Omega 104 (2021) 102493

Contents lists available at ScienceDirect

Omega

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

Algorithms based on branch and bound for the flying sidekick traveling salesman problem[☆]

Mauro Dell'Amico, Roberto Montemanni[*], Stefano Novellani

University of Modena and Reggio Emilia, Italy

ARTICLE INFO

Article history:
Received 21 August 2020
Accepted 14 May 2021
Available online 18 May 2021

Keywords:
Traveling salesman problem
Drone-assisted deliveries
Branch and bound
Heuristic algorithms

ABSTRACT

The use of drones in urban logistics is gaining more and more interest. In this paper we consider the flying sidekick traveling salesman problem, where some customers require a delivery and they can be served either by a truck or by a drone. The aim is minimizing the total time required to service all the customers.

We present a branch and bound algorithm especially designed to efficiently target small instances up to 15 customers and a heuristic algorithm, using the branch and bound as a subroutine, to attack larger instances. Extensive experimental results suggest the effectiveness of the exact solver for small instances and shows that the heuristic is able to provide state-of-the-art results for medium/large instances.

1. Introduction

Unmanned vehicles, and aerial drones in particular, are becoming of great interest in many sectors ranging from precision agriculture to catastrophic events management and freight delivery. Logistics operators are concretely considering the use of these vehicles for deliveries to customers, due to the potential economical advantages and flexibility of such a solution (Otto et al. [23], Carlsson and Song [8], Pei et al. [24], Boeck et al. [5], Dönmez et al [15]).

The first optimization work involving the use of one truck and one drone was the seminal paper by Murray and Chu [22], where the Flying Sidekick Traveling Salesman Problem (FSTSP) was firstly introduced. In this problem, the truck and the drone cooperate to serve customers. The drone operates from/to the truck when the latter is stationary, and the drone can serve one customer at a time. There is a synchronization phase when a truck has to collect back a drone. The objective function is to minimize the time required to serve all the customers and to go back to the depot. Some customers can be visited only by the truck, due to the characteristics of the delivery. In this form of the problem, the drone cannot return at the launching point, each drone flight is limited by a given battery endurance, and there are handling times to launch and collect the drone. Finally each customer can be visited only once. In [22] a Mixed Integer Linear Programming (MILP) formulation and three basic heuristic methods are proposed. Dell'Amico et al. [14] propose a random restart local search matheuristic approach for the FSTSP, while de Freitas and Penna [10] propose a randomized variable neighborhood descent and a hybrid general variable neighbourhood search algorithm for a variant of the FSTSP. Yurek and Ozmutlu [30] propose an iterative algorithm based on a decomposition approach. Ha et al. [17] solve a FSTSP whose objective function minimizes the costs depending on travelled distance and waiting times of both the truck and the drone, in contrast with the original version that minimizes the completion time of all the operations. They propose a MILP based on the Murray and Chu one and two heuristic algorithms. One heuristic first solves the TSP and then includes drone service with local search procedures. The second one is a greedy randomized adaptive search procedure that splits a TSP solution so to include drone flights and then improves the solution thanks to local search procedures. Ha et al. [18] propose a hybrid genetic algorithm to solve the minimum cost and the minimum completion time versions of the FSPTSP. Dell'Amico et al. [11] solve the FSTSP and some variants. They propose two mixed integer linear programming formulations that substantially refine the one of [22]. The same authors also propose in [13] further improved formulations able to solve larger instances.

Another problem involving the use of a truck and a drone is the TSP with drone (TSP-D), introduced in Agatz et al. [1]. It shares many characteristics with the FSTSP, but there are some substantial differences: the customers can be visited multiple times by the truck if it is a convenient location for launching and collecting a drone, the drone can return to the same place it has been launched, the battery endurance of the drone is unlimited and

[☆] Area: Production Management, Scheduling and Logistics. This manuscript was processed by Associate Editor Otto.

[*] Corresponding author.
E-mail address: monteman@unimore.it (R. Montemanni).

https://doi.org/10.1016/j.omega.2021.102493
0305-0483/© 2021 Elsevier Ltd. All rights reserved.

launching and rendezvous times are not considered. An integer linear programming model and some heuristics are presented by the authors. Of particular interest for the current research are the heuristic methods, since they can be easily adapted to the FSTSP. The most promising of them will be used as a reference later on in the experimental section. Bouman et al. [6] extend the work by Agatz et al. [1] by solving the TSP-D with dynamic programming that first enumerates the shortest paths that a truck can do, then combines some of the truck paths with drone flights, and eventually it combines the components to provide the optimal solution. To shorten the computing times the authors also propose a heuristic method obtained by restricting on the number of nodes visited by the truck during a drone flight.

Poikonen et al. [25] propose a branch and bound for a variation of the TSP-D that considers a maximum endurance for drone missions and forbids multiple visits by the truck to the same customer, but allows the truck to be stationary at a customer for a whole drone delivery. The logic of the branch and bound approach is to treat ordered sequences of customers that are then split between the truck and the drone in the most convenient way. This approach is diametrically opposed to the one we will propose in Section 3, where we work directly on possible truck missions to which we add drone launches. The authors of [25] also propose several heuristic algorithms, some strictly derived from the branch and bound solver (by heuristically using non valid lower bounds in a clever way, a major contribution of the paper) and another one based on a divide-and-conquer paradigm.

A comprehensive literature review on optimization problems with drones, fully covering the problems mentioned above and many others, can be found, together with perspectives for future research in the surveys by Otto et al. [23], Viloria et al. [29], Chung et al. [9] and Macrina et al. [20]. A substantial research effort is currently spent on distribution problems with trucks and drones. In parallel with the present research more relevant papers have been submitted or published. We mention the exact methods based on mathematical programming models that appeared in Schermer et al. [27] (new MILP formulations and branch and cut method), El-Adle et al. [16] (a MILP formulation improved enriched by valid inequalities and bounds), Vásquez et al. [28] (a novel approach based on Bender's decomposition), Boccia et al. [4] (a new MILP formulation solved by a column and row generation method), and Roberti and Ruthmair [26] for variants of the TSP-D. The paper presents a new compact formulation and a solving technique based on branch and price. The latter approach isable to solve to optimality instances with up to 39 customers.

In this paper we present some algorithms for the FSTSP. A first contribution is a branch and bound algorithm characterized by not completely specified solutions in the search tree, that are later fully determined by solving an Assignment Problem [7]. Such a choice limits the size of the search tree, but on the other hand tends to weaken lower bounds. Experimental results show, however, that the choice pays off for instances of limited size, leading to very good results in terms of speed for instances up to 10–15 customers. The same branch and bound algorithm is later used as a subroutine for a simple heuristic algorithm, which is the second contribution of this work. Again, experimental results prove that such a heuristic approach is extremely competitive on large instances, being able to effectively deal with instances with up to 229 customers in a reasonably short time.

2. Problem description

The FSTSP can be formally described as follows. A set of customers $C = \{1, \ldots, n\}$ has to be visited. The truck starts from the depot 0 and returns to the final depot $n+1$. It is equipped with an autonomous flying drone that can be used to visit customers in parallel to the truck while the latter is traveling or serving other customers. A drone service is called *sortie*, defined by a launching node (corresponding to either depot 0 or a customer), one served customer, and a rendezvous node (again corresponding to either a customer or depot $n+1$). Note that only one customer can be served in a sortie. All customers of C can be served by the truck, but only a given subset $C' \subseteq C$ can be served alternatively by the drone. The problem is modelled via a digraph $G = (N, A)$, where the node set $N = \{0, 1, \ldots, n+1\}$ represents all the customers and depots, and A be the set of all the arcs $(i, j), i \in N_0, j \in N_+, i \neq j$, with $N_0 = \{0, 1, \ldots, n\}$ and $N_+ = \{1, \ldots, n+1\}$. Each arc (i, j) is associated with two non-negatives traveling times: τ^T_{ij} and τ^D_{ij}. They represent, respectively, the time for traveling arc (i, j) by the truck and by the drone. Serving times at customers for both drone and truck are included in the travel times, while the time for preparing the drone at launch is given by σ_L and the time to collect a landing drone is given by σ_R. No launch time is considered when a sortie starts from the depot 0. A battery limit (*endurance*) of E time units characterizes the drone. Rendezvous time σ_R contributes to the endurance computation while σ_L does not. The rationale is that the drone lies on the truck when it is prepared for the launch, while it flies during the rendez-vous operations. The objective of the optimization is to minimize the completion time, that is the moment when the last vehicle arrives at the final depot $n+1$, with all the customers having been visited.

Note that the drone can be launched from the truck only when the truck is stopped at a customer or at the depot; the drone cannot leave the depot before the truck starts its route; the truck has to keep serving customers while the drone is performing a sortie. So the so-called loops, with a drone performing a delivery while the truck is parked, are forbidden. A synchronization phase is implied by the previous settings: the vehicle (drone or truck) that arrives first at a rendezvous point has to wait for the other, consuming battery in case of the drone.

3. A branch and bound exact algorithm

The main concept in the Branch and Bound algorithm (BB) we propose is that of mission. A *mission* can be either the move of the truck from one customer to another, or a phase where the truck and the drone operate in parallel. In the latter case the truck visits a sequence of customers, with the first customer being the launching point of the drone, and the last customer being where the drone is collected. In our branch and bound, a partial solution is incrementally augmented with new missions at each level of the search tree, until the final depot is reached. Note that in our design customers are assigned to drones only in a final stage, as described in the remainder of this section. The choice has been made to limit the size of the search-tree, and to avoid an excessive explosion of it. In a final stage an Assignment Problem [7] is solved to complete a solution with drone deliveries in the most convenient feasible way (if any). The whole algorithm is designed more towards execution speed than quality of the bounds. The rationale of this choice is that we purposely target instances with a limited number of missions (corresponding to 10–15 customers). The secondary aim for the branch and bound method is to use it as a subroutine for the heuristic solver that will be discussed in Section 4.

A mission M is formally characterized by the following elements:

- $C(M) = \left(c^M_1, c^M_2, \ldots, c^M_{|C(M)|}\right)$ is an ordered list of customers visited by the truck in the given sequence;
- $T(M) = \sum_{i=1}^{|C(M)|-1} \tau^T_{c^M_i, c^M_{i+1}}$ is the travel time required by the truck to visit the customers of $C(M)$;

- $DT(M)$ is the drone handling time, and is defined as follows, assuming the drone will not be used if $|C(M)| = 2$ (this assumption is eventually retracted later when evaluating a solution):

$$DT(M) = \begin{cases} 0 & \text{if } |C(M)| = 2 \\ \sigma_R & \text{if } |C(M)| > 2 \text{ and } c_1^M = 0 \\ \sigma_L + \sigma_R & \text{if } |C(M)| > 2 \text{ and } c_1^M \neq 0 \end{cases}$$

The logic adopted within the algorithm is that if $|C(M)| > 2$, then there is always a drone delivery associated with the mission M. In case $|C(M)| = 2$, a drone delivery is possible (and in this case the drone handling times will be added a posteriori), but not compulsory. The rationale behind the choice of postponing the decision wether there is a drone deliveries associated with missions with exactly two truck customers is motivated by computational reasons: deciding already within the search tree wether a mission M with $|C(M)| = 2$ has or not a drone delivery associated would substantially increase the size of the search tree itself (doubling the nodes corresponding to these missions). Preliminary experiments very clearly suggested to avoid the latter strategy.

The customers visited by the drone are however not assigned inside the branch and bound tree, but only at the end, when truck paths from 0 to $n+1$ have been built, using several missions, and the complete solutions are evaluated by assigning the customers to the drone's sorties (see Section 3.3). This design choice is motivated by the reduction of the size of the search tree, that would have been much more severe otherwise. A side effect is a worsening of the lower bounds quality, but preliminary experiments suggested to go for the chosen trade off.

Note also that the method we describe in the remainder of this section is devised to target efficiently small/medium instances with up to 15 customers. This represented the limit of current state-of-the-art solvers at the time the present study was devised. Very recently a solver able to provide exact solutions for instances with up to 39 customers have been proposed [26].

3.1. Feasible missions and dominanated missions

A mission M with more than 2 customers is *feasible* only if at least a customer c not visited in the current partial solution and that can be visited by the drone exists such that $\tau^D_{c_1^M,c} + \tau^D_{c,c^M_{|C(M)|}} + \sigma_R \leq E$, and anyway $T(M) + \sigma_R \leq E$, i.e. the constraint about the battery endurance of the drone is fulfilled. These missions are intrinsically characterized by a very few customers, given the current technological restrictions on batteries.

Given a new mission identified by a first and last customers and by a set of customers to visit in between by the truck, we only focus on the order of the customers associated with the shortest route. All the other permutations are *dominated* and therefore not considered. Note that calculating the best route is not prohibitive from a computational viewpoint since only feasible missions (that are intrinsically short) have to be considered.

We will only consider non-dominated missions in the algorithm we propose.

3.2. Search tree node structure

A search tree node ν represents a partial feasible solutions (later evolving into a full one), and contains the following elements:

- $MI(\nu) = \left(M_1, M_2, \ldots, M_{|MI(\nu)|}\right)$: an ordered list of feasible missions. Note that the first customer of the first mission has to be the depot and that given two consecutive missions, the last customer of the former one has to coincide with the first of the latter one;
- $TT(\nu) = \sum_{i=1}^{|MI(\nu)|} (T(M_i) + DT(M_i))$ is the current estimation for the time required by the partial solution associated with node ν. Only truck time and drone handling times for missions with more than 2 customers are considered at this stage, and any eventual truck waiting time is neglected. The real cost of a complete solution will be calculated only at the end, as described in Section 3.3;
- $LB(\nu)$ is a lower bound on the total time required by a complete solution built upon the partial solution associated with the current search tree node. Details on the calculation of the lower bound will be provided in Section 3.4.

Note that only non-dominated feasible partial solutions, involving only non-dominated feasible missions, are considered (see Section 3.1).

3.3. Evaluation of a solution

A leaf of the branch and bound tree is a node associated with a solution with the destination depot $n+1$ as the last element of the last mission, with all the customers in $C \backslash C'$ already assigned to the truck, and with a number of *unassigned* customers – i.e. customers not involved in any mission. Note that for feasible solutions the number of unassigned customers is greater than or equal to the number of missions with more than 2 customers, and anyway less than or equal to the total number of missions. Leaves not fulfilling these properties are not generated, being not associated with feasible solutions.

In Fig. 1 a solution is depicted, where 0 and 7 represent the starting and ending depots, customers 1 and 2 are unassigned, and the solution is composed of four missions, one of which (the black one with customers 3, 4, 6) must have an associated drone mission, while the others (red) have an optional drone mission associated. In the context of our algorithm, we need to assign unassigned customers to the missions.

Once a search tree leaf is encountered, the corresponding solution has to be evaluated both for feasibility and cost (total time requirement). In our implementation, the evaluation is carried out by solving a Linear Min Sum Assignment Problem [7], operation requiring polynomial time. An Assignment Problem can be defined as follows. There are two sets A and B of equal size, and a cost g_{ij} for each $i \in A$ and $j \in B$. The objective is to find a bijection $f : A \longrightarrow B$ such that the quantity $\sum_{i \in A} g_{i,f(i)}$ is minimized. Note that in case $|A| \neq |B|$, polynomial-time manipulation techniques are known to transform the problem into a problem with identically sized sets.

The Assignment Problem to be solved is as follows. Let set A contain unassigned customers (note that they can be visited by the drone by definition, otherwise the solution under investigation would not be feasible), and set B contain the missions in $MI(\nu)$. In order to simplify the next formulae, let us redefine the sequence of customers visited by a mission M_j, as $C(M_j) = \left(c_1^j, c_2^j, \ldots, c^j_{|C(M_j)|}\right)$. Note that generating only partial feasible solution we always have $|A| \leq |B|$. Before giving the rest of the details, let us also provide the following definitions:

$$\alpha_{ij} = \max\left\{T(M_j); t^D_{c_1^j,i} + t^D_{i,c^j_{|C(M_j)|}}\right\} \tag{1}$$

$$\beta_{ij} = \max\left\{t^D_{c_1^j,i} + t^D_{i,c^j_{|C(M_j)|}} - T(M_j); 0\right\} \tag{2}$$

Value α_{ij} reports the maximum traveling time of the truck or drone, from c_1^j to $c^j_{|C(M_j)|}$, while β_{ij} gives the possible time the truck must wait for the drone in $c^j_{|C(M_j)|}$.

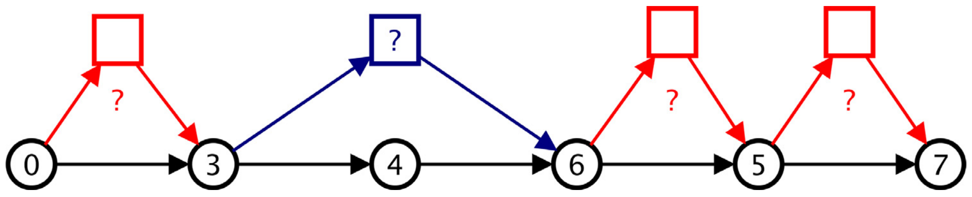

Fig. 1. Partial solution associated with a leaf of the branch and bound tree. Nodes 0 and 7 represent the starting and ending depots, respectively; customers 1 and 2 are unassigned; the association of drone sorties to red missions is optional; the customer visited by the drone in the black mission is still unassigned, but compulsory.

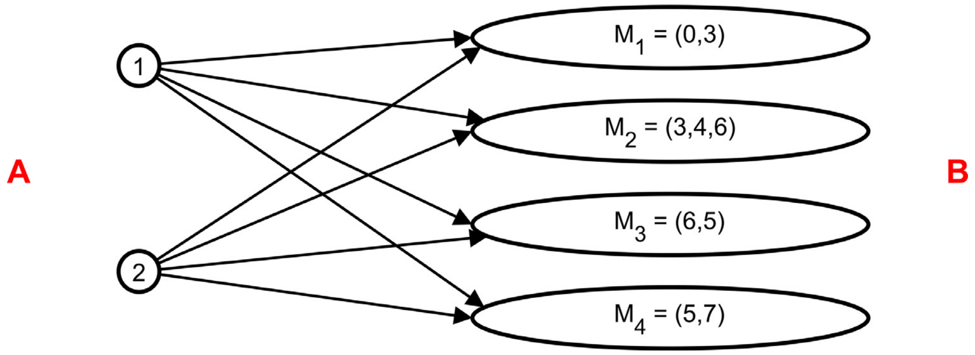

Fig. 2. Sketch of the Assignment Problem solved to complete the partial solution depicted in Fig. 1. Edge costs g are assigned based on the characteristics of the instance, as described in Section 3.3.

For each $i \in A$ and $j \in B$ the cost g_{ij} for the assignment problem is given as follows:

$$g_{ij} = \begin{cases} +\infty & \text{if } \alpha_{ij} + \sigma_R > E \\ \beta_{ij} & \text{if } \alpha_{ij} + \sigma_R \le E \text{ and } |C(M_j)| > 2 \\ \beta_{ij} + \sigma_R + Q & \text{if } \alpha_{ij} + \sigma_R \le E \text{ and } |C(M_j)| = 2 \text{ and } c_1^j = 0 \\ \beta_{ij} + \sigma_L + \sigma_R + Q & \text{if } \alpha_{ij} + \sigma_R \le E \text{ and } |C(M_j)| = 2 \text{ and } c_1^j \ne 0 \end{cases} \quad (3)$$

Where Q is a sufficiently large constant (e.g. $Q = \sum_{(i,j)\in A}(t_{ij}^T + t_{ij}^D)$) inserted to guarantee that already defined missions with more than 2 customers get assigned a drone before any other assignment happens. The first option of (3) covers unfeasible assignments; the second compulsory assignments to the missions with at least 3 customers; the third and the forth assignments to missions with exactly 2 customers (distinguishing the special case where the first node corresponds to the starting depot 0).

In Fig. 2 the Assignment Problem corresponding the solution depicted in Fig. 1 is provided.

Let γ denote the value of the optimal solution of the Assignment Problem and θ the number of missions with more than 2 customers in $MI(\nu)$. If $\gamma \ge Q \cdot (|A| - \theta + 1)$, then no feasible solution exists at node ν, since not all the missions with more than 2 customers have been assigned an element of A. Otherwise, the value of the best complete solution associated to ν is as follows:

$$cost(\nu) = TT(\nu) + \gamma - Q \cdot (|A| - \theta)$$

3.4. Calculation of the lower bound $LB(\nu)$

Given a search tree node ν, associated with a partial solution still not reaching the destination depot, the value of the lower bound $LB(\nu)$ is computed as described in the remainder of the section. An example of partial incomplete solution is provided in Fig. 3, where customers 1, 2 and 5 are unassigned together with the destination depot 7.

The idea behind the lower bound is to solve an Assignment Problem built in such a way that some unassigned customers are associated with missions requiring a drone customer, while the rest are assigned in a convenient way, either to the truck or to the drone. The bound can be seen as an extension of the evaluation

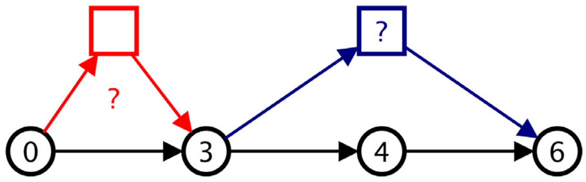

Fig. 3. Partial solution associated with an inner node of the branch and bound tree. Nodes 0 and 7 (unassigned) represent the starting and ending depots, respectively; customers 1, 2, 5 are unassigned; the association of a drone sortie to the red mission is optional; the customer visited by the drone in the black mission is still unassigned, but compulsory.

method described in Section 3.3, where unassigned customers are assigned in the cheapest possible way according to some simplified rules easy to compute. These assignments potentially lead to infeasible solutions, and therefore the resulting cost is a lower bound. The overall idea is mainly designed to be fast, with less emphasis on retrieving tight bounds. Preliminary experiments suggested to adopt this strategy.

In the context of the lower bound, we need to assign unassigned customers either as drone customers for missions previously defined, or to be in new missions served by truck or drone. For ease of notation, we define a set H to contain all unassigned customers, the arrival depot, and $\hat{c}$, the last customer visited in the partial solution $MI(\nu)$. We have a set A containing unassigned customers (note that in this case some of them can be visited by the drone, others not). Note that $H = A \cup \{\hat{c}, n+1\}$. Finally, a set $B = B_1 \cup B_2 \cup B_3$ is defined as follows:

B_1: Each mission $M_j \in MI(\nu)$ is associated with a node $b_j^1 \in B_1$. Note that this set corresponds to B in the evaluation of a complete solution, see Section 3.3;

B_2: All possible unordered pairs of customers from the set H are contained in the set B_2. The pairs represent potential previous and next customers for an unassigned customer assigned to be visited by the truck. Note that by definition $|B_2| = \frac{|H|\cdot(|H|-1)}{2}$;

B_3: Possible drone visits for unassigned customers during the part of the solution still unknown are contained in the set

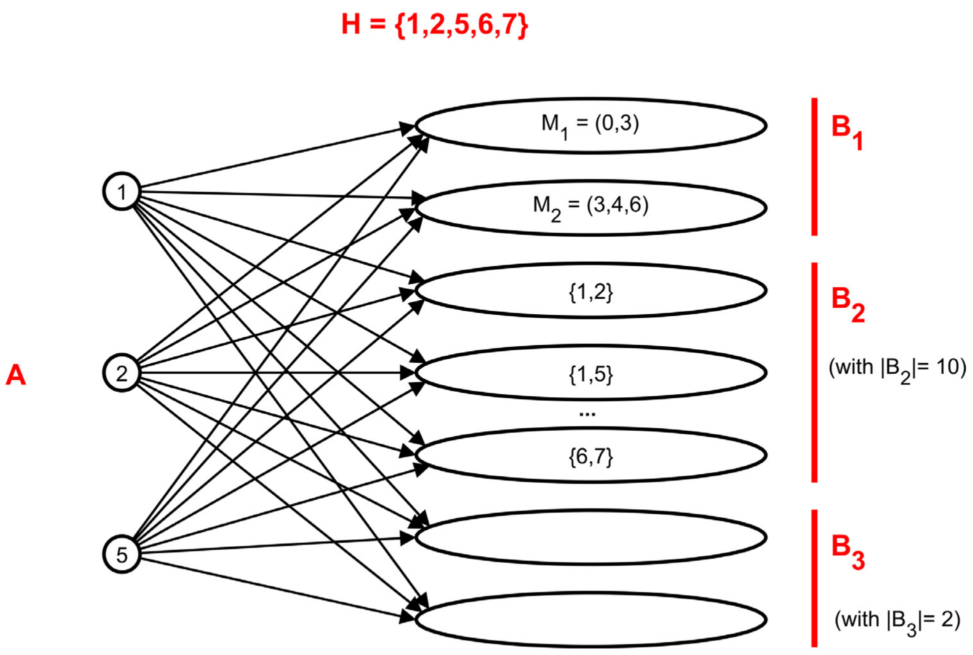

Fig. 4. Sketch of the Assignment Problem solved to complete the partial solution depicted in Fig. 2. Edge costs g are assigned based on the characteristics of the instance, as described in Section 3.4.

B_3. By construction $|B_3| = \left\lfloor \frac{|H|-1}{2} \right\rfloor$, since it represents a trivial upper bound for the number of drone missions in the part of the solution still to be completed.

For $i \in A$, $j \in B_1$, the cost g_{ij} is defined according to (3).

Referring to the pair of customers associated to the node j as $\{c_a^j, c_b^j\}$ for ease of notation, For $i \in A$, $j \in B_2$, we have:

$$g_{ij} = \begin{cases} +\infty & \text{if } i = c_a^j \text{ or } i = c_b^j \text{ or } i \notin C' \\ \frac{1}{2} \cdot \min\left\{ \tau^T_{c_a^j,i} + \tau^T_{i,c_b^j}; \tau^T_{c_b^j,i} + \tau^T_{i,c_a^j} \right\} + Q & \text{otherwise} \end{cases} \tag{4}$$

The rationale for the infinite cost is that a customer cannot be before or after itself. In the second option, the cost is given by the truck time required to have the customer i inserted between the two customers associated with node j. The division by 2 is related to the calculation of the lower bound after the solution of the Assignment Problem, as described later in this section: the contribution of each customer assigned to the truck is taken as half of the cost of its incoming arc and half of the cost of its outgoing arc. The meaning and the role of Q are the same as in Section 3.3.

For $i \in A$, $j \in B_3$, we have the following definition:

$$g_{ij} = \begin{cases} +\infty & \text{if } i \notin C' \\ \sigma_L + \sigma_R + Q & \text{if } i \in C' \end{cases}$$

This case covers the situation where the customer $i \in C'$ is assigned to the drone for the chunk of solutions still undisclosed: we have no element for judging any synchronization issue, so the cost only consider launching and collecting costs.

Figure 4 contains an example of the Assignment Problem solved to calculate the lower bound for the partial solution provided in Figure 3. Also in this case, costs are not instantiated in the figure, and they depend on the characteristics of the instance.

Let again θ be the number of missions with more than 2 customers in $MI(\nu)$. If the optimal solution of the Assignment Problem has a cost larger than $Q \cdot (|A| - \theta + 1)$, then no feasible expansion is possible for the partial solution associated with node ν, and pruning can be triggered. The infeasibility is motivated by the impossibility of assigning customers to be served by the drone to all the missions with more than 2 customers. Otherwise, if the optimal assignment has a finite cost γ, then a lower bound for the cost of the best extension of the partial solution associated with node ν can be derived as follows (we racall $\hat{c}$ is the last customer of the last mission of the partial solution associated with the search tree node ν):

$$LB(\nu) = TT(\nu) + (\gamma - Q \cdot (|A| - \theta)) + \frac{1}{2} \cdot \left(\min_{i \in H \setminus \{\hat{c}\}} \tau^T_{\hat{c},i} + \min_{i \in H \setminus \{n+1\}} \tau^T_{i,n+1} \right) \tag{5}$$

The second term being the solution of the Assignment Problem minus the penalty based on Q corresponding to a feasible solution. The last term being the cost for completing the estimation of the cost of the truck tour, given by half of the cost of the most convenient outgoing arc from $\hat{c}$ plus half of the cost of the most convenient incoming arc for $n+1$. Note that both in (4) and (5), the rationale behind the factor $\frac{1}{2}$ is that giving the hypothetic truck tour leading to the lower bound, we split the cost of each of its arcs between the extremes of the arc itself, being them either customers or depots.

If $LB(\nu)$ is not lower than the cost of the best solution retrieved so far, then node ν can be pruned.

3.5. Starting solution

Given an instance, a feasible solution for the FSTSP can be easily computed by considering only truck deliveries, which means to solve a classic Traveling Salesman Problem (TSP, see [3]) associated with the customers of the problem. Given a TSP solution $S = (0 = s_0, s_1, \ldots, s_n, \ldots, s_{n+1} = n+1)$, we can translated it into a solution for the FSTSP of cost $\sum_{i=0}^{n} \tau^T_{s_i, s_{i+1}}$.

Note that the order of the customers induced by the starting solution has an important role within the branch and bound solver, since the customers will be analysed according to this order for branching purposes (see Section 3.6). The rationale is that the sequence provided by the TSP should vaguely resemble the shape of the truck route of the optimal FSTSP solution.

3.6. Branching and visiting strategies

The root of the search tree is associated with an empty solution, and all its children nodes will be associated with a mission starting at the depot 0.

Given the partial solution associated with a node v at a generic level L of the search tree (which means there are L missions in the current partial solution), new search tree nodes are created at level $L+1$ in such a way that all the non-dominated feasible missions are considered as possible expansions. The number of new nodes can be exponential, being a new mission with more than 2 customers only limited by the availability of a customer for a drone mission and by the total truck travel time, that cannot be longer than the drone endurance E. The explosion is however not dramatic for instances with up to 15 nodes (the designed target for the method we propose).

The expanded search tree nodes are generated from the largest to the smallest (in terms of number fo customers). The search-tree nodes are visited in a depth-first fashion, and the nodes are expanded in the order they are created. In this way the nodes with a longer last mission will be expanded first. The combination of the branching and visiting rules described above is motivated by practical considerations: long missions are likely to better exploit the drone, and consequently are more promising and likely to lead to better solutions earlier.

The overall branch and bound algorithm can be summarized as follows:

- Step 1: Set $BestSol$ = TSP solution for the customers of the FSTSP instance (see Section 3.5);
  We set $L=0$ as the current level of the search tree;
  We set $PSol(i)=\emptyset$ for all $i \in \{1,\dots,n+1\}$ as the stacks (see [2]) containing the partial solutions to expand at each level of the search tree;
  We insert the solution containing only the starting depot into the stack of level 0: $Push(\{0\}, PSol[0])$;
- Step 2: If $PSol[L]=\emptyset$ then $L=L-1$ and goto Step 5;
- Step 3: Set $Partial = Pop(PSol[L])$ as the partial solution to expand;
  For each feasible and not dominated mission M extending solution $Partial$:
  - We set $Temp$ as the solution obtained by appending M to $Partial$: $Temp = Append(M, Partial)$;
  - If $Temp$ ends in depot $n+1$, then we calculate its cost (see Section 3.3) and if it improves the cost of $BestSol$, then $BestSol = Temp$;
  - If $Temp$ does not end in depot $n+1$, then we calculate the lower bound (see Section 3.4) and if it is lower than the cost of $BestSol$, then we add the partial solution to be examined at the next level: $Push(Temp, PSol[L+1])$;
- Step 4: $L=L+1$;
- Step 5: If $L>0$ goto Step 2.

4. An iterative heuristic algorithm

The heuristic method we propose is heavily based on the branch and bound exact method described in Section 3, which is here employed as a subroutine.

Given an initial feasible TSP solution $S=(0=s_0,s_1,\dots,s_n,\dots,s_{n+1}=n+1)$ retrieved as described in Section 3.5, we can have an initial FSTSP solution composed by the set of missions $MI=(M_1,M_2,\dots,M_{n+1})$ with $M_i=(s_{i-1},s_i), i=1,2,\dots,n+1$. The idea is to repeatedly apply the branch and bound method presented in Section 3 on chunks of the current solution. The aim is to produce improved incumbent solutions, that are then used as reference solutions in the subsequent iterations. The rationales behind this general idea are:

- the order provided by a TSP high-quality solution is intuitively a good starting point from which to remove customers from the truck tour and assigning them to the drone [1];
- reoptimizing short chunks of a solution with about 10–12 customers is normally enough to cover a few drone sorties, while computation times remain substantially short (see Section 5.1);
- iteratively and repeatedly locally reoptimizing solutions chunks of different length should create a global optimization after a few applications.

Formally, the heuristic algorithm we propose - and we will refer to it as *HeuBB* - works based on parameters a and b, that control the length of the subproblems solved to optimality. The overall method can be summarized as follows:

- Step 1: A TSP is solved on the customers of the FSTSP instance, as described in Section 3.5, to have a starting reference solution;
- Step 2: We set $CurMis=1$, as the index of the next mission to analyze;
- Step 3: An integer r such that $a \le r \le b$ is picked at random;
- Step 4: $NextMis$ is set to the index of a mission in such a way that $NextMis = \underset{d:CurMis \le d \le |MI|+1}{\arg\min} \left\{ \sum_{i=CurMis}^{d-1} (|C(M_i)|-1) \ge r \right\}$.
  If such an index does not exist, $NextMis$ is set to $|MI|+1$, which corresponds to the end of the current solution;
- Step 5: The subproblem characterized by the customer set $C = \bigcup_{i=CurMis}^{NextMis-1} C(M_i)$ is solved to optimality by the algorithm described in Section 3, where the starting depot is artificially given by the first customer of M_{CurMis} and the ending depot by the last node of $M_{NextMis-1}$. In case the artificial starting depot is not 0, eventual drone sortie starting from it will require to pay the launching time σ_L. The optimized solution MI becomes the new reference solution, and in case the total number of its missions $|MI|$ is changed, the index $NextMis$ is updated consequently;
- Step 6: If $NextMis \ne |MI|+1$ then $CurMis = NextMis$ and goto Step 3;
- Step 7: If the exit condition is not met, go to Step 2.

In our implementation the exit condition is given by a maximum computation time.

5. Experimental results

The algorithms have been implemented in ANSI C. An Intel Core i3-2100 CPU computer, with 3.10 GHz and 8.00 GB of RAM, has been used for the experiments. The LKH solver described in [19] has been used to solve the Traveling Salesman Problem.

Results for several instance sets previously adopted in the literature are reported, and a comparison with previously appeared method is provided. We first test in Section 5.1 the exact BB algorithm on small instances. Section 5.2 is instead devoted to the heuristic method HeuBB, which is tested on medium and large size instances. Detailed results for all the experiments are available, together with the instances, upon request to the authors.

5.1. Results of the exact algorithm BB

This section is devoted to the assessment of the exact algorithm described in Section 3. The method is compared with some exact methods based on the solution of MILP models available from the

Table 1
Results on 180 instances with 10 customers from Murray and Chu [22] and Dell'Amico et al. [13].

Instances			MILPs [13]	BB		
E	Depot Position	Nr. of Inst.	Sec Total	Sec Opt	Sec Total	Tree Nodes
20	a	9	179.2	0.6	5.9	206533.22
20	b	9	29.7	0.5	3.8	120324.67
20	c	9	4.6	0.3	2.4	67421.78
20	d	9	7.8	0.7	3.2	114873.89
Average			55.3	0.5	3.8	127288.39
40	a	9	597.7	0.5	6.0	125675.67
40	b	9	167.0	0.3	3.7	63280.44
40	c	9	61.5	0.4	3.9	87140.22
40	d	9	73.6	1.6	5.2	145142.33
Average			225.0	0.7	4.7	105309.67
60	a	9	345.3	0.4	4.2	69577.11
60	b	9	120.8	0.0	1.9	77084.33
60	c	9	133.4	0.0	1.4	42874.33
60	d	9	229.5	0.0	1.7	67777.67
Average			207.3	0.1	2.3	64328.36
80	a	9	331.2	0.4	4.2	69585.89
80	b	9	128.1	0.3	2.5	34679.33
80	c	9	87.4	0.3	2.8	50937.56
80	d	9	251.1	1.2	4.1	94775.11
Average			199.5	0.6	3.4	62494.47
100	a	9	371.5	0.4	4.2	69585.89
100	b	9	120.8	0.3	2.5	34679.33
100	c	9	87.1	0.3	2.8	50937.56
100	d	9	201.1	1.2	4.1	94775.11
Average			195.1	0.6	3.4	62494.47

literature, and later tested on increasingly larger instances, originally proposed for a similar problem, in order to take the method to its limits.

The first set of experiments is carried out on the 36 randomly generated benchmark setups with 10 customers proposed in [22] (details on the instances can be found in this paper). The customers are positioned in a 4 x 4 square, the position of the depot is of interest for our study: according to [13], position 'a' indicates that the depot is near the center of gravity of the customers; positions 'b', 'c' and 'd' have instead the following (x, y) coordinates, respectively: (4.0,2.7), (4.0,0.0), and (4,−2.7). The endurance E was set either to 20 or 40 in [22]. In [13] the set has been enlarged considering the extra values 60, 80 and 100, leading to a total of 180 instances. An optimal solution is known for each of the instances. The results for this set of instances are provided in Table 1, where for each value of E the results are grouped by the four possible positions of the depot. For the statistics from [13], we consider for each instance the average computation time over 9 instances for the fastest method reported in the paper, representing the best results reported so far in the literature (up to our knowledge). For the BB method we detail the average time to retrieve the optimal solution, the average time to prove its optimality and the average number of search tree nodes expanded.

The results of Table 1 mainly suggest that the novel *BB* is very consistent in its results, being only marginally influenced by the characteristics of the instances, and always providing very short computation times: the optimal solution is normally retrieved on average within the first second and confirmed soon after. The number of search-tree nodes visited by the method varies proportionally to the computation times, and shows a very high rate of iterations per second. In terms of relative performance with respect to the methods proposed in [13], *BB* appears substantially faster, often providing computation times shorter by orders of magnitude. Another interesting property that it is possible to observe by the experiments, is the substantial absence of correlation between the endurance E and the computation times of *BB*. Such a correlation is known to exist for MILP-based approaches (see [13,26,27]). This

Table 2
Results on 708 instances with 9, 14 and 19 customers from Poikonen et al. [25].

Instances			BB (max 3600s)			
$\|C\|$	Drone Speed Factor	E	Solved Intances	Sec Best	Sec Total	Tree Nodes
9	2	20	100/100	0.1	0.4	17281.9
9	2	30	100/100	0.0	0.1	3069.8
9	3	20	100/100	0.0	0.0	5272.2
9	3	30	100/100	0.0	0.0	1155.7
14	2	20	52/52	43.1	78.3	838965.4
14	2	30	52/52	8.3	14.8	87130.4
14	3	20	52/52	8.6	19.9	127727.4
14	3	30	52/52	3.3	7.6	38248.4
19	2	20	2/25	345.1	2715.3	2294581.5
19	2	30	5/25	688.3	2676.0	2233685.2
19	3	20	10/25	630.0	1289.5	1288558.2
19	3	30	14/25	525.3	1574.3	1261888.3

property makes therefore *BB* suitable for applications where drones are equipped with long-lasting batteries.

A second experiment is carried out on 177 benchmark setups with 9, 14 and 19 customers originally proposed for the variation of the TSP-D tackled in [25]. The customers with index $5i + 1$, with $0 \leq i \leq n/5 - 1$ are not drone-eligible. We considered $\sigma_L = \sigma_R = 0$, the values 2 and 3 for a parameter regulating the ratio between the speed of the drone and the speed of the truck, and values of endurance E of 20 and 30, in order to have some variations, for a total of 708 instances. The results are reported in Table 2, where a maximum computation time of 3600 s is considered. Each row reports the averages over 100, 52 and 25 instances, for 9, 14 and 19 customers, respectively. The columns report the characteristics of the instances, the number of solved instances, the CPU time to find the best solution, the average running time and the average number of search tree nodes expanded.

A direct comparison between BB, the exact branch and bound method proposed in [25] and the approached proposed in [13] is not possible since the objective function is different (the one of TSP-D appears to produce more challenging instances). As a reference, the exact solver of [25] was taking 77.8 s to certify optimality for some instances with 9 customers on a standard computer, while no result for the exact solver is reported for larger instances. On the same version of TSP-D, the best of the approaches discussed in [13] required 3.5 s for the same instances with 9 customers and 621.3 s for the same instances with 14 customers. On the set of TSP-D instances proposed in [25], the newly-presented MILP-based approaches discussed in [27] required in the best case 1.6, 273.5 and 1836.1 s to solve some of the instances with 9, 14 and 19 customers, respectively (on a computer substantially faster than the one we adopted). These indications, together with the results of BB presented in Table 2, suggest that the method we propose is competitive, and often better better than previously published approaches. The methods discussed in [26] appear to have similar performance to BB on the instances with 9 customers, and to be superior on instances with 19 customers. It would be interesting to carry out a comparison on the instances with 14 customers, but no result is presented in [26] for this size, since the focus there is on large instances up to 39 customers (out of our applicability domain).

Table 2 suggests that the method we propose is very effective on small instances, while its scalability on large instances is questionable: although still effective on the instances with 14 customers, the increase in computation times is remarkable, and as a consequence only a few of the instances with 19 customers are closed within the time limit. Indeed, this set of instances appear easier than the one used for Table 1, when 10 customers are con-

Table 3
Results by depot position on 240 instances with 20 customers from Murray and Chu [22] with $|C| = 20$.

Instances				TSP-ep [1]		HeuBB		HeuBB
				$a = 7$, $b = 10$		$a = 9$, $b = 13$		
E	Dep. Pos.	Nr. of Inst.	Gap %	Sec	Gap %	Sec	Gap %	Sec
20	centered	40	0.43	0.0	0.60	0.2	0.00	77.9
20	edge	40	0.56	0.0	0.36	0.2	0.01	85.9
20	origin	40	0.96	0.0	0.89	0.2	0.28	117.9
40	centered	40	−2.13	0.0	−0.22	0.3	−2.20	68.1
40	edge	40	−1.01	0.0	1.31	0.2	0.21	14.6
40	origin	40	−0.57	0.0	1.49	0.3	−0.28	66.1

sidered. Another observation is about the impact of the characteristics of the instances on the solving times and on the number of nodes visited: instances with higher speed for the drone and longer endurance appear to be easier to solve by the method we propose. This might depend on the presence of a few dominant missions that characterize good solutions, making the others clearly suboptimal.

5.2. Results of the heuristic algorithm HeuBB

In this section we focus our attention on HeuBB, the heuristic algorithm described in Section 4. The experiments will be on medium/large size instances previously adopted in the literature for the FSTSP or for similar problems. Since, up to our knowledge, no other heuristic is available in the literature for the FSTSP we consider, our method is compared with the adaptation of the heuristic TSP-ep originally proposed for a variation of D-TSP in [1], and proving very effective in that context. The method starts from a TSP solution then applies one-point moves (repositioning of a customer) and two-point moves (classic 2-opt moves) to generate further solutions. Each of the TSP solutions generated is evaluated in terms of FSTSP by splitting the customers between the truck and the drone via a very efficient exact partitioning method. The one-point and two-points operators are applied to every new improved solution eventually found, until no further improvements are possible, or a maximum computation time has elapsed. Full details can be found in [1]). We have implemented the method in ANSI C, adapted to the FSTSP. Also in this context the LKH solver described in [19] has been used to provide the initial TSP solution. For all the tests, a maximum computation times of 3600 s is considered for TSP-ep (although it often concludes the computation earlier, and this happens consistently for small instances), while 720 s are considered for the different versions of HeuBB. HeuBB is given a shorter time because it has been observed that it rarely improves after the given 720 s, independently of the characteristics of the instances.

The first set of instances considered is formed by 120 benchmarks setup with 20 customers originally proposed in [22] for the Parallel Drone Scheduling Traveling Salesman Problem (PDSTSP), a problem in which a fleet of drones can serve a set of customers only departing from the depot, the remaining customers are served by a truck [12]. These instances could be easily adapted to the FSTSP, having the very same characteristics. Two values of endurance are considered: 20 and 40 time units, for a total of 240 instances. The same instances had already been considered in [13], where upper bounds are provided. The results are summarized in Table 3, where a percentage gap with respect to the best results from [13] is reported for the adaptation to FSTSP of the TSP-ep heuristic from [1] and HeuBB, together with statistics on the time when the best solutions were retried. Along the paper, percentage gap are calculated as $100 \cdot (UB_H - UB_B)/UB_B$, where UB_H is the upper bound provided by the algorithm under evaluation and UB_B is the upper bound provided by the baseline algorithm (the best results from [13] in this case). Negative values indicate improvements over the baseline. Two different settings of the HeuBB are considered: one with $a = 7$ and $b = 10$ is designed to be fast (BB is run on short subproblem chunks), while the one with $a = 9$ and $b = 13$ is designed to be more precise (BB is run on longer chunks). The parameter settings leading to these variations have been found after some preliminary tests, and are reported in the table. The results are grouped by endurance, and by the different positions of the depot considered: 'centered' means the depot is centrally located with respect to the customers; 'edge' means the depot is at the edge of the squared area induced by the positions of the customers; 'origin' means the depot is positioned at the origin of the axes of the whole area considered.

A few conclusions can be drawn from the results reported in Table 3. The methods TSP-ep and HeuBB with fast settings retrieve their best solutions in almost negligible time, with comparable quality on the instances with $E = 20$, and with results in favour of TSP-ep when $E = 40$ (which is able to sensibly improve some of the best known upper bounds from [13]). The method HeuBB with more precise settings is able to improve the results of TSP-ep for $E = 20$ and to partially close the gap with TSP-ep for $E = 40$. The computation time to retrieve the best solution increases substantially, however. In general, TSP-ep seems to be preferable on these medium size instances.

A second set of experiments is run on some of the instances proposed in [21] for the PDSTSP, here re-considered as FSTSP instances. These instances have between 48 and 229 customers, with several percentage of drone eligible nodes, with several drone speeds, and with two different depot locations (we refer the reader to [21] for a detailed description of the settings). A total of 60 instances is considered. The results are presented in Table 4. The first columns define each instance based on the characteristics and on the name of the benchmark setup (the last digit of the name indicates the number of customers). Then, similarly to what already seen in Table 3, the results for the different methods are summarized, with the difference that now for each instance we also report the cost of the best solution retrieved be each method and the results of TSP-ep are the baseline for the calculation of the percentage gaps. All the available results of the Random Restart Local Search (RRLS) matheuristic discussed in [14] for this set of instances, also appear in the table. It is important to point out that these results have been obtained with a maximum computation time of 720 s per run on a computer (Intel Xeon E5-2620 v4 running at 2.10 GHz) approximately 15% faster than our reference machine[1], so the comparison is slightly biased in favour of RRLS.

Table 4 suggests that on these instances HeuBB outpeforms TSP-ep and RRLS often obtaining better results already when its fast version is considered, in a fraction of the time. On the other hand, when the more precise version is taken into account, the results improve even further (still with computation times shorter than

[1] Source http://www.cpubenchmark.net.

Table 4
Results on 60 instances with 48 to 229 customers from Mbiadou Saleu et al. [21].

Instances				TSP-ep [1]		RRLS [14]			HeuBB (720s)			HeuBB (720s)		
Name	a	b	c	(max 3600s)		(720s)			$a = 7$, $b = 10$			$a = 9$, $b = 13$		
				UB	Sec	UB	Gap %	Sec	UB	Gap %	Sec	UB	Gap %	Sec
att48	20	2	1	39348.00	1.8	38662.0	−1.74	78.9	40082.00	1.87	0.0	40082.00	1.87	22.3
berlin52	20	2	1	9385.00	5.1	9350.0	−0.37	150.9	9385.00	0.00	0.0	9350.00	−0.37	3.3
eil101	20	2	1	755.00	260.0	755.0	0.00	191.2	759.00	0.53	0.6	755.00	0.00	248.3
gr120	20	2	1	1818.00	1148.8	1812.0	−0.33	411.8	1825.81	0.43	0.4	1770.00	−2.64	534.8
pr152	20	2	1	84304.00	2588.9	84623.0	0.38	225.9	83172.00	−1.34	1.7	82734.00	−1.86	19.8
gr229	20	2	1	1918.70	3342.1	-	-	-	1893.57	−1.31	0.8	1889.54	−1.52	130.2
att48	40	2	1	36014.00	5.4	33348.0	−7.40	58.4	34832.00	−3.28	0.3	32688.00	−9.24	52.1
berlin52	40	2	1	8480.00	12.9	8410.0	−0.83	45.7	8638.75	1.87	0.1	8455.00	−0.29	7.2
eil101	40	2	1	690.00	354.6	687.0	−0.43	14.3	701.00	1.59	0.5	694.29	0.62	18.6
gr120	40	2	1	1698.00	1071.9	1718.0	1.18	46.7	1676.35	−1.28	0.9	1649.16	−2.88	60.6
pr152	40	2	1	78512.00	3532.0	82586.0	5.19	108.2	78827.23	0.40	5.5	77740.80	−0.98	41.0
gr229	40	2	1	1853.81	2679.4	-	-	-	1807.02	−2.52	2.5	1794.68	−3.19	157.4
att48	60	2	1	34030.00	7.2	34190.0	0.47	223.0	34030.00	0.00	0.1	31264.00	−8.13	124.9
berlin52	60	2	1	7670.00	13.2	7943.7	3.57	27.3	7842.69	2.25	1.0	7670.00	0.00	183.6
eil101	60	2	1	615.37	472.0	599.3	−2.61	168.4	644.64	4.76	0.6	615.37	0.00	218.7
gr120	60	2	1	1489.48	1957.9	1582.6	6.25	499.5	1571.98	5.54	2.4	1489.48	0.00	648.7
pr152	60	2	1	74376.88	3217.4	79294.0	6.61	228.5	77978.92	4.84	1.7	75873.92	2.01	27.0
gr229	60	2	1	1733.99	3534.8	-	-	-	1703.32	−1.77	1.1	1698.67	−2.04	105.4
att48	80	1	1	34710.36	5.5	34011.6	−2.01	131.4	35874.35	3.35	2.5	34136.42	−1.65	27.0
berlin52	80	1	1	7193.97	12.1	7339.0	2.02	128.3	7418.84	3.13	0.4	7331.02	1.91	37.3
eil101	80	1	1	626.68	418.8	605.0	−3.46	38.8	599.73	−4.30	0.6	598.80	−4.45	50.3
gr120	80	1	1	1478.76	1006.2	1503.5	1.68	491.5	1531.37	3.56	0.5	1437.77	−2.77	638.3
pr152	80	1	1	75394.58	3119.7	77622.1	2.95	265.9	74759.07	−0.84	2.5	73413.35	−2.63	165.3
gr229	80	1	1	1728.34	3306.8	-	-	-	1728.34	0.00	1.8	1728.34	0.00	223.7
att48	80	2	1	31711.95	8.8	32666.0	3.01	35.9	31127.55	−1.84	1.3	30948.37	−2.41	46.7
berlin52	80	2	1	6735.55	20.0	7151.3	6.17	55.3	6915.19	2.67	1.5	6784.75	0.73	61.5
eil101	80	2	1	544.41	717.4	547.5	0.56	268.3	556.92	2.30	1.5	539.45	−0.91	348.5
gr120	80	2	1	1284.84	1695.8	1400.0	8.96	42.0	1325.18	3.14	2.2	1240.43	−3.46	54.5
pr152	80	2	1	72556.08	2988.0	77062.2	6.21	13.9	69291.02	−4.50	2.2	68680.32	−5.34	51.4
gr229	80	2	1	1693.42	3404.1	-	-	-	1626.38	−3.96	3.2	1608.87	−4.99	117.9
att48	80	2	2	30164.00	12.4	32483.6	7.69	10.6	30381.10	0.72	2.3	29849.80	−1.04	13.8
berlin52	80	2	2	7260.00	10.6	7695.6	6.00	32.5	7708.73	6.18	0.1	6862.08	−5.48	42.2
eil101	80	2	2	563.58	748.2	525.0	−6.85	267.6	574.56	1.95	0.8	518.06	−8.08	347.5
gr120	80	2	2	1505.27	1904.9	1404.6	−6.69	39.2	1472.01	−2.21	5.6	1111.72	−26.14	50.9
pr152	80	2	2	71962.08	2823.5	76158.0	5.83	157.9	70820.27	−1.59	1.7	67971.46	−5.55	20.8
gr229	80	2	2	1570.60	3136.3	-	-	-	1570.60	0.00	1.6	1615.39	2.85	289.9
att48	80	3	1	30164.00	15.4	32680.0	8.34	28.6	32053.38	6.26	1.7	29382.40	−2.59	37.2
berlin52	80	3	1	6616.35	17.3	7158.8	8.20	6.1	6813.09	2.97	0.1	6054.92	−8.49	56.0
eil101	80	3	1	531.09	663.4	523.0	−1.52	22.0	532.81	0.32	0.7	519.00	−2.28	28.6
gr120	80	3	1	1262.58	1904.3	1400.0	10.88	22.6	1296.61	2.70	1.1	1106.08	−12.40	29.4
pr152	80	3	1	71728.00	3022.8	76980.0	7.32	69.8	68422.67	−4.61	1.9	66158.48	−7.76	45.0
gr229	80	3	1	1692.49	3599.6	-	-	-	1593.71	−5.84	3.9	1560.60	−7.79	86.4
att48	80	4	1	31782.00	9.2	33350.0	4.93	20.7	32880.00	3.45	0.2	30120.00	−5.23	26.9
berlin52	80	4	1	6725.00	16.0	7180.0	6.77	11.1	6616.21	−1.62	0.1	6045.00	−10.11	14.5
eil101	80	4	1	531.00	604.3	517.2	−2.60	92.4	535.00	0.75	0.5	503.43	−5.19	120.1
gr120	80	4	1	1174.29	2448.6	1443.5	22.92	90.4	1237.06	5.35	1.0	1122.00	−4.45	10.6
pr152	80	4	1	71922.00	3286.4	77246.0	7.40	95.4	68464.16	−4.81	1.9	64917.46	−9.74	60.0
gr229	80	4	1	1692.49	3555.5	-	-	-	1585.89	−6.30	2.7	1556.00	−8.06	647.7
att48	80	5	1	31596.00	8.2	33024.0	4.52	8.5	32578.00	3.11	0.1	27174.68	−13.99	11.1
berlin52	80	5	1	6600.00	17.2	7245.0	9.77	16.3	6589.97	−0.15	0.2	6176.57	−6.42	21.2
eil101	80	5	1	527.00	658.4	596.9	13.27	380.6	533.71	1.27	1.7	490.24	−6.98	494.3
gr120	80	5	1	1170.00	2486.1	1294.6	10.65	5.9	1250.00	6.84	0.9	1083.87	−7.36	7.7
pr152	80	5	1	71867.71	3024.3	76756.0	6.80	85.3	67507.25	−6.07	2.9	66397.93	−7.61	56.7
gr229	80	5	1	1692.49	3313.3	-	-	-	1575.06	−6.94	2.9	1536.90	−9.19	136.4
att48	100	2	1	27174.68	9.1	27905.8	2.69	96.8	26124.76	−3.86	0.8	26053.32	−4.13	16.3
berlin52	100	2	1	5830.34	17.1	6610.0	13.37	254.7	5962.40	2.27	0.4	5851.55	0.36	21.5
eil101	100	2	1	490.24	573.5	467.3	−4.67	13.9	465.98	−4.95	1.0	452.05	−7.79	18.1
gr120	100	2	1	1209.66	1917.6	1334.1	10.29	172.9	1206.67	−0.25	1.3	1166.37	−3.58	53.0
pr152	100	2	1	71774.02	2888.4	76586.5	6.70	189.0	68452.58	−4.63	3.0	67468.59	−6.00	70.3
gr229	100	2	1	1563.70	3422.7	-	-	-	1422.16	−9.05	4.4	1387.93	−11.24	99.9
Average					1450.4		3.76	122.8		−0.06	1.5		−4.43	122.7

TSP-ep and comparable with RRLS). This shows that by playing with parameters a and b it is possible to modulate the trade-off between performance and computation times. Looking at the results for single instances, it can be observed how in several cases TSP-ep finds the best solution, but apparently it has a worst exploration of the search space with respect to HeuBB, with the latter appearing more robust already in the less accurate but faster version. On the other hand, RRLS seems to be the less robust method, sometimes finding best-known solutions, sometimes performing extremely bad. This is particularly evident for the instances with many drone-eligible customers, and on large instances in general. HeuBB appears to definitely perform better on larger instances, providing substantially and systematically better results than TSP-ep and RRLS as the number of customers increases. All the heuris-

Table 5
Results on 24 instances with 50 to 199 customers from de Freitas and Penna [10].

Instances	TSP-ep [1] (max 3600s)		RRLS [14] (720s)			HeuBB (720s) $a = 7$, $b = 10$			HeuBB (720s) $a = 9$, $b = 13$		
	UB	Sec	UB	Gap %	Sec	UB	Gap %	Sec	UB	Gap %	Sec
berlin52	199.75	6.5	202.85	1.55	159.0	203.66	1.96	0.2	199.75	0.00	2.9
bier127	3502.12	702.9	3509.99	0.22	117.9	3506.58	0.13	1.2	3505.40	0.09	569.6
ch130	183.52	335.4	187.78	2.32	23.9	185.51	1.09	0.3	184.52	0.54	387.4
d198	461.86	2910.4	-	-	-	462.12	0.06	1.5	461.23	−0.14	482.6
eil51	13.45	0.0	13.45	0.00	1.0	13.45	0.00	0.0	13.45	0.00	0.0
eil76	16.90	0.0	16.90	0.00	3.4	16.90	0.00	0.0	16.90	0.00	0.0
kroA100	540.81	410.6	582.03	7.62	40.0	560.90	3.71	0.7	540.81	0.00	225.4
kroA150	722.12	704.5	716.65	−0.76	166.4	731.51	1.30	4.9	717.44	−0.65	546.0
kroA200	835.73	3122.7	-	-	-	850.63	1.78	2.5	832.10	−0.43	139.8
kroB150	706.00	688.7	823.65	16.66	24.8	708.83	0.40	2.7	694.06	−1.69	670.2
kroB200	806.43	3398.8	-	-	-	825.72	2.39	5.1	813.49	0.87	153.0
kroC100	556.38	468.9	586.71	5.45	250.2	579.11	4.09	1.6	564.28	1.42	519.2
kroD100	565.95	409.5	591.13	4.45	424.0	590.54	4.34	1.0	560.14	−1.03	674.2
kroE100	590.70	456.8	618.55	4.71	290.5	599.91	1.56	2.5	589.02	−0.28	211.0
lin105	387.62	339.7	396.31	2.24	219.0	401.24	3.51	9.5	387.62	0.00	217.7
pr107	1054.37	668.4	1063.96	0.91	251.9	1073.66	1.83	3.6	1044.11	−0.97	249.0
pr124	1620.06	165.8	1620.03	0.00	330.5	1620.23	0.01	0.8	1620.23	0.01	8.7
pr136	2618.70	667.3	2661.21	1.62	155.1	2569.83	−1.87	2.5	2525.62	−3.55	234.6
pr144	1676.75	54.4	1688.75	0.72	24.4	1676.75	0.00	1.0	1676.75	0.00	28.8
pr152	1992.44	519.8	2123.95	6.60	24.7	1988.69	−0.19	2.1	1981.94	−0.53	10.2
rat99	37.45	0.0	37.45	0.00	12.1	37.45	0.00	0.0	37.45	0.00	0.0
rat195	71.50	0.0	-	-	-	71.50	0.00	0.0	71.50	0.00	0.0
rd100	229.15	175.2	235.37	2.71	37.4	235.62	2.82	2.2	221.53	−3.33	163.0
st70	21.00	7.9	21.00	0.00	3.5	21.00	0.00	0.0	21.00	0.00	0.0
Average		675.6		2.85	128.0		1.21	1.9		−0.40	228.9

Table 6
Results on 60 instances with 50 (B^*, C^*, D^*) or 100 (E^*, F^*, G^*) customers from Ha et al. [17].

Inst.	TSP-ep [1] (max 3600s)		HeuBB (720s) $a = 9$, $b = 13$			Inst.	TSP-ep [1] (max 3600s)		HeuBB (720s) $a = 9$, $b = 13$		
	UB	Sec	UB	Gap %	Sec	UB	Sec	UB	Gap %	Sec	
B1	121.48	3.5	121.48	0.00	0.5	E1	188.32	182.5	188.28	−0.03	658.5
B2	119.54	1.8	119.54	0.00	3.4	E2	189.56	221.4	189.56	0.00	25.1
B3	116.66	3.6	116.66	0.00	17.2	E3	189.05	71.4	189.05	0.00	82.2
B4	125.47	6.2	125.47	0.00	43.3	E4	188.38	106.9	188.24	−0.08	16.4
B5	118.77	2.7	118.77	0.00	1.5	E5	189.05	288.9	189.47	0.22	199.4
B6	118.03	5.3	116.99	−0.88	408.6	E6	190.05	178.0	190.05	0.00	31.2
B7	119.29	9.0	119.29	0.00	2.8	E7	191.38	148.0	191.09	−0.15	23.4
B8	119.14	4.1	119.21	0.06	22.5	E8	190.12	141.9	190.45	0.18	662.3
B9	121.34	6.0	121.34	0.00	4.6	E9	190.78	141.1	190.43	−0.18	56.4
B10	120.15	3.6	120.34	0.16	11.7	E10	189.45	106.4	189.45	0.00	33.1
C1	218.28	6.4	218.28	0.00	14.2	F1	335.84	351.8	330.50	−1.59	44.7
C2	210.86	3.8	210.82	−0.02	53.2	F2	315.30	388.5	316.31	0.32	142.7
C3	213.13	2.7	213.13	0.00	2.2	F3	328.36	422.0	328.31	−0.01	473.9
C4	220.11	4.4	220.11	0.00	4.4	F4	324.05	247.8	322.06	−0.61	47.6
C5	235.98	2.7	235.98	0.00	56.1	F5	331.65	426.1	331.65	0.00	27.9
C6	240.61	0.9	240.61	0.00	3.5	F6	300.52	252.2	300.52	0.00	261.1
C7	226.69	3.5	224.22	−1.09	11.5	F7	314.45	338.3	314.90	0.14	57.1
C8	238.58	7.0	238.58	0.00	26.8	F8	331.93	426.6	330.71	−0.37	691.8
C9	236.14	4.6	235.99	−0.07	17.1	F9	335.12	358.4	334.31	−0.24	299.2
C10	228.77	3.5	229.77	0.44	39.7	F10	318.78	407.6	318.78	0.00	19.0
D1	322.32	2.7	322.32	0.00	2.7	G1	422.17	317.2	419.29	−0.68	220.0
D2	319.12	3.5	316.59	−0.79	15.7	G2	399.65	376.4	400.94	0.32	64.1
D3	303.23	5.2	296.70	−2.15	6.6	G3	437.02	293.0	434.74	−0.52	31.9
D4	329.21	3.6	329.21	0.00	11.2	G4	449.30	184.4	442.99	−1.40	375.0
D5	321.84	5.2	321.84	0.00	8.4	G5	428.36	352.5	429.58	0.28	13.2
D6	316.00	4.4	316.00	0.00	13.4	G6	434.47	351.4	434.47	0.00	35.4
D7	317.37	7.8	317.37	0.00	10.4	G7	411.75	282.7	411.75	0.00	15.6
D8	296.58	6.1	296.58	0.00	7.9	G8	418.35	412.9	418.35	0.00	18.0
D9	336.07	4.4	336.21	0.04	143.9	G9	450.57	422.4	451.74	0.26	114.0
D10	306.89	6.1	302.92	−1.29	30.8	G10	449.80	426.9	449.80	0.00	59.7
Agerage			4.5	−0.19	33.2			287.5		−0.14	160.0

tics do not seem to be very much affected by the other parameters of the instances.

A third set of instances considered in these experiments is the one proposed in [10] (where details can be found), which is composed of 24 instances with a number of customers ranging between 50 and 199 (the number in the names of the instances is the number of customers augmented by 1). A direct comparison with the results reported in [10] is unfortunately not possible[2]. Re-

[2] We infer the problem solved is slightly different – although the differences are not detailed in [10] – because according to our model, some of the results they report are super-optimal.

sults are summarized in Table 5, and the meaning of the column is analogous to that of Table 4. Also in this case, all the results available for the method RRLS from [14] (again obtained on the slightly faster machine) are presented in the table.

Table 5 confirm all the observation reported on the previous experiments, with the exception that on these instances TSP-ep appears more competitive than before and on the other hands RRLS seems to perform poorly. Only the more precise version of HeuBB is able to improve the average results of TSP-ep (however with considerably shorter computation times).

A forth set of instances considered in these experiments is the one proposed in [17] (where details can be found) and later used again in [18], which is composed of 60 instances with 50 or 100 customers. In the settings considered here the drone endurance E is $13.\overline{3}$ and handling times σ_L and σ_R are both $0.\overline{6}$. A direct comparison with the results reported in [18] is again not possible[3]. Results are summarized in Table 6, and the meaning of the column is analogous to that of Table 4. For this experiment, results are not available for RRLS, which was however dominated in the previous experiments. Moreover, only the more precise version of HeuBB is considered.

Table 6 confirm all the observation reported on the previous experiments. The main finding here is that TSP-ep tends to be slower than HeuBB on the larger instances of this set and that HeuBB seems – on top of being slightly better – to be more robust of the counterpart, since it never have particularly large positive gaps, but it sometimes produces substantial improvements above 1.5. On the larger instances TSP-ep appears also to be slower than HeuBB.

As a final note on the results reported for the heuristic algorithms, it is worth mentioning that a better validation of the methods would have been possible by comparing the costs of the solutions retrieved against valid lower bounds. Unfortunately, viable lower bounds are currently not available for large instances, so only comparative considerations among different techniques like those we reported are possible.

6. Conclusions

A branch and bound method has been presented for the Flying Sidekick Traveling Salesman Problem, and it has been shown how it can be used within a heuristic routine. The methods are designed to exploit the characteristics of the problem and to provide effective and efficient solving methods. Experimental results corroborate this conclusion, showing improved state-of-the-art results for the heuristic algorithm.

Future work might involve the use of other exact solvers within the heuristic framework we propose, in particular the method discussed in Roberti and Ruthmair [26]. It would be interesting to investigate wether considering larger solution chunks for re-optimization could lead to better solution, even if this would probably imply significantly longer converging times. Another research line in need of a deeper investigation is about the development of effective lower bounds for large instances. They would allow to better evaluate the heuristics algorithms currently available and at the same time lead to exact algorithms for such instances.

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.

[3] From [18] it is possible to infer that the problem is different.

Acknowledgement

The authors are grateful to Prof. Hoàng Minh Hà for the useful discussions about the instances from Ha et al. [17].

References

[1] Agatz N, Bouman P, Schmidt M. Optimization approaches for the traveling salesman problem with drone. Transp Sci 2018;52(4):965–81.
[2] Aho A, Ullman J, Hopcroft J. Data structures and algorithms. USA: Pearson; 1983. ISBN 0201000237
[3] Applegate DL, Bixby RE, Chvatal V, Cook WJ. The traveling salesman problem: a computational study (Princeton Series in Applied Mathematics). USA: Princeton University Press; 2007. ISBN 0691129932
[4] Boccia M, Masone A, Sforza A, Sterle C. A column-and-row generation approach for the flying sidekick travelling salesman problem. Transp Res Part C 2021;124:102913.
[5] Boeck KD, Decouttere C, Vandaele N. Vaccine distribution chains in low- and middle-income countries: a literature review. Omega 2020;97:102097.
[6] Bouman P, Agatz N, Schmidt M. Dynamic programming approaches for the traveling salesman problem with drone. Networks 2018;72(4):528–42.
[7] Burkard R, Dell'Amico M, Martello S. Assignment Problems. Soc Ind Appl Math; 2012.
[8] Carlsson J, Song S. Coordinated logistics with a truck and a drone. Manage Sci 2018;64(9):3971–4470.
[9] Chung SH, Sah B, Lee J. Optimization for drone and drone-truck combined operations: a review of the state of the art and future directions. Comput Oper Res 2020;123:105004.
[10] de Freitas JC, Penna PHV. A variable neighborhood search for flying sidekick traveling salesman problem. Int Trans Oper Res 2020;27:267–90.
[11] Dell'Amico M, Montemanni R, Novellani S. Drone-assisted deliveries: new formulations for the flying sidekick traveling salesman problem. Optim Lett 2019.
[12] Dell'Amico M, Montemanni R, Novellani S. Matheuristic algorithms for the parallel drone scheduling traveling salesman problem. Ann Oper Res 2020b;289:211–26.
[13] Dell'Amico M., Montemanni R., Novellani S.. Models and algorithms for the flying sidekick traveling salesman problem. arXiv preprint arXiv: 1910025592020a;.
[14] Dell'Amico M., Montemanni R., Novellani S.. A random restart random restart local search matheuristic for the flying sidekick traveling salesman problem. In: Proceedings of the 8th international conference on industrial engineering and applications (ICIEA), ACM.
[15] Dönmez Z, Kara BY, Özlem Karsu, da Gama FS. Humanitarian facility location under uncertainty: critical review and future prospects. Omega 2021;102:102393.
[16] El-Adle AM, Ghoniem A, Haouari M. Parcel delivery by vehicle and drone. J Oper Res Soc 2021;72(2):398–416.
[17] Ha QM, Deville Y, Pham QD, Hà MH. On the min-cost traveling salesman problem with drone. Transp Res Part C 2018;86:597–621.
[18] Ha QM, Deville Y, Pham QD, Hà MH. A hybrid genetic algorithm for the traveling salesman problem with drone. J Heuristics 2020;26:219–47.
[19] Helsgaun K. An effective implementation of the Lin-Kernighan traveling salesman heuristic. Eur J Oper Res 2000;126(1):106–30.
[20] Macrina G, Di Puglia Pugliese L, Guerriero F, Laporte G. Drone-aided routing: a literature review. Transp Res Part C 2020;120:102762.
[21] Mbiadou Saleu RG, Deroussi L, Feillet D, Grangeon N, Quilliot A. An iterative two-step heuristic for the parallel drone scheduling traveling salesman problem. Networks 2018;72(4):459–74.
[22] Murray CC, Chu AG. The flying sidekick traveling salesman problem: optimization of drone-assisted parcel delivery. Transp Res Part C 2015;54:86–109.
[23] Otto A, Agatz N, Campbell J, Golden B, Pesch E. Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: a survey. Networks 2018;72(4):411–58.
[24] Pei Z, Dai X, YilunYuan, Du R, Liu C. Managing price and fleet size for courier service with shared drones. Omega 2021;102482.
[25] Poikonen S, Golden B, Wasil EA. A branch-and-bound approach to the traveling salesman problem with a drone. INFORMS J Comput 2019;31(2):335–46.
[26] Roberti R, Ruthmair M. Exact methods for the traveling salesman problem with drone. Transp Sci 2021;55(2):315–35.
[27] Schermer D, Moeini M, Wendt O. A branch-and-cut approach and alternative formulations for the traveling salesman problem with drone. Networks 2020;76(2):164–86.
[28] Vásquez SA, Angulo G, Klapp MA. An exact solution method for the TSP with drone based on decomposition. Comput Oper Res 2021;127:105127.
[29] Viloria DR, Solano-Charris EL, Muñoz-Villamizar A, Montoya-Torres JR. Unmanned aerial vehicles/drones in vehicle routing problems: a literature review. Int Trans Oper Res 2021;28:1626–57.
[30] Yurek EE, Ozmutlu HC. A decomposition-based iterative optimization algorithm for traveling salesman problem with drone. Transp Res Part C 2018;91:249–62.