspective would be to find more insights on the added-value of drone delivery and on the respective sizing of the drone and vehicle fleets. This would basically require additional experiments with modified values for the two fleets, but would only be meaningful with more realistic instances and problem definition.

Acknowledgments

This work was sponsored by a public grant overseen by the French National Research Agency as part of the ”Investissements d’Avenir” through the IMobS3 Laboratory of Excellence (ANR-10-LABX-0016) and the IDEX-ISITE initiative CAP 20–25 (ANR-16-IDEX-0001). Financial support was also received from the European Union through the European Regional Development Fund program (ERDF - AURA region) and by the Auvergne-Rhône-Alpes region.

We thank reviewers for very accurate comments that helped a lot improving the quality of this paper.

References

ABCNews (2017). UPS tests launching drones from trucks equipped with battery chargers. https://abcnews.go.com/Business/ups-tests-launching-drones-trucks-equipped-battery-chargers/story?id=45650029. Accessed: 1 April 2021.

Agatz, N., Bouman, P., & Schmidt, M. (2018). Optimization approaches for the traveling salesman problem with drone. *Transportation Science, 52*(4), 965–981.

Amazon (2013). Amazon.com, Inc. Amazon Prime Air. http://www.amazon.com/primeair. Accessed: 1 April 2021.

Bin Othman, M. S., Shurbevski, A., & Nagamochi, H. (2017). Routing of carrier-vehicle systems with dedicated last-stretch delivery vehicle and fixed carrier route. *Journal of Information Processing, 25*, 655–666.

Bouman, P., Agatz, N., & Schmidt, M. (2018). Dynamic programming approaches for the traveling salesman problem with drone. *Networks, 72*(4), 528–542.

Chang, Y. S., & Lee, H. J. (2018). Optimal delivery routing with wider drone-delivery areas along a shorter truck-route. *Expert Systems with Applications, 104*, 307–317.

Cheng, C., Adulyasak, Y., & Rousseau, L.-M. (2018). *Formulations and exact algorithms for drone routing problem*. CIRRELT, Centre interuniversitaire de recherche sur les réseaux d'entreprise.

Choi, Y., & Schonfeld, P. M. (2017). Optimization of multi-package drone deliveries considering battery capacity. In *Proceedings of the 96th annual meeting of the transportation research board, washington, dc, usa* (pp. 8–12).

Climaco, J. C. N., & Martins, E. Q. V. (1982). A bicriterion shortest path algorithm. *European Journal of Operational Research, 11*(4), 399–404.

Daknama, R., & Kraus, E. (2017). Vehicle routing with drones. arXiv:1705.06431.

Dayarian, I., Savelsbergh, M., & Clarke, J.-P. (2020). Same-day delivery with drone resupply. *Transportation Science, 54*(1), 229–249.

Dell'Amico, M., Montemanni, R., & Novellani, S. (2020). Matheuristic algorithms for the parallel drone scheduling traveling salesman problem. *Annals of Operations Research*, 1–16.

DHL (2014). DHL International GmbH. (2014, Sep.) DHL parcelcopter launches initial operations for research purposes. http://www.dhl.com/en/press/releases/releases_2014/group/dhl_parcelcopter_launches_initial_operations_for_research_purposes.html. Accessed: 1 April 2021.

Dorling, K., Heinrichs, J., Messier, G. G., & Magierowski, S. (2017). Vehicle routing problems for drone delivery. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 47*(1), 70–85.

Dpdgroup (2014). DPDGroup Drone Delivery. https://www.dpd.com/home/insights/drone_delivery. Accessed: 1 April 2021.

Ford Jr, L. R., & Fulkerson, D. R. (2015). *Flows in networks*: 54. Princeton University Press.

Francis, H. (2016). Australia Post tests drones for parcel delivery. http://www.smh.com.au/technology/innovation/australia-post-tests-drones-for-parcel-delivery-20160415-go77a4.html. Accessed: 1 April 2021.

de Freitas, J. C., & Penna, P. H. V. (2020). A variable neighborhood search for flying sidekick traveling salesman problem. *International Transactions in Operational Research, 27*(1), 267–290.

Ha, Q. M., Deville, Y., Pham, Q. D., & Hà, M. H. (2018). On the min-cost traveling salesman problem with drone. *Transportation Research Part C: Emerging Technologies, 86*, 597–621.

Ham, A. M. (2018). Integrated scheduling of m-truck, m-drone, and m-depot constrained by time-window, drop-pickup, and m-visit using constraint programming. *Transportation Research Part C: Emerging Technologies, 91*, 1–14.

Helsgaun, K. (2000). An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European Journal of Operational Research, 126*(1), 106–130.

Kelion, L. (2015). Alibaba begins drone delivery trials in China. http://www.bbc.com/news/technology-31129804. Accessed: 1 April 2021.

Kim, S., & Moon, I. (2018). Traveling salesman problem with a drone station. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 49*(1), 42–52.

Kitjacharoenchai, P., Ventresca, M., Moshref-Javadi, M., Lee, S., Tanchoco, J. M., & Brunese, P. A. (2019). Multiple traveling salesman problem with drones: Mathematical model and heuristic approach. *Computers & Industrial Engineering, 129*, 14–30.

Lourenço, H. R., Martin, O. C., & Stützle, T. (2003). Iterated local search. In *Handbook of metaheuristics* (pp. 320–353). Springer.

Luo, Z., Liu, Z., & Shi, J. (2017). A two-echelon cooperated routing problem for a ground vehicle and its carried unmanned aerial vehicle. *Sensors, 17*(5), 1144.

Marinelli, M., Caggiani, L., Ottomanelli, M., & Dell'Orco, M. (2017). En route truck–drone parcel delivery for optimal vehicle routing strategies. *IET Intelligent Transport Systems, 12*(4), 253–261.

Mathew, N., Smith, S. L., & Waslander, S. L. (2015). Planning paths for package delivery in heterogeneous multirobot teams. *IEEE Transactions on Automation Science and Engineering, 12*(4), 1298–1308.

Mbiadou Saleu, R. G., Deroussi, L., Feillet, D., Grangeon, N., & Quilliot, A. (2018). An iterative two-step heuristic for the parallel drone scheduling traveling salesman problem. *Networks, 72*(4), 459–474.

Mourelo Ferrandez, S., Harbison, T., Webwer, T., Sturges, R., & Rich, R. (2016). Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm. *Journal of Industrial Engineering and Management, 9*(2), 374–388.

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.

Murray, C. C., & Raj, R. (2020). The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones. *Transportation Research Part C: Emerging Technologies, 110*, 368–398.

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*(4), 411–458.

Pinedo, M., & Hadavi, K. (1992). Scheduling: Theory, algorithms and systems development. In *Proceedings of the Operations research 1991* (pp. 35–42). Springer.

Poikonen, S., & Golden, B. (2020). Multi-visit drone routing problem. *Computers & Operations Research, 113*, 104802.

Poikonen, S., Wang, X., & Golden, B. (2017). The vehicle routing problem with drones: Extended models and connections. *Networks, 70*(1), 34–43.

Ponza, A. (2016). Optimization of drone-assisted parcel delivery. Master's thesis.ItalyUniversity of Padova.

Pugliese, L. D. P., & Guerriero, F. (2017). Last-mile deliveries by using drones and classical vehicles. In *Proceedings of the International conference on optimization and decision science* (pp. 557–565). Springer.

San, K. T., Lee, E. Y., & Chang, Y. S. (2016). The delivery assignment solution for swarms of UAVs dealing with multi-dimensional chromosome representation of genetic algorithm. In *Proceedings of the IEEE annual ubiquitous computing, electronics & mobile communication conference (UEMCON)* (pp. 1–7). IEEE.

Schermer, D., Moeini, M., & Wendt, O. (2018). Algorithms for solving the vehicle routing problem with drones. In *Proceedings of the Asian conference on intelligent information and database systems* (pp. 352–361). Springer.

Song, B. D., Park, K., & Kim, J. (2018). Persistent UAV delivery logistics: MILP formulation and efficient heuristic. *Computers & Industrial Engineering, 120*, 418–428.

Sundar, K., & Rathinam, S. (2014). Algorithms for routing an unmanned aerial vehicle in the presence of refueling depots.. *IEEE Trans. Automation Science and Engineering, 11*(1), 287–294.

Troudi, A., Addouche, S.-A., Dellagi, S., & Mhamedi, A. (2018). Sizing of the drone delivery fleet considering energy autonomy. *Sustainability, 10*(9), 3344.

Tu, P. A., Dat, N. T., & Dung, P. Q. (2018). Traveling salesman problem with multiple drones. In *Proceedings of the ninth international symposium on information and communication technology* (pp. 46–53). ACM.

Ulmer, M. W., & Thomas, B. W. (2018). Same-day delivery with heterogeneous fleets of drones and vehicles. *Networks, 72*(4), 475–505.

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

XCompany (2014). Wing : Transforming the way goods are transported. https://x.company/projects/wing/. Accessed: 1 April 2021.

Yurek, E. E., & Ozmutlu, H. C. (2018). A decomposition-based iterative optimization algorithm for traveling salesman problem with drone. *Transportation Research Part C: Emerging Technologies, 91*, 249–262.

Zipline (2016). The future of healthcare is out for delivery. http://flyzipline.com/. Accessed: 1 April 2021.