Development of Heuristic Approaches for Last-Mile Delivery TSP with a Truck and Multiple Drones

drones

MDPI

Article

Development of Heuristic Approaches for Last-Mile Delivery TSP with a Truck and Multiple Drones

Marco Rinaldi [1], Stefano Primatesta [1], Martin Bugaj [2], Ján Rostáš [2] and Giorgio Guglieri [1,*]

[1] Department of Mechanical and Aerospace Engineering, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy; marco_rinaldi@polito.it (M.R.); stefano.primatesta@polito.it (S.P.)
[2] Air Transport Department, University of Žilina, Univerzitná 8215/1, 010 26 Žilina, Slovakia; martin.bugaj@uniza.sk (M.B.); jan.rostas@uniza.sk (J.R.)
* Correspondence: giorgio.guglieri@polito.it

Abstract: Unmanned Aerial Vehicles (UAVs) are gaining momentum in many civil and military sectors. An example is represented by the logistics sector, where UAVs have been proven to be able to improve the efficiency of the process itself, as their cooperation with trucks can decrease the delivery time and reduce fuel consumption. In this paper, we first state a mathematical formulation of the Travelling Salesman Problem (TSP) applied to logistic routing, where a truck cooperates synchronously with multiple UAVs for parcel delivery. Then, we propose, implement, and compare different sub-optimal routing approaches to the formulated mFSTSP (multiple Flying Sidekick Travelling Salesman Problem) since the inherent combinatorial computational complexity of the problem makes it unattractable for commercial Mixed-Integer Linear Programming (MILP) solvers. A local search algorithm, two hybrid genetic algorithms that permutate feasible and infeasible solutions, and an alternative ad-hoc greedy method are evaluated in terms of the total delivery time of the output schedule. For the sake of the evaluation, the savings in terms of delivery time over the well-documented truck-only TSP solution are investigated for each proposed routing solution, and this is repeated for two different scenarios. Monte Carlo simulations corroborate the results.

Keywords: TSP; genetic algorithm; last-mile delivery; task scheduling; local search algorithm; truck and drones; UAV; heuristics; routing; urban air mobility; logistics; mFSTSP

Citation: Rinaldi, M.; Primatesta, S.; Bugaj, M.; Rostáš, J.; Guglieri, G. Development of Heuristic Approaches for Last-Mile Delivery TSP with a Truck and Multiple Drones. *Drones* **2023**, *7*, 407. https://doi.org/10.3390/drones7070407

Academic Editor: Diego González-Aguilera

Received: 5 May 2023
Revised: 13 June 2023
Accepted: 14 June 2023
Published: 21 June 2023

1. Introduction

In recent years, last-mile delivery has witnessed an enormous increase in quality demand in a continuously growing market, especially due to the rapid spread of online shopping platforms. Usually, e-commerce retailers are required to deliver a huge number of parcels within small time windows, to several customers and possibly without charging any delivery fees. These requirements are obviously conflicting and they shall be tackled with efficient management.

By definition, last-mile delivery refers to the final stage of the logistical process, when the package is transported from the transportation hub to the destination of the final client. It follows the main stages of the last-mile delivery process: (1) when a consumer places an order, it is entered into a digital system where it can be tracked; (2) at the transportation hub, where orders are waiting to be delivered, the logistic process starts; (3) orders are strategically assigned to delivery employees based on delivery addresses with the aim of optimizing the process itself; (4) orders are scanned before being loaded onto delivery trucks in order to monitor the parcel on both the customer and corporate sides; (5) orders are delivered successfully to their intended location, and the delivery status is updated after a proof of accurate shipment is requested.

The fundamental paradox is also that, despite the fact that customers expect quick and possibly free delivery services, the last-mile delivery process is the most expensive one

along the entire logistical chain. Last-mile shipping costs can represent 53% of the entire cost of a product, as reported in [1]. As a result, nowadays companies are pushing R&D initiatives to incorporate new technology into their last-mile delivery operations in order to outperform the competitors.

The deployment of autonomous vehicles is one of several trends that is most likely to have a remarkable impact on last-mile delivery in the future. The use of autonomous robots and Unmanned Aerial Vehicles (UAVs) is only one of several upcoming innovations that will have an impact on last-mile delivery.

An example of this new trend in logistics and, in particular, last-mile delivery is represented by "Prime Air", Amazon's drone delivery service, first introduced in 2013. The original idea was to employ UAVs to deliver packages, which would take off from warehouses and use GPS to navigate to the customer's location. The program's most recent announcement took place in 2019, when they unveiled the drone design that will be used in the "Prime Air" program. This drone can carry packages under 5 pounds and serve customers in less than 30 min, with a battery autonomy up to 15 miles. In order to boost safety that would not be assured just by communication systems for situational awareness, it also includes powerful artificial intelligence (AI) technology, as described in [2]. As a result, Amazon was given a certificate by the FAA allowing it to fly its aircraft in permitted airspace. Furthermore, the company claims that by using drones for deliveries, it will reach 50% of shipments with environmental zero-impact by 2030 [3].

Considering the logistics sector of the urban air mobility context, as highlighted in [4], UAVs employment has been demonstrated to be able to optimize the logistics process and induce the reduction in conventional resources' usage, i.e., manned ground vehicles, traffic, and CO_2 emissions. Drones have already been demonstrated to be able to bring a significant strategic advantage in terms of delivery; indeed, in traffic conditions, UAVs are generally faster than conventional ground delivery vehicles. Furthermore, UAVs are not subject to any route restriction, unless flight-restricted areas are present. Their flexibility represents a key factor to manage the delivery process optimally.

UAVs' adaptability is essential for managing the delivery time well, as this is one of the most crucial requirements of the process. Nevertheless, the payload capacity, the flight range, and the battery life, are the key technological constraints on the widespread use of UAVs, therefore their use is not without drawbacks. Furthermore, managing risks associated with devices' failures in operative conditions is still challenging. For the sake of completeness, it is worth noticing that there are still some barriers that limit the widespread adoption of drones in urban scenarios, mainly due to security and safety issues, social acceptance issues, and regulatory aspects. Refer to the works in [5–12] for a more detailed introduction to logistics and urban air mobility perspectives and challenges. For the sake of clarity, the addressed problem of this work considers a scenario where those barriers have been addressed properly, at least for the majority part.

The tandem truck-drone delivery method has been growingly studied over the last decade to be used in the last-mile delivery process in order to partially address the battery and payload capacity issues. Due to the complementarities between the truck's and the drone's characteristics in terms of speed, weight, payload capacity, and energy consumption, the tandem truck-drone solution is promising, as reported in [13].

The drone's flight range can be effectively extended by the truck transporting it, as it would otherwise only be able to fly within a range proportional to its endurance duration around the distribution center. This final situation would be crucial because moving depots to densely populated areas would be expensive for the businesses. Instead, in the case of the truck-drone tandem, the UAV can be carried close to clients by the truck and it can make deliveries while the vehicle is driving to another customer, resulting in an effective time management of the delivery process. Additionally, the orders can be assigned to either the truck or the drone according to their weight, location, and time due date, seeking for an optimization of the delivery process.

Comprehensive literature reviews of the applications of drones in the last-mile delivery domain are proposed in [14–16]. An example of an application is represented by the multiple Flying Sidekick Travelling Salesman Problem (mFSTSP), where a fleet of UAVs is used in combination with a truck that simultaneously performs other delivery tasks and can also serve as a platform for the launch and the recovery of the drones. A similar application is considered in this work.

The problem of scheduling parcel delivery tasks to a truck operating in combination with a fleet of UAVs is a variant of the Travelling Salesman Problem (TSP). In computer science, the TSP is still an open problem to be solved in a globally optimal way due to the intractability of its inherent combinatorial nature. A problem is said to be intractable if the computational complexity is superpolynomial. The TSP is defined as an NP (Nondeterministic Polynomial) problem, which means that a solution can be verified in polynomial time, but the optimal solution cannot be computed in polynomial time. However, it has not been proven yet that a polynomial resolution algorithm does not exist for these kinds of problems. In this case, scheduling parcel delivery tasks to a delivery system made by a truck and many drones belongs to the class of NP-hard problems, i.e., problems hard at least as the hardest problem in the NP-class of decision problems. The fact that the considered TSP problem can be formulated as a decision problem is straightforward. Due to the intractability issue of the optimal solving algorithms of the mFSTSP, and any TSP problem in general, several optimization methods have been developed over the last decades to address NP-hard problems. They are divided mainly into four categories based on the process of inspiration: evolution (e.g., genetic algorithms [17]), hunting (e.g., particle swarm optimization [18,19]), nature science (e.g., simulated annealing [20]) and behavioral/strategic processes (e.g., colony search optimization [21,22]). Refer to [23] for a comprehensive survey about the TSP problem's applications, the state-of-the-art solution approaches, and the complete taxonomy.

In this paper, we propose a mathematical programming statement of a particular extension of the TSP: the mFSTSP. Trivially, this problem becomes increasingly difficult if realistic constraints are considered and added to the formulation, such as the modeling of the tasks of launching and retrieving the drones, the parcel service time to complete the delivery, the drones' battery discharge, the synchronization between the truck and drones, the presence of more than one drone in the fleet, etc. Thus, we believe our work provides a significant contribution to the current state-of-the-art about the mFSTSP formulations and solution methods, since our formulation accounts also for some last-mile delivery specific issues, such as the service delivery time of the parcels, which is an inefficiency factor of the process, as highlighted in [10].

Due to the NP-hard nature of the addressed problem, heuristics approaches can often provide efficient and effective solutions. Heuristics algorithms do not guarantee an optimal solution but can provide good approximations in a reasonable amount of time. On the contrary, commercial solvers may not be suitable to solve our mFSTSP problem due to the addition of constraints and variables associated with the drones and their interactions with the truck. The problem becomes more complex, especially as the number of drones and the size of the problem increases. As also discussed in [24,25], the computational requirements of finding an exact solution to the mFSTSP using commercial solvers can become impractical or time-consuming, even for small-scale instances. For these reasons, we implement and evaluate a few sub-optimal heuristic resolution approaches: a local search solution, an evolutionary-based solution with two variants, and a lightweight ad-hoc greedy solution.

The main contributions of this work are (i) the formulation of a mathematical programming representation of a well-defined delivery scheduling problem with a truck and multiple drones and considering numerous constraints, which are generally partially evaluated at the state-of-the-art, (ii) the design of a hybrid evolutionary-based approach to be compared to both a computationally fast greedy solution method and the well-known local search method, (iii) the presentation of realistic simulation results in an urban environment.

This paper is organized as follows. Section 2 discusses the related work conducted in recent years concerning routing and task scheduling algorithms for last-mile delivery with truck and drones. Section 3 presents the problem formulation and the assumptions, as well as the proposed solution approaches and the simulation settings. Section 4 presents and discusses the simulation results obtained by comparing the performances of the proposed methods in terms of the total completion time of the delivery cycle for two scenarios of reference. Our conclusions and future research directions are drawn in Section 5.

2. Related Work

A literature review of heuristic routing solutions and mathematical formulations for tandem truck-drone problems is presented in the following.

The work in [26] proposes a mathematical formulation of TSP for an energy-efficient delivery system based on the use of UAVs in cooperation with the public transport system, showing the potential to optimize drones' energy consumption and battery charge by exploiting public transport vehicles as carriers. In [27], taking into account truck and drone's synchronization and minimization of total delivery time, two heuristic algorithms are proposed for solving large-size problems: Drone Truck Construction (DTC) and Large Neighborhood Search (LNS). The DTC algorithm tackles the routing problem of the first truck route from the depot to the scheduled customers, and the LNS algorithm schedules the drone routes considering the truck as a movable depot. The research in [28] suggests a hybrid genetic search algorithm based on a split algorithm, crossover, and local search operations. A restore strategy is designed to enhance the convergence properties of the algorithm, as well as an adaptive penalization mechanism to dynamically balance the search between feasible and infeasible solutions in order to solve the TSP-D (Travelling Salesman Problem with Drones), in the single drone variant. Simulation results show the capability of the evolutionary-based method to efficiently handle both min-cost and min-time optimization. Another hybrid genetic algorithm is proposed in [24] to tackle the mFSTSP, since commercial solvers like CPLEX cannot compute the optimal solution for more than fifteen customers due to computation time explosion. In this case, the proposed algorithm includes a sweep strategy as local search method, such that several neighborhoods of the solutions are generated for inserting and swapping orders. A combined drone-vehicle collaborative problem is solved in [29] by, firstly, planning truck and drones' routes independently while accounting for vehicle capabilities and restricted areas, and, secondly, exploiting a genetic search for finding the optimal task scheduling. The research in [30] proposes a genetic algorithm to compute the truck route in between the drone launch sites and a K-means clustering algorithm to compute the optimal launch sites of the UAVs, minimizing the total delivery time and the energy consumed by the vehicles. A genetic algorithm is also proposed in [31] to tackle the formulated mFSTSP that also includes the possibility of carrying Autonomous Transport Vehicles (ATVs) on the truck. Randomly generated instances demonstrate the validity of the approach. A genetic algorithm in combination with a pseudo-random heuristic rule in the crossover phase enhances the exploration capability of the search space in the work proposed in [32]. The proposed approach finds both the take-off points and the delivery schedules of the UAVs while minimizing the total completion time using the evolutionary-based strategy. Another genetic algorithm is proposed in [33] to address a scheduling problem of an mFSTSP, minimizing the total distance traveled. In this case, the Clarke and Wright's savings algorithm is used sequentially with the genetic algorithm to enhance the performances. Other genetic-based routing and scheduling solutions can be found in [34–36]. In [35], a particle swarm algorithm solution is compared to an evolutionary-based solution, showing the superiority of the latter in finding solutions corresponding to a higher level of optimality. The work in [36] implements a multipath genetic algorithm for allocating parcel delivery tasks in emergencies, with some nodes having higher priorities over others. An alternative to evolutionary-based approaches is represented by computationally lighter greedy solutions that, on the other hand, are less likely to produce a more optimal allocation with respect to

genetic algorithms. The study in [37] demonstrates the capability of a fast heuristic method to effectively handle an mFSTSP: the methodology combines (i) a multi-armed bandit selection for selecting the truck's customers' partitions from the whole customers' set, (ii) a stochastic local search algorithm that explores the search space, and (iii) a greedy scheduler for estimating the solution quality. The works [38–40] show an example of how to address a mFSTSP (in the single drone formulation) and the corresponding heuristic resolution method design in [38], the extension of the formulation to a fleet of multiple drones in [39], and, finally, a further formulation extension for optimizing the drones' speeds within the decision variables domain in [40]. These works show how heavily simplifications and assumptions can influence the formulation and the resolution of such problems. A Truck and Drone Routing Algorithm (TDRA) based on Adaptive Large Neighborhood Search (ALNS) is proposed in [41], since the MILP formulation yields to a drastic computation time increase with more than eight customers. It is worth noticing that, beside the optimization methodologies that sub-optimally solve such problems, a way to address the scalability issues of the commercial solvers is to design new mathematical statements of the problem. An example is represented by the study in [25], where a set of mFSTSP formulations are analyzed with the aim of enhancing the size of the largest instances solved in the literature.

3. Materials and Methods

3.1. Assumptions and Problem Formulation

The single truck-multiple drones routing problem, i.e., the mFSTSP, is the problem of allocating a set of parcel delivery tasks either to a truck or to the multiple UAVs that operate in coordination with it. A parcel delivery task consists of a customer, i.e., a location or node, to be visited exactly once either by the truck or by one of the UAVs of the fleet, which is assumed to be homogenous (all drones have the same characteristics). It is possible that not all of the customers are feasibly serviceable by the UAVs: the set of such customers is defined as the *truck only nodes* set. The reasons behind the definition of this set may be due to the parcel's weight that exceeds the UAVs' capacity, the difficult landing conditions at the customer's location, the requirement of customer's signature, etc. *truck node* and *drone node* denote nodes exclusively served by the truck or a UAV respectively, while *drone eligible node* is the set of nodes that can be served either by a truck or a UAV. *truck route* (or *tsp tour*) denotes, given a set of customers, the minimum cost sequence of nodes visited by the truck.

The delivery process starts with the truck and the drones departing from the depot and ends with the return of all of the vehicles to the depot itself. Each UAV of the fleet can travel toward customers being transported by the truck, conserving its battery life, or can perform a *drone sortie*, consuming the energy stored in the battery. A *drone sortie* consists of a set of three nodes from which the UAV is launched, delivers the parcel, and is recovered by the truck. A *drone sortie* is feasible if the UAV does not exceed its endurance limit; the loaded UAV may be launched from the initial depot or from a customer's node; the recovery node may be another customer's node or the final depot, in the latter case no synchronization between the truck and the UAV is needed since it is assumed that some personnel are always available to recover the drone. Multiple drone sorties can be performed by each UAV in the delivery cycle. *drone travel* defines a UAV operation moving from one node to another. During the execution of a *drone sortie*, the truck may go from the launch node to the recovery node or it can accomplish multiple deliveries (accounting for the synchronization constraints that impose on the truck to be present at the recovery node before the battery life of the UAV expires). $sub-route$ denotes the sub-set of the *truck route* delimited by the node where the drone is launched and the node where the drone is retrieved. The set of all $sub-route$ of a given *tsp tour* is defined as the *truck* $sub-route$. Some truck driver's service times must be considered in order to model a realistic drone-based last-mile delivery service: the service time for, possibly, charging a UAV and loading it with the parcel, a service time to physically deliver the parcel to the customer, and a service time to allow the recovery of a landing UAV.

The adopted notation is defined as follows:

- $Q = \{1, 2, \ldots, v\}$: set of available UAVs for the delivery process.
- $C = \{1, 2, \ldots, c\}$: set of customers to be served either by the truck or a UAV.
- $C' = \{1, 2, \ldots, c'\} \subseteq C$: sub-set of customers that may be served by a UAV.
- $N = \{0, 1, \ldots, c+1\}$: set of nodes to be visited exactly once in the delivery process (initial and final node correspond to the same location, i.e., the depot).
- $N_0 = \{0, 1, \ldots, c\}$: set of nodes from which the vehicles can depart from.
- $N_+ = \{1, 2, \ldots, c+1\}$: set of nodes to which the vehicles can head.
- τ_{ij}: truck travel time from node $i \in N_0$ to node $j \in N_+$.
- τ'_{ijq}: travel time of UAV q from node $i \in N_0$ to node $j \in N_+$.
- s_L: service time for launching a UAV from a customer's node.
- s_R: service time for recovering a UAV at a customer's node.
- ε: UAVs' flight endurance.
- δ_t: time to physically deliver the parcel to a customer.
- M: sufficiently large number for constraints' linearization, if needed.
- $(i, j, k)_q$: *drone sortie* of UAV q, i.e., tuple describing the operation of UAV q being launched at node i, delivering the parcel at node j and being recovered at node k.
- P_q: set of all possible tuples $(i, j, k)_q$ feasibly flown by UAV q.
- $x_{ij} = \{0, 1\}$: decision variable representing whether the truck travels from node $i \in N_0$ to node $j \in N_+ (x_{ij} = 1)$ or not $(x_{ij} = 0)$.
- $y_{ijkq} = \{0, 1\}$: decision variable representing whether UAV q flies from node $i \in N_0$ to node $j \in N_+$ and is recovered at node $k \in \{N_+ : (i, j, k) \in P\}$ $(y_{ijkq} = 1)$ or not $(y_{ijkq} = 0)$.
- t_j: truck arrival time at node $j \in N_+$.
- t'_{jq}: UAV q arrival time at node $j \in N_+$.
- $p_{ij} = \{0, 1\}$: decision variable representing whether customer $i \in C$ has been visited before customer $j \in \{C : i \neq j\}$ by the truck $(p_{ij} = 1)$ or not $(x_{ij} = 0)$.
- $1 \leq u_i \leq c{+}2$: integer position variable of node $i \in N_+$ in the truck's travel.

It follows the mathematical programming-based formulation of the addressed synchronized single truck-multiple UAVs problem, with notation and assumptions stated above.

$$min\ (t_{max})\ ,\ \ subject\ to: \tag{1}$$

$$\sum_{\substack{i \in N_0 \\ i \neq j}} x_{ij} + \sum_{q \in Q} \sum_{\substack{i \in N_0 \\ i \neq j}} \sum_{\substack{k \in N_+ \\ (i,j,k)_q \in P_q}} y_{ijkq} = 1;\ \ \forall j \in C, \tag{2}$$

$$\sum_{j \in N_+} x_{0j} = 1, \tag{3}$$

$$\sum_{i \in N_0} x_{i,c+1} = 1, \tag{4}$$

$$u_i - u_j + 1 \leq (c+2)\big(1 - x_{ij}\big);\ \ \forall i \in C,\ j\ \in \{N_+ : j \neq i\}, \tag{5}$$

$$\sum_{\substack{i \in N_0 \\ i \neq j}} x_{ij} = \sum_{\substack{k \in N_+ \\ k \neq j}} x_{jk};\ \ \forall j \in C, \tag{6}$$

$$\sum_{j \in C} \sum_{\substack{j \in C \\ (i,j,k)_q \in P_q}} y_{ijkq} \leq 1;\ \ \forall i \in N_0,\ q \in Q, \tag{7}$$

$$\sum_{\substack{i \in N_0 \\ i \neq k}} \sum_{\substack{j \in C \\ (i,j,k)_q \in P_q}} y_{ijkq} \leq 1; \ \forall k \in N_+, \ q \in Q, \tag{8}$$

$$2y_{ijkq} \leq \sum_{\substack{h \in N_0 \\ h \neq i}} x_{hi} + \sum_{\substack{l \in C \\ l \neq k}} x_{lk}; \ \forall i \in C, \ j \in \{C : j \neq i\}, \ k \in \left\{N_+ : (i,j,k)_q \in P_q\right\}, \ q \in Q, \tag{9}$$

$$y_{0jkq} \leq \sum_{\substack{h \in N_0 \\ h \neq k}} x_{hk}; \ \forall j \in C, \ k \in \left\{N_+ : (0,j,k)_q \in P_q\right\}, \ q \in Q, \tag{10}$$

$$u_k - u_i \geq 1 - (c+2)\left(1 - \sum_{\substack{j \in C \\ (i,j,k)_q \in P_q}} y_{ijkq}\right); \ \forall i \in C, \ k \in \{N_+ : k \neq i\}, \ q \in Q, \tag{11}$$

$$t'_{iq} \geq t_i - M\left(1 - \sum_{j \in C} \sum_{\substack{k \in N_+ \\ (i,j,k)_q \in P_q}} y_{ijkq}\right); \ \forall i \in C, \ q \in Q, \tag{12}$$

$$t'_{iq} \geq t_i + M\left(1 - \sum_{j \in C} \sum_{\substack{k \in N_+ \\ (i,j,k)_q \in P_q}} y_{ijkq}\right); \ \forall i \in C, \ q \in Q, \tag{13}$$

$$t'_{kq} \geq t_k - M\left(1 - \sum_{i \in N_0} \sum_{\substack{j \in C \\ (i,j,k)_q \in P_q}} y_{ijkq}\right); \ \forall k \in N_+, \ q \in Q, \tag{14}$$

$$t'_{kq} \leq t_k + M\left(1 - \sum_{i \in N_0} \sum_{\substack{j \in C \\ (i,j,k)_q \in P_q}} y_{ijkq}\right); \ \forall k \in N_+, \ q \in Q, \tag{15}$$

$$\begin{aligned} t_k \geq t_h + \tau_{hk} + s_L\left(1 - \sum_{q \in Q} \sum_{l \in C} \sum_{\substack{m \in N_+ \\ (k,l,m)_q \in P_q}} y_{klmq}\right) + \\ s_R\left(\sum_{q \in Q} \sum_{i \in N_0} \sum_{\substack{j \in C \\ (i,j,k)_q \in P_q}} y_{ijkq}\right) + \delta_t - M(1 - x_{hk}); \ \forall h \in N_0, \ k \in \{N_+ : k \neq h\}, \end{aligned} \tag{16}$$

$$t'_{jq} \geq t'_{iq} + \tau'_{ijq} + \delta_t - M\left(1 - \sum_{\substack{k \in N_+ \\ (i,j,k)_q \in P_q}} y_{ijkq}\right); \ \forall j \in C', \ i \in \{N_0 : j \neq i\}, \ q \in Q, \tag{17}$$

$$t'_{kq} \geq t'_{jq} + \tau'_{jkq} + s_R - M\left(1 - \sum_{\substack{i \in N_0 \\ (i,j,k)_q \in P_q}} y_{ijkq}\right);\ \forall j \in C',\ k \in \{N_+ : j \neq k\},\ q \in Q, \tag{18}$$

$$t'_{kq} - \left(t'_{jq} - \tau'_{ijq}\right) \leq e + M\left(1 - y_{ijkq}\right);\ \forall k \in N_+,\ j \in \{C : j \neq k\},\ i \in \left\{N_0 : (i,j,k)_q \in P_q\right\},\ q \in Q, \tag{19}$$

$$u_i - u_j \geq 1 - p_{ij}(c+2);\ \forall i \in C,\ j \in \{C : j \neq i\}, \tag{20}$$

$$u_i - u_j \leq -1 + (c+2)\left(1 - p_{ij}\right);\ \forall i \in C,\ j \in \{C : j \neq i\}, \tag{21}$$

$$p_{ij} + p_{ji} = 1;\ \forall i \in C,\ j \in \{C : j \neq i\}, \tag{22}$$

$$t'_{lq} \geq t'_{kq} - M\left(3 - \sum_{\substack{j \in C \\ (i,j,k)_q \in P_q \\ j \neq l}} y_{ijkq} - \sum_{\substack{m \in C \\ m \neq i \neq k \neq l}} \sum_{\substack{n \in N_+ \\ (l,m,n)_q \in P_q \\ n \neq i \neq k}} y_{lmnq} - p_{il}\right); \tag{23}$$
$$\forall i \in N_0,\ k \in \{N_+ : k \neq i\},\ l \in \{C : l \neq i \neq k\},\ q \in Q,$$

$$t_0 = t'_{0q} = 0, \tag{24}$$

$$p_{0j} = 1;\ \forall j \in C, \tag{25}$$

$$x_{ij} \in \{0,1\};\ \forall i \in N_0,\ j \in \{N_+ : j \neq i\}, \tag{26}$$

$$y_{ijkq} \in \{0,1\};\ \forall i \in N_0,\ j \in \{C : j \neq i\},\ k \in \left\{N_+ : (i,j,k)_q \in P_q\right\},\ q \in Q, \tag{27}$$

$$1 \leq u_i \leq c+2;\ \forall i \in N_+, \tag{28}$$

$$t_i \geq 0;\ \forall i \in N, \tag{29}$$

$$t'_{iq} \geq 0;\ \forall i \in N,\ q \in Q, \tag{30}$$

$$p_{ij} \in \{0,1\};\ \forall i \in N_0,\ j \in \{C : j \neq i\} \tag{31}$$

Equation (1) is the objective function that seeks to minimize the makespan, i.e., the total completion time, of the delivery cycle: it is equivalent to minimize the latest arrival time either of the truck or the latest UAV, i.e., $min\left(max\left(t_{c+1}, t'_{c+1,1}, t'_{c+1,2}, \ldots, t'_{c+1,v}\right)\right)$. This equivalence is justified by Constraints (14) and (15) that impose a time synchronization among the truck and the UAVs at rendezvous nodes. Constraint (2) imposes the single visit condition for each customer, (3) and (4) impose that the truck starts and ends the delivery cycle at the depot exactly once, and (5) eliminates the truck sub-tours with the bounds of the auxiliary variable u_i being set by (28). Constraint (6) imposes that if the truck visits

node j, it must depart from node j once the delivery task is performed, (7) and (8) establish that the UAVs can be launched and retrieved at any node, (9) imposes that if a UAV is launched at node i and retrieved at node k, both nodes i and k must belong to the truck tour, and (10) expresses an analogous condition for the UAVs. Constraint (11) expresses that if a UAV is launched at node i and retrieved at node k, the truck must visit i before k. Constraints (12) and (13) avoid that either a UAV or the truck are not present at a launch node; analogously, (14) and (15) avoid the lack of coordination between the truck and the UAVs at a rendezvous node. Constraint (16) regulates the time update of the truck tour if the truck travels from node $h \in N_0$ to node $k \in N_+$: the truck arrival time at node k includes the potential launch and recovery time of the UAVs, and the service time to physically deliver the parcel at node k. Constraint (17) regulates the time update for UAV q tour from node i to node j, considering the UAV flight time and the service time to physically deliver the parcel. Constraint (18) includes the travel time of UAV q from node j to node k and the recovery time of the UAV. The endurance limitations of the UAVs are expressed by Constraint (19), which becomes active if the UAVs' travels take place. Constraints (20), (21), (22) establish the correct ordered truck's path. Constraints (23) and (25) prevent that a UAV is launched from node l before being retrieved at node k. Constraint (24) defines the starting time of the truck and the UAVs. The domain of the decision variables is defined by Constraints from (26) to (31).

The MILP formulated above is inspired by the one formulated in [38]. The formulation is modified to include more than one UAV and the service times required to physically deliver the parcels. The problem is NP-hard and a few sub-optimal resolution approaches are presented in the next sections through a set of heuristic algorithms.

3.2. Local Search Algorithm

The Local Search Algorithm (LSA) is one of the first and most adopted heuristic algorithms in operation research. It is a heuristic method that evaluates a sub-space of the search space of the problem, an instance of the class of TSPs in this case, until, in general, a sub-optimal solution is found. At each step, the local search algorithm explores the neighboring solutions of a current solution to find a better solution, given an evaluation criterion. It focuses on a single solution and its immediate neighbors, rather than maintaining a population of solutions. It follows the main steps of an LSA:

- *Algorithm Initialization*: The algorithm starts the search process from an initial solution.
- *Fitness Evaluation*: The fitness or objective function value of the solution is computed; the fitness of the solution represents its quality or suitability.
- *Neighborhood Generation*: The algorithm generates solutions neighboring the current solution through small modifications of the current solution. These modifications can include swapping elements, adding and removing elements, or altering values within the encoding of the solution itself.
- *Neighbor Selection and Evaluation*: From the set of generated neighboring solutions, one neighbor is selected as the next solution to explore. The selection process can vary based on different criteria, such as selecting the neighbor with the best improvement in fitness value or choosing a neighbor randomly.
- *Solution Update*: If the neighbor solution's quality is better than the current solution, the neighbor solution is set as the new current solution. Otherwise, the current solution remains unchanged.
- *Algorithm Termination*: The algorithm repeats steps *Fitness Evaluation*, *Neighborhood Generation*, *Neighbor Selection and Evaluation*, and *Solution Update* until a termination condition is met. The termination condition can be a maximum number of iterations, reaching a satisfactory solution quality, or the lack of solution optimality improvement for a certain number of iterations.

Local search algorithms are particularly useful for solving optimization problems where the objective is to find a solution that satisfies certain constraints or criteria. They are often applied to large search spaces or problems with a high degree of complexity,

where exploring the entire solution space is computationally infeasible. However, LSAs may end up being trapped in local optima, which are suboptimal solutions with better optimality level than their immediate neighbors, but with worse optimality level than the global optimum.

Considering the formulated mFSTSP, the proposed LSA starts from the conventional $tsp\ tour$, i.e., the routing solution with a single truck and no drones obtained by a TSP solver, then either sequentially removes nodes from the $tsp\ tour$ and assigns them to one of the UAVs of the fleet or swaps the nodes of the TSP solutions seeking for an improvement of the solution itself. This process is repeated until the LSA does not find any improvement. An example of how the algorithm works is shown in Figure 1. The pseudo-code of the main body of the proposed mFSTSP-related LSA solution is shown in Algorithm 1.

Algorithm 1 Main body of the *Local Search Algorithm*

Initialize: $tsp\ tour,\ response,\ truck\ only\ nodes,\ \varepsilon,\ s_R,\ s_L,\ \delta_t,\ v$;
$max\ savings = 0;\ C' = C \backslash truck\ only\ nodes$;
$truck\ sub-routes =$ [[$tsp\ tour$] as many v are available];
$i^* =$ [[] as many v are available]; $j^* =$ [[] as many v are available]; $k^* =$ [[] as many v are available];
$sub\ route\ uav\ flag =$ [[] as many v are available];
while no improvement is produced **do**
  for $n \in v$ **do**
    for $j \in C'$ **do**
      $savings(j,t,tsp\ tour)$;
      for $sub-route \in truck\ sub-routes[n]$ **do**
        if $sub-route$ is already assigned to a UAV **do**
          $cost\ truck(j,t,savings, truck\ sub-routes, sub-route)$;
        else
          $cost\ uav(j,t,savings,sub-route)$;
        end
      end
    end
    if $max\ savings > 0$ **do**
      $perform\ update(i^*[n], j^*[n],k^*[n],served\ by\ uav[n], truck\ sub-routes, drone\ nodes)$;
      reset $max\ savings$;
    else
      exit;
    end
  end
end

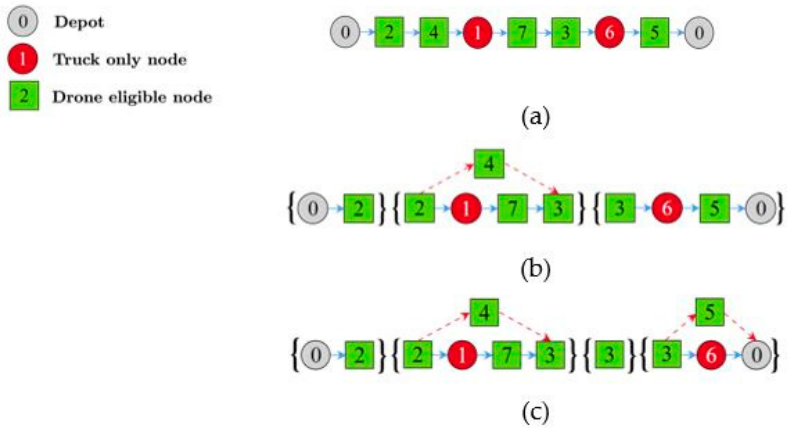


Figure 1. Representation of the functioning logic of the LSA with a truck and one drone. Grey nodes represent the depot, red nodes represent truck-only nodes and green nodes represent drone-eligible nodes. (**a**) The TSP tour computed by the TSP solver; (**b**) 1st iteration of the LSA: node 4 is removed from the TSP tour and assigned to the UAV, which is launched at node 2 and retrieved at node 3; (**c**) 2nd iteration of the LSA: node 5 is removed from the TSP tour and assigned to the UAV, which is launched at node 3 and retrieved at the depot.

The set of drone-eligible nodes C' is obtained removing the *truck only nodes* from the whole node set C. The $tsp\ tour$ is calculated through a solver that minimizes the total truck travel time given the input C. The computed $tsp\ tour$ is split into many sub-tours, called $sub-route$, that includes the truck nodes where the UAVs are launched and recovered. The set of all sub-routes of the truck is the $sub-routes$ vector; it is initialized in Line 3 and progressively modified when UAV assignments are made through the algorithm execution. Analogous considerations hold for i^*, j^* and k^*, which indicate the UAVs' launch, delivery, and recovery nodes. At each iteration and for each UAV, the LSA allocates a delivery task to the UAV with a greater value of $max\ savings$. Each UAV is considered independently, as for a single drone FSTSP problem. The $savings()$ function, shown in Algorithm 2, is called every time a *drone eligible node* j is investigated in order to compute the time reduction related to the removal of node j from the $tsp\ tour$. The $cost\ truck()$ function, shown in Algorithm 3, is called every time a *drone eligible node* cannot be assigned to a UAV because the UAV is already operating on the considered $sub-route$. The function evaluates the insertion of node j in a $sub-route$ as a *truck node* and swaps the *truck node* if this brings a reduction in the waiting time between the UAV and the truck. The $cost\ uav()$ function, shown in Algorithm 4, evaluates the potential assignment of a *drone eligible node* to a *uav sortie*, if the node j does not belong to a $sub-route$ where UAV travel has already been assigned. The $perform\ update()$ function updates the $tsp\ tour$, the truck sub-routes, the UAVs' assignments and the arrival time of the truck at each node of the $tsp\ tour$. $perform\ update()$, shown in Algorithm 5, is called once all drone eligible nodes have been evaluated and if the LSA is able to produce an improvement to the solution.

In order to ensure the synchronization between the UAVs and the truck at the recovery nodes, the $time\ update()$ function, shown in Algorithm 6, is called within the $perform\ update()$ function. $time\ update()$ regulates the awaiting of the truck by each UAV and vice versa, updating the truck arrival time vector at each node, denoted by t. For instance, if the UAV's travel time is greater than the corresponding $sub-route$, the truck should wait for the UAV recovery at a recovery node before serving the next customer. On the other hand, the UAV should hover in place waiting for the truck if the *drone travel* time is smaller than the corresponding $sub-route$.

Algorithm 2 *savings* () function

Input: $tsp\ tour,\ j,\ t$;
Output: $savings$;
find i, customer node before j in $tsp\ tour$;
find k, custormer node after j in $tsp\ tour$;
$savings = \tau_{ij} + \delta_t + \tau_{jk} - \tau_{ik}$;
if $j \in sub-route$ served by a UAV **do**
  find a: first customer node of $sub-route$;
  find b: last customer node of $sub-route$;
  find j': customer node served by UAV;
  find $t'[b]$: arrival time of truck at b, when j is removed;
  $savings = \min\left(savings, t'[b] - \left(t[a] + \tau'_{aj'} + \delta_t + \tau'_{j'b} + s_R\right)\right)$;
end

Algorithm 3 *cost truck* () function

Input: $savings,\ j,\ t,\ truck\ sub-routes,\ sub-route$;
Output: $i^*,\ j^*,\ k^*,\ served\ by\ uav,\ max\ savings$;
find a: first node of $sub-route$;
find b: last node of $sub-route$;
for $i \in sub-route$ **do**
  find k: node after i in $sub-route$;
  $cost = \tau_{ij} + \delta_t + \tau_{jk} - \tau_{ik}$;
  if $cost < savings$ **&** $t[b] - t[a] + cost \leq \varepsilon$ **&** $savings - cost > max\ savings$ **do**
    for other $truck\ sub-routes$ assigned to other UAVs **do**
      find a_D: first node of $sub-route$;
      find b_D: last node of $sub-route$;
      if $t[b_D] - t[a_D] + cost \leq \varepsilon$ **do**
        set $truck\ node\ swap$ **as** feasible, also considering other UAVs assignments;
      end
    end
    if $truck\ node\ swap$ is feasible for all UAVs assignments **do**
      $served\ by\ uav = false;\ max\ savings = savings - cost$;
      $i^* = i,\ j^* = j,\ k^* = k$;
    end
  end
end

Algorithm 4 *cost uav* () function

Input: $j,\ t,\ sub-route,\ savings$;
Output: $i^*,\ j^*,\ k^*,\ served\ by\ uav,\ max\ savings$;
for $i \in sub-route$ **do**
  for $k \in sub-route \mid k$ is after j **do**
    if $\tau'_{ij} + \delta_t + \tau'_{jk} \leq \varepsilon$ **do**
      find $t'[k]$, with j removed from $sub-route$;
      if $t'[k] - t[i] + s_R + s_L < \varepsilon$ **do**
        $cost = max\big(0, max\big(t'[k] - t[i] + s_R + s_L, \tau'_{ij} + \delta_t + \tau'_{jk} + s_R + s_L\big) - t'[k] - t[i]\big)$;
      end
      if $savings - cost > \max savings$ **do**
        $served\ by\ uav = true;\ max\ savings = savings - cost;$
        $i^* = i,\ j^* = j,\ k^* = k;$
      end
    end
  end
end

Algorithm 5 *perform update* () function

Input: $truck\ sub-routes,\ drone\ nodes,\ served\ by\ uav[n],\ i^*[n],\ j^*[n],\ k^*[n]$;
Output: $tsp\ tour,\ t,\ truck\ sub-routes$;
if $served\ by\ uav[n] = true$ **do**
  remove $j^*[n]$ from $tsp\ tour$;
  split $truck\ sub-routes[n]$;
  update $drone\ nodes[n]$;
  if $i^*[n] \in C'$ **do**
    remove $i^*[n]$ from C';
  end
  if $k^*[n] \in C'$ **do**
    remove $k^*[n]$ from C';
  end
  for all $truck\ sub-routes$ **do**
    for all $sub-route$ in each $truck\ sub-routes$ **do**
      remove $j^*[n]$ from $sub-route$;
    end
  end
  $time\ update(truck\ sub-routes, drone\ nodes)$
else
  remove $j^*[n]$ from $tsp\ tour$;
  insert $j^*[n]$ in the new position in $tsp\ tour$;
  update $truck\ sub-routes[n]$;
  for all $truck\ sub-routes$ **do**
    for all $sub-route$ in each $truck\ sub-routes$ **do**
      perform $truck\ node\ swap$ in $sub-route$, according to $j^*[n]$ location;
    end
  end
  $time\ update(truck\ sub-routes, drone\ nodes)$;
end

Algorithm 6 *time update* () function

Input: $truck\ sub-routes,\ drone\ nodes$;
Output: t;
Initialize: $t = [0]$ for $nodes \in tsp\ tour,\ index = 0$;
for all $sub-route \in truck\ sub-routes$ **do**
  $time\ truck = 0$;
  find a: first node of $sub-route$;
  find b: last node of $sub-route$;
  for $node \in sub-route$ **do**
    find i, current $node$; **find** j, next $node$ in $sub-route$; $index = index + 1$;
    if $node$ is the last of $tsp\ tour$ **do**
      $time\ truk = time\ truck + \tau_{ij}$;
      $t[index] = t[index-1] + \tau_{ij}$;
    else
      $time\ truk = \delta_t + \tau_{ij}$;
      $t[index] = t[index-1] + \delta_t + \tau_{ij}$;
    end
  end
  insert j' in $drone\ nodes$ of $sub-route$;
  $time\ drone = \tau'_{aj'} + \delta_t + \tau'_{j'b}$;
  if $time\ drone > time\ truck$ **do**
    $t[index] = time\ drone - time\ truck;\ time\ truck = time\ drone - time\ truck$;
  end
  if $time\ drone > \varepsilon\ ||\ time\ truck > \varepsilon$ **do**
    set schedule as **infeasible**;
  end
end

3.3. Hybrid Genetic Algorithm—Feasible Variant

The evolutionary-based algorithm is a heuristic algorithm that performs optimization by applying methodologies of exploration of the search space inspired by the evolution mechanisms of nature. This algorithm explores, with a progressive improvement of the optimality level, a sub-set of the solution space of the problem. According to the terminology of this method, a population made by individuals, i.e., either the delivery schedule of the truck or the delivery schedule of the UAVs in this case, gives part of its "genetic" material to the individuals of the next algorithm's iterations in such a way that a cost function's valuable improvement is followed in the next generations. During the reproduction phase, the individuals are modified in order to meet certain fitness criteria: for instance, another heuristic methodology may be adopted to generate a new set of solutions, which partially contain the information, i.e., the genetic material of the solutions of the current iteration and partially derive from the designed further exploration methodology of the search space. It follows the main steps of evolutionary search methods:

- *Algorithm Initialization*: The algorithm starts by creating a population of potential solutions to the problem. Each individual of the population is a potential solution to the problem.
- *Fitness Evaluation*: Each solution in the population is evaluated and assigned to a fitness value based on how well it solves the problem. The fitness value is represented by the total delivery time of truck and UAVs.
- *Natural Selection*: Solutions with higher fitness values are more likely to be selected for the next generation. This process mimics the natural selection of individuals with favorable traits.
- *Individuals Reproduction*: The selected solutions are used to create offspring for the next generation. This is done through crossover and mutation operations, which mimic the genetic processes of recombination and unexpected mutation in nature. The crossover phase involves combining genetic material from two parent solutions to create one or more offspring solutions. This is done by selecting a crossover point and exchanging genetic material beyond that point between the parents. The mutation phase introduces small random changes in the genetic material of offspring solutions. It helps explore new regions of the solution space that may lead to better solutions.
- *Individuals Replacement*: The offspring solutions replace some of the less fit solutions in the current population. This ensures that the population evolves over time and improves its overall fitness.
- *Algorithm Termination*: The algorithm continues to iterate through *Fitness Evaluation*, *Natural Selection*, *Individuals Reproduction*, and *Individuals Replacement* steps until a termination condition is met. This condition could be a maximum number of generations, a specific fitness threshold, or a predefined computational limit.

By iteratively repeating the steps indicated above, the evolutionary approach explores the solution space, favoring solutions with higher fitness values. Over time, the population tends to converge towards better solutions, but not, in general, to the global optimal solution. The proposed algorithm is a hybrid genetic algorithm that includes optimization and a local search-based heuristic rule during the mutation phase. For this variant of the proposed genetic algorithms, the individuals of the population of each iteration can only correspond to feasible task schedules. The fitness improvement criterion of the solution aims at the minimization of the delivery time of the delivery process. The first proposed variant of the proposed evolutionary-based routing method is the *Hybrid Genetic Algorithm—Feasible Variant*, shown in Algorithm 7.

Algorithm 7 Main body of the *Hybrid Genetic Algorithm—Feasible Variant*

Initialize: $response,\ truck\ only\ nodes,\ s_R,\ s_L,\ v,\ \varepsilon,\ \delta_t$;
generate randomly $population$ of n $individuals$; $\gamma = 0$;
while $\gamma < treshold$ **do**
  $\gamma = \gamma + 1$;
  $parents\ selection(population, fit\ values)$;
  $child\ generation(parents, drone\ deliveries)$;
  $split(child)$;
  $local\ search(split\ child)$;
  $restore(educated\ child, educated\ drone\ deliveries)$;
  $select\ survivors(restored\ child, educated\ child, educated\ drone\ deliveries, population)$;
end
$select\ fittest\ individuals(population, fitness\ values)$;

Starting from a random population of n individuals, with an individual being defined as a *tsp tour* without the initial and final depots, the *parents selection*() function, shown in Algorithm 8, selects the TSP solutions that provide the basis genetic material for the child generation phase. The *child generation*() function, shown in Algorithm 9, generates new individuals that carry the genetic material of the selected parents with some diversification, to enlarge the solution space explored. The crossover and mutation phases takes place through the $split()$, *local search*() and *restore*() functions that split, modify, and re-assemble individuals in order to generate new genetic material, seeking for an improvement of the solution optimality level with respect to the previous generations. The $split()$ function is reported in Algorithm 10, while its inner functions *create uav sorties*(), *insert infeasible customers*(), $truck\ sub-routes\ update()$ are reported in Algorithm 11, Algorithm 12, and Algorithm 13 respectively. *create uav sorties*() assigns all the *drones eligible nodes* to the UAVs, but if this results in an infeasible assignment, $insert\ infeasible\ customers()$ re-inserts the infeasible nodes as *truck nodes* in the *child* vector.

The minimum number of customers that can be served by v UAVs, n_d, is computed by means of Equation (32), with the quantity $\left\lceil \frac{c-v}{v+1} \right\rceil$ being taken from [38] and denoting the minimum number of customers to be served by the truck in presence of v UAVs.

$$n_d = c - \left\lceil \frac{c-v}{v+1} \right\rceil \ , \ if \ \ (n_d > c') \rightarrow n_d = c' \tag{32}$$

Once all nodes are assigned either to the truck or the UAVs, $truck\ sub-routes\ update()$ updates the solution. The population generated by the *Hybrid Genetic Algorithm—Feasible Variant* considers only feasible solutions, i.e., solutions that do not violate the endurance constraints of the UAVs. When the *child* has been educated by the *local search*() function, which is the same as in Algorithm 1, the *restore*() function, shown in Algorithm 14, generates an individual coherent with the others, since the LSA returns each $truck\ sub-route$. The *select survivors*() function, shown in Algorithm 15, discards the individuals with poor characteristics either in terms of fitness values (the completion time of the delivery cycle) or in terms of diversification from the other individuals. At the end of the whole process, the $select\ fittest\ individuals()$ function selects the best solution according to the minimum makespan criterion.

Algorithm 8 *parents selection* () function

Input: $population,\ fit\ values;$

Output: $parents;$

for $i = 1,2$ **do**

  sample randomly $k = 2\ individuals$ in $population;$

  select $fittest\ individual$ among $k\ individuals;$

  assign $fittest\ individual$ to $parents[i];$

end

Algorithm 9 *child generation* () function

Input: $parents$, $drone\ deliveries$;
Output: $child$;
Initialize $r \in [0,10]$ $as\ integer\ variable$;
insert depots to $parents$;
if $r < 5$ || $drone\ deliveries = \emptyset$ **do**
  select first parent, $parents[1]$;
  sample $[a, b]$ from $parents[1]$, with a before b in $tsp\ tour$;
  assign $nodes$ of $parents[1]$ between a and b to $child$;
  if $drone\ deliveries \in [a, b]$ **do**
    skip $nodes \in drone\ deliveries$;
  end
  assign remaining $nodes$ to $child$ from $parents[2]$;
else
  sample $[a, b]$ from $drone\ deliveries[1]$, with a before b in $drone\ deliveries$;
  assign $nodes$ of $drone\ deliveries[1]$ between a and b to $child$;
  if TSP $nodes$ of $parents[1] \in [a, b]$ **do**
    skip $nodes$ of $parents[1]$;
  end
  assign remaining $nodes$ to $child$ from $parents[2]$;
end

Algorithm 10 *split* () function

Input: $child$;
Output: $split\ child$, t, $split\ truck\ sub-routes$, $split\ drone\ deliveries$;
select n_d as in Equation (32);
sample randomly n_d $drone\ nodes$ from C';
remove assigned $drone\ nodes$ from $child$; **compute** t for $child$;
$create\ uav\ sorties(child, drone\ nodes, n_d, t)$;
while $infeasible\ customer \neq \emptyset$ **do**
  $insert\ infeasible\ customer(child, infeasible\ customer)$;
  reset assigned $drone\ nodes$;
  update t;
  $create\ uav\ sorties(child, drone\ nodes, n_d, t)$;
end
reset $drone\ nodes$, C';
$truck\ sub-routes\ update(child, uav\ sorties, v)$;
$time\ update(truck\ sub-routes, drone\ nodes)$;

Algorithm 11 *create uav sorties* () function

Input: $child,\ t,\ drone\ nodes,\ n_d$;
Output: $uav\ sorties,\ infeasible\ customers$;
Initialize: $number\ of\ options$ = [[0] as many $nodes$ in $tsp\ tour$];
for $n \in range(n_d)$ **do**
  for $j \in drone\ nodes$ **do**
    for $i \in tsp\ tour$ **do**
      find $node\ k$ in $tsp\ tour$ after i;
      if $\tau'_{ij} + \delta_t + \tau'_{jk} < \varepsilon$ **&** $\tau_{ik} < \varepsilon$ **do**
        $number\ of\ options[j] = number\ of\ options[j] + 1$;
      end
    end
  end
end
while unassigned $nodes \neq \emptyset$ **do**
  $wait\ time = \infty$; **pick** j as $node$ with minimum $number\ of\ options$;
  if $number\ of\ options[j] = 0$ **do**
    insert j in $infeasible\ customers$;
  else
    $j = j'$;
    for $i \in tsp\ tour$ **do**
      for $n \in available\ uav[i]$ **do**
        find k: node after i in $tsp\ tour$;
        if $\tau'_{ij} + \delta_t + \tau'_{jk} < \varepsilon$ **&** $\tau_{ik} < \varepsilon$ **do**
          $w = \tau'_{ij} + \delta_t + \tau'_{jk} - t[k] - t[i]$;
          if $wait\ time \geq 0$ **&** $w < wait\ time$ || $0 < w < wait\ time$ **do**
            $wait\ time = w;\ [n^*, i^*, j^*, k^*] = [n, i, j', k]$;
          end
        end
      end
    end
    if $wait\ time = \infty$ **do**
      insert j' in $infeasible\ customers$;
    else
      assign $uav\ sortie = (n^*, i^*, j^*, k^*)$;
    end
  end
end

Algorithm 12 *insert infeasible customers* () function

Input: $tsp\ tour,\ infeasible\ customers;$
Output: $tsp\ tour;$
while $infeasible\ customers \neq \emptyset$ **do**
  for $i \in infeasible\ customers$ **do**
    for $k \in tsp\ tour$ **do**
      find p: position of k in $tsp\ tour;$
      find $node\ h$ before $node\ k;$
      $(p^*, i^*) = \arg \min_{(p,i) \in tsp\ tour} (\tau_{hi} + \delta_t + \tau_{ik} - \tau_{hk});$
    end
  end
  insert i^* in position p^* of $tsp\ tour;$
end

Algorithm 13 *truck sub-routes update* () function

Input: $child,\ uav\ sorties,\ v;$
Output: $truck\ sub-routes,\ drone\ nodes;$
for $n \in range(n_d)$ **do**
  for each $uav\ sortie$ **do**
    $uav\ sortie = (n, i, j, k);$
    remove i,j,k from C';
    for $sub-route \in truck\ sub-routes[n]$ **do**
      remove j from $sub-route;$
      split $truck\ sub-routes;$
      update $drone\ nodes;$
      exit;
    end
  end
end

Algorithm 14 *restore* () function

Input: *educated child, educated drone deliveries*;
Output: *restored child*;
for $sub-route \in truck\ sub-routes$ **do**
  find a: first node of $sub-route$;
  find b: last node of $sub-route$;
  insert randomly *drone delivery* associated to $sub-route$ between a,b;
  remove *depots*;
end

Algorithm 15 *select survivors* () function

Input: *restored child, educated drone deliveries, population, educated child*;
Output: *population*;
if each *individual* $\neq$ *restored child* **do**
  if *fitness value* of *restored child* $<$ *fitness value* of *individual* **do**
    remove *individual* from *population*;
    insert *restored child* in *population*;
    set *tentative solution* as *educated drone deliveries*;
  else
    ignore *restored child*;
  end
else
  remove *individual* with maximum *fitness value*;
  insert *restored child* in *population*;
  set *tentative solution* as *educated drone deliveries*;
end

3.4. Hybrid Genetic Algorithm—Infeasible Variant

The second proposed variant of the evolutionary-based solution approach of the formulated mFSTSP is the *Hybrid Genetic Algorithm —Infeasible Variant*, shown in Algorithm 16. The main difference with the *Hybrid Genetic Algorithm—Feasible Variant* is that the requirement of feasible solution output of the $split()$ function, shown in Algorithm 17, does not hold, and the definition of the additional $repair()$ function, shown in Algorithm 18, is needed. The second variant considers infeasible individuals with the aim of diversifying the exploration of the search space and avoiding incurring local minima.

Algorithm 16 Main body of the *Hybrid Genetic Algorithm—Infeasible Variant*

Initialize: $response,\ truck\ only\ nodes,\ s_R,\ s_L,\ v,\ \varepsilon,\ \delta_t,\ \omega,\ \eta,\ \mu$;
generate randomly $population$ of n $individuals$; $\gamma = 0$;
while $\gamma < treshold$ **do**
  $\gamma = \gamma + 1$;
  $parents\ selection(population, fit\ values)$;
  $child\ generation(parents, drone\ deliveries)$;
  $split(child)$;
  $local\ search(split\ child)$;
  $restore(educated\ child, educated\ drone\ deliveries)$;
  if $split\ child$ is infeasible **do**
    sample randomly $P \in [0,100]$;
    if $P > 50$ **do**
      $repair(restored\ truck\ troute)$;
      $restore(repaired\ child, repaired\ drone\ deliveries)$;
      update $population$;
    else
      update $population$; **increase** ω;
    end
  else
    update $population$; **decrease** ω;
  end
  if $infeasible\ individuals > \eta\ ||\ feasible\ individuals > \mu$ **do**
    $select\ survivors(population)$;
  end
end
$select\ fittest\ individuals(population, fitness\ values)$;

In this case, since $tsp\ tour$ and $drones\ deliveries$ are generated pseudo-randomly, both feasible and infeasible solutions are considered. Then, according to a well-defined probability P, the $local\ search()$ function is entitled to either improving the pseudo-random feasible mFSTSP solution or repairing the infeasible solution. η and μ denote the maximum number of infeasible and feasible individuals in the population. The penalizing term ω is defined as a penalizing factor for the travel time computation when the feasibility constraints are violated, as follows:

$$t_p = t + \omega \cdot \max\left(0,\ \sum_{j \in sub-route}^{i=j-1} \tau_{ij},\ \ \tau'_{ijq} + \tau'_{jkq}\right) \tag{33}$$

where t_p is a penalizing cost in the truck time vector τ at each node, t is the actual truck travel time at the node, $\sum_{j \in sub-route}^{i=j-1} \tau_{ij}$ denotes the truck travel time for serving all customers of the $sub-route$ and $\tau'_{ij} + \tau'_{jk}$ is the travel time of a UAV in order to complete the $uav\ sortie$: $(i, j, k)_q$. In case many infeasible solutions are being produced by the $split()$ function,

increment variable reduces the number of nodes randomly assigned to the UAVs. This contains the number of infeasible solutions in the customers node set if the node locations are too far to be served by the UAVs. The *create uav sortie*() function assigns all the customers to the *uav sortie* independently on their feasibility.

Algorithm 17 *split* () function of *Hybrid Genetic Algorithm—Infeasible Variant*

Input: $child$;
Output: $split\ child,\ t,\ split\ truck\ sub-routes,\ split\ drone\ deliveries$;
select n_d as in Equation (32);
sample randomly n_d $drone\ nodes$ from C';
if $number\ of\ infeasible\ solutions = \eta\ ||\ increment\ variable = 0$ **do**
    $increment\ variable = increment\ variable + 1$;
    if $increment\ variable = \frac{n_d}{2}$ **do**
        $increment\ variable = 0;\ n_d = 1$;
    end
end
$n_d = n_d - increment\ variable$;
remove assigned $drone\ nodes$ from $child$; **compute** t for $child$;
reset $drone\ nodes,\ C$';
$create\ uav\ sorties(child, drone\ nodes, n_d, t)$;
$truck\ sub-routes\ update(child, uav\ sorties, v)$;
$time\ update(truck\ sub-routes, drone\ nodes)$;

Algorithm 18 *repair* () function

Input: *restored truck route*;
Output: *repaired child*, *repaired drone deliveries*;
Initialize: *drone deliveries* = ∅;
add depots to *restored truck route*;
local search(restored truck route, drone deliveries);

3.5. Ad-hoc Method

The alternative solution method *Ad-hoc Method*, shown in Algorithm 19, is a greedy algorithm that differs from the other optimization algorithms that iteratively try to improve a set of sub-optimal solutions, and generates a unique solution with a systematic procedure. A greedy algorithm is a simple, intuitive, and often efficient algorithmic approach that makes locally optimal choices at each step, aiming at finding an optimal solution. At each step of the algorithm, a well-defined strategy builds the best possible solution at the current step, without considering the overall consequences of that choice. It follows a general outline of how a greedy algorithm works:

- *Algorithm Initialization*: The algorithm starts with an empty or partial solution.
- *Greedy Strategy*: At each step, the algorithm makes a locally optimal choice based on a certain rule. It selects the most favorable available at the current stage of the process, without considering its impact on future steps.
- *Solution Feasibility and Update*: The algorithm checks if the selected choice is feasible or satisfies certain constraints. If the choice violates any constraints, it may be discarded,

and an alternative option may be considered. The selected choice is added to the current solution or modifies the partial solution.

- *Algorithm Termination*: The algorithm iterates through *Greedy Strategy* and *Solution Feasibility and Update* steps until a termination condition is met. This condition could be reaching a final solution, satisfying a specific objective, or exhausting the available choices.

Greedy algorithms do not guarantee an optimal solution for all problems, as in this case. Therefore, the correctness and optimality of a greedy algorithm depend on the specific problem being solved, the methodology exploited by the greedy algorithm itself and the properties of the problem instance. Greedy algorithms are often used when finding an optimal solution for a problem requires a lot of computational resources or is not feasible within a reasonable time frame. Additionally, greedy algorithms are commonly used as building blocks or components within more complex algorithmic approaches.

The greedy approach proposed in this work for the formulated mFSTSP is inspired by the work in [38], which also formulated a greedy strategy for a mFSTSP. The proposed greedy solution consists of a poor computationally expensive ad-hoc resolution method of the formulated mFSTSP problem that computes a feasible schedule solution according to a specific methodology of synchronization between the UAVs' delivery schedules and the truck's delivery schedule.

Algorithm 19 Main body of the *Ad-hoc Method*

Initialize: $response,\ truck\ input,\ s_R,\ s_L,\ v,\ \varepsilon,\ \delta_t,\ truck\ only\ nodes;$
$LTL = \left\lceil \frac{c-v}{v+1} \right\rceil;\ low\ bound = \infty;$
while $LTL <$ length($truck\ input$) **do**
  $C' = C \backslash truck\ only\ nodes;$
  initialize $tsp\ tour = [0,0];$
  $cost\ truck\ calculation(tsp\ tour, truck\ input, C', LTL);$
  $customer\ division(tsp\ tour, C');$
  $time\ update(tsp\ tour);$
  $create\ uav\ sorties(tsp\ tour, uav\ customers, time, v);$
  $insert\ infeasible\ customers(tsp\ tour);$
  $create\ truck\ sub-routes(tsp\ tour, drone\ nodes);$
  if $makespan > low\ bound$ **&** $LTL \leq$ length($truck\ input$) **&** $assignment$ is feasible **do**
    save mFSTSP solution;
    update $low\ bound$; **increase** LTL;
  else
    increase LTL;
  end
end

LTL (Lower Truck Limit) denotes the minimum number of customers that can be served by the truck if v UAVs are available. Maximizing the number of customers to be served by the UAVs aims at reducing the total completion time as much as possible and, at the same time, LTL may be increased for feasibility reasons if the maximum number of customers cannot be assigned to the UAVs. C' and $tsp\ tour$ are initialized at every loop start and the $cost\ truck\ calculation()$ function, shown in Algorithm 20, outputs the

shortest possible *tsp tour* given LTL customers. Firstly, the $truck - only\ nodes$ are inserted in the $tsp\ tour$, since they cannot be assigned to any UAV; secondly, the *customers nodes* are inserted in the $tsp\ tour$ from the $truck\ input$.

The $customer\ division()$ function decouples the nodes to be served by the truck and the nodes to be served by the UAVs. Then, the proposed algorithm tentatively creates a sub-optimal mFSTSP solution according to the available $tsp\ tour$ and $uav\ customers$, and repeats the process increasing LTL if the solution is not feasible or if a reduction in the makespan can be made, even if the solution is feasible.

Algorithm 20 *cost truck calculation* () function

Input: $tsp\ tour,\ truck\ input,\ truck\ only\ nodes,\ LTL;$
Output: $tsp\ tour;$
while $index < LTL$ **do**
  $min\ cost = \infty;$
  if $truck\ only\ nodes \neq \emptyset$ **do**
    insert j in $infeasible\ customers;$
    for $j \in truck\ only\ nodes$ **do**
      for $i \in tsp\ tour$ **do**
        find k: node after i in $tsp\ tour;$
        $cost = \tau_{ij} + \delta_t + \tau_{jk} - \tau_{ik};$
        if $cost < min\ cost$ **do**
          $j^* = j;\ k^* = k;\ min\ cost = cost;$
        end
      end
    end
    insert j^* before $k^*;\ index = index + 1;$
  else
    for $j \in truck\ input$ **do**
      for $i \in tsp\ tour$ **do**
        find k: node after i in $tsp\ tour;$
        $cost = \tau_{ij} + \delta_t + \tau_{jk} - \tau_{ik};$
        if $cost < min\ cost$ **do**
          $j^* = j;\ k^* = k;\ min\ cost = cost;$
        end
      end
    end
    insert j^* before $k^*;\ index = index + 1;$
  end
end

3.6. Simulation Settings

The proposed greedy algorithms are implemented by means of the *Python 3.9* programming language. In particular, the *DEAP* library is exploited for ease of management of

the genetic algorithm variables. The settings of the evolutionary algorithms' parameters are as follows: $\gamma = 1000$, $n = 20$, $\omega = 1$, $\eta = 15$, $\mu = 25$. The parameter settings for the fleet of UAVs are reported in Table 1. Note that all the customers nodes are considered eligible for the UAVs. As far as the optimization of the process' makespan, this choice allows us to highlight the benefits of drones' usage in cooperation with a truck.

Table 1. Fleet of UAVs' parameters setting.

Parameter	Value
ε [s]	1500
s_R [s]	60
s_L [s]	60
UAV average speed [m/s]	11.1
c'	c

The delivery operational area of either the truck or the drones is defined as a rectangular portion of the city of Turin, Italy. The fleet is assumed to be composed of $v = 2$ UAVs. δ_t is set to 200 seconds. The truck-only TSP solutions are obtained with the *Google OR tools* solver [42], with the parameter *local search metaheuristic* set as *guided local search* and the parameter *time limit* set to *30 s*. The capability of the proposed approaches of managing the formulated mFSTSP is tested considering two scenarios and the results are corroborated by Monte Carlo simulations. *Scenario A* consists of $c = 10$ customers to be served (plus the depot, denoted with 0, from which the truck departs and returns to end the delivery cycle), randomly sampled in the operational area. Table 2 shows the customers' coordinates. Such coordinates are given to the *Bing Maps API REST Services* [43] in order to obtain the truck travel times and the drones travel times, encoded as cost matrixes. *Scenario B* is defined analogously, this time considering $c = 20$ customers to be served in a larger rectangular area defined in the city of Rome, Italy. This is done to test the proposed solutions with a more challenging scenario, with a higher number of customers and greater travel distances. Table 3 shows the randomly sampled customers' coordinates, the corresponding cost matrixes are derived with the same procedure adopted for *Scenario A*.

Table 2. Customer nodes' coordinates for *Scenario A* (10 customers and 1 depot), generated randomly via [44] in a rectangular map comprehending the Turin urban area.

Customer Node ID	Latitude [°]	Longitude [°]
0	45.0902	7.8638
1	45.0047	7.8187
2	45.9967	7.5445
3	45.1337	7.6307
4	45.0593	7.7530
5	45.0595	7.7048
6	45.1062	7.6360
7	45.1539	7.8095
8	45.0488	7.6658
9	45.0314	7.6910
10	45.0925	7.8468

Table 3. Customer nodes' coordinates for *Scenario B* (20 customers and 1 depot), generated randomly via [44] in a rectangular map comprehending the Rome urban area.

Customer Node ID	Latitude [°]	Longitude [°]
0	41.8958	12.6744
1	41.8908	12.6671
2	41.7953	12.6604
3	41.0029	12.4694
4	41.8829	12.4764
5	41.8375	12.6567
6	41.9343	12.3555
7	41.8671	12.5108
8	41.8006	12.5615
9	41.8086	12.5555
10	41.9577	12.3447
11	41.9377	12.3337
12	41.8076	12.5186
13	41.9187	12.4850
14	41.9137	12.5509
15	41.9083	12.3323
16	41.9459	12.4253
17	41.8403	12.4315
18	41.9021	12.4312
19	41.8344	12.5388
20	41.8054	12.3754

4. Results and Discussion

Considering either *Scenario A* or *Scenario B*, the improvement that is obtained in terms of total completion time adopting the UAVs in combination with the truck with respect to the conventional truck-only TSP solution is shown in Table 4. Considering a single iteration of the proposed algorithms, Table 4 also shows the level of optimality in terms of the makespan of the delivery cycle that can be achieved by each routing method for the two scenarios of references.

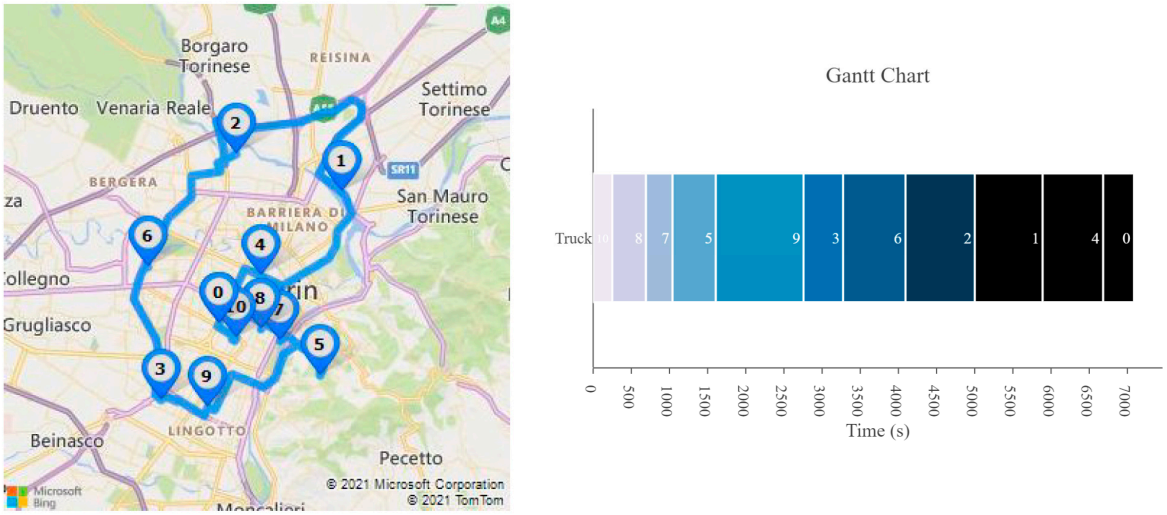
Table 4. Makespan and savings over truck-only TSP of each proposed solution for *Scenario A* and *Scenario B*.

Scenario	Algorithm	Makespan [s]	Savings over Truck-Only TSP [%]
A	LSA	6700	5.5
	HGA-Feasible Variant	3028	57.3
	HGA-Infeasible Variant	3028	57.3
	Ad-hoc Method	3223	54.5
B	LSA	18,689	5.6
	HGA-Feasible Variant	13,254	33.1
	HGA-Infeasible Variant	15,392	22.2
	Ad-hoc Method	18,215	8

The results show that the enhanced solution space exploration capability of the genetic algorithms results in the highest savings over the TSP solutions among the proposed algorithms, regardless of the instance size. This is also confirmed in the works [31–34] where evolutionary-based methods outperform most of the alternative approaches.

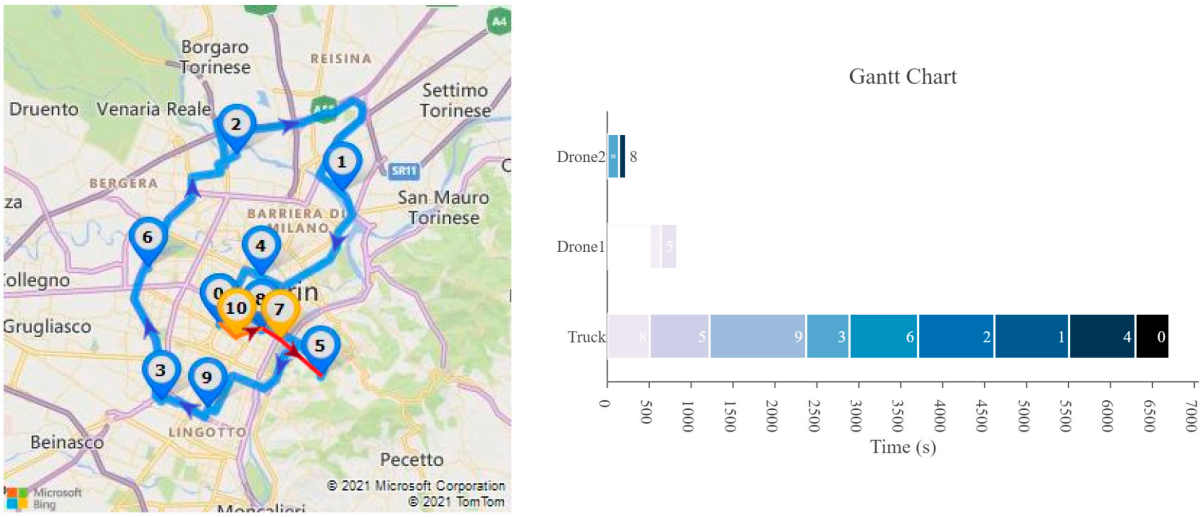
Nevertheless, the computational complexity of the evolutionary approach is not justified for small size instances. This is confirmed by the results of *Scenario A*: the greedy method generates a delivery schedule with a slightly lower optimality level (~3%) with respect to the evolutionary methods, but with a noticeable gain of computational time. This is no longer the case when a bigger instance size is considered, as shown by the

results of *Scenario B*. The differences between the two variants of the evolutionary approach emerge in *Scenario B*, as the feasible variant is more likely to find a better optimal solution when the maximum number of iterations is limited. When the search space is broader, the genetic material of feasible solutions that is passed from one population to the next one represents a sub-optimal search direction of the solution space. If the size of the problem instance increases, this is most likely to produce a better optimal solution with respect to the infeasible variant. For the sake of clarity, a visual representation of the solutions obtained with the TSP solver and the proposed routing algorithms is given in Figures 2–6 for *Scenario A*, starting from the conventional TSP solution up to the *Ad-hoc Method* solution. Each of the mentioned figures shows the truck and drones' travels on the map of the portion of the city of Turin, Italy, and the corresponding Gantt Chart with the timed schedules.



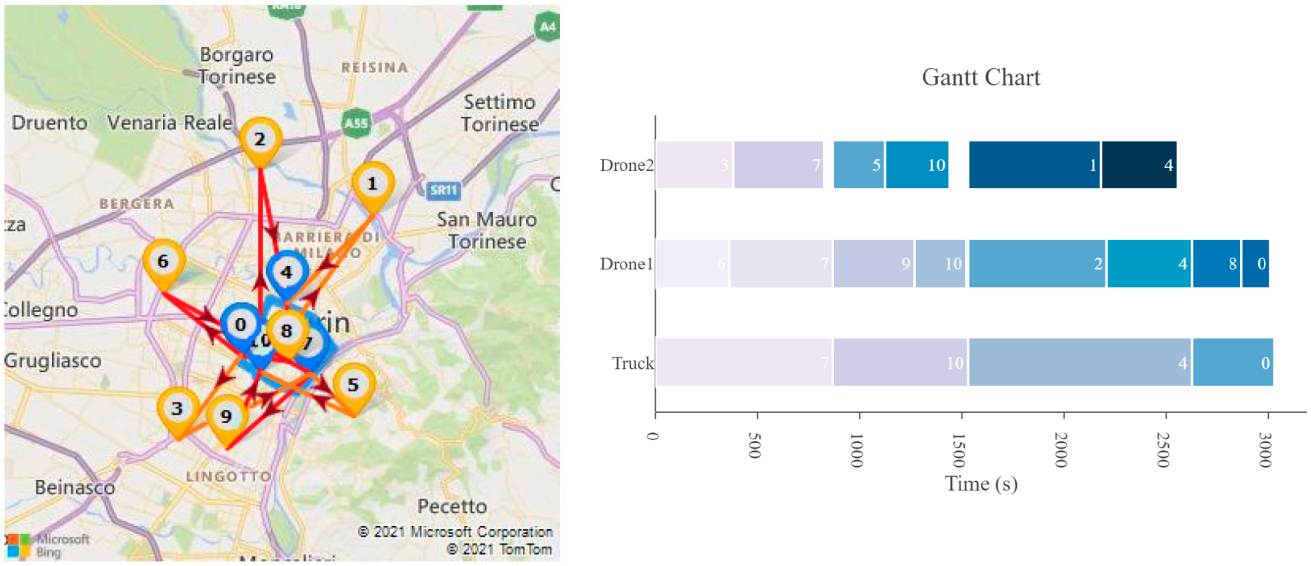
(**a**) (**b**)

Figure 2. TSP solution representations of *Scenario A*. (**a**) TSP tour reported on the map of the operational area, with the paths of the truck in blue; (**b**) Gantt chart of the TSP solution. The numbers from 1 to 10 in the map and the Gant chart represent the Customer Node IDs.



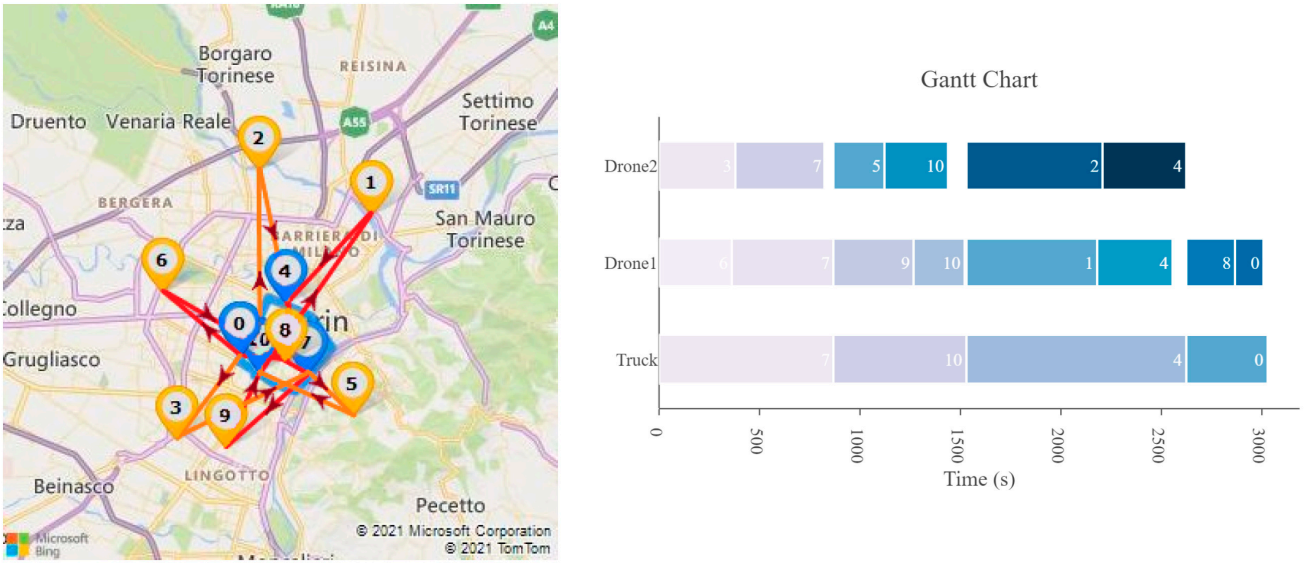
(**a**) (**b**)

Figure 3. *Local Search Algorithm* solution representations of *Scenario A*. (**a**) mFSTSP tour reported on the map of the operational area, with the paths of the truck (blue), drone 1 (orange) and drone 2 (red); (**b**) Gantt chart o f the mFSTSP solution. The numbers from 1 to 10 in the map and the Gant chart represent the Customer Node IDs.



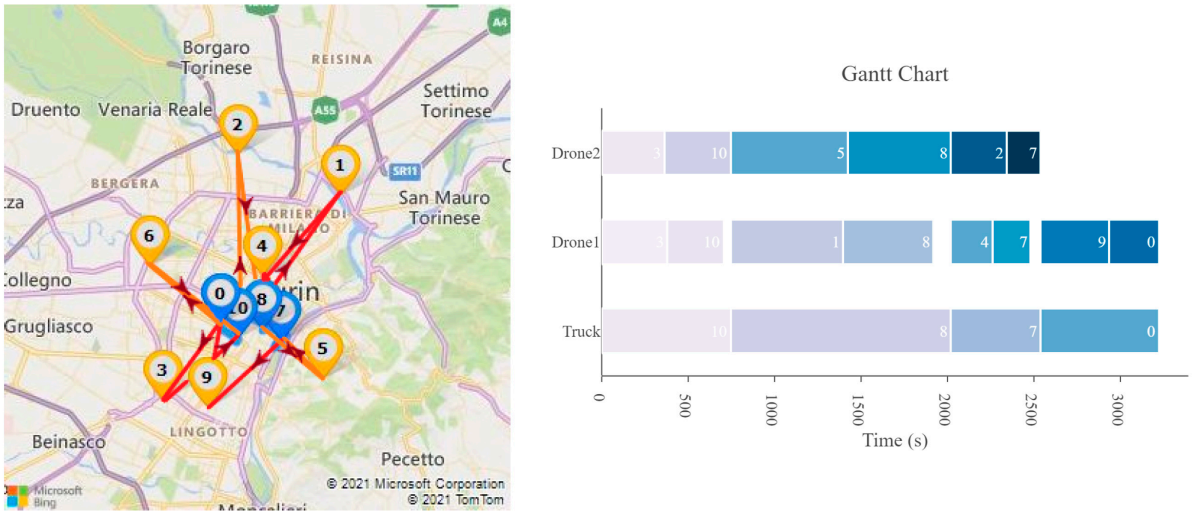
(**a**) (**b**)

Figure 4. *Hybrid Genetic Algorithm—Feasible Variant* solution representations of *Scenario A*. (**a**) mFSTSP tour reported on the map of the operational area, with the paths of the truck (blue), drone 1 (orange) and drone 2 (red); (**b**) Gantt chart of the mFSTSP solution. The numbers from 1 to 10 in the map and the Gant chart represent the Customer Node IDs.



(**a**) (**b**)

Figure 5. *Hybrid Genetic Algorithm—Infeasible Variant* solution representations of *Scenario A*. (**a**) mFSTSP tour reported on the map of the operational area, with the paths of the truck (blue), drone 1 (orange) and drone 2 (red); (**b**) Gantt chart of the mFSTSP solution. The numbers from 1 to 10 in the map and the Gant chart represent the Customer Node IDs.



(**a**) (**b**)

Figure 6. *Ad-hoc Method* solution representations of *Scenario A*. (**a**) mFSTSP tour reported on the map of the operational area, with the paths of the truck (blue), drone 1 (orange) and drone 2 (red); (**b**) Gantt chart of the mFSTSP solution. The numbers from 1 to 10 in the map and the Gant chart represent the Customer Node IDs.

Considering either *Scenario A* or *Scenario B*, one hundred Monte Carlo simulations are performed for each scenario in order to corroborate the capability of the proposed evolutionary algorithms to handle the formulated mFSTSP. Table 5 shows the results. The infeasible variant of the evolutionary-based task scheduling method has enhanced exploration capability of the solution space and it is less likely to stop at local minima too early. This is demonstrated with *Scenario A*, where both the minimum and the maximum makespan results are smaller than the ones related to the feasible variant. Nevertheless, with a more complex scenario and a bigger operational area, i.e., as in *Scenario B*, the optimality level of the solution produced by the *Hybrid Genetic Algorithm—Infeasible Variant* is worse. This is due to the fact that, when the search space is much broader as for this scenario, the constraint of considering only feasible schedules during the crossover phase restricts the investigated solutions to the neighborhood of the feasible solutions and this is more likely to generate a more optimal schedule, given a "reasonable" maximum number of iterations of the genetic algorithms. As expected, and accordingly to the search strategies of the two evolutionary-based methods, the variability of the results is smaller for the feasible variant than for the infeasible variant. By paying the price of heavier computational loads, both the variants of the evolutionary-based methods ensure a greater solution's optimality level than either the greedy *Local Search Algorithm* or the *Ad-hoc Method*, for both *Scenario A* and *Scenario B*.

Table 5. Monte Carlo simulation results: total completion time of the delivery process.

Scenario	**Algorithm**	**Minimum [s]**	**Maximum [s]**	**Average [s]**
A	LSA	-	-	6700
	HGA-Feasible Variant	2951	3127	3041
	HGA-Infeasible Variant	2916	3107	3040
	Ad-hoc Method	-	-	3223
B	LSA	-	-	18,689
	HGA-Feasible Variant	12,679	14,029	13,254
	HGA-Infeasible Variant	13,163	20,242	15,393
	Ad-hoc Method	-	-	18,215

As far as the problem addressed in this work is concerned, the choice of the most suitable method between the feasible and the infeasible variants depends on the instance of the problem itself and on the available computational resources. Considering both the inherent features of the proposed routing algorithms and the simulation results, the summarizing considerations of this work are reported in Table 6.

Table 6. Summarizing consideration about the proposed routing algorithms.

Algorithm	Advantages	Disadvantages
Local Search	• Flexibility • Good computational efficiency with respect to neighborhood size	• Local optimality • Limited explainability • Poor hill climbing capability
Evolutionary	• Enhanced exploration capabilities of search space	• Limited explainability
	• Hill climbing through randomness and solution mutation	• Computational complexity
	• Directional exploration of search space • Parallel computing • Solution optimality	• Extensive tuning of evolutionary parameters (population dimension, probability of mutation, . . .)
Greedy	• Intuitiveness • Computational efficiency • Explainability	• Local optimality • Poor flexibility • Difficulty in recognition of bad decisions • Deep understanding of the problem

5. Conclusions

In this paper, a few heuristic methods were designed, implemented, and compared for a last-mile delivery (LMD) Travelling Salesman Problem with a truck and multiple drones. Firstly, a mathematical formulation of the problem was proposed considering the case of a truck that cooperates synchronously with a fleet of Unmanned Aerial Vehicles. The problem formulation accounts for most of the key representative elements of an LMD service, such as the delivery due dates of the parcels, the service time required to process the orders, the battery capacities of the UAVs, etc. The goal was to find a combined truck-UAVs feasible schedule to minimize the total completion time of the delivery process while respecting the constraints related to either the delivery tasks or the vehicles' capacities. Secondly, given the addressed NP-hard problem, we designed a local search solution approach, two variants of an evolutionary-based solution and an ad-hoc heuristic solution specifically designed for addressing the formulated problem. The latter was conceptually different from the other proposed methods as it generated a solution systematically, considering the features of the specific problem. The evolutionary-based strategy was split into two variants depending on the feasibility feature of the solutions that were permutated at each execution step. The proposed solutions were evaluated over two realistic urban air mobility scenarios. The capability of the methods to produce an optimal schedule was corroborated through Monte Carlo simulation campaigns.

Future research directions will include the extension of the problem formulation to multiple trucks and heterogenous drones and the design of a dynamic task allocation strategy in order to account for environmental uncertainties and order modifications during task execution. The design and implementation of other routing methods may also be investigated and compared to the algorithms proposed in this work.

Author Contributions: Conceptualization, M.R. and S.P.; methodology, M.R. and S.P.; software, M.R.; investigation, M.R.; data curation, M.R.; writing—original draft preparation, M.R. and S.P; writing—review and editing, M.R., S.P., J.R. and G.G.; supervision, G.G.; project administration

M.B., J.R. and G.G.; funding acquisition, M.B., J.R. and G.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used in the current study are available from the corresponding author upon reasonable request.

Acknowledgments: The authors wish to thank the Piedmont Aerospace District for supporting the PhD activity of Marco Rinaldi.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. What Is Last Mile Delivery? Costs & How to Optimize. Available online: https://optimoroute.com/last-mile-delivery/ (accessed on 24 May 2023).
2. Amazon's New Prime Air Drone Features a Weird Tailsitter Design. Available online: https://spectrum.ieee.org/amazon-redesigned-prime-air-drone (accessed on 24 May 2023).
3. A Drone Program Taking Flight. Available online: https://www.aboutamazon.com/news/transportation/a-drone-program-taking-flight/ (accessed on 24 May 2023).
4. Eskandaripour, H.; Boldsaikhan, E. Last-Mile Drone Delivery: Past, Present, and Future. *Drones* **2023**, *7*, 77. [CrossRef]
5. Cohen, A.; Shaheen, A.; Farrar, E. Urban Air Mobility: History, Ecosystem, Market Potential, and Challenges. *IEEE Trans. Intell. Transp. Syst.* **2022**, *22*, 6074–6087. [CrossRef]
6. Nawaz, H.; Ali, H.M.; Massan, S.U.R. Applications of unmanned aerial vehicles: A review. *3c Tecnol. Glosas Innovac. Apl. Pyme* **2019**, 85–105. [CrossRef]
7. Mohsan, S.A.H.; Khan, M.A.; Noor, F.; Ullah, I.; Alsharif, M.H. Towards the Unmanned Aerial Vehicles (UAVs): A Comprehensive Review. *Drones* **2022**, *6*, 147. [CrossRef]
8. Al-Dosari, K.; Hunaiti, Z.; Balachandran, W. Systematic Review on Civilian Drones in Safety and Security Applications. *Drones* **2023**, *7*, 210. [CrossRef]
9. Rábago, J.; Portuguez-Castro, M. Use of Drone Photogrammetry as an Innovative, Competency-Based Architecture Teaching Process. *Drones* **2023**, *7*, 187. [CrossRef]
10. Skrinjar, J.P.; Skorput, P.; Furdic, M. Application of Unmanned Aerial Vehicles in Logistic Processes. In Proceedings of the New Technologies and Applications (NT-2018), Sarajevo, Bosnia-Herzegovina, 28–30 June 2018.
11. Li, X.; Tupayachi, J.; Sharmin, A.; Martinez Ferguson, M. Drone-Aided Delivery Methods, Challenge, and the Future: A Methodological Review. *Drones* **2023**, *7*, 191. [CrossRef]
12. Aurambout, J.P.; Gkoumas, K.; Ciuffo, B. Last mile delivery by drones: An estimation of viable market potential and access to citizens across European cities. *Eur. Transp. Res. Rev.* **2019**, *11*, 30. [CrossRef]
13. Wang, D.; Hu, P.; Du, J.; Zhou, P.; Deng, T.; Hu, M. Routing and Scheduling for Hybrid Truck-Drone Collaborative Parcel Delivery with Independent and Truck-Carried Drones. *IEEE Internet Things J.* **2019**, *6*, 10483–10495. [CrossRef]
14. Chung, S.H.; 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. [CrossRef]
15. Macrina, G.; Pugliese, L.D.; Guerriero, F.; Laporte, G. Drone-aided routing: A literature review. *Transp. Res. Part C-Emerg. Technol.* **2020**, *120*, 102762. [CrossRef]
16. Khoufi, I.; Laouiti, A.; Adjih, C. A Survey of Recent Extended Variants of the Traveling Salesman and Vehicle Routing Problems for Unmanned Aerial Vehicles. *Drones* **2019**, *3*, 66. [CrossRef]
17. Potvin, J.Y. Genetic algorithms for the traveling salesman problem. *Ann. Oper. Res.* **1996**, *63*, 339–370. [CrossRef]
18. Wang, K.P.; Huang, L.; Zhou, C.G.; Pang, W. Particle swarm optimization for travelling salesman problem. In Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat.No.03EX693), Xi'an, China, 5 November 2003.
19. Yan, T.; Lu, F.; Wang, S.; Wang, L.; Bi, H. A hybrid metaheuristic algorithm for the multi-objective location-routing problem in the early post-disaster stage. *J. Ind. Manag. Optim.* **2023**, *19*, 4663–4691. [CrossRef]
20. Zhan, S.H.; Lin, J.; Zhang, Z.J.; Zhong, Y.W. List-Based Simulated Annealing Algorithm for Traveling Salesman Problem. *Comput. Intell. Neurosci.* **2016**, *2016*, 1712630. [CrossRef]
21. Wen, H.; Wang, S.X.; Lu, F.Q.; Feng, M.; Wang, L.Z.; Xiong, J.K.; Si, M.C. Colony search optimization algorithm using global optimization. *J. Supercomput.* **2022**, *78*, 6567–6611. [CrossRef]
22. Lu, F.; Feng, W.; Gao, M.; Bi, H.; Wang, S. The Fourth-Party Logistics Routing Problem Using Ant Colony System-Improved Grey Wolf Optimization. *J. Adv. Transp.* **2020**, *2020*, 8831746. [CrossRef]
23. Cheikhrouhou, O.; Khoufi, I. A Comprehensive Survey on the Multiple Travelling Salesman Problem: Applications, Approaches and Taxonomy. *Comput. Sci. Rev.* **2021**, *40*, 100369. [CrossRef]

24. Euchi, J.; Sadok, A. Hybrid genetic-sweep algorithm to solve the vehicle routing problem with drones. *Phys. Commun.* **2021**, *44*, 101236. [CrossRef]
25. Dell'Amico, M.; Montemanni, R.; Novellani, S. Modeling the flying sidekick traveling salesman problem with multiple drones. *Networks* **2021**, *78*, 303–327. [CrossRef]
26. Moadab, A.; Farajzadeh, F.; Fatahi Valilai, O. Drone routing problem model for last-mile delivery using the public transportation capacity as moving charging stations. *Sci. Rep.* **2022**, *12*, 6361. [CrossRef]
27. Kitjacharoenchai, P.; Min, B.; Lee, S. Two echelon vehicle routing problem with drones in last mile delivery. *Int. J. Prod. Econ.* **2020**, *225*, 107598. [CrossRef]
28. Ha, Q.M.; Deville, Y.; Pham, Q.D.; Hà, M.H. A hybrid genetic algorithm for the traveling salesman problem with drone. *J. Heurist.* **2020**, *26*, 219–247. [CrossRef]
29. Song, K.; Ho, M.J. Ground Vehicle and Drone Collaborative Delivery Planning using Genetic Algorithm. *J. Aerosp. Syst. Eng.* **2020**, *14*, 1–9.
30. Ferrández, S.; Harbison, T.; Weber, T.; Sturges, R.H.; Rich, R. Optimization of a truck-drone in tandem delivery network using K-means and genetic algorithm. *J. Ind. Eng. Manag.* **2016**, *9*, 374–388.
31. Moeini, M.; Salewski, H. A Genetic Algorithm for Solving the Truck-Drone-ATV Routing Problem. In Proceedings of the 6th World Congress on Global Optimization (WCGO 2019), Metz, France, 8–10 July 2019.
32. Hazama, Y.; Iima, H.; Karuno, Y.; Mishima, K. Genetic algorithm for scheduling of parcel delivery by drones. *J. Adv. Mech. Des. Syst. Manuf.* **2021**, *15*, 69–80. [CrossRef]
33. Özoğlu, B.; Çakmak, E.; Koç, T. Clarke & Wright's Savings Algorithm and Genetic Algorithms Based Hybrid Approach for Flying Sidekick Traveling Salesman Problem. *Eur. J. Sci. Technol.* **2019**, 185–192. [CrossRef]
34. Peng, K.; Du, J.; Lu, F.; Sun, Q.; Dong, Y.; Zhou, P.; Hu, M. A Hybrid Genetic Algorithm on Routing and Scheduling for Vehicle-Assisted Multi-Drone Parcel Delivery. *IEEE Access* **2019**, *7*, 49191–49200. [CrossRef]
35. Lin, M.; Lyu, J.; Gao, J.; Li, L. Model and Hybrid Algorithm of Collaborative Distribution System with Multiple Drones and a Truck. *Sci. Program* **2020**, 1–16. [CrossRef]
36. Peng, X.; Peng, S.; Zhang, L. Optimization of a truck-UAVs delivery based on FSTSP model with multipath genetic algorithm. In Proceedings of the 22nd Annual IEEE International Conference on Electro Information Technology (eit2022), Mankato, MN, USA, 19–21 May 2022.
37. Delazeri, G.; Ritt, M. Fast Heuristics for Traveling Salesman Problems with Multiple Flying Sidekicks. In Proceedings of the 2021 IEEE Congress on Evolutionary Computation (CEC2021), Kraków, Poland, 28 June–1 July 2021.
38. Murray, C.C.; Raj, R. The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones. *Transp. Res. Part C-Emerg. Technol.* **2020**, *110*, 368–398. [CrossRef]
39. Raj, R.; Murray, C.C. The multiple flying sidekicks traveling salesman problem with variable drone speeds. *Transp. Res. Part C-Emerg. Technol.* **2020**, *120*, 102813. [CrossRef]
40. Murray, C.C.; Chu, A.G. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transp. Res. Part C-Emerg. Technol.* **2015**, *54*, 86–109. [CrossRef]
41. Moshref-Javadi, M.; Hemmati, A.; Winkenbach, M. A truck and drones model for last-mile delivery: A mathematical model and heuristic approach. *Appl. Math. Model.* **2020**, *80*, 290–318. [CrossRef]
42. Google OR-Tools. Available online: https://developers.google.com/optimization/routing/routing_options?hl=en (accessed on 3 April 2023).
43. Bing Maps REST Services. Available online: https://learn.microsoft.com/en-us/bingmaps/rest-services/ (accessed on 3 April 2023).
44. Random Point Generator. Available online: http://www.geomidpoint.com/random/ (accessed on 3 April 2023).