Transportation Research Part C 114 (2020) 657–680

Contents lists available at ScienceDirect

Transportation Research Part C

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

Truck-drone team logistics: A heuristic approach to multi-drop route planning

Pedro L. Gonzalez-R[a,*], David Canca[a], Jose L. Andrade-Pineda[b], Marcos Calle[a], Jose M. Leon-Blanco[a]

[a] Department of Industrial Engineering and Management Science, School of Engineering, University of Seville, Seville, Spain
[b] Robotics, Vision & Control Group, School of Engineering, University of Seville, Seville, Spain

ARTICLE INFO

Keywords:
Drones
Delivery
Tandem truck-drone
Optimization
Heuristic

ABSTRACT

Recently there have been significant developments and applications in the field of unmanned aerial vehicles (UAVs). In a few years, these applications will be fully integrated into our lives. The practical application and use of UAVs presents several problems that are of a different nature to the specific technology of the components involved. Among them, the most relevant problem deriving from the use of UAVs in logistics distribution tasks is the so-called “last mile” delivery.

In the present work, we focus on the resolution of the truck-drone team logistics problem. The problems of tandem routing have a complex structure and have only been partially addressed in the scientific literature. The use of UAVs raises a series of restrictions and considerations that did not appear previously in routing problems; most notably, aspects such as the limited power-life of batteries used by the UAVs and the determination of rendezvous points where they are replaced by fully-charged new batteries. These difficulties have until now limited the mathematical formulation of truck-drone routing problems and their resolution to mainly small-size cases.

To overcome these limitations we propose an iterated greedy heuristic based on the iterative process of destruction and reconstruction of solutions. This process is orchestrated by a global optimization scheme using a simulated annealing (SA) algorithm. We test our approach in a large set of instances of different sizes taken from literature. The obtained results are quite promising, even for large-size scenarios.

1. Introduction

The capability of unmanned aerial vehicles (UAVs or drones) to travel without a link to the road infrastructure is leading to new working models in a variety of logistic scenarios. For instance, in emergency and disaster management situations, UAVs equipped with cameras allow for viewing disaster scenes promptly, collecting critical data for the response teams on the ground. In the healthcare sector, UAVs can also be used to transport diagnostic samples or medical laboratory items between facilities. In all these situations, the use of drones instead of other transportation modes requires dealing with the limited life of the electrical batteries powering each UAV. Generally, the cooperative use of drones with another vehicle acting as a moving base is the most versatile solution, at the cost of the need of determining the most effective locations of rendezvous points. Beyond the development of new and better technology solutions for UAVs, a real transference to industry still requires the resolution of certain tactical and operational issues.

* Corresponding author.
E-mail address: pedroluis@us.es (P.L. Gonzalez-R).

https://doi.org/10.1016/j.trc.2020.02.030
Received 4 July 2019; Received in revised form 26 February 2020; Accepted 29 February 2020
Available online 14 March 2020

Currently, most of the civil uses of drones are based on the cooperative employment of a drone and a truck, as has been done for decades in the military field (Garone et al., 2011). Typically, the envisioned applications incorporate an electrical multirotor vehicle which is carried by the truck to the appropriate position, to take off and start its mission in a remotely piloted mode (see e.g., Chang and Lee, 2018; Ha et al., 2018, or Yurek and Ozmutlu, 2018). Hence, a moving truck acting as a base allows the enlargement of the action radius of the drone's operation, at the cost of complicating the determination of landing locations.

Hybrid truck-drone systems have also been reported in other practical applications where the crucial feature is the drone's ability to travel directly between two points of interest. These include: intelligence, surveillance and reconnaissance (ISR) missions to visit a set of locations (Manyam et al., 2018); first healthcare aid, especially in developing countries (Wen et al., 2016) and in emergency aid in order to reduce the worker's exposure to danger and/or facilitate a quick and flexible access to the locations of interest (Rabta et al., 2018). The latter takes special importance in situations where the transportation network is severely compromised by natural disasters, aimed at collecting crucial information for the efficient development of the emergency mission itself (see e.g., Chowdhury et al., 2017, or Greenwood, 2015).

Over the last years, such practical applications have motivated drone-related optimization approaches with objectives around to both, time and cost. The latter, mainly aimed at the degree of difficulty of operations and/or the resources consumption, oftentimes applied throughout the use of cost-drivers as the distance travelled, the number of drones or the number of batteries that a certain solution requires. However, as recently showed in Schermer et al. (2019), the more usual performance indicators in hybrid drone-truck scenarios are time-related: the minimal completion time for serving the set of locations, the minimal time in which the later vehicle retrieval occurs at the end of operations or the saving time compared to service by truck only.

Our research covers some of the gaps identified in the conceptualization of new working models around the use of drones in the above-mentioned logistic scenarios. In particular, we focus on the applications where a drone trip is capable of serving several locations (every trip comprises of a launching, the visit to a typically small number of locations and a landing in the ground vehicle at the next rendezvous point). This differs from the commonly accepted hypothesis in last-mile delivery literature which considers that only one single location is visited at each drone's trip – with the exception of only a few works, see e.g., Wang and Sheu (2019), or Karak and Abdelghany (2019). However, notice that this limitation is not necessarily applicable; for instance, current drone technology allows for picking-up small items –e.g., vaccines, water purification tablets, or medicines- and delivering them one-at-a-time: a sequence of locations are then served by the same drone's trip. In our approach to the problem, every location stands for a customer that can be served either by drone or by truck. We assume the truck stops at certain customer sites (which are not predefined) and hence, the drone is only allowed to merge with the ground vehicle at these locations. Once landed, the drone always gains a fully charged battery and would then be ready to start a new trip or stay at the truck while the truck carries it to a new services area.

The main contributions of our paper can be summarized as follows:

- We present a novel truck-drone team logistic (TDTL) mathematical formulation, allowing multi-drop routes for the drone. Under the assumption of infinite travel autonomy for the truck and a limited battery life for the drone, we plan the synchronization events –namely, the sites wherein the drone will gain a fully-charged new battery- with the objective of minimizing the makespan. We assume that the truck-drone team starts from an origin (depot) and has the mission of reaching an end node, visiting the entire set of locations (customers), either by the truck or the drone. All the routes in the problem are open in the sense that are not predefined.
- In order to solve real-life instances of the above challenging mixed integer model (MIP), we propose a heuristic method with an innovative coding scheme, a fast and high-quality constructive two-step procedure for obtaining the initial solution and two original local search procedures for improving solutions. We code the solutions with a tuple formed by a node-sequence vector (defining the order of visits) and a resource-type vector containing the type of resource that visits each node (truck, drone or truck-drone) in the node-sequence vector.
- We compare the resolution of the instances by using the Gurobi's Branch-and-Cut algorithm with the heuristic approach in a set of 1080 literature-based instances. The heuristic procedure exhibits a good performance in solving all the instances in a reduced time while providing good solutions.
- The developed heuristic allows its transfer to real-life cases, since it is able to obtain solutions of relatively good quality for large-size problems in short computation time.

2. Literature review

The last-mile delivery is by far the most prominent research stream on hybrid truck-drone systems. Amazon, DHL, Google and Alibaba have conceptualized their logistics transportation solution around this new cooperative approach, so that a truck loaded with one or several drones gets near the delivery locations, which are ultimately served by a drone. In view of the new method for crossing the last-mile to the customer sites, the delivery service plan is to be built on the assumption that the truck acts as a mobile depot to pick the parcel up. While some authors give an active role to the trucks in the delivery process in the sense that they are able to deliver the products to the customers (see e.g., Murray and Chu, 2015, or Ha et al., 2018), others assume that the truck does not visit customers (see e.g., Karak and Abdelghany, 2019, or Ferrandez et al., 2016). The former forces the planner to face a synchronization problem, since the truck and the drone need to wait for each other. In other works (see Agatz et al., 2018) such a synchronization issue is mitigated by using a policy of "drone following the same road network as the truck", although this is somehow contradictory with fully exploiting the last-mile drone delivery advantage.

The problem of pairing drones with traditional delivery trucks was first devised by Murray and Chu (2015). These authors study two types of truck-drone delivery problems. The first one, the flying sidekick traveling salesman problem (FSTSP), concerns the case in which the distribution center (DC) is far from the customer locations. In this problem the truck must visit some fixed customers due to the weight of the parcels involved, while the rest of the parcels can be delivered by the drone carried by the truck. The second one is the parallel drone scheduling traveling salesman problem (PDSTSP). This problem corresponds with the situation in which a significant proportion of customers are located within a drone's flight range from the DC. In this model, drones depart from the depot, deliver parcels to clients and return to the depot. At the same time, the truck services some clients without carrying any drone. Both drone and truck are working simultaneously in parallel. Authors propose simple heuristics to obtain solutions to these problems. Recently, Murray and Raj (2020) have addressed the so-called multiple FSTSP, which considers an arbitrary number of heterogeneous UAVs along with the truck, with attention to real-life issues as it is the scheduling of launching and landing activities.

Traditionally, the studies on the transformed last-mile delivery are focused on the routing decisions of truck and drones. The problem that arises is called Travelling Salesman Problem with Drones (TSP-D). Chang and Lee (2018) consider that a single truck carries the drones to some centers from which drones fly to serve the customers. Moshref-Javadi and Lee (2017) use a truck-multi-drone cooperative delivery system to minimize delays in a customer-oriented distribution system. At each stop site, the truck will wait until all drones come back. The truck then carries drones to the next node. Ferrandez et al. (2016) analyze a cooperative truck-drone tandem, considering that the truck follows the route generated by initially solving a traveling salesman problem (TSP). Then, at each truck stop one or more drones can be launched to deliver parcels. In this way, each truck stop is like a hub for drone deliveries. Authors first applied a K-means clustering algorithm to determine truck stops (by grouping customers) and then applied a genetic algorithm to determine the truck TSP tour. Again, concerning the parallel drone scheduling problem, Ham (2018) extends the original PDSTSP proposed by Murray and Chu (2015) by assuming that drones can perform not only delivery operations but also pick-up activities. After a drone finishes the delivery task, it can fly back to the depot and then begin a new trip, or it can fly to the next client to pick up a returned parcel, in such a way, each drone can visit only two consecutive nodes and no synchronization with the truck occurs at all. Moreover, as in the original PDSTSP, trucks and drones' routes are decoupled. Boysen et al. (2018) provide an integer model and a complexity analysis of several variants of the PDSTSP (considering robots instead of drones) based on the number of available robots and whether or not they are launched. The paper develops scheduling procedures which determine the truck route along robot depots and drop-off points where robots are launched, such that the weighted number of late customer deliveries is minimized. Customers are only served by robots, and the truck route visits several robot depots where the truck can replenish robots. Authors propose a heuristic approach which is compared against the Gurobi Branch-and-Cut solver for small instances (up to 10 nodes).

Wang et al. (2017), Poikonen et al. (2017), Schermer et al. (2019) and Sacramento et al. (2019) have considered a last-mile delivery around a fleet of homogeneous ground vehicles (trucks) equipped with multiple drones, when stating the so-called Vehicle Routing Problem with Drones (VRP-D). Wang et al. (2017) and Poikonen et al. (2017) make a comparison between the amounts of time (best case scenario) that can be saved using a cooperative scenario (truck-drone) when compared with the delivery using only trucks. Sacramento et al. (2019) aim their operational costs minimization approach at the equalization of the working time of truck drivers (all the routes restricted to a maximum duration). Schermer et al. (2019) have provided a comprehensive approach to the makespan minimization VRP-D, firstly incorporating several sets of valid inequalities to its MILP formulation and secondly proposing a matheuristic procedure that exploits the structure of the VRP-D to leveraging the commercial MILP solvers. The use of multiple trucks and multiple UAVs is considered also by Kitjacharoenchai et al. (2019), which is an extension of the FSTSP, but without endurance limitations and launch/delivery time considerations. While UAVs can be launched from and retrieved at different trucks, only one UAV can be launched or retrieved at any customer location. They present a MIP formulation and propose an insertion-based heuristic to solve problems with up to 50 customers. Ulmer and Thomas (2018) study a same-day delivery problem with trucks and drones, in which customer orders come dynamically during a work shift. The involved decisions concern the acceptance or rejection of customer orders for same-day delivery, and, if accepted, the method of delivery, by selecting between a truck or a drone. Pugliese and Guerriero (2017) incorporate time windows constraints for delivery goods. In their experiments, although the use of drones is not economically positive, they conclude that a cooperative strategy can reduce negative environmental impacts and improve service quality. Campbell et al. (2017) formulate and optimize continuous approximation models of hybrid truck-drone delivery, where both trucks and drones make deliveries simultaneously and compare their performance with respect to truck-only delivery scenarios. Wang and Sheu (2019) propose a MIP for the VRP-D and provide a branch-and-price algorithm for its exact resolution using randomly generated instances up to 15 nodes.

We can find approaches in which the role of drones is even more relevant. Dayarian et al. (2017) focused on a same-day delivery problem with drone resupply, where trucks oversee delivery orders and the role of drones is committed to resupplying the trucks. Authors present a heuristic to solve the problem with only one truck and one drone. Furthermore, Cheng et al. (2018) address a distribution problem where there are no trucks at all as a VRP which is further solved using a Branch-and-Cut algorithm. The valid cuts introduced come from the nonlinear (convex) energy function modeling drones' energy consumption (which is influenced by payload and travel distance). Dorling et al. (2017) proposed a linear approximation to the non-linear drone energy consumption, linearly depending on the drone payload and the battery weight, considered as a decision variable. In their model, each drone, departing from a given depot, can perform multiple trips and visit multiple customers per trip, in a similar way than a VRP problem, but considering battery endurance constraints. The model is approximately solved using a SA heuristic for different scenarios up to 500 customers. We refer to Dorling et al. (2017) for the variety of factors that intervene in the maximum operation time or endurance of a flight mission.

Henceforth, we concentrate our revision two-fold: (i) in the use of a single truck equipped with a single drone, both cooperating to serve a set of locations and (ii) in the resolution methods.

The tandem truck-drone has been studied for urban contexts where only the drone is responsible for delivering parcels to customers: the truck serves the aim of traveling along the street network and the drone's purpose is to reach the customer's doorstep (see Mathew et al., 2015, and bin Othman et al., 2017). Mathew et al. (2015) consider that the truck can wait at a node until the drone returns or go to the next vertex to rendezvous with the drone. The same assumption is considered in bin Othman et al. (2017), which in addition start from a fixed route for the truck –i.e. predefined for supply issues- and define the routing decisions for the drone. In contrast with both references, in our research we extend the FSTSP model to allow the drone to visit several customers per trip, between two consecutive rendezvous with the truck. Certain similarities arise with Luo et al. (2017), which study an ISR problem with a truck and its carried drone. However, while they force the truck to select from among a set of candidates stopping points that are adequate for rendezvous operations, in our paper the possible rendezvous nodes are not-a-priori defined.

As regards the relevant resolution methods published, they mostly focus on the FSTSP. Ponza (2016) proposes a SA heuristic to solve the FSTSP. Bouman et al. (2017) use a dynamic programming (DP) approach for the FSTSP. Marinelli et al. (2017) extend the FSTSP by allowing launching and rendezvous operations along the route arcs. The authors propose a greedy randomized adaptive search procedure for solving this problem. Agatz et al. (2018) extend the FSTSP, allowing the truck the possibility of waiting for the return of the drone at the start node. They propose a route-first cluster-second heuristic method. More recently, Jeong et al. (2019) extend the FSTSP problem to include two practical considerations concerning the drone operations: the existence of non-fly zones and the effect of parcel weight on drone energy consumption.

Based on the conducted review, we can state that papers addressing the truck-drone cooperative system mostly assume that during each trip a drone can only visit one customer, except for Ham (2018), Luo et al. (2017) for the FSTSP and Cheng et al. (2018) for the PDSTSP (the latter involving a pure drone-delivery scenario where no truck intervenes at all).

Our research work assumes a set of locations to which we need either deliver lightweight relief items or visit for an ISR mission. Like us, Luo et al. (2017) and Wang and Sheu (2019) assume that both the truck and the drone can serve the customers. Luo et al. (2017) have evolved the FSTSP, when designing a truck-drone cooperative ISR mission over a set of surveillance target locations. Wang and Sheu (2019) have addressed the routing decisions for trucks and drones in an urban last-mile delivery problem under the following practical operational considerations: (i) the existence of a predetermined set of docking hubs –i.e. large areas to carry out a controlled landing -, (ii) the possibility for drones of taking-off directly from the depot, from a docking hub or from a truck and, (iii) the latter is only allowed at one of the customer locations. However, both works limit the locations where the synchronization among trucks and drones can occur: Wang and Sheu (2019) schedule it at the docking hubs, whereas Luo et al. (2017) schedule the rendezvous points at truck stops among a preselected subset of candidate locations (a simplification that also appears in Sacramento et al. (2019) and Karak and Abdelghany (2019). On the contrary, in our approach, all the locations are open to be visited either by the drone or by the truck; therefore, all of them are potential points for both stop and rendezvous. Besides, while Sacramento et al. (2019) impose the truck cannot wait for the drone at the same location it was launched, in our approach we get the same result without enforcing this constraint: owing to the active role played by the truck, it would not be of interest that a launch and a rendezvous occur at the same location. Indeed, for us the synchronization issue is crucial, since it turns the problem into finding the chain of customers to be served by each vehicle while specifying the locations where the battery swaps will take place.

We next describe the formulation of our TDTL problem, a generalization of the FSTSP model proposed by Murray and Chu (2015) capable of planning multi-visit per drone's trip in tandem with a truck that actively serves some a priori unknown locations, acting in addition as a ground base (recharging and rendezvous).

3. Problem description

This problem consists of finding the routes for a drone and a truck that collaboratively have to visit a set of customers, with different starting and finishing points, as is typically required in practical ISR missions.

Since a single drone cannot visit all customers (owing to a short flight range due to the limited battery life), the truck must assume the role of a battery swap station. While this fact has been typically addressed by assigning a mothership role to the truck – with the only mission of carrying the drone, see e.g., Poikonen et al. (2017), and Wang et al. (2017)–, in our study the truck itself participates also in serving customers. Moreover, we use a flexible definition of the truck routes, in contrast to what is often assumed in the literature, where the truck route must be completely pre-defined when determining the operation of the drone as in bin Othman et al. (2017), or partially defined (see e.g., Murray and Chu, 2015, Mathew et al., 2015, or Luo et al., 2017) owing to certain customers requiring a visit by truck. The latter makes some sense in delivery operations (e.g., due to heavy deliveries, as in Sacramento et al., 2019), but not in the case of small items delivery or in the case of reconnaissance missions. In short, in our study we are interested in covering the more general case in which each customer can be served or visited either by truck or by drone.

In addressing this more general problem, we make the following assumptions:

1. We disregard the maximum distance the truck can operate (the truck can finish its route without the need of refueling).
2. Drones can only merge with a truck at a customer node (rendezvous locations) and not at intermediate locations. The change of the drone's battery is made at rendezvous points (see Fig. 1). The truck and drone routes are synchronized at the rendezvous locations –i.e. if the truck arrives before at the customer, the truck must wait for the drone, and vice versa.
3. There is no limitation to the number of batteries or swap operations made on the drone, the swap-time being negligible as compared with the total time of the mission. Moreover, since the battery is replaced by a fully charged new one each time the drone and the truck merge, the drone battery is completely full when the drone takes off from the truck.
4. As long as the drone has enough battery, it can visit multiple customers on each trip that ends by merging with the truck at a new

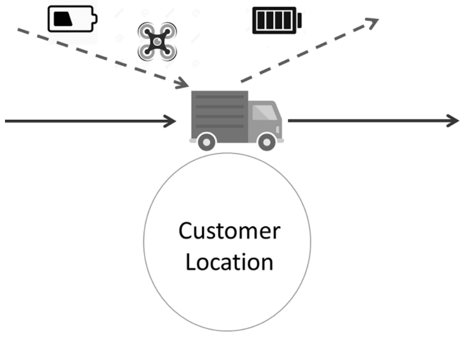


Fig. 1. Merging procedure at customer locations.

customer location.

5. The drone, if needed, can travel a sub-route transported in the truck, in which case, there is no consumption of battery.
6. Each trip is composed of segments interconnecting the locations, measured by the Euclidean distance between them. They are crossed at constants speeds by both the truck and the drone.
7. For the sake of simplicity, the pattern of drone's energy consumption is assumed to be linear. We do not consider the possible excess of consumption during take-off or landing operations, nor the non-linear pattern which therefore appears as an effect of the load carried.
8. Each customer must be served either by a drone or by a truck.

4. Mathematical formulation

This section presents the mathematical model for the described problem. The different sets needed for modelling the problem, the parameters and variables are firstly defined. Then, the formulation is presented, grouping the constraints according to their specific function.

4.1. Notation

Sets:

$G = \{N, A\}$	Graph defining the set of locations or nodes to be visited and the set of directed links connecting them.
N	Set of nodes of graph G.
o	The origin node of the mission, $o \in N$.
e	The ending node of the mission, $e \in N$.
A	Set of directed links in G
$\delta^{+}(i)$	Set of nodes that can be reached from node $i \in N$using links in A.
$\delta^{-}(i)$	Set of nodes that can be used to reach node $i \in N$ using links in A.

Parameters:

Q	Maximum level of drone battery charge (just when it is replaced), expressed in time units.
t_{ij}^{T}	Truck running time at link $(i, j) \in A$.
t_{ij}^{D}	Drone running time at link $(i, j) \in A$.
M	A big enough constant.

Variables:

u_{ij}	Binary variable equal 1 if the link $(i, j) \in A$ is traversed by the truck.
v_{ij}	Binary variable equal 1 if the link $(i, j) \in A$ is traversed by the drone.
s_i	Continuous variable that measures the earliest departure time from node $i \in N$, once it is served either by the drone or by the truck.
b_i^{-}	Continuous variable that measures the drone battery level when the drone is just coming to the node $i \in N$.
b_i^{+}	Continuous variable that measures the drone battery level when the drone is just departing from node $i \in N$.

4.2. Model formulation

We can formulate the TDTL problem as the next MIP model, where the variable s_e is minimized:

$$\text{Objective Function} \quad Min \;\; s_e \tag{1}$$

st:

(A) Truck shortest path constraints

$$\sum_{j\in\delta^-(i)} u_{ji} \le 1 \quad i\in N\backslash\{o, e\} \tag{2}$$

$$\sum_{j\in\delta^+(i)} u_{ij} - \sum_{j\in\delta^-(i)} u_{ji} = 0 \quad i\in N\backslash\{o, e\} \tag{3}$$

$$\sum_{j\in\delta^+(o)} u_{oj} = 1 \tag{4}$$

$$\sum_{i\in\delta^-(e)} u_{ie} = 1 \tag{5}$$

(B) Drone shortest path constraints

$$\sum_{j\in\delta^-(i)} v_{ji} \le 1 \quad i\in N\backslash\{o, e\} \tag{6}$$

$$\sum_{j\in\delta^+(i)} v_{ij} - \sum_{j\in\delta^-(i)} v_{ji} = 0 \quad i\in N\backslash\{o, e\} \tag{7}$$

$$\sum_{j\in\delta^+(o)} v_{oj} = 1 \tag{8}$$

$$\sum_{i\in\delta^-(e)} v_{ie} = 1 \tag{9}$$

(C) Truck-Drone shortest path constraints

$$\sum_{i\in\delta^-(j)} u_{ij} + \sum_{i\in\delta^-(j)} v_{ij} \ge 1 \quad j\in N\backslash\{o\} \tag{10}$$

(D) Synchronization constraints

$$s_j \ge s_i + t_{ij}^T\cdot u_{ij} - M\cdot(1-u_{ij}) \quad (i, j)\in A \tag{11}$$

$$s_j \ge s_i + t_{ij}^D\cdot v_{ij} - M\cdot(1-v_{ij}+u_{ij}) \quad (i, j)\in A \tag{12}$$

$$s_o = 0 \tag{13}$$

(E) Battery level constraints

$$b_j^- \le Q + M\cdot(2-v_{ij}-u_{ij}) \quad (i, j)\in A \tag{14}$$

$$b_j^- \ge Q - M\cdot(2-v_{ij}-u_{ij}) \quad (i, j)\in A \tag{15}$$

$$b_j^+ \le Q + M\cdot(2-v_{ij}-u_{ij}) \quad (i, j)\in A \tag{16}$$

$$b_j^+ \ge Q - M\cdot(2-v_{ij}-u_{ij}) \quad (i, j)\in A \tag{17}$$

$$b_j^- \le b_i^+ - t_{ij}^D + M\cdot\left(1 - v_{ij} + u_{ij} + \sum\nolimits_{k\neq i} u_{kj}\right) \quad (i, j)\in A \tag{18}$$

$$b_j^- \ge b_i^+ - t_{ij}^D - M\cdot\left(1 - v_{ij} + u_{ij} + \sum\nolimits_{k\neq i} u_{kj}\right) \quad (i, j)\in A \tag{19}$$

$$b_j^+ \le b_j^- + M\cdot\left(1 - v_{ij} + u_{ij} + \sum\nolimits_{k\neq i} u_{kj}\right) \quad (i, j)\in A \tag{20}$$

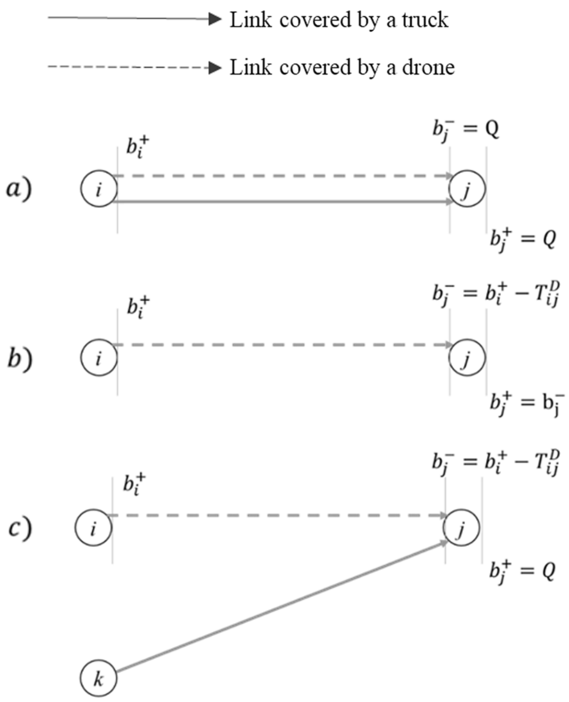


Fig. 2. Drone battery constraint types.

$$b_j^+ \geq b_j^- - M \cdot \left(1 - v_{ij} + u_{ij} + \sum\nolimits_{k \neq i} u_{kj}\right) \quad (i, j) \in A \tag{21}$$

$$b_j^- \leq b_i^+ - t_{ij}^D + M \cdot \left(1 - v_{ij} + u_{ij} + 1 - \sum\nolimits_{k \neq i} u_{kj}\right) \quad (i, j) \in A \tag{22}$$

$$b_j^- \geq b_i^+ - t_{ij}^D - M \cdot \left(1 - v_{ij} + u_{ij} + 1 - \sum\nolimits_{k \neq i} u_{kj}\right) \quad (i, j) \in A \tag{23}$$

$$b_j^+ \leq Q + M \cdot \left(1 - v_{ij} + u_{ij} + 1 - \sum\nolimits_{k \neq i} u_{kj}\right) \quad (i, j) \in A \tag{24}$$

$$b_j^+ \geq Q - M \cdot \left(1 - v_{ij} + u_{ij} + 1 - \sum\nolimits_{k \neq i} u_{kj}\right) \quad (i, j) \in A \tag{25}$$

$$b_o^+ = Q \tag{26}$$

Constraints (2) ensure that a node can be visited by one truck as maximum. Constraints (3) correspond to node balances between incoming and outgoing links for the truck. Constraints (4) and (5) enforce that the number of outgoing links from node o and the number of incoming links at node e must be equal 1, respectively. Constraints (6)–(9) are equivalent to the previous ones but they are relative to the drone instead the truck. Constraints set (10) ensures that each node must be visited by a drone or a truck (or both). Constraints (11) and (12) are in charge of calculating the departure time at each node. Constraint (13) enforces the departure time at the origin node. Constraints sets (14)–(25) are used to compute the level of drone's battery at each node. Finally, constraint (26) set the initial battery load in the origin node at its maximum level Q. To better understand how these constraints work, we distinguish three cases, according to Fig. 2.

Constraints (14)–(17) are used to model the case (a). If the link (i, j) is traversed simultaneously for the truck and the drone, the level of drone battery at the end of the link (before coming to node j) is Q and the battery level after leaving node j is also Q. Constraints (18)–(21) model the case (b). In this situation, the drone battery level just coming to the node j must be $b_j^- = b_i^+ - t_{ij}^D$ and the battery level when the drone leaves node j must be equal to b_j^-. The case (c) is modelled by constraints (22)–(25). In this case, when the drone is coming to the node j, its battery level must be $b_i^+ - t_{ij}^D$, however, after leaving node j, the battery must have been replaced and consequently the battery level will be Q. The objective function (1) consists of minimizing the total travel time which is represented by the variable s_e.

Table 1
Feasible values of elements in the resource-type vector (π_t).

	$\pi_t(i) = k$
$k = 1$	The node at position i in π_s is only visited by a drone
$k = 2$	The node at position i in π_s is only visited by a truck
$k = 0$	The node at position i in π_s is visited by both, the drone and the truck

5. Solution' coding, initial solution and search space considerations

This section presents the algorithm designed to solve the TDTL problem. We first introduce the coding scheme used to represent the solutions to our problem, explaining the way of recognizing synchronization events. Then, we deal with the mechanism designed to ensure that the algorithm will select feasible solutions. In order to illustrate de degree of difficulty on solving the TDLT problem, the section concludes by analyzing the differences in the solutions' search space among our problem and the TSP and PFSP (permutation flowshop scheduling problem).

5.1. Solution's coding scheme

Each solution π is encoded with a tuple of two array components (π_t, π_s). The first one is a resource-type vector (π_t) containing at the i position a value k standing for the type of vehicle –i.e. truck, drone, or both- visiting the subsequent node stored into the node-sequence vector (π_s) at the same position. Thus, $\pi_t(i) = k$ means that the node at position i in π_s is only visited by a drone if $k = 1$, only visited by a truck if $k = 2$, or visited by both a truck and a drone if $k = 0$ (see Table 1). For a problem with n nodes, the $(n - 2)$ elements in the middle part of the array π_s can be ordered as a permutation from the set $S = \{1, 2, \cdots, n - 1\}$, whereas the first and last components in the sequence $\pi_s$will always be the origin node (o), and the end node (e).

Next, in Fig. 3 we show the coding scheme applied to an illustrative example. It can be observed that, for instance, $\pi_s(5) = 1$ and $\pi_t(5) = 0$ indicates that the fourth served location is that corresponding to node 1, which is visited by both, the drone and the truck.

5.2. Synchronization nodes

We code the existence of a synchronization event by means of a zero value in the corresponding element in π_t, assuming that the output from the involved node occurs after the latest vehicle arrive. Therefore, in this situation, a truck may have to wait for the drone or alternatively, the drone must wait for the truck. It has been considered without loss of generality that there is no set-up time between the arrival of the last vehicle and the departure from the node. At those synchronization nodes the battery change is made. Moreover, with respect to π_t, it must be considered that both the first and the last elements (0 and n) are always 0, since a synchronization is necessary for both: in the exit of the starting node (o), as well as in the arrival to the final node (e). See the example shown previously in Fig. 3.

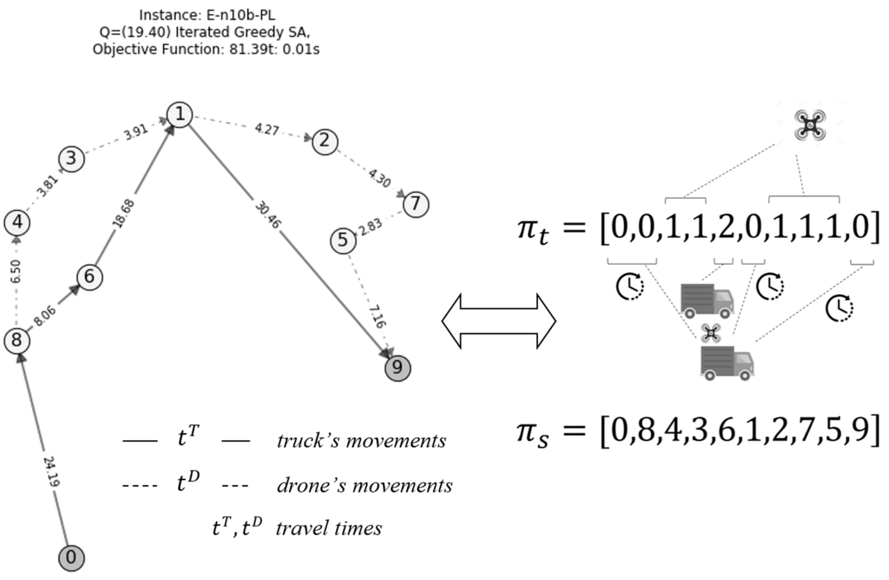


Fig. 3. Codification example.

5.3. Feasible solutions

Notice that, at any point of the drone's trip, the remaining level of battery must be enough to further visiting the remaining nodes. Thus, for any partial mission (or sub-route) finishing at certain node i, the drone should be synchronized again with the truck to charge or replace its battery, hence, at this node $\pi_t(i) = 0$. Let q_r be the necessary battery life for a certain route r in the set of a priori unknown routes R made by the drone, measured in time units. If the maximum capacity of a battery is Q, the next set of restrictions must be satisfied:

$$q_r \le Q, \qquad r \in R \tag{27}$$

All the feasible solutions to the problem have to meet the battery life requirements for all of its drone sub-routes. In what follows we specify how these constraints are incorporated in the solving procedure.

5.4. Objective function

The objective function of our model minimizes the variable s_e, which measures the earliest departure time from the node e, i.e., the time needed for visiting all the nodes in the graph.

However, from a solving point of view, when generating different permutations (candidate solutions) of the type of nodes in the array π_s, regarding the battery capacity constraints, some of these permutations could be not feasible. In order to avoid the need of checking the feasibility of the solutions that will be successively generated by our heuristic procedure we have proceeded to penalize the deficit of battery life in the problem's objective function, which would be formulated as follows:

$$minf = s_e + p\cdot\max_{r\in R}(0,\, q_r - Q) \tag{28}$$

Therefore, for a given solution the objective function will contain a penalty term only if there is not enough battery capacity to complete all the sub-routes, whereas, if the solution is feasible, the penalty term turns to be zero.

As a value for the penalty weight(p), the ratio of the speed of the drone to the speed of the truck (v_D/v_t) has been taken, assuming that in practice $v_d > v_t$. Then, if a certain solution is not feasible, the solution is penalized translating the deficit of battery life in terms of time at the speed of the slowest vehicle. In this way, the problem of getting solutions with low objective function values but at the expense of selecting non-feasible drone's sub-routes (long routes without intermediate battery replacement), is solved.

5.5. The search space

In order to obtain a deep insight about the difficulty of solving the TDTL problem, we make a comparison of its solution's search space (SS) with respect to that of the TSP and PFSP. To this end, we first consider these three problems as pure permutation problems (the usual representation when a heuristic or metaheuristic solving procedure is followed). Later, from the point of view of their mathematical formulation, we compare the minimum number of binary variables required to formulate and solve the PFSP, TSP and TDTL problems.

Comparison as a permutation problem

According to the proposed solution's coding scheme, the TDTL problem search space consists of a permutation of $n - 2$ elements with 3 different states, i.e., $SS_{PER}(n) = (n-2)!3^{n-2}$.

Note that a PFSP with n tasks has a solutions space of size $n!$. It is worthy to mention that, as the number of nodes increases, the solutions' search space of the TDTL problem becomes quite larger than the one of the PFSP, and therefore solving the truck-drone problem is much more complex. For a small number of nodes (10 nodes) the size of the solutions search space of the TDTL problem is 72.9 times the space of the PFSP, this quantity increases up to more than 1 million times for the case of 20 nodes, according to the expression:

$$\frac{SS_{PER}(n)}{n!} = \frac{3^{n-2}}{n(n-1)} \tag{29}$$

This situation is even worse when comparing with the TSP, since a TSP with n nodes presents a solution space of size $(n-1)!$ if the depot has been fixed. Table 2a summarizes the search space dimension for different instance sizes according to the number of nodes.

Comparison in terms of the number of binary variables

Table 2a
Comparison of search space sizes vs number of nodes (Permutations).

	TDTL vs PFSP	TDTL vs TSP
n	$SS_{PER}(n)/n!$	$SS_{PER}(n)/(n-1)!$
5	1.35	6.75
10	72.9	729
20	> 1M	> 2M
50	> 3.25E+19	> 1.62E+21

Table 2b
Comparison of search space sizes vs number of nodes (MIP).

	TDTLvsPFSP	*TDTL*vs*TSP*
n	$\frac{SS_{MIP}(n)}{2^{n^2}}$	$\frac{SS_{MIP}(n)}{2^{n(n-1)}}$.
5	33,554,432	1,073,741,824
10	$>$ 1.26E+30	$>$ 1.29E+33
20	$>$ 2.58E+120	$>$ 2.7E+126

In general, the number of solutions in the search space of a MIP model depends on both the number of variables and constraints. In order to point the reader to the differentiated computational efforts involved in exactly addressing the above mentioned problems, we make a comparison among them solely in terms of the number of variables, as a reasonable indicator about the difficulty of commercial solvers in managing such kind of problems.

The MIP formulation uses $n(n-1)$ binary variables for the TSP and $n^2$for the PFSP (continuous variables are not considered). The TDTL problem uses two binary variables u_{ij} and v_{ij} for (i, j) in A (set of directed links in the graph), and therefore $2n^2$ binary variables. That means that the number of binary variables used in the TDTL problem $SS_{MIP}(n)$ is always twice than the PFSP and greater than twice, $2 + 2/(n-1)$ when compared against the TSP problem. Since the considered variables are binary, the unconstrained search space of these problems can be obtained as 2 raised to the number of variables.

Table 2b highlights the number of times that the space of solutions of the PFSP or TSP is contained in $SS_{MIP}(n)$ of the TDTL for different instance sizes according to the number of nodes n and both problems (PFSP and TSP).

The two above comparisons point out that solving the TDTL problem, either as a permutation problem or by mathematical programming, is much more complex than solving a PFSP or TSP of similar size.

Fig. 4 shows the ratio between the search space using MIP and the one using the permutation approach for the TDTL, i.e. $SS_{MIP}(n)/SS_{PER}(n)$.

It can be observed that for $n \geq 10$ the search space by the MIP approach exponentially increases with respect to the permutation approach. This suggests not only the burden found for a MIP approach for instances of size greater than 10, but also that a permutation approach exhibits smaller increases on the search space beyond this limit. In practice this means that the greater the instances, the greater the comparative advantage of using a heuristic (combinatorial) method on them.

6. The algorithm

6.1. The iterated greedy approach

To solve the problem and given the similarity of the proposed coded solution with permutation problems, we propose an adaptation of an iterative greedy search algorithm (IG) that was used efficiently for task completion time minimization in a permutation flowshop scheduling problem (see Ruiz and Stützle, 2007). The IG approach was inserted into a SA framework to guide the global search, and specifically adapted to our problem as described in Algorithm 1.

The process starts with an initial solution πthat feeds the block carried out by the SA. Once in the block, the procedure begins with a solution destruction and a solution construction mechanism which gives rise to a new solution π''. After this phase, a local search is performed, which works over the two arrays $\pi_s$and $\pi_t$of π'' through a local insertion mechanism over $\pi_s$and a search for promising solutions on π_t. In the next, each of the algorithm steps is described in detail. Concerning the SA, as in previous works we select an initial temperature based on the traveling costs (distances or times) between nodes and the number of nodes (see e.g., Ruiz and Stützle, 2007). The initial temperature T used, for a given adjustable parameter Ω is as follows:

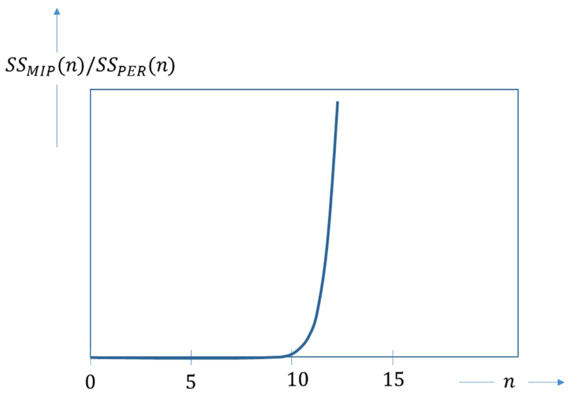


Fig. 4. Ratio between SS_{MIP} and SS_{PER}.

$$TStart = \Omega \frac{\sum_{\forall (i,j) \in A} t_{ij}^{T} + \sum_{\forall (i,j) \in A} t_{ij}^{D}}{n} \tag{30}$$

where t_{ij}^{T} is the travel time by truck and t_{ij}^{D} the travel time by drone for every $(i, j) \in A$. The cooling factor (β) considered to decrease the temperature at each iteration has been set to 0.995. Moreover, in this work $\Omega = 1$ and $T_{end} = 0$.

Algorithm 1 (*IG procedure*).

```
procedure IG:
  $T: =TStart$
  $\pi$: = Initial_solution
  repeat (while termination criterion not satisfied):
    $\pi'$: = Destruction ($\pi$)
    $\pi''$: = Construction ($\pi'$)
    $\pi'''$: = Combined_Local_search ($\pi''$)
    if $f(\pi''') < f(\pi)$:
      $\pi: =\pi'''$
      if $f(\pi) < f(\pi_{best})$ and $\pi$ is feasible:
        $\pi_{best}: =\pi$
    elseif $random(0, 1) < \exp\left(-\frac{(f(\pi''') - f(\pi))}{T}\right)$:
      $\pi: =\pi'''$
    update $T: =\beta \cdot T$
return $\pi_b$
```

6.2. Computing a feasible initial solution

Given the absence of direct methods to get a good initial solution to the addressed TDTL problem, we have devised a procedure for generating high quality initial solutions. It is a constructive routine that starting from an initial drone-only serving solution, evolves it by using a two-step heuristic combining the 2-opt and Dijkstra's algorithms for identifying the rendezvous points that finally have to be part of the truck route.

The idea of generating an initial solution from solving a TSP problem on all the nodes has formerly been used in Murray and Chu (2015) and Luo et al. (2017). Murray and Chu (2015) take as a starting route that obtained from solving a TSP assuming that initially all the nodes are visited by the truck. Later some of the nodes are selected so that they become part of the sub-routes made by the drone. In contrast, Luo et al. (2017) start from the route obtained by solving a TSP supposing all the nodes must be visited by the drone and subsequently some nodes of this route are added to the route followed by the truck. We start from this latter idea and solve the involved TSP to obtain a base route by using a 2-opt heuristic procedure which gives good quality solutions in a short computation time. Inspired by this latter idea and using a 2-opt heuristic procedure we are able to obtain good quality initial TDTL solutions in a short computation time.

Despite finding a local optimum may (in the worst case) take a sub-exponential time of $2^{O(n)}$, in practice the 2-opt heuristic converges faster. If only uncrossing moves are allowed, the 2-opt algorithm will perform at most in $O(n^3)$ moves (Johnson and McGeoch, 1997). In our problem, a starting solution is denoted as a sequence π_{TSP}^{2-opt} (e.g., $\pi_{TSP}^{2-opt} = (0, 8, 4, 3, 6, 1, 2, 7, 5, 9)$ is the sequence obtained after the resolution of the TSP wherein all the nodes are initially visited by drones in the illustrative instance in Fig. 3). It is important to note that an optimal or very high-quality solution in this phase does not guarantee a good final solution of the problem under study and, therefore, the use of more complex and time-consuming heuristics is ruled out.

In general, the π_{TSP}^{2-opt} solution will not meet the battery capacity constraints for the drone, and therefore it will not be a feasible solution for the problem under study. It is therefore necessary to establish some synchronization truck-drone nodes to replace (or load) the battery. This new problem can be represented by a graph where each arc connects two possible synchronization nodes. This graph $G'(N, A')$ is formed by the set N containing all the nodes in the initial problem and a subset of edges, $A' \subseteq A$, which is constructed respecting battery capacity constraints according to the procedure described in Algorithm 2. Let t_{ij}^{T} be the travel time by truck from node i to $j \in N$, and let t_{ij}^{Dr} be the travel time by drone, according to the drone route given by the π_{TSP}^{2-opt}. Then the arc cost, t_{ij}, represents the more restrictive transportation time from node i to j, i.e., $t_{ij} = \max(t_{ij}^{T}, t_{ij}^{Dr})$. In this way, each feasible arc in A' is weighted with the most restrictive travel time by comparing the travel time by truck and the drone route established in the previous phase while fulfilling the battery capacity constraint.

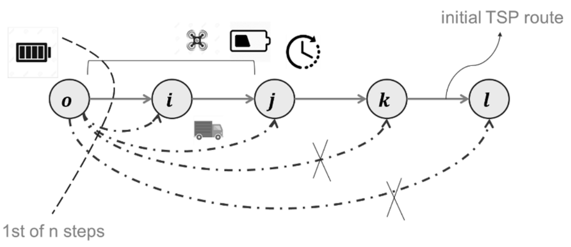


Fig. 5. 1st step in A′ set construction phase.

Algorithm 2 (*Set A′ construction phase*).

```
procedure Construction of set $A'$:
  for $i = 0$ to $n - 1$:
    for $j = i + 1$ to $n - 1$:
      if $tD^{r}_{\pi_S^0(i),\pi_S^0(j)} \leq Q$:
        Append $e = (\pi_S^0(i), \pi_S^0(j))$ to $A'$
        $t_{\pi_S^0(i),\pi_S^0(j)} = \max\left(t^{T}_{\pi_S^0(i),\pi_S^0(j)}, t^{Dr}_{\pi_S^0(i),\pi_S^0(j)}\right)$
        exit for
      endif
    endfor
endfor
```

Therefore, the problem of completing the construction of a feasible initial solution π_0 becomes in getting the shortest path ($mins_e$) between the origin and end nodes in G', which can be efficiently solved in $O(n^2)$ using Dijkstra's algorithm (Dijkstra, 1959). Fig. 5 illustrates the first of the n steps necessary to compute all possible paths between every node and its successors using the previous π_{TSP}^{2-opt} route and taking into account the battery capacity constraints.

For the illustration used so far (shown in Fig. 3), the shortest path, and therefore our initial solution π_0, is shown in Fig. 6, where every activated link appears labelled with the necessary travel time (either by drone in dotted lines or by truck in solid lines). Observe that when the drone is transported by the truck along links $(0, 8)$ and $(5, 9)$, the solid line is drawn and the travel time of these links is that of the truck.

Proceeding as explained, we have obtained a feasible initial solution to the problem π_0, with $s_e = 70.62$, which is coded as: $\pi_t = [0, 0, 1, 0, 0, 1, 1, 1, 0, 0]$ and $\pi_s = [0, 8, 6, 4, 3, 1, 2, 7, 5, 9]$.

Once an initial good feasible solution is obtained, the heuristic must evolve by searching for improved solutions. To this end, each iteration uses a destruction-construction mechanism on the last candidate solution in order to generate a new one. Then, it proceeds with a combined local search to improve the fitness of the new solution. Thus, a new candidate solution emerges from the described two-step process, which is compared with the best stored solution, as reported in Algorithm 1. In the next subsection, we explain destruction/construction and combined local search.

6.3. The destruction and construction phases

One of the most important steps in the IG algorithm consists of the destruction and construction mechanisms. Different schemes of solution's destruction and its subsequent reconstruction are applicable, as for example the one used by Ruiz and Stützle (2007). In that work a given number of elements in the sequence were randomly extracted obtaining a partial sequence π_R (destruction phase) – some authors also order it according to a given criterion. Later, the sequence was reconstructed adding each of the extracted elements one by one in the same order than π_R, but following a local search procedure for the insertion. This method has proven not to be competitive enough in the preliminary tests performed for our problem. It is important to highlight that while in flowshop scheduling problems the solutions are just encoded in one array, the TDTL problem requires two different arrays, what makes the direct adaptation of the previous referred method difficult. Besides, it is also important to note that the quality of the results and the computing time are both sensitive to the correct setting of the algorithm parameters, e.g., the number of elements to extract, or the sorting method. As an alternative to the previous method, another mechanism based on reverse swapping has been sought, which has proven to be robust and computationally efficient.

The destruction mechanism designed for the TDTL problem is as follows: Two random numbers (i, j) are selected within the sequence π_s so that $i \neq j$ and $(i, j) \neq \{o, e\}$. The partial sequence $\pi_R(i, j)$ is extracted from i to j. The vector of node types π_t remains the same.

As a construction mechanism, the partial subsequence previously obtained $\pi_R(i, j)$ is inserted in the same position as before but in reverse order. When changing the order of the sequence and not the type of node, it is possible to improve the solution by modifying π_t. In order to improve the construction phase, the process ends with a local search over the node type. The local search procedure which modify the array π_t is reported in the Algorithm 3:

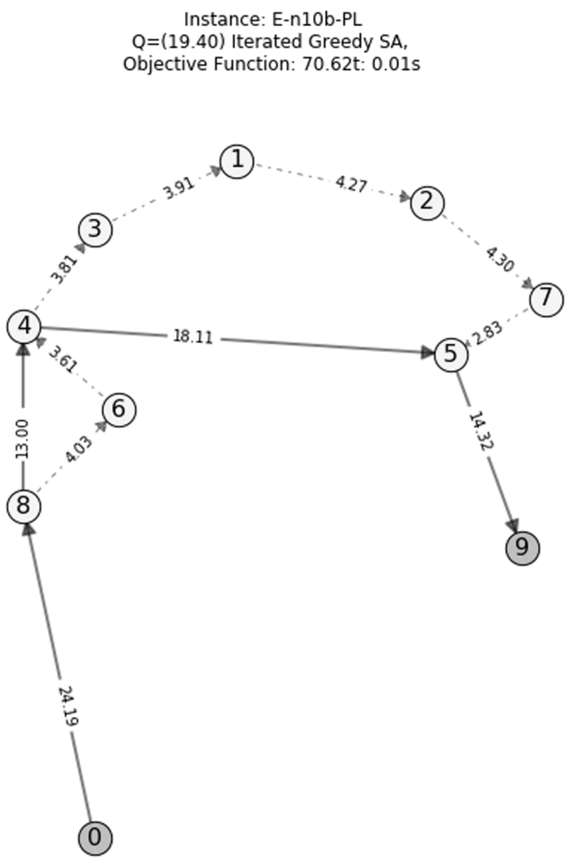


Fig. 6. The initial solution for the previous example.

Algorithm 3 (*Local search for ending the construction phase***).**

```
procedure Local_search_in_t:
  improve := True
  repeat (while improve = True):
     improve := False
     for $i = 1$ to $n - 2$:
       Select $i$ at random from $i = 1$ to $n - 2$ (without repetition)
       $\pi'$: =best solution obtained for every type of node {0, 1, 2} at position $i$ in $\pi_t$.
       if $f(\pi') < f(\pi)$:
          $\pi$: $=\pi'$
          improve := True
         if $f(\pi) < f(\pi_{best})$ and $\pi$ is feasible:
            $\pi_{best}$: $=\pi$
       endif
     endfor
  return $\pi$
```

It should be noted that this mechanism does not need the adjustment of parameters, which makes it more robust. Fig. 7 illustrates this procedure for a particular example.

6.4. The combined local search

To carry out the local search, the mechanism of local search insertion (see e.g., Ruiz and Stützle, 2007) is used. This technique has shown better performance than swapping or exchange techniques in flowshop scheduling problems. However, unlike the flowshop scheduling, the problem addressed in this paper also considers, apart from the sequence array π_s, the type of node in a second array π_t, so that for a certain insertion at position i, it is necessary to select the best value of $\pi_t(i)$ among the possible values in $\{0, 1, 2\}$. The procedure is complemented running again, for the more promising solutions, the search reported in Algorithm 2 over π_t. This final enhancement has improved the quality

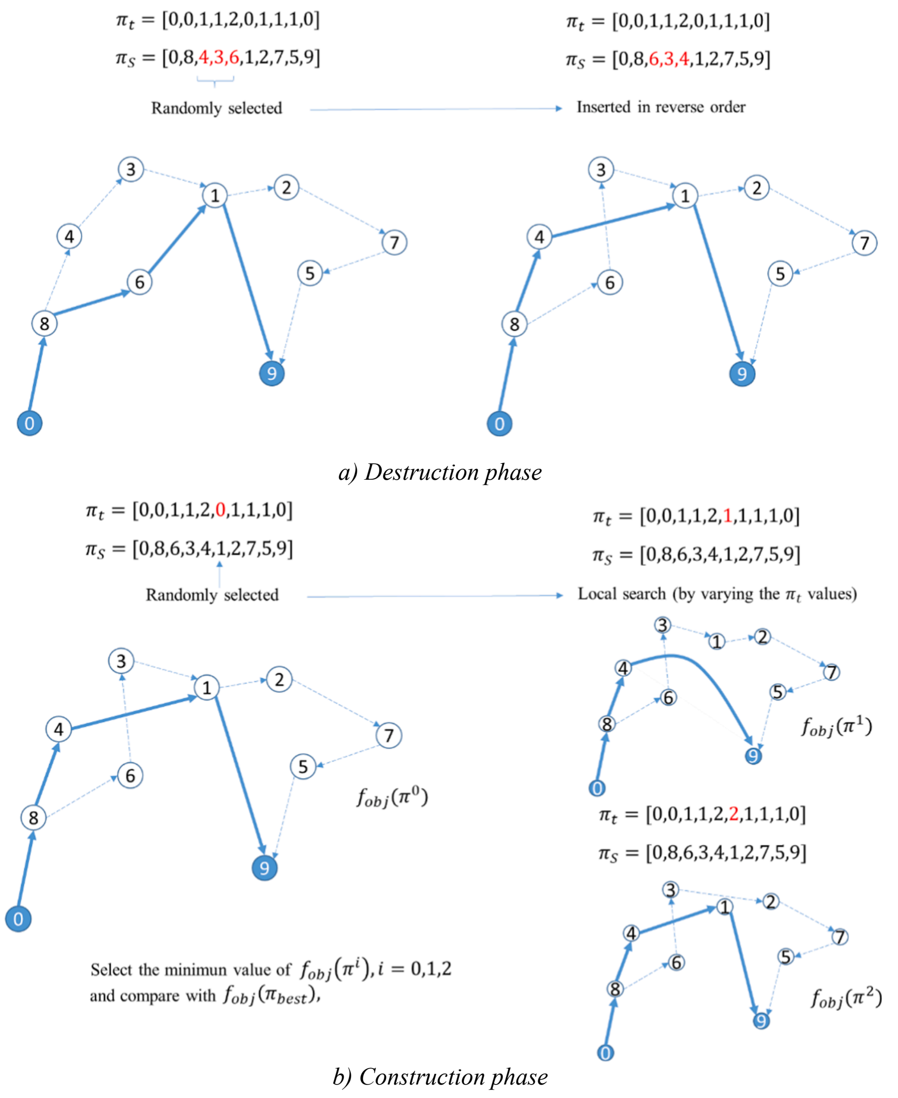

$\pi_t = [0,0,1,1,2,0,1,1,1,0]$
$\pi_S = [0,8,4,3,6,1,2,7,5,9]$
Randomly selected
Inserted in reverse order
$\pi_t = [0,0,1,1,2,0,1,1,1,0]$
$\pi_S = [0,8,6,3,4,1,2,7,5,9]$
a) Destruction phase
$\pi_t = [0,0,1,1,2,0,1,1,1,0]$
$\pi_S = [0,8,6,3,4,1,2,7,5,9]$
Randomly selected
Local search (by varying the π_t values)
$\pi_t = [0,0,1,1,2,1,1,1,1,0]$
$\pi_S = [0,8,6,3,4,1,2,7,5,9]$
$f_{obj}(\pi^0)$
$f_{obj}(\pi^1)$
$\pi_t = [0,0,1,1,2,2,1,1,1,0]$
$\pi_S = [0,8,6,3,4,1,2,7,5,9]$
$f_{obj}(\pi^2)$
Select the minimun value of $f_{obj}(\pi^i), i = 0,1,2$ and compare with $f_{obj}(\pi_{best})$,
b) Construction phase

Fig. 7. The destruction-construction procedure.

of the best solution so far. The combined local search procedure is shown below in Algorithm 4:

Algorithm 4 (*Local search for improving the solution***).**

```
procedure Combined_local_search:
  improve := True
  repeat (while improve = True):
    improve := False
    for $i = 1$ to $n - 2$:
      Remove node $k$ at random from $\pi_s$, $k \neq \{o, e\}$, (without repetition)
      $\pi'$: = is the solution obtained by the best permutation in $\pi_s$ by inserting $k$ in all possible positions of $\pi_s$ and for every type of node $\{0, 1, 2\}$.
      if $f(\pi') < f(\pi)$:
        $\pi: = \pi'$
        improve := True
      if $f(\pi) < f(\pi_{best})$ and $\pi$ is feasible:
        $\pi_{best}: = \pi$
        $\pi: =$Local_search_in_t ($\pi_{best}$)
      endif
    endfor
return $\pi$
```

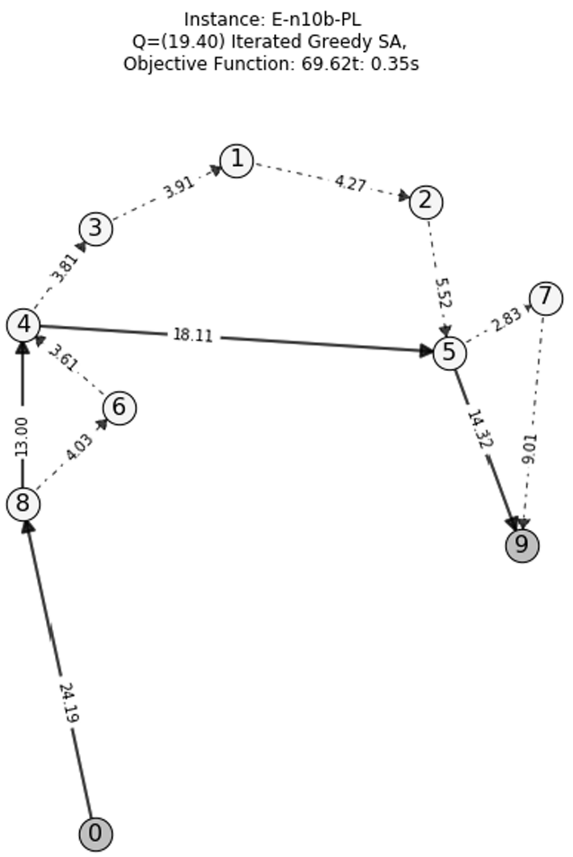


Fig. 8. Illustration of the IG final solution in a 10-nodes network.

The solution obtained by the adapted IG algorithm for the example followed until now is shown in Fig. 8.

7. Computational study

In order to test the performance of proposed heuristic, we have conducted an experimental study on a literature-based set of instances with different characteristics. The experiments have been solved on an Intel Core i7 2.8 GHz with 16 GB of RAM.

As explained in Section 5.5, it is expected the Branch-and-Cut not being capable of solving medium and large size problems. This circumstance has arisen when performing a set of preliminary experiments on several instances of size greater than 20 nodes. Specifically, we have modelled and solved the TDTL problem by using the Python interface and the standard Branch-and-Cut solver algorithm from Gurobi 7.0.0 along a 24 h computation time. Table 3 shows the obtained results.

Secondly, we have addressed the same set of instances from 20 to 250 nodes but this time providing the MIP solver with an initial solution arising from considering that all the nodes are only served using the truck and stopping its execution at 1800 s. The results are reported in Table 4, along with the performance provided by our IG procedure when its time limit is set at 60 s and the maximum number of iterations without improvement is set at 50.

According to the conducted preliminary experiments, we can conclude that the Branch-and-Cut procedure was not expected to be able to solve medium and large problems.

Henceforth, we present a comprehensive test bed for testing the performance of our heuristic proposal. Coded in standard Python,

Table 3
Results of the Gurobi Solver after 24 h of computation time.

Instance size (nodes)	Optimality Gap (after 24 h of computation time)
20	87.1%
50	–
75	–
100	–
175	–
250	–

Table 4
Results of the Gurobi Solver after feeding the problem with an initial solution.

Instance size (nodes)	Objective function value at the initial solution (only trucks)	Best solver solutions after 1800 s	Number of improved solutions	Objective value of the heuristic after 60 s	% of difference between the initial solution and the 60 s heuristic solution
20	329.617	**182.277**	**2**	174.012	47.207
50	564.955	564.955	0	317.814	43.745
75	564.955	564.955	0	363.221	35.707
100	794.478	794.478	0	395.899	50.168
175	1002.578	1002.578	0	506.982	49.432
250	1161.264	1161.264	0	604.862	47.913

it addresses these new set of experiments adjusted to a 600 s time limit (IG_TL) and to a maximum number of iterations without improvement equal to 50. For the sake of comparison, we also report the Branch-and-Cut performance on the new set of experiments with time limit truncated to a value MIP_TL of 1800 s.

7.1. Instances

The set of instances used in our study is based in that previously reported by Agatz et al. (2018) and accessible in Bouman et al. (2018). The authors proposed a different set of randomly generated instances where every node is in a plane. The travel time between nodes is considered proportional and constant between every pair of nodes. Three different sets are considered: uniform, 1-center, and 2-center. In the uniform set, the nodes are located uniformly according an integer uniform distribution (1,100). The 1-center and 2-center set of instances try to mimic the density of customers in a city with one or two circular city centres respectively. Details about the construction of the center scenarios can be found in Agatz et al. (2018). In our study, for all the instances, we have considered a constant battery life (Q) proportional to the average drone travel time between each pair of nodes. Therefore $Q = \frac{2}{n}\sum_{\forall(i,j)\in A} tD_{ij}$.

Our research is based on the premise of making customer visits between a source and a destination node, the former been coincident with the depot in the set of instances taken from literature. Besides, the last node in every instance specification has been taken as the destination node.

For each one of the sets of instances, we apply the heuristic procedure by varying the penalization factor $p = v_D/v_T$ in the objective function for equal speed between truck and drone, twofold and threefold. The bed of instances contains 10 problems for each value of the number of clients n. We have divided this set in two different groups i) problems of small size, for $n = \{5, 6, 7, 8, 9, 10\}$ and ii) problems of medium/large size with $n = \{20, 50, 75, 100, 175, 250\}$. In total, 1080 MIP instances have been solved (or partially solved), computing 120 cases (10 for each one of the 12 problem sizes) by varying $\alpha = \{1, 2, 3\}$ and considering 3 scenarios (uniform, 1-center, and 2-center). Concerning to the heuristic, since by its nature it is a non-deterministic procedure, 10 replications have been made for each one of the 1080 instances, solving a total amount of 10,800 cases. Figs. 9, 10 and 11 depict three solutions for uniform instances of 75, 100 and 250 nodes respectively.

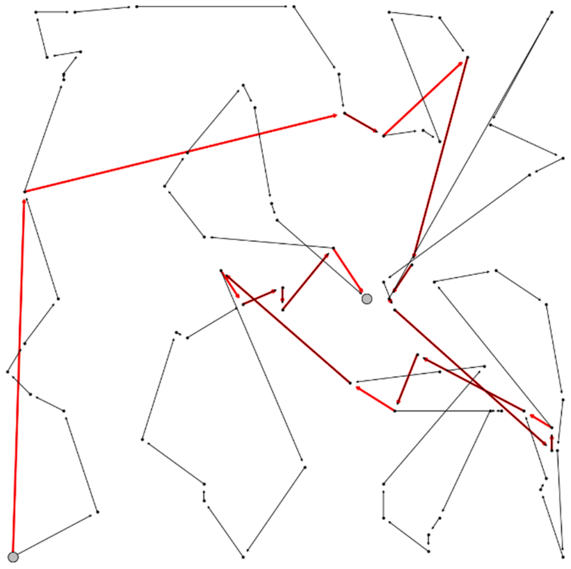


Fig. 9. Uniform 75 nodes. Q = 54.63, Obj. function = 402.21. Comp. time = 163.28 s.

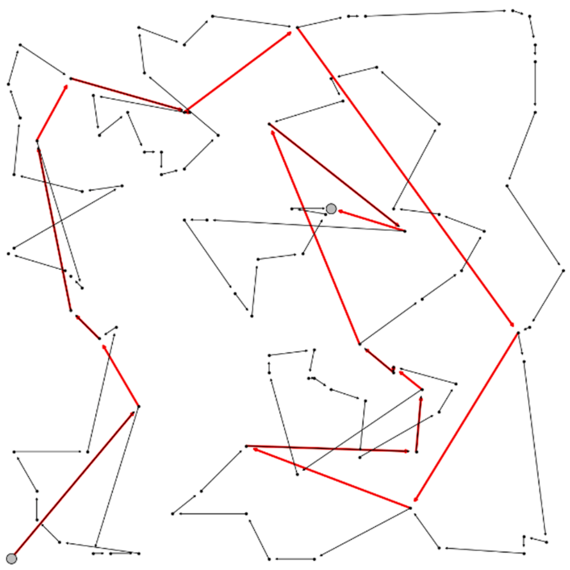


Fig. 10. Uniform 100 nodes. Q = 53.24, Obj. function = 431.10. Comp. time = 384.81 s.

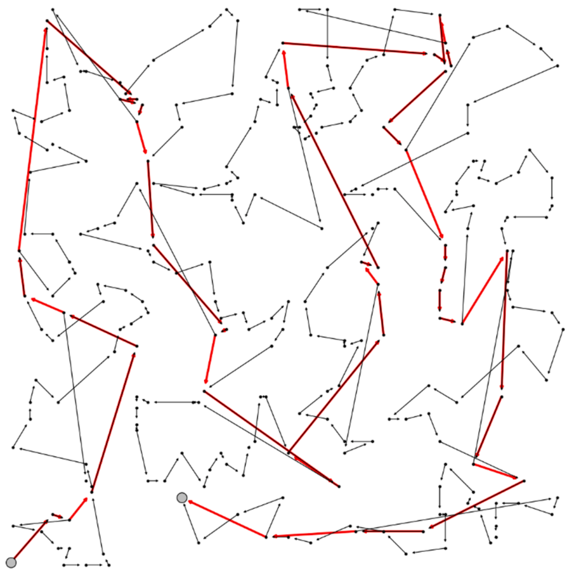


Fig. 11. Uniform 250 nodes. Q = 51.56, Obj. function = 584.12, Comp- time = 605.19 s.

7.2. Performance measures

As in many other challenging problems addressed in the scientific literature, the basic quality measure of the results is the relative percentage deviation (*RPD*) over the best-known solution. For a given solution π and a best-known solution π_b with objective values $s_e(\pi)$ and $s_e(\pi_b)$ respectively, the *RPD* is obtained as follows:

$$RPD = \frac{s_e(\pi) - s_e(\pi_b)}{s_e(\pi_b)} \times 100 \tag{31}$$

Other quality measures are the optimality of the solution (if known, which can be achieved only for small problems), and the number of times that the optimal (or best solution) is reached by in the replications of the heuristic. Furthermore, the average computing time will be also shown for all the cases. We also report the performance of both approaches (exact and heuristic) over the time of a medium size instance (n = 20).

Finally, a nonparametric analysis of Kruskal-Wallis is also used to provide more insights about the influence of the α parameter in the performance of the heuristic approach and the three different environments studied, i.e. uniform, 1-center and 2-center.

Table 5
Uniform scenario, $\alpha = \{1, 2, 3\}$.

Uniform ($\alpha = 1$)

	MIP				IG			
Instance Size(n)	Avg MIPGAP(%)	Avg Time	# Optimal Sols	Avg Feas. Sols	Avg RPD(%)	Avg Time	# Optimal and Best Sols	Avg Feas. Sols
5	0.0	0.1	**10**/10	4.2	3.1	0.1	**80**/100	1.6
6	0.0	0.3	**10**/10	6.5	1.4	0.2	**90**/100	3.7
7	0.0	1.3	**10**/10	8.1	0.1	0.3	**97**/100	4.9
8	0.0	6.3	**10**/10	9.2	1.4	0.5	**64**/100	5.8
9	0.0	130.2	**10**/10	10	2.1	0.7	**49**/100	6.3
10	59.1	1018.4	**8**/10	10	2.4	0.9	**35** + 7*/100	7.8
20	892.8	MIP_TL	0/10	10	2.6	3.8	74*/100	14.1
50	–	MIP_TL	0/10	0	5.4	47.6	14*/100	28.5
75	–	MIP_TL	0/10	0	4.4	163.2	12*/100	43.0
100	–	MIP_TL	0/10	0	4.5	387.2	13*/100	51.0
175	–	MIP_TL	0/10	0	3.3	IG_TL	18*/100	87.7
250	–	MIP_TL	0/10	0	2.9	IG_TL	17*/100	115.1

Uniform ($\alpha = 2$)

	MIP				IG			
Instance Size(n)	Avg MIPGAP(%)	Avg Time	# Optimal Sols	Avg Feas. Sols	Avg RPD(%)	Avg Time	# Optimal and Best Sols	Avg Feas. Sols
5	0.0	0.1	**10**/10	5	0.0	0.1	**100**/100	0.8
6	0.0	0.3	**10**/10	6.8	0.2	0.2	**95**/100	1.3
7	0.0	1.0	**10**/10	8.4	0.3	0.3	**86**/100	1.3
8	0.0	7.7	**10**/10	10	0.3	0.3	**89**/100	0.9
9	0.0	139.8	**10**/10	10	2.8	0.5	**47**/100	1.8
10	63.9	1170.7	**8**/10	10	0.8	0.7	**66** + 16*/100	1.7
20	682.7(9)	MIP_TL	0/10	9	0.2	3.4	95*/100	3.7
50	–	MIP_TL	0/10	0	1.5	48.2	47*/100	10.6
75	–	MIP_TL	0/10	0	1.9	163.2	27*/100	17.4
100	–	MIP_TL	0/10	0	1.9	387.0	17*/100	21.1
175	–	MIP_TL	0/10	0	2.1	IG_TL	20*/100	40.2
250	–	MIP_TL	0/10	0	2.1	IG_TL	15*/100	53.6

Uniform ($\alpha = 3$)

	MIP				IG			
Instance Size(n)	Avg MIPGAP(%)	Avg Time	# Optimal Sols	Avg Feas. Sols	Avg RPD(%)	Avg Time	# Optimal and Best Sols	Avg Feas. Sols
5	0.0	0.1	**10**/10	5.1	0.0	0.1	**100**/100	0.9
6	0.0	0.3	**10**/10	6.2	0.4	0.2	**94**/100	1.2
7	0.0	1.2	**10**/10	8.5	0.6	0.3	**95**/100	1.2
8	0.0	10.7	**10**/10	10	0.1	0.4	**93**/100	1.2
9	0.0	221.3	**10**/10	10	4.1	0.6	**56**/100	1.8
10	141.4	1255.6	**6**/10	10	1.3	0.6	**41** + 31*/100	1.3
20	–	MIP_TL	0/10	0	4.4	3.6	54*/100	3.3
50	–	MIP_TL	0/10	0	0.8	49.1	65*/100	7.2
75	–	MIP_TL	0/10	0	1.0	166.1	43*/100	9.1
100	–	MIP_TL	0/10	0	1.0	394.2	45*/100	11.8
175	–	MIP_TL	0/10	0	0.8	IG_TL	39*/100	23.2
250	–	MIP_TL	0/10	0	0.9	IG_TL	29*/100	30.3

7.3. Results

Tables 5–7 summarize the results for each one of the three studied scenarios, each table reporting what arises from the different values of α, ranged from 1 to 3. On the left-hand side every table presents the results obtained by the MIP while the results obtained with the heuristic are shown on the right-hand side. Importantly, in both cases, the values correspond to the 10 problems of each size included in the original instances bed (average or total amount).

The report of the solver performance in addressing the MIP with the commercial solver is provided presenting the average MIPGAP (%) – i.e. the RPD between the upper and lower bounds obtained by the solver-, the average computation time, the number of optimal solutions found out of the 10 problems (in bold face), along with the average number of feasible solutions found by the

Table 6
'1-Center' scenario, $\alpha = \{1, 2, 3\}$

1-center ($\alpha = 1$)

	MIP				IG			
Instance Size(n)	Avg MIPGAP(%)	Avg Time	# Optimal Sols	Avg Feas. Sols	Avg RPD(%)	Avg Time	# Optimal and Best Sols	Avg Feas. Sols
5	0.0	0.1	**10**/10	4.1	1.1	0.1	**78**/100	2.0
6	0.0	0.3	**10**/10	5.7	0.0	0.2	**99**/100	3.1
7	0.0	1.6	**10**/10	8.5	0.6	0.3	**88**/100	4.9
8	0.0	7.4	**10**/10	9.6	0.8	0.5	**70**/100	6.4
9	0.0	116.2	**10**/10	10	1.6	0.6	**43**/100	7.4
10	102.6	939.2	**9**/10	10	1.8	0.8	**30**/100	8.1
20	1885.2	MIP_TL	0/10	10	0.7	3.5	91*/100	12.8
50	–	MIP_TL	0/10	0	5.8	46.4	11*/100	33.1
75	–	MIP_TL	0/10	0	3.6	157.5	16*/100	44.7
100	–	MIP_TL	0/10	0	3.6	380.3	15*/100	57.9
175	–	MIP_TL	0/10	0	3.0	IG_TL	15*/100	100.5
250	–	MIP_TL	0/10	0	1.6	IG_TL	12*/100	64.9

1-center ($\alpha = 2$)

	MIP				IG			
Instance Size(n)	Avg MIPGAP(%)	Avg Time	# Optimal Sols	Avg Feas. Sols	Avg RPD(%)	Avg Time	# Optimal and Best Sols	Avg Feas. Sols
5	0.0	0.1	**10**/10	4.1	0.0	0.1	**100**/100	1.4
6	0.0	0.3	**10**/10	7.3	0.5	0.2	**95**/100	1.1
7	0.0	1.5	**10**/10	8.3	1.1	0.3	**86**/100	2.6
8	0.0	6.8	**10**/10	9.7	0.5	0.4	**75**/100	2.6
9	0.0	145.1	**10**/10	10	2.1	0.5	**46**/100	1.6
10	153.9	1239.9	**4**/10	10	1.4	0.7	**24** + 38*/100	1.2
20	1979.5(3)	MIP_TL	0/10	3	0.0	3.6	100*/100	3.8
50	–	MIP_TL	0/10	0	0.6	50.4	61*/100	7.2
75	–	MIP_TL	0/10	0	1.0	170.4	28*/100	16.1
100	–	MIP_TL	0/10	0	0.9	406.6	35*/100	19.4
175	–	MIP_TL	0/10	0	1.0	IG_TL	24*/100	43.4
250	–	MIP_TL	0/10	0	0.9	IG_TL	31*/100	54.9

1-center ($\alpha = 3$)

	MIP				IG			
Instance Size(n)	Avg MIPGAP(%)	Avg Time	# Optimal Sols	Avg Feas. Sols	Avg RPD(%)	Avg Time	# Optimal and Best Sols	Avg Feas. Sols
5	0.0	0.1	**10**/10	4.8	0.5	0.1	**98**/100	1.2
6	0.0	0.3	**10**/10	6.8	0.4	0.2	**98**/100	1.0
7	0.0	1.5	**10**/10	7.5	0.4	0.3	**93**/100	2.3
8	0.0	8.2	**10**/10	9.9	0.6	0.4	**68**/100	2.0
9	0.0	183.9	**10**/10	10	1.8	0.6	**46**/100	1.9
10	345.2	1322.2	**3**/10	10	1.9	0.7	**5** + 38*/100	1.6
20	–	MIP_TL	0/10	0	0.0	3.5	100*/100	1.8
50	–	MIP_TL	0/10	0	0.7	50.7	63*/100	4.6
75	–	MIP_TL	0/10	0	0.5	171.1	74*/100	10.3
100	–	MIP_TL	0/10	0	0.5	412.1	64*/100	12.4
175	–	MIP_TL	0/10	0	0.7	IG_TL	54*/100	23.6
250	–	MIP_TL	0/10	0	0.7	IG_TL	48*/100	31.3

solver during the optimization process. Similarly, when reporting the heuristic performance we present the average RPD (%) with respect to the best solution found for the 10 replications conducted for each one of the 10 problems, the average computation time, the number of instances in which the heuristic has reached the optimum along with the average number of solutions found by the heuristic, apart from the initial one, which is the same for all the instances (in bold face). Furthermore, those solutions that have reached a better performance than the best solution obtained by the MIP approach have been marked with (*).

The insights from the experimentation conducted can be discussed according to the size of the instances. For those instances of size (n) less than or equal to 9, the resolution through the MIP approach has been successful – i.e. the exact resolution is reached in all cases-, using an average computing time less than the time limit (MIP_TL) and reaching. For instances of size n=10 the MIPGAP(%) differs from zero, since there are some instances in which the optimum has not been reached in the specified time limit. Although the

Table 7
'2-Center' scenario, $\alpha = \{1, 2, 3\}$

2-center ($\alpha = 1$)

	MIP				IG			
Instance Size(n)	Avg MIPGAP(%)	Avg Time	# Optimal Sols	Avg Feas. Sols	Avg RPD(%)	Avg Time	# Optimal and Best Sols	Avg Feas. Sols
5	0.0	0.1	**10**/10	3.9	0.9	0.1	**90**/100	2.6
6	0.0	0.3	**10**/10	6.3	0.0	0.1	**100**/100	3.8
7	0.0	1.2	**10**/10	7.4	0.0	0.3	**98**/100	4.3
8	0.0	7.1	**10**/10	9.7	0.5	0.5	**81**/100	6.5
9	0.0	87.9	**10**/10	9.5	1.1	0.6	**65**/100	7.6
10	27.7	1313.1	**6**/10	10	1.8	0.7	**27** + 23*/100	7.5
20	1778.2	MIP_TL	0/10	10	1.0	3.8	77*/100	16.5
50	–	MIP_TL	0/10	0	6.7	46.3	11*/100	30.1
75	–	MIP_TL	0/10	0	4.9	163.0	13*/100	48.6
100	–	MIP_TL	0/10	0	5.6	385.9	15*/100	65.6
175	–	MIP_TL	0/10	0	3.9	IG_TL	14*/100	102.7
250	–	MIP_TL	0/10	0	3.7	IG_TL	19*/100	134.7

2-center ($\alpha = 2$)

	MIP				IG			
Instance Size(n)	Avg MIPGAP(%)	Avg Time	# Optimal Sols	Avg Feas. Sols	Avg RPD(%)	Avg Time	# Optimal and Best Sols	Avg Feas. Sols
5	0.0	0.1	**10**/10	4.8	0.1	0.1	**97**/100	1.1
6	0.0	0.2	**10**/10	7.4	0.1	0.2	**97**/100	0.9
7	0.0	1.2	**10**/10	9.5	0.3	0.2	**92**/100	1.1
8	0.0	7.3	**10**/10	9.4	0.9	0.4	**70**/100	3.1
9	0.0	111.7	**10**/10	9.7	0.3	0.4	**81**/100	1.4
10	62.7	1465.9	**6**/10	10	1.5	0.6	**29** + 28*/100	3.0
20	1371.6	MIP_TL	0/10	10	0.0	3.4	100*/100	5.1
50	–	MIP_TL	0/10	0	1.7	49.6	22*/100	15.8
75	–	MIP_TL	0/10	0	1.7	172.7	22*/100	20.9
100	–	MIP_TL	0/10	0	2.1	412.4	16*/100	29.5
175	–	MIP_TL	0/10	0	1.9	IG_TL	19*/100	49.2
250	–	MIP_TL	0/10	0	1.6	IG_TL	20*/100	66.5

2-center ($\alpha = 3$)

	MIP				IG			
Instance Size(n)	Avg MIPGAP(%)	Avg Time	# Optimal Sols	Avg Feas. Sols	Avg RPD(%)	Avg Time	# Optimal and Best Sols	Avg Feas. Sols
5	0.0	0.1	**10**/10	5.2	0.0	0.1	**100**/100	0.9
6	0.0	0.2	**10**/10	6.6	0.6	0.2	**93**/100	1.1
7	0.0	1.2	**10**/10	8.4	0.1	0.2	**95**/100	0.8
8	0.0	7.0	**10**/10	10	0.3	0.4	**81**/100	2.1
9	0.0	119.1	**10**/10	10	0.3	0.4	**83**/100	1.2
10	99.8	1078.9	**7**/10	10	1.8	0.6	**38** + 15*/100	2.1
20	1176.0(4)	MIP_TL	0/10	4	0.0	3.5	100*/100	3.7
50	–	MIP_TL	0/10	0	1.9	48.8	41*/100	9.8
75	–	MIP_TL	0/10	0	2.1	168.3	21*/100	14.7
100	–	MIP_TL	0/10	0	1.5	401.3	31*/100	18.8
175	–	MIP_TL	0/10	0	1.3	IG_TL	28*/100	33.5
250	–	MIP_TL	0/10	0	0.8	IG_TL	27*/100	43.4

number of instances in which the optimum has been reached (#Optimal Sols) is, in general, reasonably high ($>$ 7 out of 10), the computation time increases markedly. In fact, for instances of larger size ($n = 20$) the Branch-and-Cut algorithm exhausts the time limit specified (MIP_TL) and feasible solutions are only reached in the instances with $\alpha = 1$, while for $\alpha = 2$ feasible solutions are only reached in several instances (the number is shown in parentheses). No feasible solutions are found for $\alpha = 3$. It is important to note that, as previously explained in Section 5.5, the complexity of the problem increases noticeably with n, as illustrated in the column of feasible solutions found -"Avg. Feas. Sols "-, in which the MIP exact solving procedure fails to find solutions in problems with $n > 20$ for all the proposed scenarios. With respect to the optimality, it is important to highlight that no optimal solution have been obtained for instances of size greater than or equal to 20 clients.

As for the heuristic performance observed, when addressing small instances ($n \leq 10$) it obtains a fairly low average RPD(%) in a

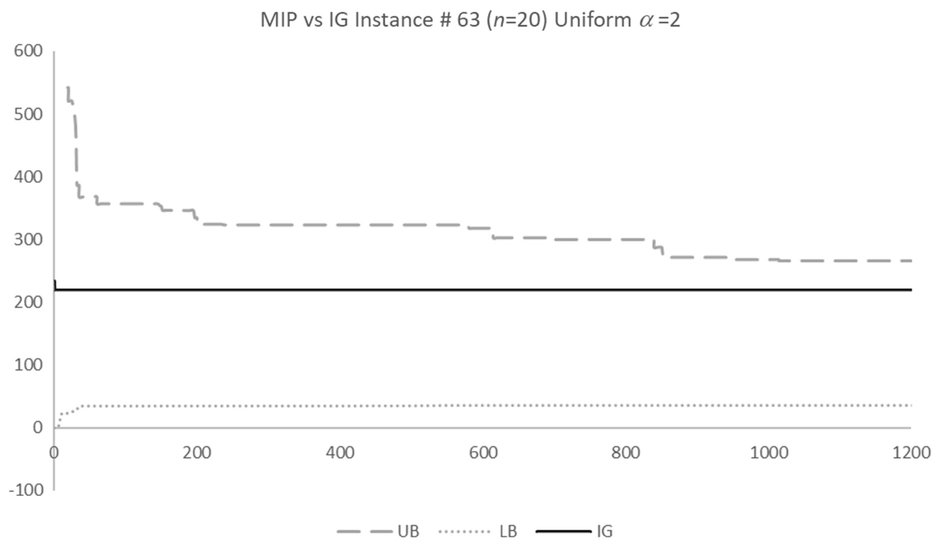


Fig. 12. Comparison of the MIP exact algorithm and the heuristic for the uniform set. instance #63, n=20 ($\alpha = 2$).

noticeably less computing time than the exact resolution procedure. It should be noted that, with regard to the quality of the solutions, the number of optimal solutions found is also high, which is indicative of the robustness of the heuristic. For instances with $n = 10$, even some solutions reached by the MIP approach have been improved what it is indicated by marking the number of improved solutions by an (*) in the column "#Optimal and best sols". For sizes greater than or equal to 20, the best solutions have always been obtained by the heuristic in all the scenarios under study. Furthermore, the heuristic is able to find better solutions than the initial one, as shown in the column "Avg. Feas. Sols", illustrating the capacity of the solution generation mechanism (through the destruction-construction process and the combined local search) for improving the objective function value in a process globally guided by a SA structure. Finally, it should be noted that for small problems, in general, the number of feasible solutions is small, which, considering the high optimality rate achieved, it is an indicator of the high quality of the initial solutions designed for our heuristic.

Fig. 12 shows both the upper and lower bounds (UB and LB respectively) of the MIP resolution over time for a medium-sized type instance ($n = 20, \ \alpha = 2$), as well as the evolution of the results obtained by the heuristic.

As shown, the Branch-and-Cut upper and lower bounds converge, but around 1000 s the convergence stagnates and the difference between upper and lower bounds remains the same until reaching the MIP time limit (MIP_TL = 1800 s). On the other hand, it can be observed that the initial solution used by the heuristic reaches a value reasonably below than the value of the upper bounds reached by the MIP exact resolution algorithm over the solving time. The heuristic quickly finds solutions in its initial search, which, for this type of instances, turns into an average computing time of 3.4 s (see Table 1). In the picture, the value of the best solution, found after 3.4 s, has been graphically depicted as a horizontal line in order to better represent the difference with respect to the upper and lower bound obtained by the MIP.

To facilitate the interpretation of the influence of the α-parameter {1, 2, 3}, and the type of scenario (uniform, 1-center and 2-center) two nonparametric analysis of Kruskal-Wallis has been carried out for the RPD(%) values and the small-size instances ($n \leq 10$). For each αwe have 18 values, forming a sample size of 54 values. The average ranges are shown in the Table 8:

The Kruskal-Wallis test statistic (H), is calculated as 2486. At 0.05 level of significance, the critical value obtained from the chi-square table with 2 degrees get a p-value of 0.288 (>*level_of_significance*), and therefore we cannot reject the null Hypothesis of equal Avg. RPD(%) medians among groups.

The same conclusions can be obtained when performing the Kruskal-Wallis test with respect to the analysis of the different types of scenarios under study. See Table 9.

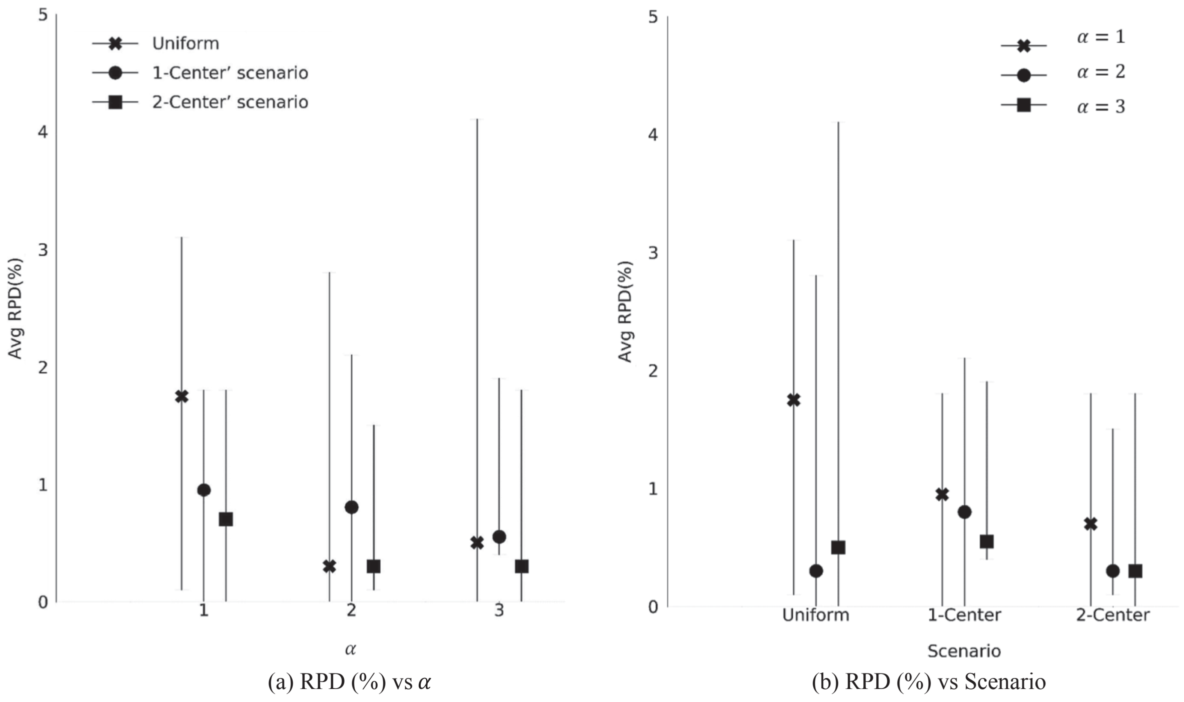
In this case, the Kruskal-Wallis test statistic (H), is calculated as 3313. At 0.05 level of significance, the critical value obtained from the chi-square table with 2 degrees get a p-value of 0.191 (>*level_of_significance*), and therefore we cannot reject the null

Table 8
Average Range values, $\alpha = \{1, 2, 3\}$.

α	N	Avg Range
1	18	32.19
2	18	24.47
3	18	25.83
Total	54	

Table 9
Average Range, Uniform, 1-center, and 2-center scenarios.

Scenario	*N*	Avg Range
Uniform	18	29.58
1-center	18	30.86
2-center	18	22.06
Total	54	



(a) RPD (%) vs α (b) RPD (%) vs Scenario

Fig. 13. Average RPD(%) values.

Hypothesis of equal Avg. RPD(%) medians among groups.

As conclusion, in both cases, at 0.05 level of significance, there is insufficient evidence to conclude that at least one of the groups differs with respect to the median Avg. RPD(%).

The Fig. 13 shows the results of the medians (marked with different symbols) for the average RPD (%) values grouped by α, Fig. 13(a), or by type of scenario, Fig. 13(b). In addition to the median, the ranges of variation of the data (minimum and maximum values) are also depicted in the graphs.

The previous analysis shows that the average RPD(%) obtained by the heuristic in the small instances, of which the deviation from the optimum is known in most cases, does not vary significantly according to the Kruskal-Wallis analysis. In addition, values are, in all cases, inferior to 4.1%, and the median values are less than 2%, and very close to 0. In conclusion, we can say that the analysis reinforces the characteristics of robustness and quality of the proposed heuristic.

8. Conclusions

In this paper, we propose a mathematical formulation for the TDTL problem, determining the best routes in the use of a single truck equipped with a single drone, with both cooperating to serve a set of locations. The problem lies in determining the best routes for both, truck and drone, between two different nodes (origin and destination), without predefining the waypoint or rendezvous locations for either the truck or the drone.

Our formulation is indeed an extension of the FSTSP model, allowing the drone to visit several customers per trip between two consecutive rendezvous with the truck. We go beyond the general assumption in hybrid truck-drone systems concerning that the locations are served one at a time. Although the above general assumption can be acceptable in the last-mile delivery case, it makes no sense in other scenarios such as the delivery of lightweight relief items in first aid scenarios or in ISR missions. Aside from the multi-drop assumption other features of our novel model is the absence of a pre-established route (open) for both truck and drone and the consideration of each location as a potential synchronization point between them. We suppose infinite travel length capacity for the truck and a limited battery life for the drone, but rechargeable at certain (a priori undetermined) rendezvous points.

Since the MIP model stated becomes intractable for medium or large-size instances, we have also provided an approximate IG heuristic method to solve it in an efficient way. To this aim, we have defined an original coding scheme of solutions based on a tuple formed by two vectors, a sequence vector and a vector containing the type of resource (truck and/or drone) used to visit every node. We have taken advantage of this coding scheme, firstly in developing a fast-constructive two-step procedure for generating high-quality initial solutions; and secondly, in improving the solution through a destruction-construction mechanism followed by two original local search procedures. This process is conducted by a global optimization scheme using a SA algorithm.

We have checked the developed heuristic, addressing a set of 1080 problems from literature-based scenarios. In this benchmark set of instances, we have compared the heuristic's performance with the exact resolution using the standard Branch-and-Cut algorithm of the Gurobi solver. As the extensive computational study reveals that the heuristic shows a good performance in both its ability to solve all the instances in a short computation time, and the provision of high-quality solutions. Since our proposal is able to produce good solutions for large-sized problems, we can state it is ready for application in solving current real-life truck-drone routing problems.

CRediT authorship contribution statement

Pedro L. Gonzalez-R: Conceptualization, Methodology, Software. **David Canca:** Data curation, Writing - original draft. **Jose L. Andrade-Pineda:** Investigation, Writing - review & editing. **Marcos Calle:** Formal analysis, Visualization, Investigation. **Jose M. Leon-Blanco:** Software, Validation.

References

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

Othman, M.S., Shurbevski, A., Karuno, Y., Nagamochi, H., 2017. Routing of carrier-vehicle systems with dedicated last-stretch delivery vehicle and fixed carrier route. J. Inform. Process. 25, 655–666.

Bouman, P., Agatz, N., Schmidt, M., 2017. Dynamic programming approaches for the traveling salesman problem with drone. ERIM Report Series Research in Management (ERS-2017-011–LIS).

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

Boysen, N., Schwerdfeger, S., Weidinger, F., 2018. Scheduling last-mile deliveries with truck-based autonomous robots. Eur. J. Oper. Res. 271 (3), 1085–1099.

Campbell, J.F., Sweeney, D., Zang, J., 2017. Strategic design for delivery with trucks and drones. Technical Report. SCMA-2017-0201. College of Business Administration. University of Missouri. St. Louis. MO USA.

Chang, Y.S., Lee, H.J., 2018. Optimal delivery routing with wider drone-delivery areas along a shorter truck-route. Expert Syst. Appl. 104, 307–317.

Cheng, C., Adulyasak, Y., Rousseau, L.M., 2018. Formulations and Exact Algorithms for Drone Routing Problem. Technical report. 2018-2018-31. CIRRELT. Canada.

Chowdhury, S., Emelogu, A., Marufuzzaman, M., Nurre, S.G., 2017. Drones for disaster response and relief operations: a continuous approximation model. Int. J. Prod. Econ. 188, 167–184.

Dayarian, I., Savelsbergh, M., Clarke, J.P., 2017. Same-day delivery with drone resupply, Optimization Online, http://www.optimization-online.org/DB_FILE/2017/09/6206.pdf.

Dijkstra, E.W., 1959. A note on two problems in connection with graphs. Numer. Math. 1, 83–89.

Dorling, K., Heinrichs, J., Messier, G.G., Magierowski, S., 2017. Vehicle routing problems for drone delivery. IEEE Trans. Syst. Man Cybernet. 47 (1), 70–85.

Ferrandez, S.M., Harbison, T., Wieber, T., Stnuges, R., Rich, R., 2016. Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm. J. Ind. Eng. Manage. 9 (2), 374–388.

Garone, E., Naldi, R., Casavola, A., 2011. Traveling salesman problem for a class of carrier-vehicle systems. J. Guid. Control Dyn. 34 (4), 1272–1276.

Greenwood, F., 2015. Above and Beyond: Humanitarian Uses of Drones, World Politics Review, Available from: http://www.worldpoliticsreview.com/articles/16750/above-and-beyond-humanitarian-uses-of-drones [accessed on December 15. 2018].

Ha, Q.M., Deville, Y., Pham, Q.D., Hà, M.H., 2018. On the min-cost traveling salesman problem with drone. Transp. Res. Part C: Emerg. Technol. 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. Transp. Res. Part C: Emerg. Technol. 91, 1–14.

Jeong, H.Y., Song, B.D., Lee, S., 2019. Truck-drone hybrid delivery routing: payload-energy dependency and no-fly zones. Int. J. Prod. Econ. 214, 220–233. https://doi.org/10.1016/j.ijpe.2019.01.010.

Johnson, D.S., McGeoch, L.A., 1997. The traveling salesman problem: a case study in local optimization. In: Aarts, E.H.L., Lenstra, J.K. (Eds.), Local Search in Combinatorial Optimization. John Wiley & Sons, New York, pp. 215–310.

Karak, A., Abdelghany, K., 2019. The hybrid vehicle-drone routing problem for pick-up and delivery services. Transp. Res. Part C: Emerg. Technol. 102, 427–449.

Kitjacharoenchai, P., Ventresca, M., Moshref-Javadi, M., Lee, S., Tanchoco, J.M.A., Brunese, P.A., 2019. Multiple traveling salesman problem with drones: mathematical model and heuristic approach. Comput. Ind. Eng. 129, 14–30.

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, 1144.

Manyam, S.G., Sundar, K., Casbeer, D.W., 2018. Cooperative Routing for an Air-Ground Vehicle Team–Exact Algorithm, Transformation Method, and Heuristics, arXiv preprint arXiv: 1804.09546.

Marinelli, M., Caggiani, L., Ottomauelli, M., Dell'Orco, M., 2017. En-route truck-drone parcel delivery for optimal vehicle routing strategies. IEE Intell. Transp. Syst. l2 (4), 253–261.

Mathew, N., Smith, S.L., Waslander, S.L., 2015. Planning paths for package delivery in heterogeneous multirobot teams. IEEE Trans. Autom. Sci. Eng. 12, 1298–1308.

Moshref-Javadi, M., Lee, S., 2017. Using drones to latency in distribution systems. In: IIE Annual Conference, Proceedings. Institute of Industrial and Systems Engineers (IISE), pp. 235–240.

Murray, C.C., Chu, A.G., 2015. The flying sidekick traveling salesman problem: optimization of drone-assisted parcel delivery. Transp. Res. Part C: Emerg. Technol. 54, 86–109.

Murray, C., Raj, R., 2020. The multiple flying sidekicks traveling salesman problem: parcel Delivery with Multiple Drones. Transp. Res. Part C: Emerg. Technol. 110, 368–398.

Poikonen, S., Wang, X., Golden, B., 2017. The vehicle routing problem with drones: extended models and connections. Networks 70 (1), 34–113.

Ponza, A., 2016. Optimization of Drone-assisted Parcel delivery. Master Thesis. University of Padova, Italy.

Pugliese, L.D.P., Guerriero, F., 2017. Last-mile delivery by using drones and classical vehicles. In: International Conference on Optimization and Decision Science. Springer, pp. 557–565.

Rabta, B., Wankmüller, C., Reiner, G., 2018. A drone fleet model for last-mile distribution in disaster relief operations. Int. J. Disaster Risk Reduct. 28, 107–112.

Ruiz, R., Stützle, T., 2007. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. Eur. J. Oper. Res. 177, 2033–2049.

Sacramento, D., Pisinger, D., Ropke, S., 2019. An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones. Transp. Res. Part C: Emerg. Technol. 102, 289–315.

Schermer, D., Moeini, M., Wendt, O., 2019. A matheuristic for the vehicle routing problem with drones and its variants. Transp. Res. Part C: Emerg. Technol. 106, 166–204.
Ulmer, M.W., Thomas, B.W., 2018. Same-day delivery with heterogeneous fleets of drones and vehicles. Networks 72, 475–505.
Wang, X., Poikonen, S., Golden, B., 2017. The vehicle routing problem with drones: Several worst-case results. Optimiz. Lett. 11 (4), 679–697.
Wang, Z., Sheu, J.B., 2019. Vehicle routing problem with drones. Transp. Res. Part B: Methodol. 122, 350–364.
Wen, T., Zhang, Z., Wong, K.K.L., 2016. Multi-objective algorithm for blood supply via unmanned aerial vehicles to the wounded in an emergency situation. PloS ONE 11 (5), 1–22.
Yurek, E.E., Ozmutlu, H.C., 2018. A decomposition-based iterative optimization algorithm for traveling salesman problem with drone. Transp. Res. Part C: Emerg. Technol. 91, 249–262.