Received: 25 June 2018 | Accepted: 28 June 2018
DOI: 10.1002/net.21846

SPECIAL ISSUE ARTICLE

WILEY

An iterative two-step heuristic for the parallel drone scheduling traveling salesman problem

Raïssa G. Mbiadou Saleu[1] | **Laurent Deroussi**[1] | **Dominique Feillet**[2] | **Nathalie Grangeon**[1] | **Alain Quilliot**[1]

[1]Université Clermont Auvergne, LIMOS UMR CNRS 6158, 63178 Aubière Cedex, France
[2]Mines Saint-Étienne and LIMOS UMR CNRS 6158, CMP Georges Charpak, F-13541 Gardanne, France

Correspondence
Dominique Feillet, Mines Saint-Étienne and LIMOS UMR CNRS 6158, CMP Georges Charpak, F-13541 Gardanne, France.
Email: feillet@emse.fr

A recent evolution in urban logistics involves the usage of drones. In this article, we address a heuristic solution of the parallel drone scheduling traveling salesman problem, recently introduced by Murray and Chu. In this problem, deliveries are split between a vehicle and drones. The vehicle performs a classical delivery tour, while the drones are constrained to perform back and forth trips. The objective is to minimize completion time. We propose 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 trips for the drones. Decoding is expressed as a bicriteria shortest path problem and is carried out by dynamic programming. Experiments conducted on benchmark instances confirm the efficiency of the approach and give some insights on this drone delivery system.

KEYWORDS
city logistics, drone delivery, dynamic programming, heuristic, PDSTSP, vehicle routing problem

1 | INTRODUCTION

Last mile logistics has become a popular area of interest for retailers. Companies are always searching for fast and cost-efficient ways to deliver goods to their customers. It is from this perspective that in 2013, the chief executive officer of the largest online retailer Amazon revealed a revolutionary project of the company called *Prime Air* [1], a drone designed to deliver packages in just 30 minutes (Figure 1). In August 2014, Google followed by revealing its drone delivery project called *wing*, with experiments run in Australia. In September 2014, DHL Express launched its *parcelcopter*, a helicopter-style drone which could deliver medications and urgently needed goods to the remote North Sea island of Juist [7]. In 2015, the online retailer Alibaba tested the usage of drones to deliver online orders of tea to customers in China [15]. In April 2016, Australia post started testing the usage of remotely piloted drones internally for parcel delivery, with the support of the Civil Aviation Safety Authority [4]. In July 2016, convenience store chain 7-eleven made their first commercial delivery with drone, partnered with the drone start-up Flirtey [3]. United Parcel Service (UPS) company is testing drones to deliver medicine to an island near Boston [22]. Still in July 2016, the first humanitarian drone called *Zipline* arrived in Rwanda. *Zipline* is a drone of about 10 k that is capable of carrying 1.3 k of medicines or blood [24]. In August 2016, Domino's Pizza announced their intention to extend to France their experiments on pizza delivery by drone, as initiated in New Zealand [14]. Many other companies not mentioned above have launched similar projects.

Previously exclusively used in military applications, drones are becoming a part of the last-mile delivery concept which can significantly accelerate delivery times and reduce human intervention. Indeed, drones are lightweight, they consume less

FIGURE 1 Amazon prime air Source: amazon.com [Colour figure can be viewed at wileyonlinelibrary.com]

energy than standard vehicles and they are not subject to congestion problems as they do not follow the road network. On the other hand, drones are limited in terms of endurance, capacity, and payload. Despite several regulatory and safety standards barriers, the idea of using unmanned aerial vehicle (UAV) for parcel delivery is gaining ground. Companies that want to use drones in their delivery process are confronted by air control administrations like the FAA (Federal Aviation Administration) in the United States or the DGAC (Direction Générale de l'Aviation Civile) in France. The scientific community is getting more and more interested in investigating the design of drone delivery systems.

In this article, we are interested in the development of efficient solution methods for a specific drone delivery system. The remainder of the article is organized as follows. In Section 2, we present the related literature. In Section 3, we formally describe the problem addressed in this article and provide an integer linear programming formulation. We propose a heuristic to solve the problem in Section 4. In Section 5, we discuss the results obtained from our computational experiments. Finally, we conclude with some directions for future researches in Section 6.

2 | RELATED WORK

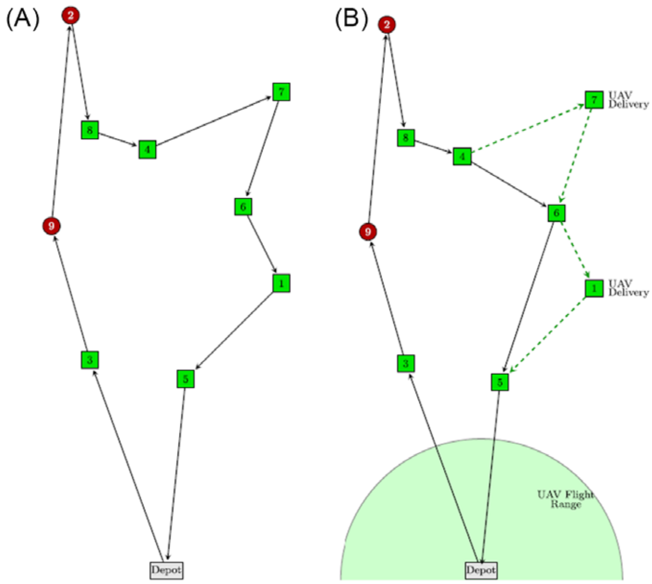
The problem of parcel delivery with drone has received increasing attention these last years. Murray and Chu [5] introduced the flying sidekick traveling salesman problem (FSTSP) and the parallel drone scheduling traveling salesman problem (PDSTSP). In both problems, the optimization criterion is the minimization of the completion time, that is, the time elapsed between the beginning and the end of the delivery process.

In the FSTSP, deliveries are performed by a single vehicle and a single drone working in tandem. The vehicle carries the drone and when the vehicle is at a customer location, it can decide to launch the drone to visit another customer location. After this drone delivery, the vehicle will then have to retrieve the drone at another customer location. Formally, a drone trip is represented as a triplet $<i,j,k>$, where i is the launching point, j is the service point, and k is the rendez-vous point ($i \neq j$, $j \neq k$, and $i \neq k$). Figure 2 shows an illustration of the synchronization mechanism in the FSTSP. The authors proposed a mixed-integer linear programming (MILP) formulation and a heuristic that first constructs a traveling salesman problem (TSP) tour visiting all the customer locations and next, repeatedly run a relocation procedure to assign some customer requests to the drone.

The second problem introduced by Murray and Chu [5], the PDSTSP, simplifies the FSTSP setting by relaxing the synchronization issue between the vehicle and the drone. Conversely, it considers a fleet of drones instead of a single drone. The vehicle visits a subset of customers with a TSP tour and the drones serve the remaining customers directly from the depot with back and forth trips. Again, the authors proposed a MILP formulation and a heuristic. The principle of the heuristic is to partition the set of customers into two subsets (a *vehicle set* and a *drone set*). Then, a TSP tour is computed for the customers assigned to the vehicle and a parallel machine scheduling (PMS) problem is solved to assign customer requests (jobs) to drones (machines). At the end, an improvement step is performed to reassign customers either to the drone set or to the vehicle set in order to better balance vehicle and drones activities.

In his master thesis, Ponza [2] addresses the FSTSP and proposes a simulated annealing heuristic.

Wang et al. [25] studied an extension of the FSTSP called the vehicle routing problem with drones (VRPD) in which they considered a homogeneous fleet of trucks, each one of them carrying several identical drones. A drone launched from a truck must be picked up by the same truck at the same or at a different location. The objective is still to minimize the completion time.



(A) An optimal truck delivery sequence, without the aid of a UAV.

(B) The UAV is launched from a delivery truck, delivering parcels to two eligible customers.

FIGURE 2 FSTSP: Illustration of the synchronization mechanism Source: [5] [Colour figure can be viewed at wileyonlinelibrary.com]

The authors proposed worst-case analyses and obtained theoretical bounds on the benefits achieved using drones. This line of research is pursued by the same authors in [23].

In Ref. [19], Agatz et al. introduced another variant of the FSTSP that they called the traveling salesman problem with drone (TSP-D). Contrary to the FSTSP, the launching point of a drone and the rendezvous point with the vehicle are not necessarily different. Furthermore, the objective is to find the shortest tour to serve all customer locations either by the vehicle or the drone. The authors proposed an integer programming (IP) formulation and a *route first–cluster second* heuristic.

Ha et al. [20] adopted the name TSP-D for the FSTSP. They proposed a *cluster first–route second* and a *route first–cluster second* heuristic. In Ref. [21], the same authors investigated a different objective function: the minimization of the total transportation cost for both the vehicle and the drone. They coined this problem as min-cost TSP-D. They proposed a MILP model and two heuristics: one based on a Greedy Randomized Adaptive Search Procedure (GRASP) and another one called TSP-LS adapted from the work of Murray and Chu [5].

Dorling et al. [13] introduced a very different problem that they called drone delivery problem (DDP). They considered two variants of the DDP: the *minimum cost drone delivery problem* (MC-DDP) that aims at minimizing the total delivery cost and the *minimum time drone delivery problem* (MT-DDP) that aims at minimizing the total delivery time. In these problems, deliveries are only performed by drones, and the energy consumption is explicitly considered for the drones, taking into account their payload. The authors established the relationship with the multitrip vehicle routing problem (MTVRP). They proposed a MILP formulation and a simulated annealing (SA) heuristic.

Ulmer and Thomas [17] studied a dynamic variant 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).

Table 1 gives a summary of the main papers on drone delivery problems found in the literature.

In connection with the literature on drone delivery, we can also mention the Carrier-Vehicle TSP, addressed by Garone et al. [8] or Gambella et al. [6]. In this problem, two vehicles with different capabilities cooperate. The carrier is slow but has a virtually infinite operating range. Its role is to deploy and recover a second vehicle, typically an aircraft, which is faster but has a reduced operating range.

TABLE 1 Optimization of drone delivery systems

Reference	Problem	Vehicles	Drones	Synchro.	Solution method
Murray and Chu [5]	FSTSP	1	1	Yes	MILP, Heuristics
	PDSTSP	1	*n*	No	
Wang et al. [25]	VRPD	*m*	*n*	Yes	
Poikonen et al. [23]	VRPD	*m*	*n*	Yes	
Agatz et al. [15]	TSP-D	1	1	Yes	IP, Route first–Cluster second
Ha et al. [20]	TSP-D(FSTSP)	1	1	Yes	Cluster first–Route second, Route first–Cluster second
Ha et al. [21]	min-cost TSP-D	1	1	Yes	MILP, GRASP, TSP-LS
Dorling et al. [13]	DDPs	0	*n*	No	MILP, SA
Marlin et al. [17]	SDDPHF	*m*	*n*	No	Adaptive dynamic programming (PFA)

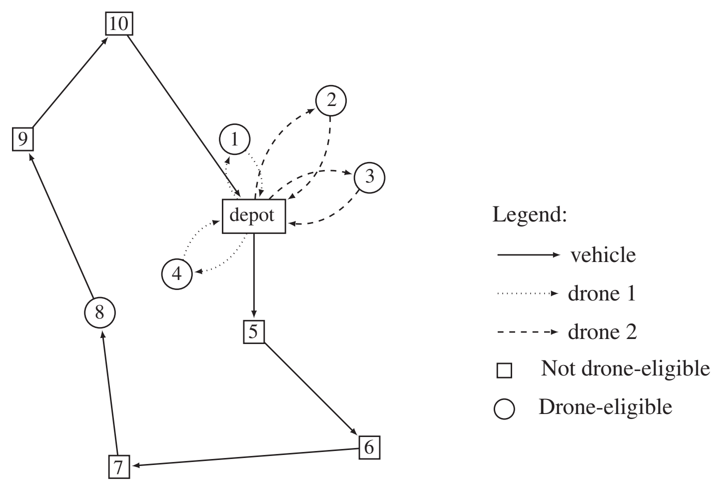


FIGURE 3 A set of 10 customers served by 1 vehicle and 2 drones

3 | THE PARALLEL DRONE SCHEDULING TRAVELING SALESMAN PROBLEM

In this article, we focus on the PDSTSP, presented in Ref. [5], that we found more relevant on a short term in view of actual delivery practices and that has yet drawn less attention in the literature. The PDSTSP is formally defined below.

3.1 | Problem definition and notation

We consider a complete directed graph $G = (N \cup \{0\}, A)$ where $N = \{1, \ldots, n\}$ is a customer set and 0 is the depot. A vehicle and a fleet of M drones are available to deliver parcels to customers. The vehicle delivers customers with a single tour starting from the depot, visiting a subset of customers, and returning back to the depot. Drones operate back and forth trips between the depot and the customers, delivering a single customer in each trip. Not all customers are 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 article. Figure 3 depicts a solution of the PDSTSP on an instance with 10 customers, among which 5 are drone-eligible.

A travel time d_{ij} is incurred when the vehicle goes through an arc $[i,j] \in A$. A travel time $\widehat{d}_i$ is incurred when a drone serves a customer $i \in N_d$. Assuming that both the vehicle and the drones can start from the depot at time 0, the objective of the PDSTSP is to minimize the delivery completion time, that is, the time at which the vehicle and all drones are back to the depot, with the service of all customers carried out.

3.2 | MILP formulation

In this Section, we express the PDSTSP with a MILP formulation. We introduce the following binary decision variables:

- $z_i = 1$ if customer i is visited by the vehicle, 0 if it is visited by the drone ($i \in N$);

- $x_{ij} = 1$ if arc (i,j) belongs to the vehicle tour, 0 otherwise $((i,j) \in A)$;
- $y_{im} = 1$ if customer i is assigned to drone m, 0 otherwise $(i \in N_d,\ 1 \le m \le M)$.

In addition, we introduce a nonnegative variable T that represents the completion time.

The model is:

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

subject to

$$T \geq \sum_{(i,j)\in A} d_{ij} x_{ij} \tag{2}$$

$$T \geq \sum_{i\in N_d} \hat{d}_i y_{im} \qquad (1 \leq m \leq M) \tag{3}$$

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

$$\sum_{(i,j)\in A} x_{ij} = z_i \qquad (i \in N) \tag{5}$$

$$\sum_{1\leq m\leq M} y_{im} = 1 - z_i \qquad (i \in N_d) \tag{6}$$

$$\sum_{(0,j)\in A} x_{0j} \leq 1 \tag{7}$$

$$\sum_{(i,j)\in A} x_{ij} = \sum_{(k,i)\in A} x_{ki} \qquad (i \in N) \tag{8}$$

$$\sum_{i\in S} \sum_{j\notin S} x_{ij} \geq \sum_{i\in S} z_i + 1 - |S| \qquad (S \subseteq N, S \neq \emptyset) \tag{9}$$

$$z_i \in \{0,1\} \qquad (i \in N) \tag{10}$$

$$x_{ij} \in \{0,1\} \qquad ((i,j) \in A) \tag{11}$$

$$y_{im} \in \{0,1\} \qquad (i \in N_d, 1 \leq m \leq M) \tag{12}$$

$$T \geq 0 \tag{13}$$

The objective (1) is to minimize the completion time T. Constraints (2, 3) give lower bounds on T expressed using vehicle assignment and drone assignment variables. Constraints (4) ensure that all the customers that are not drone-eligible are served by the vehicle. Constraints (5, 6) guarantee that every customer is served either by the vehicle or a drone, exactly once. Constraint (7) stipulates that the vehicle leaves the depot at most once. Constraints (8) ensure flow conservation for the vehicle tour. Subtour elimination constraints are provided by (9). These constraints ensure that given a non-empty subset of customers $S \subseteq N$, if all the customers in S are visited by the vehicle, there is at least one outgoing arc from S. Finally, constraints (10)-(13) define the decision variables.

4 | ITERATIVE TWO-STEP HEURISTIC

The PDSTSP can be considered as a bi-level problem in which:

- at the first level, customers are *partitioned* between the vehicle and the fleet of drones;
- the second level manages the *routing optimization*, which consists in solving two NP-Hard problems, namely: a TSP for the vehicle and a parallel machine scheduling (PMS) problem for drones.

Our iterative two-step heuristic alternates between these two levels. Given a solution of the PDSTSP, a coding step transforms this solution into a customer sequence. Then, a decoding step decomposes this sequence into a tour for the vehicle and series of trips for the drones. These tours/trips are optimized and the process is repeated. The general solution framework does not depend on the number of drones, but the decoding procedure does. When $M = 1$, that is, a single drone is available, the decoding scheme is exact. When $M > 1$, it is heuristically extended.

In Section 4.1, we describe the general scheme of the heuristic. The decoding procedure is detailed in Section 4.2 when $M = 1$. We explain how it is adapted to $M > 1$ in Section 4.3. In these sections, we adopt the following notation. A solution S is represented as a vector of $M + 1$ customer sequences $(\tau_{vehicle}, \tau_1, \ldots, \tau_M)$, where $\tau_{vehicle}$ indicates the visit order of the customers for the vehicle and τ_1 to τ_M this order for the M drones. We denote $c^1(S)$ the completion time for the vehicle in solution S, and $c^2(S)$ the completion time for the fleet of drones. Finally, $c(S) = \max(c^1(S), c^2(S))$ is the solution cost.

4.1 | General scheme of the iterative two-step heuristic

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

Algorithm 1 General scheme of the iterative two-step heuristic

1: $\tau \leftarrow solveTSP()$
2: $bestSol \leftarrow (\tau, \emptyset, \ldots, \emptyset)$
3: **while** $bestSol$ is improved **do**
4: $\quad(\tau_{vehicle}, \tau_{drones}) \leftarrow split(\tau)$
5: $\quad\tau^{opt}_{vehicle} \leftarrow reoptimizeTSP(\tau_{vehicle})$
6: $\quad(\tau^{opt}_1, \ldots, \tau^{opt}_M) \leftarrow optimizePMS(\tau_{drones})$
7: $\quad$**if** solution $(\tau^{opt}_{vehicle}, \tau^{opt}_1, \ldots, \tau^{opt}_M)$ is better than $bestSol$ **then**
8: $\qquad bestSol \leftarrow (\tau^{opt}_{vehicle}, \tau^{opt}_1, \ldots, \tau^{opt}_M)$
9: $\qquad \tau \leftarrow bestInsertion(\tau^{opt}_{vehicle}, \tau^{opt}_1, \ldots, \tau^{opt}_M)$
10: $\quad$**end if**
11: **end while**

In lines 1 and 2, we build a giant TSP tour τ visiting the depot and all the customers. For that matter, we apply a nearest-neighbor construction procedure. The TSP tour provides a starting solution $(\tau, \emptyset, \ldots, \emptyset)$ for our algorithm, where all the customers are visited by the vehicle in the order of sequence τ and no customer is assigned to the drones.

The main part of the algorithm is given by lines 4 to 9 that are repeated as long as they enable finding better solutions. A solution S is considered to be better than another solution S' if one of the two following conditions hold:

- $c(S) < c(S')$
- or $c(S) = c(S')$ and $\min(c^1(S), c^2(S)) < \min(c^1(S'), c^2(S'))$

Decoding procedure *split* on Line 4 decomposes τ in two complementary subsequences: one assigned to the vehicle ($\tau_{vehicle}$) and one assigned to the fleet of drones (τ_{drones}). This procedure, which is central in the algorithm, is detailed below. Vehicle route $\tau_{vehicle}$ is then reoptimized, with Helsgaun's implementation of Lin-Kernighan heuristic [12] (Line 5) and a PMS solution is obtained from τ_{drones} (see Section 4.3.3). The new solution $(\tau^{opt}_{vehicle}, \tau^{opt}_1, \ldots, \tau^{opt}_M)$ is compared with *bestSol* and the latter is updated if needed. Finally, in this case, all the customers from $\tau^{opt}_1, \ldots, \tau^{opt}_M$ are reinserted in $\tau^{opt}_{vehicle}$, in this order and with a best insertion policy (Line 9). Otherwise the algorithm stops.

Note that when $M = 1$, the PMS problem is trivial, hence one can replace Line 6 with $\tau^{opt}_1 \leftarrow \tau_{drones}$.

4.2 | Decoding procedure when $M = 1$

We first consider the case $M = 1$. Procedure $split(\tau)$ computes the best partition of the customer set between the vehicle and the unique drone with the constraint that the customers in the vehicle tour follows the same order as in sequence τ. This problem is modeled as a bicriteria shortest path problem [11] in an acyclic directed graph and is solved by dynamic programming.

4.2.1 | Definition of the acyclic graph

We introduce $G^\tau = (V^\tau, A^\tau)$ an acyclic graph defined as follows. $V^\tau = \{0, 1, 2, \ldots, n, n + 1\}$ represents the set of customers completed by two copies of the depot: the original depot 0 and its copy $n + 1$. In the bicriteria 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:

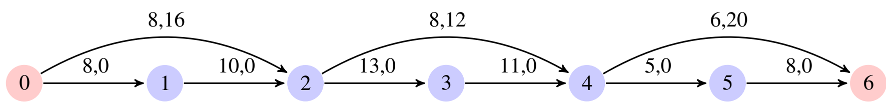


FIGURE 4 Graph G^τ for the instance of Table 2A,B, with $\tau = (1, 2, 3, 4, 5)$ [Colour figure can be viewed at wileyonlinelibrary.com]

TABLE 2 Illustrative instance

(A) Vehicle cost matrix d_{ij}

	0	**1**	**2**	**3**	**4**	**5**	**6**
0	0	8	8	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 $\hat{d}_i$

Customer	1	3	5
Drone cost	16	12	20

- $i = 0$, or $j = n + 1$, or i is before j in τ (in the three cases we note $i <_\tau j$)
- all the customers between i and j in τ are drone-eligible: $i <_\tau k <_\tau j \Rightarrow k \in N_d$

With every arc $(i,j) \in A^\tau$, we associate a cost vector (c_{ij}^1, c_{ij}^2). The first cost component c_{ij}^1 represents the cost incurred for the vehicle if it travels directly from i to j: $c_{ij}^1 = d_{ij}$. The second cost component c_{ij}^2 represents the corresponding cost induced for the drone. If the vehicle travels directly from i to j, all customers k in-between (which by definition of A^τ are all drone-eligible) are assigned to the drone: $c_{ij}^2 = \sum_{\{k \in V^\tau:\ i <_\tau k <_\tau j\}} \hat{d}_k$.

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

The set of paths from 0 to $n + 1$ in G^τ exactly matches the set of acceptable routes for the vehicle that can be extracted from sequence τ. Furthermore, given a path P, the cost vector $(c^1(P), c^2(P)) = \left(\sum_{(ij) \in P} c_{ij}^1, \sum_{(ij) \in P} c_{ij}^2\right)$ corresponds to the costs of the vehicle and the drone, if the route of the vehicle follows P and the remaining customers are assigned to the drone. Taking again the example depicted with Figure 4, path $P = (0, 2, 4, 5, 6)$ should be interpreted as a solution S with a vehicle tour (0, 2, 4, 5, 0) and customers 1 and 3 assigned to the drone; $c^1(P) = 29$, $c^2(P) = 28$, $c(S) = \max(29, 28) = 29$.

For that reason, the aim of the *split* procedure is to find the path P from 0 to $n + 1$ in G^τ that minimizes $\max(c^1(P), c^2(P))$. We now describe the dynamic programming procedure developed to compute this path.

4.2.2 | Dynamic programming scheme

To understand the procedure applied to find that shortest path, let us first consider the following definitions:

- *partial path*: a path going from 0 (origin depot) to any node i of graph G^τ;
- *label*: the cost vector (c^1, c^2) of a partial path calculated by adding the costs of the crossed arcs.

The shortest path is found by applying the procedure 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 1). The procedure then starts by assigning label (0, 0) to $\mathcal{L}(0)$ (Line 2).

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. For a given vertex i, list $\mathcal{L}(i)$ is constructed by adding a new label for every label in the list of labels of a predecessor node (Lines 6-13). The new label is simply defined by adding the arc cost (Line 8). To limit the number of labels generated in the lists as the procedure advance in the graph, some bounding mechanisms (Line 9) and a dominance rule (Line 10) are introduced. These two components are detailed below.

At the end, the shortest path is retrieved through a backtrack mechanism considering the best label found in $\mathcal{L}(n+1)$. This shortest path constitutes the vehicle tour $\tau_{vehicle}$ and the nodes out of the path are assigned to the drone.

4.2.3 | Dominance rule

The dominance rule is very simple. A label $L_a = (c_a^1, c_a^2)$ dominates a label $L_b = (c_b^1, c_b^2)$, if ($c_a^1 < c_b^1$ and $c_a^2 \le c_b^2$) or ($c_a^1 \le c_b^1$ and $c_a^2 < c_b^2$). Before adding a new label to a list of label, dominance tests are tried 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 guarantees that the number of labels attached to a node is limited by the number of values that can take label components c^1 or c^2. As this number can still be very large, bounding mechanisms are helpful to limit the combinatorial explosion.

4.2.4 | Upper bound generation

The bounding mechanisms rely on the computation of several upper bounds. A first upper bound UB_1 is given by the value of the best solution found so far. A second upper bound UB_2 is obtained by applying Algorithm 2 with the following modifications:

- the bounding mechanisms of Line 9 are not considered;
- the dominance rule is changed to: L_a dominates L_b if $c(L_a) \le c(L_b)$; with this new rule, a single label is kept at each node of the graph.

Algorithm 2 Procedure $split(\tau)$

1: $\mathcal{L}(i) \leftarrow \emptyset$ for $0 \le i \le n+1$
2: $\mathcal{L}(0) \leftarrow \{(0,0)\}$
3: $i \leftarrow 0$
4: **while** $i <_\tau n+1$ **do**
5:     increment i
6:     **for all** j such that arc $(j,i) \in A^\tau$ **do**
7:         **for all** label $L \in \mathcal{L}(j)$ **do**
8:             $L' \leftarrow (c^1(L) + c_{ji}^1, c^2(L) + c_{ji}^2)$
9:             **if** L' is not pruned with the bounding mechanisms **then**
10:                 $\mathcal{L}(i) \leftarrow addWithDominance(\mathcal{L}(i), L')$
11:             **end if**
12:         **end for**
13:     **end for**
14: **end while**
15: $bestLabel \leftarrow$ label L in $\mathcal{L}(n+1)$ with minimum completion time $\max(c^1(L), c^2(L))$
16: $\tau_{vehicle} \leftarrow$ path that led to label $bestLabel$
17: $\tau_{drones} \leftarrow$ nodes that are not in the path

The third upper bound UB_3 results from the following greedy heuristic. The nodes are considered in the order defined by $<_\tau$. Nodes are added to the vehicle tour $\tau_{vehicle}$ except when the two following conditions hold: node i is drone-eligible and adding i to the drone does not increase the completion time of the current partial solution. In this case, node i is assigned to the drone. UB is set as the best (minimal) upper bound among these three bounds: $UB = \min(UB_1, UB_2, UB_3)$.

Figure 5 represents the application of these algorithms to compute UB_2 and UB_3 on the example introduced with Figure 4. Assuming that no previous solution is known ($UB_1 = +\infty$), we obtain $UB_2 = 29$, $UB_3 = 42$, and $UB = 29$.

4.2.5 | Lower bound generation

We introduce two lower bounds, $LB^{tot}(i)$ and $LB^{veh}(i)$, at each customer node of the graph. Lower bound $LB^{tot}(i)$ indicates the minimal total cost $c^1(P) + c^2(P)$ of any path P between i and $n+1$ in G^τ. Lower bound $LB^{veh}(i)$ indicates the minimal vehicle cost $c^1(P)$ of any path P between i and $n+1$ in G^τ. Both bounds are computed simultaneously with a backward exploration of the acyclic (topologically ordered) graph G^τ.

Figure 6 details these lower bounds on the example introduced with Figure 4.

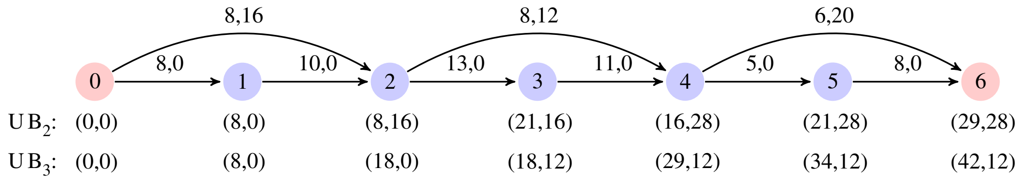


FIGURE 5 Upper bounds for the graph of Figure 4 [Colour figure can be viewed at wileyonlinelibrary.com]

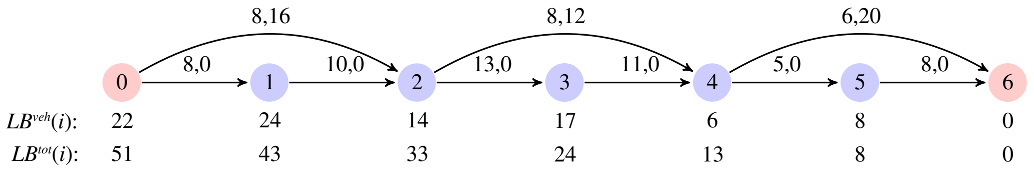


FIGURE 6 Lower bounds for the graph of Figure 4 [Colour figure can be viewed at wileyonlinelibrary.com]

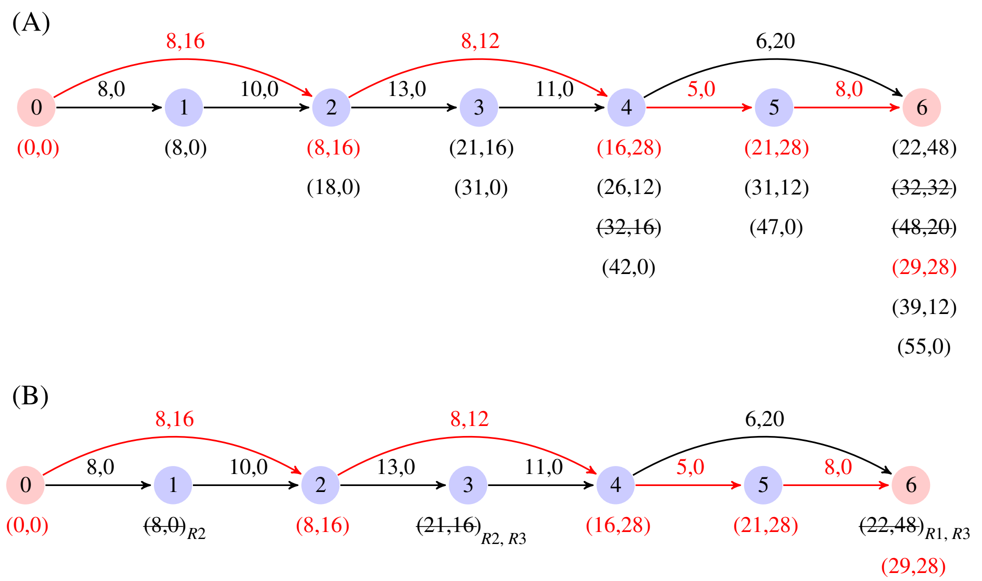


FIGURE 7 Illustration of the *split* procedure on the graph of Figure 4, without or with the bounding mechanisms [Colour figure can be viewed at wileyonlinelibrary.com]

4.2.6 | Bounding mechanisms

We define the three following bounding rules:

- **R1**: prune label L if $c^2(L) \geq UB$
- **R2**: prune label L if $c^1(L) + LB^{veh}(i) \geq UB$
- **R3**: prune label L if $c^1(L) + c^2(L) + LB^{tot}(i) \geq 2 \times UB$

Rule R1 identifies labels whose completion time is already at least UB, because of the customers already assigned to the drone. Rule R2 identifies labels whose vehicle tour cannot finish before UB, because of the subtour already assigned to the vehicle and the minimal remaining path to $n+1$. Rule R3 considers both the vehicle and drone completion times. It is based on the fact that whatever the path P extending the subpath associated with L to $n+1$, the completion time of the resulting solution is $\max(c^1(L)+c^1(P), c^2(L)+c^2(P)) \geq \frac{c^1(L)+c^1(P)+c^2(L)+c^2(P)}{2} \geq \frac{c^1(L)+c^2(L)+LB^{tot}(i)}{2}$.

Hence, none of the labels for which R1, R2, or R3 holds, can result in a solution better than UB. They can be pruned.

Figure 7 illustrates the *split* subroutine on the graph presented in Figure 4. List $\mathcal{L}(i)$ is reported under every node i. In Figure 7A, the bounding mechanisms are not applied. Labels can only be removed by dominance. Dominated labels are crossed out. In Figure 7B, the bounding mechanisms are reinserted. The bounding rule(s) enabling to delete a label is reported on the right of the strikethrough label.

In both cases, the optimal path and the associated labels are represented in red. The optimal decomposition is to assign customer sequence (2, 4, 5) to the vehicle and customers (1, 3) to the drone. The cost of this solution is 29.

4.3 | Decoding procedure when $M > 1$

When $M > 1$, the preceding decomposition scheme could naturally be adapted by defining labels of size $M + 1$, where the first field represents the vehicle cost and the following M fields represent the cost for each drone. This would however induce a more complex labeling algorithm and, above all, many more labels, with probably intractable computing times except for fairly small instances. We preferred to handle this situation with a different approach.

We reproduce exactly Algorithm 2 with the sole modification that arc costs c^2_{ij} are divided by M. This change simulates the fact that the total drone delivery time can be shared equally between the M drones. In other words, it also means that the PMS problem is relaxed, allowing both preemption of tasks, and parallel execution of these tasks.

Sequence τ_{drones} returned by procedure *split* is then transformed into a feasible PMS solution $(\tau_1^{opt}, \dots, \tau_M^{opt})$ with a greedy heuristic. We propose to apply the well-known longest processing time (LPT) heuristic [9]. In LPT, the tasks (drone deliveries) are first sorted in the decreasing order of their processing time $(\hat{d}_i)$. Then, they are successively assigned to the drone with the minimal current completion time.

5 | EXPERIMENTS AND RESULTS

We conducted two sets of experiments. In the first set, we evaluate the efficiency of our heuristic on Murray and Chu [5]'s benchmark instances and present some further analyzes. As far as we know, these instances are the only ones existing for this problem. Unfortunately, they are of small size (10-20 customers) and it is difficult to draw definitive conclusions with them. For this reason, we introduced instances of larger size on which we conducted a second set of experiments. In this second set, we investigate in more details the behavior of our heuristic and analyze the benefits of using drones.

The environment used for the computational work is Intel core(TM) i5-6200 U CPU at 2.30Ghz 2.40Ghz; 8GB RAM; Windows 10; 64 bits. C++ language is used for the implementation part.

For the experiments, we considered two different ways of applying our two-step heuristic:

Single-start two-stepH: the algorithm is executed exactly as it is described in Section 4.

Multistart two-stepH: the algorithm is repeated several times, with a randomized initialization and until a time limit is reached. Randomization is introduced in the nearest neighbor heuristic, by randomly selecting one of the K-closest neighbors instead of the closest. The solution returned is the best solution among all the solutions found. In the experiments, we arbitrarily set $K = 3$. The computing time limit depends on the experiments and is indicated when needed

5.1 | First set of experiments: Murray and Chu's instances

Murray and Chu's [5] instances were generated with either 10 or 20 customers. The vehicle and drone speeds were fixed at the same speed of 25 miles/h. However, distances were computed with a Manhattan distance for the vehicle and with a Euclidean distance for the drones. The depot location was selected as being either near the center of all customers, near the edge of the customer region, or at the origin. Customer locations were generated such that either 20%, 40%, 60%, or 80% of all customers were located within the drone's range from the depot, with the drone having a flight endurance of 30 minutes. Finally, 10%-20% of customers were arbitrarily set as not drone-eligible because of excessive parcel weights. Each of the above parameter settings was repeated 10 times, resulting in 120 instances with 10 customers and 120 instances with 20 customers. All these instances were solved with a single truck and either 1, 2, or 3 available UAVs, resulting in 720 test instances. For these instances, completion times are indicated in minutes.

Before analyzing the performance of our heuristic, let us recall the principle of Murray and Chu's [5] PDSTSP heuristic. Their idea is to first assume that the drones will serve all the drone-eligible customers and that all remaining customers will be delivered by the vehicle. Based on this partition, a TSP and a PMS are solved for the vehicle and the drones, respectively. An improvement step reassigns individual customers to either a drone or the vehicle if it allows decreasing the overall completion time. This improvement step is implemented as follows. If the completion time for the drones exceeds the duration of the vehicle tour, a customer is moved from a drone to the vehicle. The move affording the greatest net savings is chosen. If no move yields a savings in the overall completion time a swap is investigated. The swap operator explores all pairwise exchanges of customers between the drones and the vehicle. If the completion time for the vehicle determines the overall completion time, the swap operator is again used. The process of reallocating customers between the drones and the vehicle is repeated as long as improvements are carried out. Three TSP solution methods were evaluated for computing the vehicle tour, namely: integer programming (with the guarantee that the optimal TSP solution is obtained), the savings heuristic [10], and the nearest neighbor heuristic. Similarly, 2 PMS solution methods were evaluated for the drones: integer programming (exact solution) and the LPT heuristic [9]. In Murray and Chu's experiments, mixed integer linear programs were solved with Gurobi version 5.6.0.

TABLE 3 Comparison with Murray and Chu results

	10 Customers					20 Customers				
	Gap (%)		Runtime (s)			Gap (%)		Runtime (s)		
Solution approach	Avg	Max	Avg	Max	#Sol	Avg	Max	Avg	Max	#Sol
IP/IP	0.12	10.13	2.4856	29.97	299	0.31	23.47	495	21 510	302
IP/LPT	0.12	10.13	2.3093	28.85	300	0.41	23.47	498	21 521	292
Savings/IP	1.57	20.68	0.2373	8.26	209	3.57	18.83	3.721	80.68	87
Savings/LPT	1.58	20.68	0.0003	0.01	209	3.98	18.83	0.008	0.07	80
Exact (IP)			0.3194	2.02	360			77.775	180.00	352
Single-start two-stepH	0.12	8.51	0.1660	0.32	278	0.51	23.47	0.210	0.56	225
Multi-start two-stepH	0.02	4.45	3.2060	3.26	313	0.15	23.47	3.072	3.32	337

TABLE 4 Multistart two-stepH (time limit: 3 seconds) performance on 10 customer instances

	1 drone	2 drones	3 drones
AVG gap (%)	0.00	0.05	0.00
MAX gap (%)	0.00	4.45	0.19
AVG Obj.	114.07	105.09	91.90
#SplitCall	2	2	2
#Start (3 s)	17	19	8

TABLE 5 Multistart two-stepH (time limit: 3 seconds) performance on 20 customer instances

	1 drone	2 drones	3 drones
AVG gap (%)	0.20	0.06	0.18
MAX gap (%)	23.47	2.41	6.33
AVG Obj.	141.90	139.93	139.82
#SplitCall	2	2	2
#Start (3 sec)	15	16	15

Table 3 presents a summary on the performance of our heuristics compared to Murray and Chu's. The first four lines correspond to Murray and Chu's methods. Note that we ignored the less efficient among their methods in this table. The fifth line provides information on the computation of exact solutions by Murray and Chu with their IP formulation. The results obtained with our two-step heuristic are presented for both single-start two-stepH and multistart two-stepH, with a time limit of 3 seconds for the latter. Columns #Sol give the number of times each method found the best solution among all methods. Columns Gap give the average and maximal gaps to these best solutions, for each method. Columns Runtime report computing times in seconds.

Regarding best solutions, Murray and Chu [5] indicate that their IP formulation was able to find the optimal solution for all of the 10-customer instances but was not able to solve in the imparted time 87 of the 360 20-customer instances. Column #Sol, however, shows that for these 87 instances, IP still found the best solution among all tried methods in all cases except 8.

We observe that the single-start version of the two-step heuristic and the savings/LPT are both extremely fast. In comparable amounts of time, single-start two-stepH is able to achieve far better optimality gaps than savings/LPT. Regarding the multi-start version, we can notice that with a time limit of only 3 seconds, the two-step heuristic yields very small optimality gaps. The IP/IP and IP/LPT methods are also effective but very slow. In general, this comparison shows that our two-step descent algorithm is able to converge very quickly toward very good solutions, at least for small instances with 10 or 20 customers. Clearly, the multistart version of the algorithm should be preferred to the single-start version, as it permits to reduce optimality gaps, with a controlled additional amount of time and with a very limited additional effort in the implementation.

Tables 4 and 5 show more details of the behavior of Multistart two-stepH with 1, 2, or 3 drones for instances with 10 and 20 customers, respectively.

We notice that the average completion time value decreases when the number of drones increases. This is a logical expected result but it greatly depends on the number of drone-eligible customers and it is not always very significant. At some points, indeed, all the drone-eligible customers might be assigned to drones; then, if the vehicle tours is longer than drone tours, adding new drones does not help. Results also show that the split procedure is called 2 times on average.

Seeing that our decoding scheme is optimal when $M = 1$ (a single drone) while it is not when $M > 1$, we could have also expected larger gaps for instances with several drones. This is, however, not clearly observed with these results, probably showing the quality of the approximation introduced in the multiple-drones case.

5.2 | Second set of experiments: New instances generated from TSPLIB

To evaluate our heuristic on larger instances, we generated another set of instances by using the following files from the TSPLIB library (TSPLIB is a library of sample instances for the Traveling Salesmans Problem. Website: http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html): *att48.tsp, berlin52.tsp, eil101.tsp, gr120.tsp, pr152.tsp, and gr229.tsp*. The number in the file name corresponds to the number of customers which are represented by their coordinates x and y. To stay consistent with Murray and Chu's instances, we used the Manhattan distance for the vehicle trip and the Euclidean distance for the drone trip. From each file, we generated what we call a reference instance and several instances that can be derived from this reference instance by changing some parameters. These instances are characterized by:

- *The position of the depot*: A fictitious point was added to represent the depot. The depot location was selected as being either near the center of all customers ($x_0 = \frac{\max_{i \in N}(x_i) - \min_{i \in N}(x_i)}{2}$, $y_0 = \frac{\max_{i \in N}(y_i) - \min_{i \in N}(y_i)}{2}$) or at the left-bottom corner of the region ($x_0 = \min_{i \in N}(x_i), y_0 = \min_{i \in N}(y_i)$). In reference instances, the depot is located at the center.
- *The percentage of drone-eligible customers*: Instances were generated with either 0%, 20%, 40%, 60%, 80% or 100% of drone-eligible customers. We proceeded as follows. Let K be the target percentage of drone eligible customers. We introduced L ($L = 30\%$) the percentage of nondrone-eligible customers that are not drone-eligible because of parcel weights. We selected these customers according to their index n: customers not drone-eligible because of their parcel weight are customers, such that, $n\left(mod\left\lfloor \frac{1}{(1-K)\times L} \right\rfloor\right) = 0$. Remaining customers were then set as drone-eligible in the increasing order of their distance to the depot, until reaching $K\%$ of drone-eligible customers. In reference instances, $K = 80\%$.
- *The drone speed*: The drone can go at the same speed as the vehicle or 2, 3, 4 or 5 times faster. In this case, vector $\hat{d}$ is simply divided by the speed factor. In the reference instances, the speed factor is 2. The speed of the vehicle is assumed to be one unit of native distance per unit of time.
- *The number of drones*: The drone fleet can be composed of 1, 2, 3, 4 or 5 drones. In the reference instance, a single drone is used.

Note that the method for generating drone-eligible customers might appear a little bit twisted, but it has the advantage of being fully deterministic and easily reproducible.

The first goal with these new instances is to conduct some sensitivity analyzes. We compare solutions obtained while changing the depot position, the percentage of drone-eligible customers, the drone speed factor and the number of drones, respectively. For that purpose, we consider the six reference instances and construct new instances by changing a single parameter at a time. The method used for these experiments is Multistart two-stepH with a time limit set to 5 minutes (a large amount of time is given so as to obtain optimality gaps as small as possible and make the results more reliable).

Results are presented in Tables 6–9. In these tables, columns labeled with C.T. indicate the value of the completion time and columns labeled with #D.C. indicate the number of customers assigned to the drones.

Table 6 investigates the impact of the depot location. We observe that completion times are higher when the depot is located at the left-bottom corner. Furthermore, less customers are assigned to the drone. These observations are consistent with expectations.

In complement to this table, we also investigate if the drones tend to serve customers near the depot, in other words if there are any "rules of thumb" that dictate which customers should be served by the drones. Figure 8 shows the solutions obtained

TABLE 6 Impact of depot position (80% of drone-eligible customers, 1 drone, speed factor 2)

	Center		Corner	
Instance	C.T.	#D.C.	C.T.	#D.C.
att48	29 954	17	33 798	9
berlin52	6386.48	18	7830	8
eil101	564	28	650	19
gr120	1414	30	1730	18
pr152	76 008	20	76 556	19
gr229	1794.84	25	1913.74	9

TABLE 7 Impact of the percentage of drone-eligible customers (depot at the center, 1 drone, speed factor 2)

	0%		20%		40%		60%		80%		100%	
Instance	**C.T.**	**#D.C.**	**C.T.**	**#D.C.**	**C.T.**	**#D.C.**	**C.T.**	**#D.C.**	**C.T.**	**#D.C.**	**C.T.**	**#D.C.**
att48	42 136	0	38 662	7	31 592	15	30 788.80	16	29 954	17	27 784	16
berlin52	9675	0	9350	4	8300	15	7410	25	6386.48	18	6192	14
eil101	819	0	738	18	646	31	578	27	564	28	561.41	26
gr120	2006	0	1736	18	1624	25	1494	29	1414	30	1414.80	23
pr152	86 596	0	82 504	22	77 372	23	76 786	20	76 008	20	74 468	23
gr229	2020.16	0	1862.76	19	1828.02	22	1807.50	23	1794.84	25	1498.05	11

TABLE 8 Impact of the drone speed (depot at the center, 80% of drone-eligible customers, 1 drone)

	1		2		3		4		5	
Instance	**C.T.**	**#D.C.**	**C.T.**	**#D.C.**	**C.T.**	**#D.C.**	**C.T.**	**#D.C.**	**C.T.**	**#D.C.**
att48	33 234	10	29 954	17	29 142	21	28 686	24	28 610	26
berlin52	7450	13	6386.48	18	5656.56	23	5290.65	31	5190	35
eil101	650	17	564	28	504	36	456	39	420.83	47
gr120	1592	18	1414	30	1289.27	37	1189.71	43	1112	50
pr152	80 164	12	76 008	20	72 936	31	70 412	41	67 798	31
gr229	1865	13	1794.84	25	1735.16	36	1679.33	48	1642.04	58

TABLE 9 Impact of the number of drones (depot at the center, 80% of drone-eligible customers, speed factor 2)

	1		2		3		4		5	
Instance	**C.T.**	**#D.C.**	**C.T.**	**#D.C.**	**C.T.**	**#D.C.**	**C.T.**	**#D.C.**	**C.T.**	**#D.C.**
att48	29 954	17	28 686	24	28 610	26	28 610	26	28 610	27
berlin52	6386.48	18	5299.81	31	5290	35	5190	35	5190	35
eil101	564	28	456	40	395	52	346.68	59	319.74	69
gr120	1414	30	1188.51	43	1044.65	54	946.04	65	880	73
pr152	76 008	20	70 244	42	65 062.10	41	60 027.40	56	56 336.10	60
gr229	1794.84	25	1686.75	50	1603.90	66	1518.62	84	1483.68	98

for our six reference instances (depot position: center, 80% of drone-eligible customers, drone speed: 2, one drone). The depot is represented by a triangle. Customers that are not drone-eligible are represented by squares and drone-eligible customers are represented by circles. The circle is in plain when the given drone-eligible customer is served by the drone. Images show no clear correlation between the position of a drone-eligible customer and its selection for a delivery by a drone.

Table 7 investigates the impact of the percentage of drone-eligible customers. The table shows that, in general, completion times are improved when the percentage of drone-eligible customers increases. However, it does not necessarily imply that the number of customers assigned to the drone increases (except of course for the case with 0 drone-eligible customers). Actually, at first, when a few customers are drone-eligible, they tend to be assigned to the drone to relive the vehicle. Then, when a good balance in vehicle and drone completion times is reached, two conflicting phenomena are met: adding drone-eligible customers allows a better partition of the customers between the vehicle and the drone, and thus, should result in an higher number of drones visited per unit of time; but, it also allows decreasing completion times, and thus should result in a smaller total number of customers assigned to the drone.

In Tables 8 and 9, we vary the speed and number of drones, respectively. We can notice that when the speed of the drone increases, the completion time is improved and the drone is able to visit more customers. Similar results are obtained when increasing the number of drones. Actually, this is not a surprise seeing that in the algorithm, changing the speed or the number of drones almost has the same effect. For example, the only difference between solving an instance with 2 drones having a speed ratio 2 and an instance with 1 drone of speed ratio 4, concerns the solution of the PMS. However, one should mention that when the number of drones increases the average number of customers assigned to each drone decreases. Finally, in some cases, adding drones does not allow improving the solution, for the reasons already developed in Section 4.2 (see att48 and berlin52).

With further experiments, we analyze the behavior of our two-step heuristic general scheme. For these experiments, we only consider the reference instance obtained from *gr229.tsp* and execute single-start two-stepH. Results are presented in Tables 10 and 11.

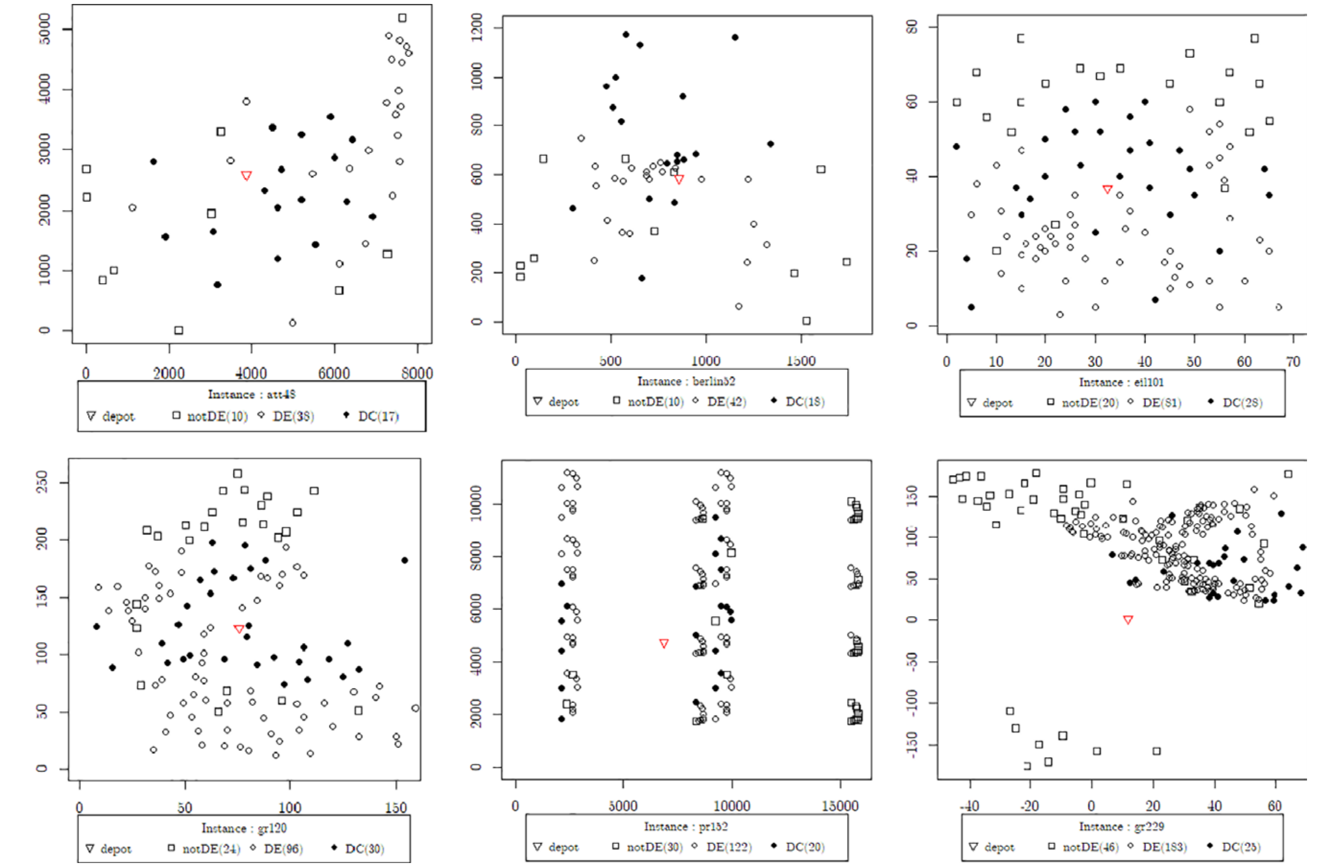


FIGURE 8 Results of the two-step heuristic for the six reference instances [Colour figure can be viewed at wileyonlinelibrary.com]

TABLE 10 Impact of the percentage of drone-eligible customers on the behavior of single-start two-sepH (depot at the center, 1 drone, speed factor 2)

Drone-eligible	C.T.	#SplitCall	#LabelsGenerated	LabelsDeleted (%)
0%	2020.16	0	0	0
20%	1867.60	3	9450	28.26
40%	1826.22	6	32 969	53.10
60%	1812.84	6	139 849	76.85
80%	1809.84	4	385 309	88.43
100%	1500.45	4	3 662 748	99.05

TABLE 11 Average gap between the drone completion time found by the split procedure and the completion time found with the LPT heuristic

No. of drones	1	2	3	4	5
Avg gap (%)	0	0.01	0.73	0.85	0.99
Max gap (%)	0	0.01	1.41	1.35	1.86

Table 10 presents the results obtained while varying the percentage of drone-eligible customers. It provides the value of the completion time, the average number of calls to the split procedure, the average number of labels generated in the split procedure (summed over all nodes and including labels eventually deleted by dominance or with the bounding mechanisms) and the percentage of labels deleted with the bounding mechanisms.

Several interesting conclusions can be drawn from this table. First, one can observe an increase in the number of calls to the split procedure following that of the number of drone-eligible customers. It indicates that, thanks to the flexibility provided by the new drone-eligible customers, the method gets less easily trapped into a local optimum. However, the method still converges quickly (a few iterations) and toward good-quality solutions: the gaps with the solutions found in 5 minutes with Multistart two-stepH (last line of Table 7) remain limited. Second, one can see a quick increase in the number of generated labels. It can clearly be explained by the shape of graph G^{τ}. Having more drone-eligible customers favors longer arcs and result in more arcs

TABLE 12 Benchmark based on reference instances with 1-5 drones applying multistart two-setpH with a time limit of 5 minutes

	1	2	3	4	5
att48	29 954	28 686	28 610	28 610	28 610
berlin52	6386.48	5299.81	5190	5190	5190
eil101	564	456	395	346.68	319.74
gr120	1414	1188.51	1044.65	946.04	880
pr152	76 008	70 244	65 062.10	60 027.40	56 336.10
gr229	1794.84	1686.75	1603.90	1518.62	1483.68

in the graph, and thus, more possibilities for extending labels. Fortunately, bounding mechanisms are capable of getting rid of most of these labels: more than 99% when all the customers are drone-eligible.

Table 11 investigates the quality of the approximation made in the split procedure when $M > 1$. It shows the average and maximum gaps between the drone completion time found by the split procedure (assuming a single drone with a speed multiplied by M) and the completion time obtained after having applied the LPT heuristic. We observe that the gaps grow when the number of drones increases, but remain very limited.

Finally, with Table 12, we provide a benchmark set of best known solutions on our new instances, for future researches. In this table, we consider our six references instances with a drone fleet size going from 1 to 5 drones. It gives rise to a new instance set of 30 instances. The results provided in the table were obtained with a single execution of Multistart two-stepH with a time limit of 5 minutes.

6 | CONCLUSIONS

Drone deliveries look like the future: unmanned aerial vehicles rapidly delivering packages to our doors, eliminating both waiting times and the cost of human labor.

In this article, we studied the combination of vehicle and drones for parcel delivery in an approach without synchronization between the vehicle and drones. We proposed an iterative two-steps heuristic that uses dynamic programming for efficiently partitioning the customers between the vehicle and the drone fleet. Results are very promising and permit a clear improvement over the existing literature.

Nevertheless, there are still a lot of possible improvements. First, we mainly focused in this study on the split procedure. Clearly the quality of the results could be slightly improved by solving better the PMS that are repeatedly solved in the solution scheme. In the current implementation, the algorithms used for solving these problems are very simple. They have the advantage to be very quick, but it is certainly possible to gain in effectiveness with a limited increase in computing times. Most importantly, the iterative algorithm could certainly beneficially be inserted within a meta-heuristic scheme. Multistart two-stepH can already be interpreted as a GRASP, but more promising meta-heuristic schemes exist. Iterated Local search, or the hybridization with evolutionary algorithms, would for example be natural candidates.

Apart from these issues, another clear limit in the evaluation of our method is the lack of lower bounds (except for small instances for which the optimal solution is known). Developing an efficient solution method, as branch-and-cut or branch-and-price, or finding tight lower bounds is certainly an important direction to follow. Branch-and-cut solution procedures have been developed for other types of routing problems where a single truck selects and visits a subset of customers (eg, the selective TSP [18] or the profitable tour problem [16]). The valid inequalities and separation algorithms used in these procedures could certainly be generalized to our problem.

Finally, more complex variants of the problem would certainly deserve to be addressed. Especially, a strong limit in the problem definition is the presence of a single vehicle. Parcel delivery typically involves many vehicles. Extending the problem to this setting would certainly be valuable, though raising new difficulties in the partitioning subproblem.

ACKNOWLEDGMENT

The authors would like to thank Chase Murray and Amanda Chu for providing their instances and their results.

ORCID

Dominique Feillet http://orcid.org/0000-0003-1246-223X

REFERENCES

[1] Amazon.com, Inc. Amazon prime air. (2013). http://www.amazon.com/primeair
[2] Andrea Ponza. Optimization of drone-assisted parcel delivery. Master's thesis, Universita Degli Studi Di Padova, Padova, Italy, 2016.
[3] Andrew Liptak. 7-Eleven just made the first commercial delivery by drone. (2016). http://www.theverge.com/2016/7/23/12262468/7-11-first-retailer-deliver-food-drone.
[4] Australia Post tests drones for parcel delivery. (2016). http://www.smh.com.au/technology/innovation/australia-post-tests-drones-for-parcel-delivery-20160415-go77a4.html
[5] C.C. Murray and A.G. Chu, *The flying sidekick traveling salesman problem: optimization of drone-assisted parcel delivery*, Transportation Research Part C: Emerging Technologies **54** (2015), 86–109.
[6] C. Gambella, A. Lodi, and D. Vigo, *Exact solutions for the carrier–vehicle traveling salesman problem*, Transportation Science **52**(2) (2017), 229–496.
[7] DHL International GmbH. DHL parcelcopter launches initial operations for research purposes. (2014). http://www.dhl.com/en/press/releases/releases_2014/group/dhl_parcelcopter_launches_initial_operations_for_research_purposes.html
[8] E. Garone, R. Naldi, and A. Casavola, *Traveling-salesman problem for a class of carrier-vehicle systems*, Journal of Guidance Control and Dynamics **34**(4) (2011), 1272–1276.
[9] M.L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 4th Edition, Springer, New York, 2016.
[10] G. Clarke and J.W. Wright, *Scheduling of Vehicles from a Central Depot to a Number of Delivery Points*, Operations Research **12** (1964), 568–581.
[11] J.C.N. Climaco and E.Q.V. Martins, *A bicriterion shortest path algorithm*, European Journal of Operational Research **11**(4) (1982), 399–404.
[12] K. Helsgaun, *An effective implementation of the Lin–Kernighan traveling salesman heuristic*, European Journal of Operational Research **126**(1) (2000), 106–130.
[13] K. Dorling, J. Heinrichs, G.G. Messier, and S. Magierowski, *Vehicle routing problems for drone delivery*, IEEE Transactions on Systems, Man, and Cybernetics: Systems **47**(1) (2017), 70–85.
[14] L'avenir est dans le ciel: Domino's et Flirtey s'associent pour livrer des pizzas par drone, conformément au code de l'aviation. (2016). https://pizza.dominos.fr/dominos-pizza/dru-drone-dominos-et-flirtey-livraison-de-pizzas
[15] Leo Kelion. Alibaba begins drone delivery trials in China. (2015). http://www.bbc.com/news/technology-31129804
[16] M.K. Jepsen, B. Petersen, S. Spoorendonk, and D. Pisinger, *A branch-and-cut algorithm for the capacitated profitable tour problem*, Discrete Optimization **14** (2014), 78–96.
[17] Marlin W Ulmer and Barrett W Thomas. Same-day delivery with a heterogeneous fleet of drones and vehicles. Technical report, Technical University of Braunschweig, 2017. https://www.researchgate.net/publication/301739923_The_vehicle_routing_problem_with_drones_Several_worst-case_results
[18] M. Gendreau, G. Laporte, and F. Semet, *A branch-and-cut algorithm for the undirected selective traveling salesman problem*, Networks **32**(4) (1998), 263–273.
[19] N. Agatz, P. Bouman, and M. Schmidt, *Optimization approaches for the traveling salesman problem with drone, Transportation Science*, in press, https://doi.org/10.1287/trsc.2017.0791.
[20] Quang Minh Ha, Yves Deville, Quang Dung Pham, and Minh Hoàng Hà. Heuristic methods for the traveling salesman problem with drone, arXiv:1509.08764. (2015).
[21] Quang Minh Ha, Yves Deville, Quang Dung Pham, et al. *On the min-cost traveling salesman problem with drone*. Transportation Research Part C: Emerging Technologies **86** (2018), 597–621.
[22] Rodrique Ngowi. UPS testing drones for use in its package delivery system. (2016). http://phys.org/news/2016-09-ups-drones-package-delivery.html
[23] S. Poikonen, X. Wang, and B. Golden, *The vehicle routing problem with drones: extended models and connections*, Networks **70**(1) (2017), 34–43.
[24] The future of healthcare is out for delivery. (2016). http://flyzipline.com/product/
[25] X. Wang, S. Poikonen, and B. Golden, *The vehicle routing problem with drones: several worst-case results*, Optimization Letters **11**(4) (2017), 679–697.

How to cite this article: Mbiadou Saleu R, Deroussi L, Feillet D, Grangeon N, Quilliot A. An iterative two-step heuristic for the parallel drone scheduling traveling salesman problem. *Networks.* 2018;1–16. https://doi.org/10.1002/net.21846