Exact and heuristic approaches to Truck–Drone Delivery Problems
Júlia C. Freitas *, Puca Huachi V. Penna, Túlio A.M. Toffolo
Federal University of Ouro Preto, Department of Computing, Brasil

A R T I C L E   I N F O
Keywords:
Unmanned Aerial Vehicle
Traveling Salesman Problem
Drone delivery
Mixed-integer Programming
General Variable Neighborhood Search

A B S T R A C T
Collaborative delivery employing drones in last-mile delivery has been an extensively studied topic in recent years. In this paper, it is studied Truck–Drone Delivery Problems (TDDPs) in which a traditional delivery truck is gathered with a drone to cut delivery times and costs. The vehicles work together in a hybrid operation involving one drone launching from a larger vehicle that operates as a mobile depot and recharging platform. The drone launches from the truck with a single package to deliver to a customer. Each drone must return to the truck to recharge batteries, pick up another package, and launch again to a new customer location. This work proposes a novel Mixed Integer Programming (MIP) formulation and a heuristic approach to address the problem. The proposed MIP formulation yields better linear relaxation bounds than previously proposed formulations for all instances, and was capable of optimally solving several unsolved instances from the literature. A hybrid heuristic based on the General Variable Neighborhood Search metaheuristic combining Tabu Search concepts is employed to obtain high-quality solutions for large-size instances. The efficiency of the algorithm was evaluated on 1415 benchmark instances from the literature, and over 80% of the best known solutions were improved.

1. Introduction

Unmanned Aerial Vehicles (UAVs), generally known as drones, have recently shown a lot of potential in a wide range of civil purposes. In the last years, with the evolution of drone technology, these vehicles and their applications have been attracting the attention of both the academic literature and large transportation companies (Mercer, 2018). Drones are employed within many applications in different fields. They are no longer limited to military use, and different businesses are now investing in these devices for faster and more responsive customer service. One of the most important purposes for these unmanned vehicles lies in disaster management, rescue operations and health-care. Drones can operate in dangerous environments that are inaccessible to humans. Another potential use concerns law enforcement, given these devices have the innate ability to hover around locations without drawing much attention from people. Thus, surveillance and public safety are also prominent applications of drones. Drone delivery is another important and innovative application of drones, one which has become particularly relevant with the social distancing requirements due to the Covid-19 pandemic (Guillot, 2020; Douglas, 2020). The potential of drone delivery is vast both concerning operational cost and customer service efficiency. However, these aerial vehicles are not a replacement for the traditional delivery trucks, due to their low payload capacity and short flying range. Drones can, however, be a useful complementary feature to the delivery process since they are not limited by road networks. The applications of unmanned vehicles spread far beyond the preceding examples. For a complete survey on drone applications and insights into emerging modeling approaches, the reader is referred to Otto et al. (2018).

This work focuses on a groundbreaking delivery system composed by a truck and drone working collaboratively named Truck–Drone Delivery Problem (TDDP). We approach two different TDDPs: the Flying Sidekick Traveling Salesman Problem (FSTSP), first introduced by Murray and Chu (2015), and the Traveling Salesman Problem with Drone (TSP-D), presented by Agatz et al. (2018). Both problems are a generalization of the Traveling Salesman Problem (TSP), one of the most well known and studied problems in operational research. While the classical TSP consists in finding an optimal route for one vehicle to deliver goods to a set of customers (Fig. 1(a)), the TDDP consists in finding optimal routes for both a drone and one traditional delivery vehicle to perform goods distribution (Fig. 1(b)). In Fig. 1, circular nodes indicate customers served by the truck while triangular nodes represent those served by the drone. Dashed and continuous lines indicate a drone trip and a truck tour, respectively. The delivery truck departs from the depot carrying a drone and all customer parcels. As the driver performs deliveries, the drone is launched from the truck, carrying the parcel for an individual customer. While the drone is on a trip, no intervention from the delivery driver is required. After

* Corresponding author.
E-mail addresses: julia.freitas@aluino.ufop.edu.br (J.C. Freitas), puca@ufop.edu.br (P.H.V. Penna), tulio@toffolo.com.br (T.A.M. Toffolo).

https://doi.org/10.1016/j.ejtl.2022.100094
Received 24 October 2021; Received in revised form 10 July 2022; Accepted 30 September 2022
Available online 22 October 2022
Published by Elsevier B.V. on behalf of Association of European Operational Research Societies (EURO). This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).
delivering the parcel, the drone returns to the truck in a new customer location. One of the advantages of the truck–drone delivery system is the efficiency enhancement, as the drone’s capability to reach more customers increases once it is launched from the truck closer to the customer delivery location.

The contribution of this paper is twofold. A Mixed Integer Programming (MIP) formulation is proposed for the problem, capable of solving several unsolved instances from the literature. Also, we addressed a heuristic capable of finding high-quality solutions to the larger instances. Both the MIP model and the proposed heuristic were designed to tackle the two variants of drone delivery problems addressed by Murray and Chu (2015) and Agatz et al. (2018). The heuristic improved the best-known solution of 1138 instances from the literature (80.4% of the total). These results show the improvements that may occur in the total delivery time when using specific neighborhood structures of the problem within the local search. The results also indicate that combining truck and drone for last-mile parcel delivery results in huge improvements in delivery time over truck tours for the instances considered.

In summary, this work presents both exact and heuristic approaches for TDDP, resulting in improvements over the best known bounds and solutions for a vast number of problem instances from the literature.

This manuscript is organized as follows. Section 2 details the problem. Section 3 presents a few drone applications and a literature overview concerning TDDP. A novel MIP formulation for the problem is proposed in Section 4. The heuristic algorithm developed is introduced and described in Section 5. Section 6 presents the computational experiments and, finally, Section 7 summarizes the conclusions.

2. Problem description

The Flying Sidekick Traveling Salesman Problem (FSTSP) presented by Murray and Chu (2015) can be described as follows. Let $G = (V, A)$ be an undirected graph with $|V| = n + 1$ nodes. Node $v_0 \in V$ represents the depot, where drone and truck must depart from and return to exactly once. The two vehicles (drone and truck) may depart and return from the depot either in tandem or independently. While traveling in tandem, the drone is transported by the truck to conserve battery. Every other node $v \in V \setminus \{v_0\}$ represents a customer that must be visited exactly once. Some customers must be served by the truck due to the travel time being superior to drone endurance, while others may be served by either the truck or the drone.

The vehicles do not necessarily follow the same distance metric. The truck is limited to the road network, while the drone can use a different network to travel between customers.

During the delivery, the drone may make multiple trips, each composed of three locations. The trip begins at the launch node, which can be either the depot or a customer location. Before launch, the setup time $s^L$ is required for the drone to have its battery changed and the parcel loaded. The second node in a trip is the delivery node, which represents a customer serviced by the drone. The final or return node is the location where the drone is collected. This node may be either the depot or a location visited by the truck. An additional setup time $s^R$ is required for the return node since the drone must be recovered. We consider that whenever the drone is launched, the package will be successfully delivered and the drone will return to the truck or the depot without any issue, and within the drone’s flight endurance limit. If the drone trip ends at the depot, the drone cannot be re-launched.

The drone can only be recovered at a different position than the launch location in the truck route. Therefore, the drone must not be launched multiple times from the same location. The drone may visit only one customer per trip. The truck may serve other customers in between launch and return locations, while the drone is serving a customer. Moreover, either of the two vehicles may have to wait for the other one depending on their arrival times at the return node. The waiting time must not exceed the drone’s battery power. Regarding vehicle capacity, the drone can carry and deliver only one parcel per trip, while no limit is imposed on the truck’s capacity.

The FSTSP objective is to minimize the time required to complete all deliveries and return both vehicles to the depot.

3. Related literature

This section provides a literature review of the potential benefit of pilotless technology. This review first presents a few promising drone applications. Next, it aims at distinguishing the different approaches and delivery problem with drones.

Delivery applications have recently received considerable media attention, mainly because of the prospect of door-to-door express deliveries at low-cost. With this new concept, delivery times and costs could be significantly reduced. Because of the overwhelming amount of announcement in drone research by the industry, the literature has been growing significantly. Hereafter, we present some approaches that address the FSTSP and similar problems.

Murray and Chu (2015) proposed a mixed integer linear programming formulation for two delivery-by-drone problems in which the delivery is performed by truck and drone. In the Parallel Drone Scheduling Traveling Salesman Problem (PDSTSP) the drone attends customers within flight range of the depot while the truck attends the remaining. The operations occur independently, i.e., while the truck follows a TSP route the drones fly to the customers and back to the depot multiple times. In the other problem, the truck and drone work collaboratively (Flying Sidekick Traveling Salesman Problem — FSTSP). The authors implemented a simple and effective heuristic approach for
both problems. A similar work is the one by Bouman et al. (2018). They addressed an exact solution approach for the TSP-D based on dynamic programming that can solve problems up to 16 customers. This number of customers is larger than the mathematical programming approaches presented in the literature thus far. Ponza (2016) based his dissertation on the FSTSP proposed by Murray and Chu (2015). He proposed a slightly different mathematical formulation to the problem and presented an analysis of several heuristics that could be used to resolve the problem. Moreover, he introduced a new set of instances to the literature.

Dell’Amico et al. (2020) approached the PDTSP proposing a MILP model and several matheuristics. The authors experiment with the algorithms on the benchmark instances introduced by Saleu et al. (2018) and Murray and Chu (2015). The computational study validates that the proposed algorithms produce competitive results in terms of both efficiency and effectiveness mainly on small and medium-size instances.

Wang et al. (2017) study the Vehicle Routing Problem with Drones (VRPD) from a worst-case point of view. The paper describes several theorems formulation for the vehicle routing problem with drones and represents bounds on maximal savings to the companies. Poikonen et al. (2017) expand the description of the theorems comparing different drone configurations in the delivery process to determine the maximum benefit. For example, the trade-off between speed and the number of drones, i.e., they compare what is better, a more substantial number of slower drones or a smaller number of faster drones.

While the works above aim to minimize the time required to complete the tour, in Ha et al. (2018) the objective is to minimize the total operational cost of a drone–truck delivery system. They proposed a MILP formulation to the problem and a heuristic called TPS-LS, both inspired on the work of Murray and Chu (2015). Furthermore, a GRASP heuristic was presented. The results showed that GRASP provides better solution quality while TPS-LS deliver a lower solution quality, yet very quickly.

Another approach for the drone–truck system is the one presented in Jeong et al. (2019). They considered two practical issues to evaluate the problem, the effect of parcel weight on drone energy consumption and restricted flying areas. Their sensitivity analysis shows that the increase in package weight and no-fly zones reduce the efficient use of drones as it limits their flight range, especially when the two factors are combined.

The FSTSP proposed by Gonzalez-R et al. (2020) allows the truck to wait for the drone where it was launched. The drone also can perform multiple visits per launch. The authors considered drone energy, i.e., the battery is changed between drone trips and it is considered fully charged after the swap. An iterative greedy search heuristic combined with simulated annealing was proposed.

de Freitas and Penna (2020) introduces new instances based on the TSPLIB and compares the HGVNS (General Variable Neighborhood Search) result with instances found in the literature (Agatz et al., 2018; Ponza, 2016). In this work, we complement the heuristics by using a list, based on the Tabu Search, to avoid cycling in the neighborhoods. Here, we also propose a MILP to solve the FSTSP in the instances proposed in Murray and Chu (2015), Ponza (2016).

The multiple flying sidekicks traveling salesman problem (mfFSTSP) introduced by Murray and Raj (2020) considers a delivery truck operating in coordination with a fleet of drones. The drones are launched from the truck to deliver a single package, then return to the truck where it can be loaded again. They employed a three-phased (I. initial truck assignments, II. create drone routes, III. combining phase I and II) heuristic solution to approach the problem. The heuristic result analysis revealed that drones with high-speed and long-range offer greater benefits in larger geographic regions, where customers are distributed over a larger area.

Ferrandez et al. (2016) introduced a truck–drone delivery system where multiples drones travel per truck. They investigated the time and energy associated with a truck–drone delivery network compared to a standalone truck or drone delivery. Besides, they proposed a k-means and a genetic algorithm to determine the optimal number of launch locations and drones per truck. Following the multiple drones per truck approach, Karak and Abdelghany (2019) present a mathematical formulation and solution methodology for the hybrid vehicle–drone routing problem (HVDRP) for pick-up and delivery services. The problem is formulated as a mixed-integer program, which minimizes the vehicle and drone routing cost to serve all customers. The formulation captures the vehicle–drone routing interactions during the drone dispatching and collection processes and accounts for drone operation constraints related to flight range and load carrying capacity limitations. A novel solution methodology is developed which extends the classic Clarke and Wright algorithm (HCHW) to solve the HVDRP. The performance of the developed heuristic is benchmarked against two other heuristics, namely, the vehicle-driven routing heuristic (VDH) and the drone-driven routing heuristic (DDH). A set of experiments are conducted to evaluate the performance of the developed heuristics. While the VDH and the DDH focus on optimizing the cost of one mode only, the HCHW is shown to outperform these two heuristics in terms of minimizing the cost of the entire multi-modal network. The network operation cost is shown to be minimum when the used drones are balanced in terms of their flight range and load carrying capacity.

Scherner et al. (2019), Wang et al. (2017), Poikonen et al. (2017), Ulmer and Thomas (2018), Pugliese and Guerriero (2017) tackled the Vehicle Routing Problem with Drones (VRPD). In Scherner et al. (2019), they formulated the VRPD as a Mixed Integer Linear Program (MILP), and introduced several sets of valid inequalities aiming to improve the performance of solvers. To address large instances, they presented a matheuristic approach that exploits the problem structure of the VRPD. The authors proposed the Drone Assignment and Scheduling Problem (DASP) defined as minimization problem that given an existing routing of trucks, looks for an optimal assignment and schedule of drones such that the makespan is minimized.

Contrasting with the others approaches, Song et al. (2018) addressed a delivery problem where drones serve all customers. The authors described the Unmanned Aerial Vehicle Routing Problem (UAVRP) presenting new features to drone delivery problem: capacity and time window. Multiple drones, package weight impacting the vehicle battery life and customers demand to be satisfied are also considered. Service stations are strategically positioned to respect the endurance of drones and minimize delivery time. In these stations vehicles can be recharged and reloaded; thus, the drones can serve customers persistently. Computation experiments evaluated a heuristic and a mixed integer linear programming (MILP) formulation. While the MILP formulation could not solve large-scale problems, the heuristic successfully derives optimal or near-optimal solutions for them in a short time. Concerning drone-only delivery (Dorling et al., 2017) presented in their work an energy consumption model for multirotor drones and provided a linear approximation for it. They proposed the Drone Delivery Problem (DDP) which seeks to minimize cost (MC-DDP) or delivery time (MT-DDP) while considering battery weight, payload weight, and drone reuse. They proposed a MILP implementation and a Simulated Annealing (SA) to solve practical scenarios with hundreds of locations. Comparing the approaches, the SA implementation consistently finds near-optimal solutions to problems with eight or fewer locations. The heuristic behavior in larger instances with 125 to 500 customers showed consistent results. Arenzana et al. (2020) work presented a strategic framework to quantify the efficiency of hospital operations. They introduced a MILP to design drone delivery network for hospital deliveries. The problem minimizes drone traveling time, battery consumption levels, vehicle investment, and infrastructure costs. The trajectories designed between hospitals conform to the latest air traffic management regulations. They presented a case study based in London. The paper shows that drones present numerous advantages in comparison with traditional road transport. With operational costs averaging 30% depending on drone model and operational
parameters, such as vehicle range and payload size, the drone-based model increases service reliability (lower variability in travel time) and overcomes initial investment.

Table 1 summarizes the related work mentioned in this section. Column **Problem class** defines the problem approached in the paper. #Drones and #Trucks describe, respectively, the number of drones and trucks considered in the problem. Column Same L/R indicates the node where the drone is launched is necessarily equal from the node it returns. Same net describes if the network the vehicles travel are the same. Then, there are three columns describing drones: (1) whether energy consumption is evaluated (Endurance), (2) whether capacity is considered (Capacity), (3) whether drone can perform multiple visits (Drone multiple visits). Column Sync indicates if trucks and drones considered in the problem. Column Same L/R indicates the node where the drone is launched and returning to the node it is available online.

At first sight, the number of variables may seem prohibitively significant. However, in practice, this number can be considerably reduced by filtering variable sets \( x \) and \( y \) to consider only feasible connections, meaning only \((i,j) \in A\) and \((i,k,j) \in D\) should be considered. Moreover, despite requiring more variables, the formulation here proposed is stronger than those proposed by Murray and Chu (2015) and Ponza (2016), yielding better linear relaxation lower bounds for all instances considered (computational results are presented in Section 6).

The formulation is presented by Eqs. (1)-(15). To simplify the notation and reduce the constraints length, we assume \( L_i' \) variables are not generated by our implementation, whose source code is available online.\(^1\)

\[\text{min.} \quad \sum_{j \in V} x_{0,j} \]
\[\text{s.t.} \quad \sum_{(i,j) \in E} x^+_{i,j} = \sum_{(k,j) \in E} x^-_{k,j} = 1 \quad \forall i \in V \]

\[^1\] The formulation’s implementation is available at [http://www.github.com/tuliotoffolo/fstsp](http://www.github.com/tuliotoffolo/fstsp).
\[ \sum_{j \in \mathcal{V}'} x_{i,j}^{(t)} = \sum_{j \in \mathcal{V}} x_{i,j}^{t-1} \quad \forall t, t' \in L \setminus \{0\} \quad (4) \]

\[ \sum_{i,j \in \mathcal{A}} x_{i,j}^{t} \leq 1 \quad \forall t' \in L \quad (5) \]

\[ \sum_{i,k \in \mathcal{A}} \sum_{j \in \mathcal{V}} y_{i,k,j}^{(t)} \leq 1 \quad \forall t, t' \in L \quad (6) \]

\[ \sum_{i,k \in \mathcal{A}} \sum_{j \in \mathcal{V}} y_{i,k,j}^{(t)} + \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} y_{i,j}^{(t)} = 1 \quad \forall t \in V' \quad (7) \]

\[ \sum_{i,k \in \mathcal{A}} \sum_{j \in \mathcal{V}} y_{i,k,j}^{(t)} \leq \sum_{i \in \mathcal{V}, t' \in L} y_{i,j}^{(t)} \quad \forall j \in V, t' \in L \quad (8) \]

\[ \sum_{i,k \in \mathcal{A}} \sum_{j \in \mathcal{V}} y_{i,k,j}^{(t)} \leq \sum_{i \in \mathcal{V}, t' \in L} y_{i,j}^{(t-1)} \quad \forall j \in V, t' \in L \quad (9) \]

\[ t' - t \leq \epsilon + M \left( 1 - \sum_{i,k \in \mathcal{A}} y_{i,k,j}^{(t')} \right) \quad \forall t' \in L \setminus \{0\}, \quad t' \in L : \epsilon' > \epsilon \quad (10) \]

The upper bound was obtained by a simple Nearest Neighbor constructive heuristic, and its value is available as part of Ponza (2016) instances.

5. The hybrid heuristic

The proposed heuristic algorithm, named Hybrid Tabu General Variable Neighborhood Search (HTGVNS), is a hybrid metaheuristic that combines the exact solution of a TSP solver and the exploration capabilities of systematical neighbourhood changes. The HTGVNS employs the Route First Cluster Second approach of Beasley (1983) in which first a TSP is solved and then clusters are created by assigning customers to the drone.

The following sections detail the heuristic approach. First Section 5.1 introduces the different neighborhood structures used within the algorithm. Then, Section 5.2 formally introduces the proposed heuristic.

5.1. Neighborhood structures

Several neighborhood structures were developed to explore the search space of the TSP, based on different moves. When a move is applied on a reference solution \( S \), a neighbor solution \( S' \) is obtained. A neighborhood structure \( N \) consists of all possible applications of a move. Considering a reference solution \( S \), this results in a set of neighbor solutions \( N(S) \), which is hereinafter referred to as neighborhood.

The neighborhood structures considered are based mostly on classical TSP moves, with some of them relying on specificities of the problem. The solution space is visited applying the Best Improvement (BI) (Hansen et al., 2016) approach, which exhaustively explores a neighborhood and returns the solution with the lowest objective value, i.e., the best neighbor.

Only feasible solutions are accepted within the proposed algorithm, meaning that infeasible neighbors are discarded. Note that to be feasible a route \((i,k,j)\) must respect the drone battery’s life, i.e. \( r^D_{i,k,j} \leq \epsilon \). Similarly, the drone battery power must endure until the truck arrival at the return node. Sub-routes are considered to avoid a drone launch before a return, as illustrated by Fig. 2, where each rectangle represents a sub-route. In Fig. 2(a), drone customers can be assigned only to sub-routes 1 and 3. A launch must not occur in sub-route 2 since a drone trip is already associated with this sub-route. It is important to note that for every new drone trip, the truck route splits into one more sub-route. Therefore, new launches can only occur in sub-routes not associated with a drone. Fig. 2(b) describes the split of sub-route 3 as the result of a new drone trip.

The reader is directed to de Freitas and Penna (2020) for more details on the neighborhood structures considered here, which are divided into two categories: intra-route and inter-route, detailed next.

5.1.1. Intra-route neighborhood structures

The intra-route neighborhood structures considered are based on classical TSP moves.

i. \textit{Swap}(1,1): It swaps one customer in the solution with another one.

ii. \textit{Swap}(2,1): It swaps two consecutive customers in the solution with another one.

iii. \textit{Swap}(2,2): It swaps two consecutive customers in the solution with other two consecutive ones.

iv. \textit{2-opt}: This move is the classical 2-opt move proposed by Croes (1958). Two edges are removed, and then, the two paths created are reconnected in the only possible way to keep a valid tour.

v. \textit{Reinsertion}: This move consists of removing a customer from its current position and reinserting it in another one.
5.1.2. Inter-route neighborhood structures

The inter-route neighborhood structures are based on moves which envision solution improvement by exploring the problem characteristics.

i. **Shift(1,0):** This move consists of removing one truck customer and subsequently inserting it into the drone route. This move requires a cubic time, $O(c^2c')$ where $c$ is the number of truck customers and $c'$ is the number of eligible customers not currently assigned to the drone. This complexity time refers to three nested loops where the first defines the launch node $i$, subsequently, the delivery node $j$ and, lately, the return node $k$. This candidate trip must not exceed drone endurance. The pair of nodes $i$ and $k$ are not necessarily adjacent, but $i$ must precede $k$. The selected combination of $(i, j, k)$ is the one that presents the more considerable decreasing cost of the truck route within the removal of customer $j$ (see Fig. 3).

ii. **Swap(1,1):** This swap move consists of simply swapping two customers in the solution. One customer belongs to the truck and the other to the drone route. The complexity is $O(c'c)$ where $c'$ and $c$ are the number of drone and truck customers, respectively. This complexity is due to the necessity of comparing all drone nodes with every truck customer to find a combination at least as good as the previous or has a lower objective value. The swap is only performed if the drone endurance is not violated, i.e., the new drone trip can be completed before the drone runs out of battery power and the truck sub-route does not exceed the drone flight limit (see Fig. 4).

iii. **Swap(0,1):** This move is used to shake the current solution by turning a drone customer into a truck customer. A customer is removed from the drone route and inserted into the truck route in a position that generates the least impact on the solution quality. This move may increase the objective value, as it adds another customer to the truck route. However, if the truck waiting time is larger than the drone trip, a decrease in the objective value may occur. This move requires combining drone customers with every position of the truck route to get the customer position that produces the smallest impact on the solution quality. Therefore, the time complexity of performing this move is $O(c'c)$ where $c'$ and $c$ are the number of drone and truck customers, respectively (see Fig. 5).

5.2. HTGVNS algorithm

The proposed algorithm first builds an optimal TSP route by employing the exact approach from Applegate et al. (2016). This results in a truck route with all customers, including eligible drone customers, which is used to build an initial solution to the FSTSP. This procedure, presented by Algorithm 1, requires two arguments: (i) an initial TSP solution $S$ and (ii) the eligible drone customers $C'$. The algorithm iterates over the eligible drone customers until no improvement occurs. For each customer, $j \in C'$ the cost of removing it from the truck’s route is computed (line 5). It is also verified for each subroute if it is associated with a drone trip (line 6). For example, in Fig. 2(a) Subroute 1 is not associated with a drone trip while Subroute 2 is. If the subroute is paired with a drone, an attempt is made to insert customer $j$ into the truck’s route between adjacent nodes $i$ and $k$ (line 8). Otherwise, the cost of serving customer $j$ by drone is computed (line 10). Each pair of nodes $(i,k)$ such that $i$ precedes $k$ in a given subroute that is not currently connected with the drone is investigated. The goal is to compute the travel time associated with the drone trip launching from node $i$, visiting node $j$, and finally returning to the truck at node $k$.

The proposed algorithm is a General Variable Neighborhood Search (GVNS), which consists of Variable Neighborhood Search (VNS) using Variable Neighborhood Descent (VND) as local search. Hansen et al. (2010) introduced the GVNS metaheuristic that relies upon a local search followed by perturbations to escape from local optima. Furthermore, to avoid cycling, it was adopted a tabu list likely in Tabu Search (Glover and Laguna, 1998) to forbid solutions that possess some attributes of recently explored solutions, especially between insertion and removal of drone customers. For example, in Fig. 2(a) customer 4...
Three arguments are required: (i) an initial solution $S$, (ii) the maximum number of interactions and (iii) static tabu neighborhood list $\mathcal{N}_{TL}$. Algorithm 3 begins by initializing solution $S$, counter $k$, perturbation level $\rho$ and tabu list $TL$ (lines 1–4). The main loop (line 5) first generates a random neighbor solution $S$ from neighborhood $\mathcal{N}_{TL}$ (line 6). If neighborhood $\mathcal{N}_{TL}(k)$ is in $\mathcal{N}_{TL}$, then solution $S$ is inserted in tabu list $TL$ (lines 7–8). Afterwards, the RVND algorithm is called (line 9). If the solution produced by RVND is better than the current best solution, the best solution is updated (lines 10–11). Note that counter $k$ and perturbation level $\rho$ are reset only if $S$ is an improving solution over $S'$ (lines 12–13). Otherwise, the value of $k$ is incremented (lines 14–15). Afterwards, the perturbation is executed (lines 16–17), which corresponds to applying neighborhood $\mathcal{N}_{TL}(k)$ (see Section 5.1) $\rho$ times to the current solution. Next, the perturbation level $\rho$ is increased (line 18) and the loop repeated. Variable $\rho$ is reset when it equals $\rho_{\text{max}}$. Once the main loop reaches its stopping criterion, the best solution is returned (line 19).

Algorithm 3: General Variable Neighborhood Search

Input: Initial solution $S$, maximum interaction $k_{\text{max}}$, maximum perturbation level $\rho_{\text{max}}$, and static tabu neighborhood list $\mathcal{N}_{TL}$

1. $S' \leftarrow S$
2. $k \leftarrow 1$
3. $\rho \leftarrow 0$
4. $TL \leftarrow \emptyset$
5. while $k \leq k_{\text{max}}$
   6. $S \leftarrow$ random neighbor solution in $\mathcal{N}_{TL}(S)$
   7. if $\mathcal{N}_{TL}(k) \in \mathcal{N}_{TL}$ then
      8. insert $S$ in list $TL$
   9. $S' \leftarrow$ RVND($S$, $TL$, $\mathcal{N}_{TL}$)
   10. if $f(S) < f(S')$ then
       11. $S' \leftarrow S$
       12. $k \leftarrow 1$
       13. $\rho \leftarrow 0$
   14. else
       15. $k \leftarrow k + 1$
   16. for $i = 0$ to $\rho$ do
       17. $S \leftarrow$ random neighbor solution in $\mathcal{N}_{TL}(S)$
   18. $\rho \leftarrow (\rho \mod \rho_{\text{max}} + 1)$
19. return $S'$

was just added to the truck route as a drone customer. Thus, it must not return to the truck route for the next $|TL|$ iterations, where $TL$ is the tabu list.

The Randomized Variable Neighborhood Descent (RVND) is an adaptation of the classic VND heuristic in which the neighborhood’s selecting order is randomized. The VND is a heuristic described in Hansen et al. (2016) and further discussed by Souza et al. (2010) and Penna et al. (2013), who showed that the randomized procedure often outperforms the deterministic approach. The RVND procedure developed for the FSTSP is presented in Algorithm 2. The algorithm begins by filling the neighborhood structure list $\mathcal{N}$ (line 1) and initializing solutions $S$ and $S'$ (line 2). Next, variable $k$ is initialized (line 3), representing the current number of neighborhoods analyzed. The main loop (line 4) first randomly selects a neighborhood structure (lines 5–8). This structure can be chosen from two different sets, $\mathcal{N}$ or $\mathcal{N} \setminus \mathcal{N}_{TL}$. The first set is selected if the current solution $S$ is not in tabu list $TL$. Otherwise, the second set is selected. Afterwards, a new neighboring solution is generated (line 9). If this solution is at least as good as the previous or has a lower objective value than the considered entry is accepted, the counter $k$ is reset to 1 and list $\mathcal{N}$ is shuffled (lines 10–13). Otherwise, the value of $k$ is incremented (lines 14–15). Once the maximum number of neighbors is reached, the best solution produced is returned (line 16).

The combination of RVND and GVNS is presented in Algorithm 3. Three arguments are required: (i) an initial solution $S$, (ii) the maximum number of interactions and (iii) static tabu neighborhood list $\mathcal{N}_{TL}$. The RVND algorithm is called (line 9). If the solution produced by RVND is better than the current best solution, the best solution is updated (lines 10–11). Note that counter $k$ and perturbation level $\rho$ are reset only if $S$ is an improving solution over $S'$ (lines 12–13). Otherwise, the value of $k$ is incremented (lines 14–15). Afterwards, the perturbation is executed (lines 16–17), which corresponds to applying neighborhood $\mathcal{N}_{TL}(k)$ (see Section 5.1) $\rho$ times to the current solution. The perturbation level $\rho$ is increased (line 18) and the loop repeated. Variable $\rho$ is reset when it equals $\rho_{\text{max}}$. Once the main loop reaches its stopping criterion, the best solution is returned (line 19).
6. Experimental analysis

In this section we present experimental results and analysis of the proposed exact and heuristic approaches. The MIP formulations were modeled using the Python-MIP package and solved using the commercial solver Gurobi 9.0 with the default configuration, while the heuristic algorithm was coded in C++ and compiled with G++ 5.3.1. All experiments were executed on Intel Core i7 3.60 GHz computers with 16 GB of RAM running Ubuntu Linux 16.04.

First, Section 6.1 presents and analyzes the results obtained by the proposed MIP formulation considering the instances from Murray and Chu (2015) and Ponza (2016).\(^3\) Next, Section 6.2 discusses HTGVNS results for different benchmarks from the literature. Section 6.2.1 considers the instances proposed by Ponza (2016), Section 6.2.2 considers those by Agatz et al. (2018) and, finally, the set introduced by de Freitas and Penna (2020) is considered in Section 6.2.3.

6.1. Formulation results

The proposed formulation models the problem addressed by Murray and Chu (2015) and Murray and Chu (2016). This problem considers, however, slightly different constraints than those considered by Murray and Chu (2015). There are two points of attention:

1. in the problem described by Murray and Chu (2015), the truck’s travel time between the drone’s launch and return can be longer than the drone endurance; the drone can therefore run out of battery while waiting for the truck;

\(^3\) Ponza’s instances can be found at: https://github.com/tuliotoffolo/fstsp.

Table 3
Average number of variables and constraints per instance-set and endurance value.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#Vars</td>
<td>#Constrs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>5</td>
<td>1,440</td>
<td>658</td>
<td>158</td>
<td>116</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>6</td>
<td>1,440</td>
<td>1,023</td>
<td>212</td>
<td>144</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>7</td>
<td>1,440</td>
<td>1,893</td>
<td>274</td>
<td>196</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>8</td>
<td>1,440</td>
<td>4,411</td>
<td>344</td>
<td>281</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>9</td>
<td>1,440</td>
<td>4,161</td>
<td>422</td>
<td>295</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>10</td>
<td>1,440</td>
<td>9,208</td>
<td>508</td>
<td>422</td>
</tr>
</tbody>
</table>

Table 4

<table>
<thead>
<tr>
<th>Inst.</th>
<th>(e = 20)</th>
<th>(e = 40)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ponza (2016)</td>
<td>56.47</td>
<td>1800.15</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
<tr>
<td>Ponza (2016)</td>
<td>57.50</td>
<td>1800.16</td>
</tr>
</tbody>
</table>
2. The formulation proposed by Murray and Chu (2015) do not consider the setup time for launching the drone as part of the drone’s flying time, and it does not count for the total completion time or the battery’s endurance, even when the drone leaves from the depot.

Formulation (1)–(15) can be adapted to obtain results comparable with those by Murray and Chu (2015) by removing Constraints (10) and altering Constraints (11) and (12). Due to these small differences in the problem definition, we report results for instances by Ponza (2016) altering Constraints (11) and (12). Due to these small differences in Formulation (1)–(15) can be adapted to obtain results comparable with Formulation results for Ponza (2016) instances.

Table 5 presents the results obtained by Formulation (1)–(15) without alterations considering Ponza (2016)’s smaller instances containing from 5 to 10 customers. No formulation was capable of solving larger instances with 50, 100, 150 and 200 customers. For these instances, the formulations presented by Ponza (2016) did not obtain any feasible solution and the formulation we propose could not be executed due to memory limitations. It is thus by no means a coincidence that these large instances have only been addressed with heuristic approaches so far. Therefore, we developed the HTGVNS to tackle the large instances found in the literature.

The formulations by Ponza (2016) also obtained linear relaxation lower bounds of value zero for all instances, and so we omitted column LB0. It is clear that the formulations previously proposed in the literature are outperformed by the one we propose, which was capable of producing significantly better lower bounds and by consequence proven optimal solutions for all small instances, which are represented by a ⊙ in the table.

6.2. HTGVNS results

In this section, solutions provided by HTGVNS for different benchmarks and the parameters of the heuristic are detailed.

The HTGVNS tabu list size |TL| and perturbation level $\rho_{\text{max}}$ were manually tuned. After a significant number of experiments considering different values for the parameters, $\rho_{\text{max}}$ was defined as $n \times \frac{1}{T_{\text{max}}}$, with $n$ being the number of customers. Two sets were selected based on their performance to be the tabu list size: $|TL| = 2$ for instances with up to 20 customers and $|TL| = 7$ for instances with more than 20 customers. The solutions are kept in the tabu list until |TL| moves involving the drone are performed. The problem-specific input parameters, such as drone endurance, service time and vehicles speed, are defined individually for each instance set (see Sections 6.2.1–6.2.3).

From all the neighborhood structures implemented, the most effective ones for all set of instances were Shift(1,0) and Swap(1,1) intra-route, which improved the solution by 78% and 43% of the times they were considered, respectively. The reinsertion moves, Reinsertion and Or-opt2, presented the worst performance when analyzed individually.

The tables presented hereafter employ the following notation. Column Inst. indicates the instance name, and BKS describes the best-known solution reported in the literature. Columns Sol., Gap and Time indicate the best solution value, the gap between the best solution found by HTGVNS and BKS and the computational time in seconds, respectively. Finally, columns $\text{Sol.}$ and $\text{Gap}$ present the average solution cost of ten runs and the gap between the average solution cost of HTGVNS and the BKS.

HTGVNS uses the optimal TSP solution to create the initial FSTSP solution. The TSP solution is obtained using Concorde solver (Applegate et al., 2016) version 3.12.19 configured with CPLEX 12.6.3. Concorde is capable of finding the optimal solution without considerably increasing HTGVNS computational time. The computational experiments show that Concorde TSP Solver uses less than 7% of HTGVNS total time.

6.2.1. Results for Ponza’s benchmark set

Ponza (2016) generated 50 instances that were split into two sets. A set of small instances considering between 5 to 10 customers, and a
set of large instances including those with 50 to 200 customers. Five instances were generated for each number of customers.

For the set of small instances, HTGVNS resulted in solutions with the same objective value as the proposed formulation (see Table 5) in all executions. Therefore, HTGVNS always resulted in optimal solutions for all instances in the small set.

The results for the set of large instances are presented in Table 6. For some of these instances, Concorde was not able to find the initial solution. Therefore, for consistency, we used the Cheapest Insertion (CI) heuristic to obtain the initial solution for every test problem. The HTGVNS algorithm was able to improve the solution quality up to 15% for instances up to 100 customers. The scenario changes when the number of customers increases as HTGVNS presents a runtime much slower and replaced with two new edges and the 1-point-move (1p) swaps two nodes in the truck route, 2-opt move where two edges are

The results presented by these tables were collected by running the other hand, trucks must respect traffic sign regulation and follow the street network. On some instances the gap is still considerable. Therefore, the algorithm still has a lot to be improved.

4 https://github.com/pcbouman-eur/Drones-TSP

6.2.2. Results for Agatz benchmark set

A variant of the FTSP is the Traveling Salesman Problem with Drones (TSP-D) introduced by Agatz et al. (2018). While, the FTSP defines endurance (ε) and service time (s^d and s^f), the TSP-D neglects those variables (ε = ∞, s^d = s^f = 0), therefore, the drone can visit all customers. Furthermore, a drone trip can start and end at the same node (launch = return). The truck waits for the drone at the launch node or visits other customers then returns to the launch node. According to the authors, this strategy is beneficial, considering the truck can visit a node for either serve a customer, or supply the drone with another parcel. The authors proposed an enormous number of instances to the problem. In Bouman et al. (2015) it is explained how they are generated. They are divided into three sets, uniform, single-center and double-center distribution. The parameter a defines the ratio of drone speed to truck speed in an instance. Variable a = 1 determines that the vehicles travel with the same speed, when a = 2 the drone travels two times faster than the truck. Lastly, for a = 3, the drone speed is three times as fast as the truck.

Tables 7–9 are broken down into scenarios stratified by the value of a. The results presented by these tables were collected by running the available code of the author’s repository. It reflects the average of 10 runs for each instance size. As parameters, we adopted the same configuration employed in HTGVNS to obtain a comparable result. The TSP initial solution was obtained by the TSP solver Concorde. The four neighborhood structures available were: 2-point move (2p) that swaps two nodes in the truck route, 2-opt move where two edges are replaced and compared with some other endpoints and 1-point move (1p) move where a node in the truck route is relocated to a new position. Finally, the last neighborhood combines all the previous moves.

According to the tables, it is possible to notice that HTGVNS improved the average solution for a = (1,2) for both uniform and single-center combination. The HTGVNS could not improve the average solution of a = 3 for double-center and single-center distribution, the later set achieved the worst improvement compared to the others distributions and values of a. Concerning computational time, the different distribution of customers does not affect runtime; however, when the vehicles present the same speed runtime increases. Agatz et al. (2018) exact partitioning algorithm outperforms HGVS computational time for instances up to 100 customers. The scenario changes when the number of customers increases as HTGVNS presents a runtime much smaller than the LS.

Fig. 6, 7, and 8 show boxplot graphics within the solution value of 10 runs of HGVS and HTGVNS. For each distribution (uniform, single-center, and double-center) exist one boxplot representing the instance size (10, 20, 50, 75, 100, 175, 250) of a certain a value. The thick line corresponds to the median, i.e., half of the values are below this line, divided into two quartiles, and the other half is above this line, divided into two quartiles. The crosses in the graph correspond to the outliers, values that present a considerable distance from the others. The plots show the tendency of the medium size instance in all distributions have a smaller gap between the minimal and maximum values. It is possible to notice that HTGVNS found better gaps than HGVS; however, in some instances the gap is still considerable. Therefore, the algorithm still has a lot to be improved.

6.2.3. Results for Freitas and Penna (2020) benchmark set

The latest FSTSP benchmark set used to test the HTGVNS performance is the one proposed by de Freitas and Penna (2020). The authors introduced 25 FSTSP instances based on the TSPLIB Symmetric Traveling Salesman Problem instances (Skorobohatyj, 1995). These instances follow the FSTSP definition of Murray and Chu (2015). A different metric is considered to the vehicles travel distance. This is a reasonable consideration since drones are not affected by congestion, and they can fly in a straight line without considering the street network. On the other hand, trucks must respect traffic sign regulation and follow street network. The Euclidean metric is used to describe the drone
Fig. 6. Results for Agatz et al. (2018) instances with uniform distribution; left boxplots represent HGVNS results while right boxplots present HTGVNS results.

Fig. 7. Results for Agatz et al. (2018) instances with single-center distribution; left boxplots represent HGVNS results while right boxplots present HTGVNS results.
Table 8
HTGVNS results for Agatz et al. (2018) instances with uniform distribution.

<table>
<thead>
<tr>
<th>(n)</th>
<th>BKS</th>
<th>LS</th>
<th>(\text{Agatz et al. (2018)})</th>
<th>(\text{de Freitas and Penna (2020)})</th>
<th>HTGVNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>286.82</td>
<td>0.00</td>
<td>0.00</td>
<td>289.82</td>
<td>0.05</td>
</tr>
<tr>
<td>20</td>
<td>286.82</td>
<td>0.00</td>
<td>0.00</td>
<td>365.38</td>
<td>0.00</td>
</tr>
<tr>
<td>50</td>
<td>698.42</td>
<td>0.00</td>
<td>0.00</td>
<td>559.20</td>
<td>1.60</td>
</tr>
<tr>
<td>75</td>
<td>905.32</td>
<td>0.00</td>
<td>0.00</td>
<td>905.32</td>
<td>0.00</td>
</tr>
<tr>
<td>100</td>
<td>1113.96</td>
<td>0.00</td>
<td>0.00</td>
<td>1135.32</td>
<td>1.92</td>
</tr>
<tr>
<td>Average</td>
<td>1.67</td>
<td>74.60</td>
<td>0.00</td>
<td>0.78</td>
<td>44.98</td>
</tr>
</tbody>
</table>

Average 1.60 63.39 1.05 54.20

The eligible drone customers are randomly generated such that for each instance there are between 80% and 90% serviceable customers. Service times \(s^l\) and \(s^h\) are unitary. Moreover, both vehicles speed is 40 km/h, and the drone endurance is 40 min.

Travel distance, and the Manhattan metric is adopted to simulate the city block, thus, determining the truck distance.

The eligible drone customers are randomly generated such that for every instance there are between 80% and 90% serviceable customers. Service times \(s^l\) and \(s^h\) are unitary. Moreover, both vehicles speed is 40 km/h, and the drone endurance is 40 min.

Table 8 provides the results obtained by HTGVNS. Note that column TSP Time indicates the runtime required to obtain the optimal TSP solution by Concorde. These results consider ten HTGVNS executions for each instance available online.\(^5\) Service time causes a large impact in instances with short travel times because the truck may need to wait a long time for the drone. Moreover, instances with customers disposed along a road are less impacted by the drone presence, since customers would be on the truck’s way and it would be natural for the truck to

\(^5\) The instances and their solutions are available at http://goal.ufop.br/fs tsp/.
<table>
<thead>
<tr>
<th>𝛼</th>
<th>BKS</th>
<th>Agatz et al. (2018)</th>
<th>de Freitas and Penna (2020)</th>
<th>HTGVNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>634.42</td>
<td>652.30</td>
<td>2.82</td>
<td>0.00</td>
</tr>
<tr>
<td>20</td>
<td>754.81</td>
<td>776.32</td>
<td>2.85</td>
<td>0.00</td>
</tr>
<tr>
<td>50</td>
<td>1149.84</td>
<td>1149.84</td>
<td>0.00</td>
<td>0.40</td>
</tr>
<tr>
<td>75</td>
<td>1430.96</td>
<td>1430.96</td>
<td>0.00</td>
<td>3.00</td>
</tr>
<tr>
<td>100</td>
<td>1556.52</td>
<td>1605.50</td>
<td>3.15</td>
<td>9.60</td>
</tr>
<tr>
<td>175</td>
<td>2072.36</td>
<td>2155.85</td>
<td>4.03</td>
<td>88.30</td>
</tr>
<tr>
<td>250</td>
<td>2523.00</td>
<td>2572.80</td>
<td>1.97</td>
<td>353.60</td>
</tr>
<tr>
<td>Average</td>
<td>2.12</td>
<td>64.99</td>
<td>1.30</td>
<td>58.53</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>𝛼</th>
<th>BKS</th>
<th>Agatz et al. (2018)</th>
<th>de Freitas and Penna (2020)</th>
<th>HTGVNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>490.64</td>
<td>490.64</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>20</td>
<td>603.31</td>
<td>603.31</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>50</td>
<td>850.31</td>
<td>864.21</td>
<td>1.63</td>
<td>1.00</td>
</tr>
<tr>
<td>75</td>
<td>1075.12</td>
<td>1075.12</td>
<td>0.00</td>
<td>5.30</td>
</tr>
<tr>
<td>100</td>
<td>1199.09</td>
<td>1202.27</td>
<td>0.27</td>
<td>17.00</td>
</tr>
<tr>
<td>175</td>
<td>1611.13</td>
<td>1611.13</td>
<td>0.00</td>
<td>149.70</td>
</tr>
<tr>
<td>250</td>
<td>1862.44</td>
<td>1934.80</td>
<td>3.89</td>
<td>585.70</td>
</tr>
<tr>
<td>Average</td>
<td>0.83</td>
<td>108.39</td>
<td>1.68</td>
<td>41.24</td>
</tr>
</tbody>
</table>

Fig. 8. Results for Agatz et al. (2018) instances with double-center distribution; left boxplots represent HGVNS results while right boxplots present HTGVNS results.

visit them. This behavior occurs in instance pr144 which prevents a significant improvement.

7. Conclusions

This work approaches a hybrid delivery system composed by truck and drone working collaboratively named Truck–Drone Delivery Problem (TDDP). The problem considers parcels distribution that seeks to coordinate a traditional delivery truck with a drone that may be launched from the larger vehicle. This class of problem considers different vehicle speeds and drone endurance to minimize the required time to serve all customers. Here we studied two TDDPs variants. The Flying Sidekick Traveling Salesman Problem (FSTSP) presented by Murray and Chu (2015) and the Traveling Salesman Problem with Drone (TSP-D) introduced by Agatz et al. (2018).
A Mixed Integer Programming formulation was proposed, solving multiple previously unsolved instances with up to 10 customers. In addition to providing new optimal solutions, it also provided stronger bounds than those obtained from formulations previously proposed for the FSTSP (Murray and Chu, 2015; Ponza, 2016). However, due to the NP-hard characteristic of the problem, only small-scale instances could be solved.

We introduced a metaheuristic to tackle the TDDPs large-sized instances. The Hybrid Tabu General Variable Neighborhood Search (HTGVNS) algorithm initially constructs a solution by using a TSP solver to build the truck tour. This solution is then improved by the metaheuristic General Variable Neighborhood Search (GVNS) mixed with a Tabu Search list to avoid cycling.

The solution approach was validated through numerical analysis which indicates the potential of the truck–drone delivery system to improve the operation. The HTGVNS was evaluated considering three benchmark sets. For the FSTSP sets, the algorithm improved all the best-known solutions. On average, these solutions were improved by 3.25%. For the TSP-D instances, the heuristic improved or achieved the same result for 74% of the instance set.

The numerical analysis indicated that the hybrid delivery system can be more efficient by employing drones traveling faster than trucks. For the TSP-D benchmark set, the best solution found considers instances in which the drone travels twice as fast as the truck, and the smallest improvements were observed for instances where both vehicles present the same speed.

### Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

### Acknowledgments

The authors would like to thank the National Council for Scientific and Technological Development (CNPq), Grant 438473/2018-3. We also thank the Federal University of Ouro Preto (UFOP), and the Coordination for the Improvement of Higher Education Personnel (CAPES), Finance Code 001, for supporting the development of the present study.

### References


