Università degli Studi di Padova

Dipartimento di ingegneria dell'informazione

UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di ingegneria dell'informazione

Corso di laurea magistrale in ingegneria informatica

An Exact and Heuristic Approach for the Traveling Salesman Problem with Drone and Variable Drone Speeds Selection

Relatore:
Prof. Roberto Roberti

Laureando:
Federico Michelotto

ANNO ACCADEMICO: 2021-2022

Data di laurea: 17 Ottobre 2022

Sommario

Grazie al rapido avanzamento tecnologico avvenuto negli ultimi anni, i sistemi di consegna via drone stanno diventando sempre più popolari. In questa tesi, proponiamo un metodo esatto e un metodo euristico per il problema del *Flying Sidekick Traveling Salesman Problem with Variable Drone Speeds Selection* (FSTSP-VDS), un'estensione del *Flying Sidekick Traveling Salesman Problem* (FSTSP) in cui un veicolo tradizionale (e.g., un furgone) è assistito da un drone che può viaggiare a velocità variabile con l'obiettivo di servire un insieme di clienti nel minore tempo possibile. In particolare, ci concentriamo sul caso generico in cui il consumo energetico del drone è caratterizzato da un modello non lineare rispetto alla velocità e al peso trasportato. Mostriamo che, se la funzione che descrive il consumo di energia per metro del drone è convessa, allora, i tempi di percorrenza minimi e massimi del drone, possono essere calcolati utilizzando metodi standard di ottimizzazione convessa. Sotto questa ipotesi, proponiamo la prima formulazione di *Programmazione Lineare Intera Mista* (PLIM) per il problema del FSTSP-VDS. La formulazione che proponiamo ha il vantaggio di essere compatta (i.e., ha un numero polinomiale di variabli e vincoli) e permette di risolvere all'ottimo instanze fino a 25 clienti. Inoltre, presentiamo un algoritmo genetico per poter affrontare istanze con un numero maggiore di clienti. La qualità dell'algoritmo genetico proposto viene validato in due modi. Prima, lo compariamo con lo stato dell'arte per il problema del FSTSP-VDS, ottenendo un miglioramento medio del 6%. Infine, testiamo l'algoritmo genetico sul problema del *Traveling Salesman Problem with Drone* (TSP-D), un problema analogo al FSTSP in cui la velocità del drone è fissata, e mostriamo che il metodo sviluppato è il primo euristico capace di trovare soluzioni ottime note per istanze con 39 clienti. In particolare, delle 227 instanze valutate, 194 sono state risolte all'ottimo (85.5%), ottenendo un scarto medio complessivo dello 0.30%.

Abstract

Drone-assisted deliveries have gained popularity in the last few years thanks to the recent technological advances. In this thesis, we propose an exact and heuristic method for the *Flying Sidekick Traveling Salesman Problem with Variable Drone Speeds Selection* (FSTSP-VDS), an extension of the *Flying Sidekick Traveling Salesman Problem* (FSTSP) in which the standard vehicle (e.g., a truck) is combined with a drone that can fly at variable speeds to deliver parcels to customers with the objective to minimize the completion time. In particular, we focus on the generic case in which the drone power consumption is modeled by a nonlinear function with respect to speed and weight. We show that if the drone energy-per-meter function is convex (and this is true for the most popular drone power consumption models in literature), then, the minimum and maximum feasible drone travel times can be computed using a generic convex optimization solver. Under this assumption, we present the first *Mixed Integer Linear Programming* (MILP) formulation for the FSTSP-VDS. The formulation we present has the advantage to be compact (i.e., it features a polynomial number of variables and constraints) and allows to solve instances with up to 25 customers. Furthermore, we propose a genetic heuristic method to tackle larger FSTSP-VDS instances. The quality of our heuristic method is validated in two ways. First, we compared it with the current state of the art FSTSP-VDS heuristic, obtaining an average improvement of 6%. Then, we tested our heuristic on the *Traveling Salesman Problem with Drone* (TSP-D), a nearly identical problem to the FSTSP in which the drone flying speed is fixed, and show that our method is the first heuristic that can find known optimal solutions of TSP-D instances with up to 39 customers. In particular, for a total of 227 TSP-D with known optimal solutions evaluated, the genetic algorithm was to able to find the optimal solution for 194 instances (85.5%), with an average optimality gap of only 0.30%.

Acknowledgements

I really want to thank my supervisor Prof. Roberti for having supported me from the beginning to the end of this journey with insightful ideas and constructive comments. Since the first meeting, he was always present for any doubt I had, and his guidance was inspirational, both from a professional side and from a human perspective. I feel very grateful to have been supervised by Prof. Roberti.

I also want to thank Prof. Fischetti that, with his enthusiasm, made me passionate about the Operations Research discipline. It was a profound joy following his courses, and something I will not forget.

Last but not least, I want to thank my family, my sister Anna, my girlfriend Giulia and my friends, for having supported me along this journey.

Contents

1 Introduction **1**

2 Problem Definition **6**

3 Literature Review **9**

4 Drone Power Consumption **12**

4.1 Considerations 14

5 Exact Method **17**

5.1 Compact Formulation 17

5.2 Valid Inequalities 20

5.3 Big-M Value 20

5.4 Cutting Plane Approach 21

6 Genetic Algorithm **23**

6.1 Solution Representation 24

6.2 Method to retrieve the best solution from an individual 27

6.2.1 Linear Time Approximated Method 28

6.3 Evaluation of Individuals 31

6.4 Parent Selection and Crossover 32

6.5 Education 35

6.6 Population Management 35

7 Computational Results **37**

7.1 Test instances 38

7.1.1 FSTSP-VDS instances 38

7.1.2 TSP-D instances 39

7.2 Computational results on single drone FSTSP-VDS instances 40

7.2.1 Sensitivity analysis on optimal drone leg speeds 41

7.3 Computational results on TSP-D instances 43

8 Conclusions and Future Work **45**

A Extended Results **47**

List of Figures

1.1 Optimal tour of 42 cities in the USA. The problem was solved by Dantzig, Fulkerson, and Johnson in 1954. 2
1.2 Example of a FSTSP solution with 10 customers. 3

2.1 Example of solution of a FSTSP instance with eight customers. . . 8

4.1 Liu et al. [11] drone power consumption with respect to the drone speed v, for different payload weights. 13
4.2 Liu et al. [11] drone range in meters of a drone with a battery capacity of 500 kJ with respect to the drone speed v, for different payload weights. 13
4.3 Liu et al. [11] energy-per-meter function $E_w^{pm}(v)$ with respect to the drone speed v, for a payload weight w of 1 kg. 14

6.1 All the 12 solutions encoded by the sequence $(0, 1, 2, 3, 0')$. 26
6.2 Example of a single point crossover with a crossover point $c = 6$ between two individuals with a chromosome length $n = 14$. The resulting offspring is the concatenation of the first $c = 6$ elements of parent A with the last $n - c = 8$ elements of parent B. During the copy of the elements of parent B, values that are already present in the offspring sequence are discarded (values 9 and 5), in order to avoid duplicates. After the combine phase, the values $R = \{8, 11\}$ are not present in the offspring sequence because they reside in the last $n - c$ elements of parent A, and in the first c elements of parent B. During the repair phase, the left over values are added in the offspring chromosome. 33

List of Tables

3.1 Related works in which drone speeds can be selected from a continuous set, and the drone energy consumption is defined by a non-linear model. 11

4.1 Coefficient values for the power consumption model of Liu et al. [11] 14

7.1 Genetic algorithm parameters. 38
7.2 Computational results on the FSTS-VDS. 40
7.3 Comparison between continuous optimal and discrete optimal min/max drone travel times on FSTSP-VDS instances with 10 customers. In this setting, the discretization step of the drone speed domain is equal to $\delta = 0.5$ m/s. 42
7.4 Computational results on the TSP-D 43
7.5 Percentage differences of the GA approach solution times relative to the BP approach. 44

A.1 Extended results on the FSTS-VDS instances with 10 customers. . 48
A.2 Extended results on the FSTS-VDS instances with 25 customers. . 49
A.3 Extended results on the FSTS-VDS instances with 50 customers. . 50
A.4 Extended results on the FSTS-VDS instances with 100 customers. 51
A.5 Extended results on the TSP-D with 9 customers. 51
A.6 Extended results on the TSP-D with 19 customers. 53
A.7 Extended results on the TSP-D with 29 customers. 55
A.8 Extended results on the TSP-D with 39 customers 57

Chapter 1

Introduction

One of the most popular problem studied in combinatorial optimization is the *Traveling Salesman Problem* (TSP). The TSP can be formulated as follows: given n cities and the traveling cost to move from one city to another, the task is to find the shortest path to visit each city and returning to the starting point. In the symmetric version of the TSP, the cost of going from the node a to node b is the same of going from node b to node a. Whereas, in the asymmetric TSP, this is not true in general. Given an example with n cities, the number of possible solutions for the asymmetric TSP is $(n-1)!$ and $(n-1)!/2$ for the symmetric TSP. Therefore, when n is large, an exhaustive search is not a computationally viable approach. To give an idea, when n, the number of cities, is 65, there are about 10^{90} different solutions for the TSP. To put this number into prospective, just think that 10^{90} is the estimated number of atoms in the observable universe [14]. An example of an optimal solution for an instance with 42 nodes of the symmetric TSP is illustrated in Figure 1.1. The TSP arises naturally in an extremely wide number of practical applications ranging from planning trips to mapping genes and scheduling space-based telescopes [23]. Unfortunately, the TSP belongs to the class of NP-hard problems, which means that no polynomial time algorithms exist for this problem (assuming that P $\neq$ NP). However, to find the optimal solution it is not necessary to compute the cost of every possible solution. In fact, the TSP can be formulated as an integer linear programming problem, and solved by using techniques such as branch-and-bound and branch-and-cut, which allow to solve instances with thousands of nodes. Currently, the largest TSP instance ever solved has 85,900 nodes and it was solved back in 2006 by Applegate et al. [10] using the Concorde code [4], a program based on the branch-and-cut technique.

Over the past few years there has been a growing interest in delivery configurations that include unmanned aerial vehicles (UAVs, or drones). For example, different delivery companies such as UPS and Amazon started in the last

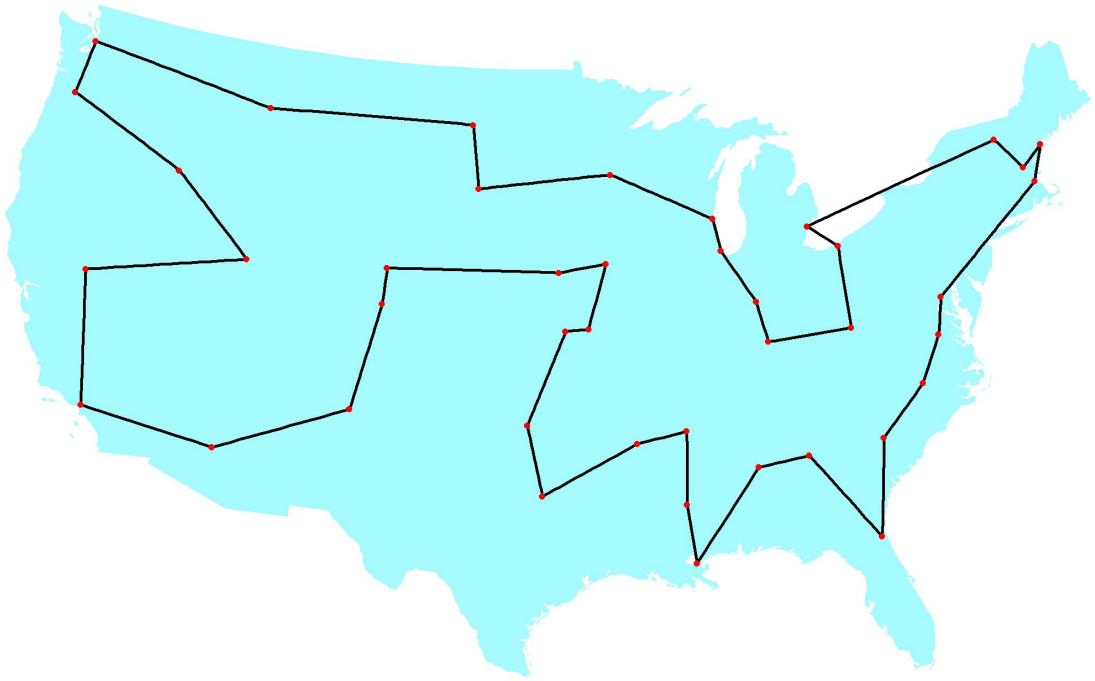

Figure 1.1: Optimal tour of 42 cities in the USA. The problem was solved by Dantzig, Fulkerson, and Johnson in 1954.

few years different projects that involve the use of drones with the aim at reducing delivery times, gas emissions and improving customer satisfaction [24] [1]. For these reasons, the research community began to study different routing problems in which the standard vehicles are supported by one or more drones. The first one of this kind was the *Flying Sidekick Traveling Salesman Problem* (FSTSP) introduced in 2015 by Murray and Chu [6]. The FSTSP is a an extension of the classical TSP in which the ground vehicle (e.g., a truck) is coupled with a drone with the goal of serving a set of customers in the minimum time or cost. In particular, the drone can be launched from the truck at a customer location or the depot, serve a customer, and then return back to the truck at a different location, with the constraint that if the drone arrives before the truck at the rendezvous location, it cannot land and wait, but it must hover until the arrival of the truck. An example of a solution for an instance of the FSTSP is shown in Figure 1.2. One common assumption of most drone-aided routing problems as the FSTSP, is that the drone endurance and flight range are constant and independent with respect to speed and payload. This hypothesis leads to the logical conclusion that increasing the drone flying speed up to the maximum allowable speed has only beneficial effects. In reality however, just like for cars, when the drone flying speed exceed a certain threshold, the drone power consumption grows approximately quadratically with respect to speed due to the wind resistance. Therefore, when dealing with more realistic drone power consumption models, reducing the drone

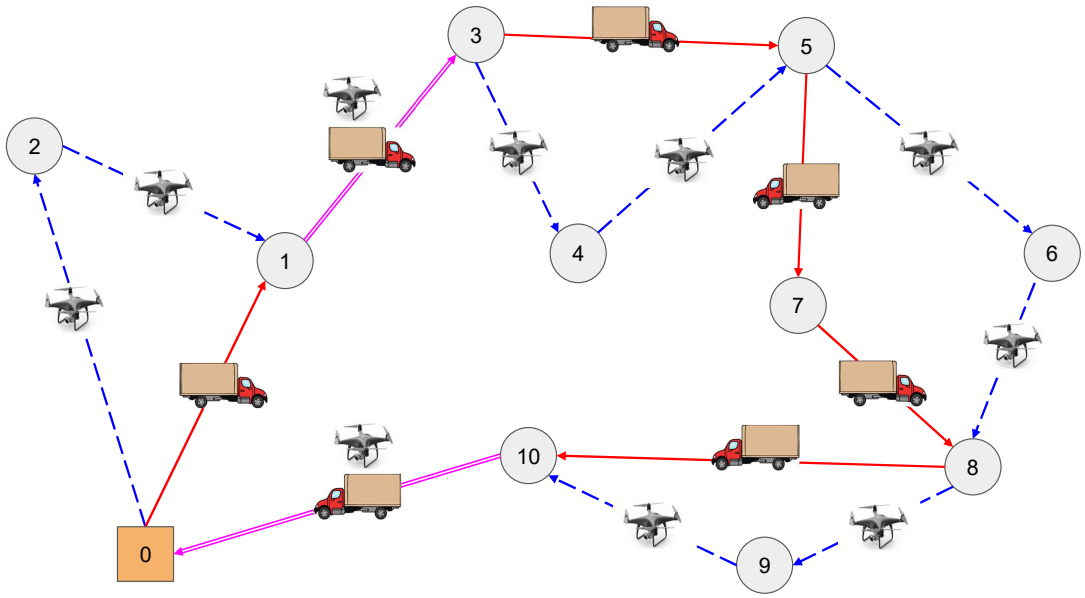

Figure 1.2: Example of a FSTSP solution with 10 customers.

flying speed could allow to increase the flight range and serve more customers with the drone, and this could reduce the overall time required to serve all customers. To this extent, in 2020 Raj and Murray introduced the *Flying Sidekick Traveling Salesman Problem with Variable Drone Speeds* (FSTSP-VDS) [7], a variant of the FSTSP in which the drone power consumption is modeled as a non-linear function, and in which the drone is allowed to fly from one location to another at different speeds.

The focus of this thesis is to devise the first exact method for the FSTSP-VDS, and a heuristic method to tackle larger instances of the FSTSP-VDS. We believe that both from a theoretical and practical point of view it is important to investigate simple truck-and-drone routing problems in which the drone flying speed is not fixed with the prospect of accelerating the development of efficient and practical solutions for real world applications. Our main scientific contributions are as follows:

- We show that the most used drone power consumption models in literature present a convex energy-per-distance function (the energy required to travel one meter at a given speed). We then show that, if such property is satisfied,

the minimum and maximum drone feasible travel times to traverse a generic drone leg $i \dashrightarrow j \dashrightarrow k$ (see Chapter 2 for the definition of drone leg) can be computed by solving a convex optimization problem.

- We introduce the first MILP formulation for the FSTSP-VDS. The formulation we present has the advantage to be compact (i.e., it features a polynomial number of variables and constraints) and allows to solve instances with up to 25 customers using the CPLEX commercial solver.

- We propose a cutting plane algorithm based on the MILP formulation above that works without requiring the exact minimum and maximum drone travel times for each drone leg in the problem, but instead, it requires only a lower and upper bound of each one of these quantities. The algorithm works by refining the minimum and maximum drone travel times only when is needed, that is, only if the MILP returns one or more infeasible drone legs. In that case, the minimum or maximum drone travel times of the infeasible drone legs are refined, and the MILP restarted.

- We present a genetic algorithm to tackle larger instances of the FSTSP-VDS, that can also be applied on single-drone single-truck problems in which the drone speed is fixed such as the FSTSP and the TSP-D. The proposed genetic algorithm allows to obtain a 6% average improvement over the current state of the art FSTSP-VDS heuristic, and to find known optimal solutions of TSP-D instances with up to 39 customers. In particular, for a total of 227 TSP-D instances with known optimal solutions evaluated, the genetic algorithm was able to find 194 out of 227 optimal solutions (85.5%), with an average optimality gap of only 0.30%.
  The effectiveness of our heuristic relies on a one-to-many encoding such that an individual maps many different solutions, and an efficient procedure to retrieve the best solution encoded by an individual. In particular, let n the instance size, we prove that, with the proposed encoding, an individual maps $\Omega(2^n)$ unique solutions. We then provide a simple dynamic programming algorithm that allows to retrieve the best solution among all the ones represented by an individual in time $\Theta(n^3)$, and a very effective approximated variant that runs in time $\Theta(n)$.

The thesis is organized as follows. Chapter 2 gives a precise definition of the FSTSP-VDS. Chapter 3 discusses the most relevant drone-aided routing optimization works related to the FSTSP and the FSTSP-VDS. Chapter 4 introduces the drone power consumption model used in this thesis, and provides crucial considerations in order to solve the FSTSP-VDS efficiently. Chapter 5 provides the first

exact method for the FSTSP-VDS. In particular, it is presented a compact *Mixed Integer Linear Programming* (MILP) that allows to solve FSTSP-VDS instances with up to 15 customers using commercial solver such as CPLEX or GuRoBi. Chapter 6 introduces a genetic algorithm as metaheuristic method for the FSTSP-VDS and for the FSTSP. Chapter 7 reports extensive computational results obtained using the exact and heuristic approach on FSTSP-VDS instances and the results obtained are compared with the ones of existing solutions. Then, the genetic algorithm is tested on instances of the TSP-D to evaluate its effectiveness against exact methods for the TSP-D. Chapter 8 draws some conclusions and addresses future work.

Chapter 2

Problem Definition

The *Flying Sidekick Traveling Salesman Problem with Variable Drone Speeds* (FSTSP-VDS) can be formally described as follows. A complete directed graph $G = (V, A)$ is given. The vertex set V is defined as $V = \{0, 0'\} \cup N$, where both 0 and $0'$ represent a single depot and N a set of customers to serve. To this extent we will use the notation $N_0 = N \cup \{0\}$ and $N_0' = N \cup \{0'\}$. The arc set A is defined as $A = \{(i,j)|i,j \in N : i \neq j\} \cup \{(0,j)|j \in N\} \cup \{(i,0')|i \in N\}$. A single truck, equipped with a single drone, is located at the depot. For each customer $i \in N$, it is associated a parcel with a weight w_i. Each customer $i \in N$ must be served exactly once, and it must be served either by the truck or by the drone. Each drone can carry only one parcel at the time, therefore it can be used to serve only one customer in each flight. The drone has a payload capacity of K kg, this implies that if a parcel exceeds this weight it can only be delivered by the truck.
The drone can leave and return to the truck at customer locations or at the depot only, but it cannot be launched and retrieved at the same node. While the drone is serving a customer, the truck can be used in parallel with the drone to serve other customers. The drone can traverse different arcs at different flying speeds, but the flying speed must remain constant during the flight, and in any case, it cannot be greater than v_{max}. The distance that the truck must cover to traverse the arc $(i,j) \in A$ is indicated with d_{ij}^T, whereas the distance that the drone must cover to traverse the same arc (i,j) is indicated with d_{ij}^D.
This distinction is motivated by the fact that the truck and the drone travel on different pathways, therefore, we typically have $d_{ij}^T \neq d_{ij}^D$. The time for the truck to traverse the arc $(i,j) \in A$ is indicated with t_{ij}^T. The minimum and maximum feasible drone traveling times to takeoff from node $i \in N_0$, deliver the parcel at node $j \in N$, and landing at node $k \in N_0'$, including potential hovering time over the landing node k, are denoted respectively with t_{ijk}^D and T_{ijk}^D. If the drone arrives before the truck at the rendezvous location, the drone cannot land and wait, but it must hover above the rendezvous location until the truck arrives, consuming

energy. Vice versa, if the truck arrives before the drone, the truck must wait the drone. The drone has a fixed battery capacity of B Joules. The endurance of the drone, expressed in seconds, is a function of the parcel weight and the drone speed. The drone must return at the landing location or at the depot, before it runs out of battery. Once returned to the truck, the drone's battery is swapped with a fully charged battery, without requiring additional time. The endurance of the truck is assumed to be infinite. The time required to prepare the launch of the drone from the truck and the time required to retrieve the drone and placing back into the truck are constants and take respectively s_L and s_R seconds, and their sum, is denoted with the constant $\mu = s_L + s_R$. The time required to deliver a parcel, once the vehicle is arrived at destination, it takes σ_T seconds for the truck, and σ_D seconds for the drone. Without loss of generality, delivery times σ_T and σ_D are included in the travel times. The drone cruise altitude is defined by the parameter h.

The goal of the FSTSP-VDS is to find a tour with the minimum completion time serving all customers either by the truck or by the drone and considering potential waiting times because of the synchronization between the truck and the drone. Figure 2.1 illustrates a solution of a FSTSP instance with eight customers (the circles) and the depot, that is, nodes 0 and $0'$ (the rectangle). The drone is launched from the depot to serve customer 3, while the truck leaves the depot to serve customer 1. After the drone has served customer 3, it travels to customer 1 to rejoin the truck. Depending on the travel times t^T_{01}, t^D_{03} and t^D_{31} either the truck waits for the drone ($t^T_{01} < t^D_{03} + t^D_{31}$) or the drone waits for the truck ($t^T_{01} > t^D_{03} + t^D_{31}$). Then, the truck moves with the drone onboard towards customer 6. After that customer 6 has been served, the drone is launched from the truck to serve customer 7, while the truck is serving customer 5. Then, the truck and the drone rejoins at customer 8. From customer 8, the drone is launched from the truck to serve customer 4, while the truck serves customer 2. Then, the truck and the drone travel back to the depot independently, without waiting for each other. The total completion time t of this solution is computed as

$$t = \max\{t^T_{01}, t^D_{03} + t^D_{31}\} + t^T_{16} + \max\{t^T_{67} + t^T_{78}, t^D_{65} + t^D_{58}\} + \max\{t^T_{82} + t^T_{20'}, t^D_{84} + t^D_{40'}\} + 3\mu.$$

In the remainder of the thesis, we will use the following concepts borrowed from the work of Roberti and Ruthmair [18]. A *truck customer* is a customer visited by the truck alone. Similarly, a *drone customer* is a customer visited by the drone alone. If instead the customer is served by the truck while the drone is onboard (hence, not operational) we say it is a *combined customer*. In the case of

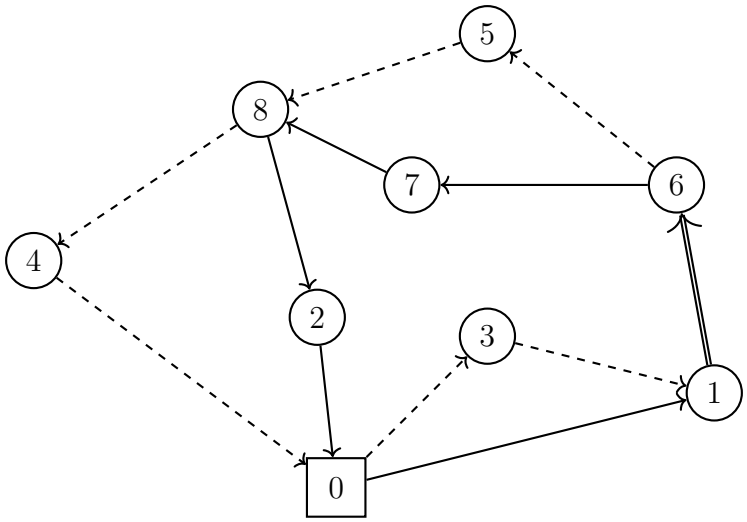

Figure 2.1: Example of solution of a FSTSP instance with eight customers.

the solution of Figure 2.1 customers 2 and 7 are truck customers, 3,4 and 5 are drone customers, and 1,6 and 8 are combined customers.

A *truck arc* (*drone arc*, respectively) is an arc traversed by the truck (drone, respectively) alone. A *combined arc* is an arc traversed by the truck while the drone is onboard. The solution of Figure 2.1 includes four truck arcs (i.e., $(0, 1)$, $(6, 7)$, $(8, 2)$ and $(2, 0')$), six drone arcs (i.e., $(0, 3)$, $(3, 1)$, $(6, 5)$, $(5, 8)$, $(8, 4)$ and $(4, 0')$), and one combined arc (i.e., $(1, 6)$).

A *truck leg* is a concatenation of truck arcs traversed by the truck alone in between two consecutive combined customers. A *drone leg* is a concatenation of exactly two consecutive drone arcs traversed by the drone alone in between two consecutive combined customers. A *combined leg* is a concatenation of combined arcs traversed by the truck and the drone together that consists of combined customers only. The solution of Figure 2.1 includes three truck legs (i.e., $0 \rightarrow 1$, $6 \rightarrow 7 \rightarrow 8$ and $8 \rightarrow 2 \rightarrow 0'$), three drone legs (i.e., $0 \dashrightarrow 3 \dashrightarrow 1$, $6 \dashrightarrow 5 \dashrightarrow 8$ and $8 \dashrightarrow 4 \dashrightarrow 0'$), and a single combined leg (i.e., $1 \Rightarrow 6$).

An *operation* consists of a truck leg and a drone leg in between the same pair of combined customers. The solution of Figure 2.1 consists of three operations: the first one consisting of the truck leg $0 \rightarrow 1$ and the drone leg $0 \dashrightarrow 3 \dashrightarrow 1$, the second one consisting of the truck leg $6 \rightarrow 7 \rightarrow 8$ and the drone leg $6 \dashrightarrow 5 \dashrightarrow 8$, and the third one consisting of the truck leg $8 \rightarrow 2 \rightarrow 0'$ and the drone leg $8 \dashrightarrow 4 \dashrightarrow 0'$. Finally, notice that a FSTSP solution can be seen as a concatenation of operations and combined legs. The power consumption model adopted in this thesis is described in Chapter 4.

Chapter 3

Literature Review

The number of works related to drone-aided routing optimization problems has grown extremely fast over the last few years, for this reason, in this section we mainly focus on works relevant to our problem, that is, works that do not impose a fixed drone speed, and that model the drone energy consumption through a non-linear model. The problem of combining a drone with a traditional ground vehicle for parcel delivery was first formally defined by Murray and Chu in 2015 [6] with the introduction of the FSTSP. The authors proposed a MILP formulation for the FSTSP capable of solving instances with up to 10 customers. In that paper, the authors also define the *Parallel Drone Scheduling TSP* (PDSTSP), a problem in which multiple drones are launched from the depot to serve nearby customers, independent of the truck delivery. For both problem, the authors devised greedy construction heuristics. A Nearly identical problem to the FSTSP has been introduced by Agatz et al. (2018) under the name *TSP with Drone* (TSP-D). The main difference between the two is that in the TSP-D it is assumed that the drone and the truck travel on the same pathways, but at different speeds, whereas, in the FSTSP the truck and the drone travel on different pathways. The authors proposed a MILP formulation that allows to solve to optimality instances with up to 12 customers, and a heuristic algorithm based on a route first-cluster second procedure. Roberti and Ruthmair (2021) [18] proposed a compact MILP for the TSP-D and additional constraints for several variants of the problem. The authors introduced dynamic programming recursions to model several TSP-D variants, and exploited these recursions to devise an exact branch-and-price approach based on a set partitioning formulation capable of solving to optimality instances with up to 39 customers. Ha et al. (2020) [9] proposed a hybrid genetic algorithm for the TSP-D based on the hybrid genetic framework of Vidal et al. [25] to either minimize the total operational cost or minimize the completion time for the truck and drone. Raj and Murray (2020) introduced the *multiple Flying Sidekicks Traveling Salesman Problem* (mFSTSP) [13], an extension of the FSTSP in which instead of

only one UAV, an heterogeneous fleet of UAVs is employed. The authors proposed a MILP formulation which allows to solve to optimality instances with up to 8 customers, and a three-phased heuristic. Poikonen and Golden (2020) [17] proposed a variant of the mFSTSP, named *k-Multi-visit Drone Routing Problem* (k-MVDRP), in which the truck acts only as a mobile depot and recharging platform for the drones, the drones can deliver multiple parcels in a row, and the takeoff and landing locations is a set of predefined locations that can differ from the customer ones. The energy consumption is modeled by the non-linear model of Stolaroff et al. [20]. A MILP formulation and a heuristic algorithm are proposed. Different drone speeds have been tested using the non-linear energy consumption model of Stolaroff et al. [20], and they concluded that the objective values are highly sensitive to drone speed when a realistic power models is used. Tamke and Buscher (2021) [22] introduced the *Vehicle Routing Problem with Drones and Drone Speed Selection* (VRPD-DSS), a variant of the classical VRP, in which trucks are equipped with multiple drones to minimize the overall operational cost. The drone speeds are not fixed, but have to be selected from a discrete set, and the drone energy consumption is characterized by the Stolaroff energy consumption model [20]. The energy levels for each speed is computed in advance, and the dominated drone speeds are eliminated in preprocessing. The authors proposed a MILP formulation that aims at minimizing the operational costs consisting of fuel consumption costs of the trucks, labor costs for the drivers, and energy costs of the drones. Dukkanci et al. (2021) introduced the *Energy Minimizing and Range Constrained Drone Delivery Problem* (ERDDP) [5] in which trucks are used exclusively as mobile depots for the drones that are used to deliver parcels to customers, and drone speeds are continuous decision variable. The set of potential parking locations for the trucks is fixed and different from the the set of customer locations. The drone energy consumption is defined by the non-linear model of Zeng et al [27]. The aim of the ERDDP is to minimize the overall operational cost. The authors formulated the problem as a second order cone programming problem that can be solved using off-the-shelf optimization solvers. Raj and Murray (2020) introduced the *multiple Flying Sidekicks Traveling Salesman Problem with Variable Drone Speeds* (mFSTSP-VDS) [7], an extension of the mFSTSP in which drones are allowed to fly at different speeds, selected from a continuous set. The non-linear model of Liu et al. [11] was used as drone power consumption model. A three-phase heuristic adapted from the solution for the mFSTSP has been proposed. Raj et. Al (2021) [16] proposed an exact branch-and-price algorithm based on a set covering based formulation for the PDSVRP. A modified version of the exact branch-and-price algorithm has been also proposed as heuristic for the PDSVRP. Then, the authors introduced the *PDSVRP with Variable Drone Speeds* (PDSVRP-VDS), an

extension of the PDSVRP that allows the drones to fly at different speeds as in the mFSTSP-VDS. The authors employed the Liu et al. [11] model as non-linear drone power consumption model, and since the PDSVRP does not require any synchronization between the trucks and the drones, Raj et al., computed the optimal drone speeds in advance, and added this information in the solution proposed for the PDSVRP.

To the best of our knowledge, the only works for drone-aided routing problems that adopt a non-linear drone energy consumption model and allow the drone speeds to be selected from a continuous set are, the work of Raj and Murray (2020) [7] for the FSTSP-VDS, the work of Raj et al., (2021) [16] for the PDSVRP-VDS, and the work of Dukkanci et al., (2021) [5] for the ERDDP. The salient characteristics of these three works can be found in Table 3.1. It can be noted that the ERDDP

	Problem						
Authors	Energy function	#Trucks	#Drones	Truck deliver parcels	Drone/Truck sync	Takeoff and rendezvous locations	Solution
Raj and Murray (2020)	Liu et al. (2017)	1	many	yes	yes	customer locations	Heuristic
Raj et al. (2021)	Liu et al. (2017)	many	many	yes	no	depot	Exact + Heuristic
Dukkanci et al. (2021)	Zeng et al. (2019)	many	many	no	no	parking locations	Exact
Our study	Liu et al. (2017)	1	1	yes	yes	customer locations	Exact + Heuristic

Table 3.1: Related works in which drone speeds can be selected from a continuous set, and the drone energy consumption is defined by a non-linear model.

of Dukkanci et al., [5] and the PDSVRP-VDS of Raj et al., [16] do not require synchronization between trucks and drones, and for this reason, these models are easier to model when dealing with variable drone speeds, since, in general, the optimal speeds could be computed in advance. However, when the problem requires the synchronization between trucks and drones, as in the case of the FSTSP-VDS, the problem becomes much harder, since, each possible truck leg is related to multiple non-linear optimization problems. Finally, the development of the genetic algorithm was inspired by the work of Vidal et al. [25], a genetic based framework that proved to be extremely effective for various classes of VRPs.

Chapter 4

Drone Power Consumption

In this chapter, we define the drone power consumption model used in this thesis and we make some considerations in order to tackle the FSTSP-VDS using *Mixed Integer Linear Programming* (MILP). For comparison purposes, we used the same power consumption model used by Raj and Murray [7], i.e., the power consumption model of Liu et al. [11], a model that is specifically designed for small multi-rotor unmanned aircraft systems. The model is defined by three functions that characterize the drone power consumption during takeoff or landing (4.1), during horizontal cruising (4.2), and during stationary hovering (4.3).

$$P_w^{tl}(v_{tl}) = k_1(W+w)g\left[\frac{v_{tl}}{2} + \sqrt{\left(\frac{v_{tl}}{2}\right)^2 + \frac{(W+w)g}{k_2^2}}\right] + c_2\left((W+w)\,g\right)^{3/2} \quad (4.1)$$

$$P_w^{c}(v) = (c1+c2)\left[\left((W+w)g - c5\left(v\cos\alpha\right)^2\right)^2 + \left(c_4v^2\right)^2\right]^{3/4} + c4v^3 \quad (4.2)$$

$$P^{h}(w) = (c_1+c_2)\left((W+w)\,g\right)^{3/2} \quad (4.3)$$

The Liu et al. [11] drone power consumption and drone range functions with respect to speed and parcel weight are illustrated in Figure 4.1 and Figure 4.2. The coefficients $k1, k2, c1, c2, c3, c4, c5$ are the parameters of the power consumption model of Liu et al. [11], and their values can be found in Table 4.1. The other parameters W, w and α define the physical characteristics of the UAV. In particular, W is the UAV frame weight, that is assumed to be 1.5 kg, w is the UAV payload weight in kg, and α is the angle of attack of the drone, which is assumed to be 10 degrees. The vertical ascending and descending speeds during takeoff and landing are assumed to be constant and equal to 10 m/s^2 and 5 m/s^2 respectively. Finally, g is the gravitational acceleration (9.8 m/s^2).

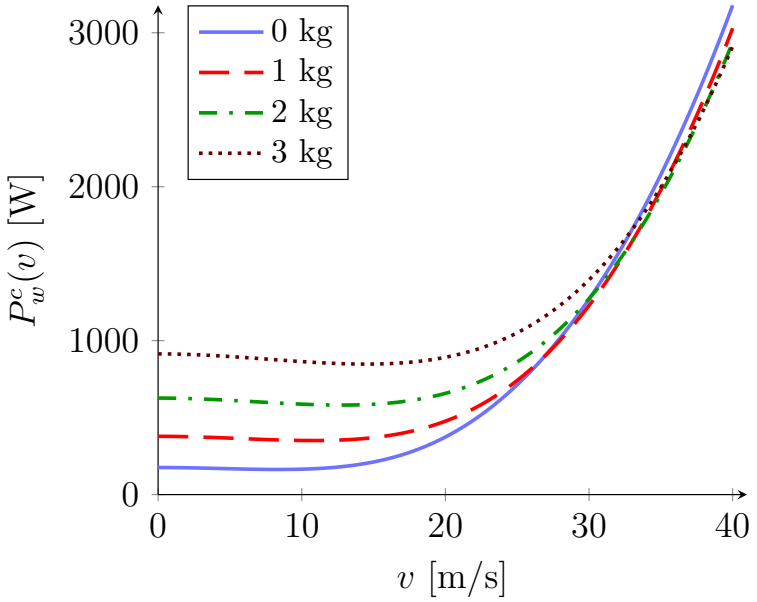

Figure 4.1: Liu et al. [11] drone power consumption with respect to the drone speed v, for different payload weights.

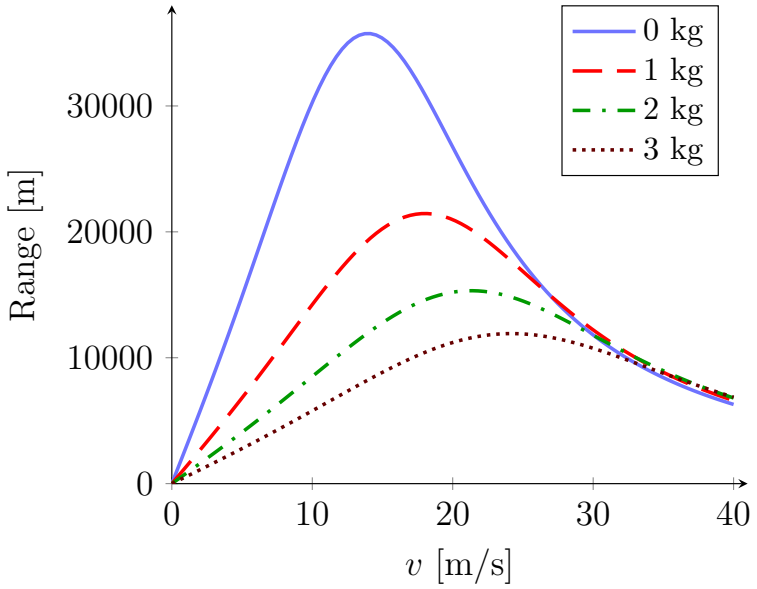

Figure 4.2: Liu et al. [11] drone range in meters of a drone with a battery capacity of 500 kJ with respect to the drone speed v, for different payload weights.

Coefficient:	k_1	k_2	c_1	c_2	c_4	c_5
Value:	0.8554	0.3051	2.8037	0.3177	0.0296	0.0279
Unit:	-	$\sqrt{kg/m}$	$\sqrt{m/kg}$	$\sqrt{m/kg}$	kg/m	N s/m

Table 4.1: Coefficient values for the power consumption model of Liu et al. [11]

4.1 Considerations

Let $E_w^{pm}(v) = (1/v)P_w^c(v)$ be the energy-per-meter function, i.e., the energy that the drone consumes to travel horizontally one meter at speed v m/s carrying a payload of w kg. From empirical observations (Figure 4.3), we can note that, for the power consumption model of Liu et al. [11], the energy-per-meter function $E_w^{pm}(v)$ is convex. We can then prove the following proposition.

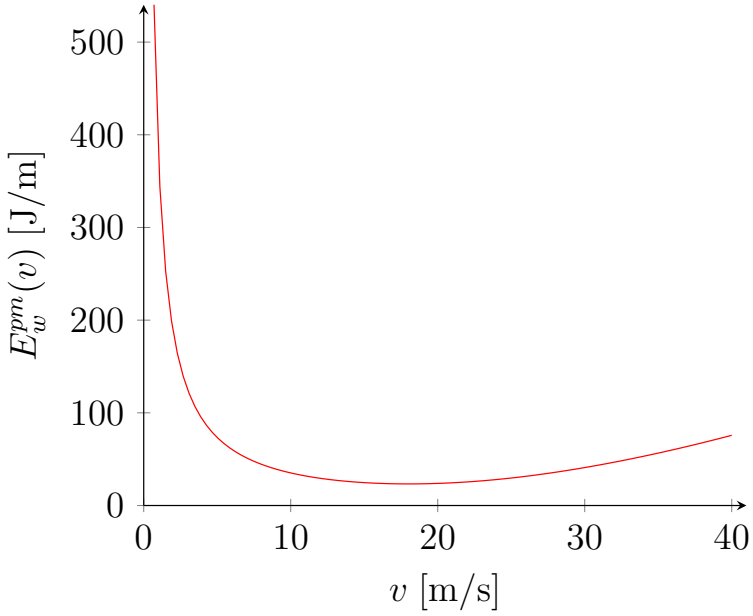

Figure 4.3: Liu et al. [11] energy-per-meter function $E_w^{pm}(v)$ with respect to the drone speed v, for a payload weight w of 1 kg.

Theorem 1. *If the energy-per-meter function $E_w^{pm}(v)$ is convex in the interval $(0, v_{max}]$ for each $w \in [0, K]$, then, for each drone leg $i \dashrightarrow j \dashrightarrow k$, each $\hat{t}_{ijk} \in [t_{ijk}^D, T_{ijk}^D]$ is a feasible drone traveling time.*

Proof. Consider the following optimization problem:

$$\min \frac{d_{ij}^D}{s_{ij}} + \frac{d_{jk}^D}{s_{jk}} + t_h \tag{4.4}$$

$$E_w^{pm}(s_{ij})d_{ij}^D + E_0^{pm}(s_{jk})d_{jk}^D + t_h P^h(0) \leq B \tag{4.5}$$

$$t_h \geq 0 \tag{4.6}$$

$$s_{ij}, s_{jk} \in (0, s_{max}]. \tag{4.7}$$

The objective function represents the drone travel time to traverse the arcs (i, j) and (j, k) at speeds s_{ij} and s_{jk} respectively, plus an hovering time over node k defined by the decision variable t_h. Inequality (4.5) imposes that the drone energy consumption to traverse the drone leg $i \dashrightarrow j \dashrightarrow k$ at speeds s_{ij} and s_{jk} plus the energy required to hover for a time t_h seconds must not exceed the battery capacity of B Joules. Both the objective function (4.4) and the left-hand side of (4.5) are convex functions because they are sums of convex functions. Therefore, the optimization problem above is convex and the optimal solution corresponds to t_{ijk}^D. Whereas, T_{ijk}^D corresponds to the feasible solution with the maximum cost. Since the feasible region of a convex optimization problem is convex too, each $\hat{t}_{ijk} \in [t_{ijk}, T_{ijk}]$ corresponds to at least one feasible solution of this optimization problem. ■

The convex optimization problem (4.4) - (4.7) returns the minimum drone travel time for the drone leg $i \dashrightarrow j \dashrightarrow k$. Instead, to compute the maximum drone travel time, this convex optimization problem cannot be used because in a maximization setting (4.4) becomes concave. In the next section, we propose a convex optimization problem to compute the maximum drone travel time and a convex optimization problem to find the minimum energy consumption to traverse a drone leg in a target travel time. The total energy consumed by the drone to travel a distance of d meters in t seconds can be expressed as follows:

$$E_d(t) = dE_w^{pm}(d/t) = d(t/d)P_w^c(d/t) = tP_w^c(d/t). \tag{4.8}$$

The mapping $f(x) = d/x$ is part of a larger group of functions called *linear-fractional* functions, and one property of these functions is that they preserve convexity [3]. Therefore, assuming $E_w^{pm}(t)$ convex, also $E_w^{pm}(d/t)$ and $E_d(t)$ are convex functions. With the considerations made above we can define a convex optimization problem to compute the maximum drone travel time of a drone leg (4.9) - (4.13), in which the speed variables s_{ij}, s_{jk} are replaced with the time variables t_{ij}, t_{jk}, t_h that denote respectively the traveling time to cross the arc (i, j), the traveling time to cross the arc (j, k) and the hovering time above the

rendezvous location k.

$$\min \; -t_{ij} - t_{jk} - t_h \tag{4.9}$$
$$E_{d_{ij}}(t_{ij}) + E_{d_{jk}}(t_{jk}) + t_h P^h(0) \leq B \tag{4.10}$$
$$t_{ij} \geq d_{ij}/v_{max} \tag{4.11}$$
$$t_{jk} \geq d_{jk}/v_{max} \tag{4.12}$$
$$t_h \geq 0 \tag{4.13}$$

Now, suppose instead that we are given a target travel time T for the drone leg $i \dashrightarrow j \dashrightarrow k$ and we are asked to compute the minimum energy to achieve such target time. The following convex optimization problem (4.14) - (4.17) can be used to accomplish such task.

$$\min E_{d_{ij}}(t_{ij}) + E_{d_{jk}}(t_{jk}) + (T - t_{ij} - t_{jk})P^h(0) \tag{4.14}$$
$$t_{ij} + t_{jk} \leq T \tag{4.15}$$
$$t_{ij} \geq d_{ij}/v_{max} \tag{4.16}$$
$$t_{jk} \geq d_{jk}/v_{max} \tag{4.17}$$

Finally, the same approach discussed in this chapter can also be used for the power consumption models of Stolaroff et al. (2020) [20] and Zeng et al. (2019) [27], since both models present a convex energy-per-meter function [5], [20].

Chapter 5

Exact Method

In chapter 4, we have shown that if the energy per distance unit is a convex function, then, the region of feasible travel times of a drone leg is also convex. Therefore, if we know the minimum and maximum travel times of the drone leg $i \dashrightarrow j \dashrightarrow k$, denoted respectively as t^D_{ijk} and T^D_{ijk}, the interval of feasible travel times is the interval $[t^D_{ijk}, T^D_{ijk}]$. This result suggests to model the non-linear energy constraint (4.5) by constraining the drone travel times of each drone leg to be in the feasible range, instead of constraining the drone speeds.

5.1 Compact Formulation

In this section, we describe the formulation for the FSTSP-VDS, under the assumption that the energy per distance unit is convex. The resulting formulation has the quality to be compact (polynomial number of constraints and variables) and can be solved with a generic MILP solver such as CPLEX or GuRoBi.
Let $x^T_{ij} \in \{0,1\}$ be a binary variable equal to 1 if the truck traverses the arc $(i,j) \in A$. Let $x^D_{ij} \in \{0,1\}$ be a binary variable equal to 1 if the drone traverses the arc $(i,j) \in A$ (no matter if it is onboard the truck or airborne). Let $y^T_i \in \{0,1\}$ be a binary variable equal to 1 if $i \in N$ is a truck customer. Let $y^D_i \in \{0,1\}$ be a binary variable equal to 1 if $i \in N$ is a drone customer. Let $y^C_i \in \{0,1\}$ be a binary variable equal to 1 if $i \in N$ is a combined customer. Let $z_{ijk} \in \{0,1\}$ be a binary variable equal to 1 if the drone traverses the drone leg $i \dashrightarrow j \dashrightarrow k$. Let T_{tl} the amount of time equal to the sum of the time required to launch the drone from the truck, and the time required to retrieve it. Finally, let $a_i \in \mathbb{R}_+$ be the arrival time at node $i \in V$ of the truck if i is a truck customer, of the drone if i is a drone customer, or of the latest vehicle if i is a combined customer.
Hence, $a_{0'} + \mu \sum_{i \in N} y^D_i$ is the quantity we want to minimize and it represents the overall tour duration. We also assume that the drone battery capacity is finite and equal to B Joules.

$$\min \quad a_{0'} + \mu \sum_{i \in N} y_i^D \tag{5.1}$$

$$\text{s.t.} \quad \sum_{(i,j)\in A} x_{ij}^T = \sum_{(j,i)\in A} x_{ji}^T \qquad i \in N \tag{5.2}$$

$$\sum_{(i,j)\in A} x_{ij}^T = y_i^T + y_i^C \qquad i \in N \tag{5.3}$$

$$\sum_{(0,j)\in A} x_{0j}^T = \sum_{(i,0')\in A} x_{i0'}^T = 1 \tag{5.4}$$

$$\sum_{(i,j)\in A} x_{ij}^D = \sum_{(j,i)\in A} x_{ji}^D \qquad i \in N \tag{5.5}$$

$$\sum_{(i,j)\in A} x_{ij}^D = y_i^D + y_i^C \qquad i \in N \tag{5.6}$$

$$\sum_{(0,j)\in A} x_{0j}^D = \sum_{(i,0')\in A} x_{i0'}^D = 1 \tag{5.7}$$

$$y_i^T + y_i^D + y_i^C = 1 \qquad i \in N \tag{5.8}$$

$$a_i + t_{ij}^T \leq a_j + M(1 - x_{ij}^T) \qquad (i,j) \in A \tag{5.9}$$

$$a_i + \min\left\{\frac{d_{ij}^D}{v_{max}}, t_{ij}^T\right\} \leq a_j + M(1 - x_{ij}^D) \qquad (i,j) \in A \tag{5.10}$$

$$x_{ij}^D + x_{ji}^D \leq y_i^C + y_j^C \qquad i,j \in N : i < j \tag{5.11}$$

$$a_i + \sum_{j \in N} (t_{ijk}^D + M) z_{ijk} \leq a_k + M \qquad i \in N_0, k \in N_0' \tag{5.12}$$

$$a_i + \sum_{j \in N} (T_{ijk}^D - M) z_{ijk} \geq a_k - M \qquad i \in N_0, k \in N_0' \tag{5.13}$$

$$\sum_{i \in N_0} \sum_{k \in N_0'} z_{ijk} = y_j^D \qquad j \in N \tag{5.14}$$

$$\sum_{k \in N_0} z_{kij} + \sum_{k \in N_0'} z_{ijk} \leq x_{ij}^D \qquad (i,j) \in A \tag{5.15}$$

$$x_{ij}^T, x_{ij}^D \in \{0,1\} \qquad (i,j) \in A \tag{5.16}$$

$$y_i^T, y_i^D, y_i^C \in \{0,1\} \qquad i \in N \tag{5.17}$$

$$z_{ijk} \in \{0,1\} \qquad i \in N_0, k \in N_0', j \in N \tag{5.18}$$

$$a_i \in \mathbb{R}_+ \qquad i \in N \cup \{0,0'\} \tag{5.19}$$

The objective function (5.1) aims at minimizing the total tour duration to serve all customers. Constraints (5.2) are flow conservation constraints for the truck. Constraints (5.3) link the x_{ij}^T variables with the variables y_i^T and y_i^C. Constraints (5.4) ensure that the truck leaves and returns to the depot exactly once. Constraints (5.5) - (5.7) are equivalent to constraints (5.2) - (5.4) but apply

to the drone. Constraints (5.8) ensure that each customer is (visited at least once) a drone customer, a truck customer, or a combined customer. Constraints (5.9) act as subtour elimination constraints for the truck, and set the truck arrival time at each node. Constraints (5.10) act as subtour elimination constraints for the drone. Constraints (5.11) ensure that the arc $(i, j) \in A$ is traversed by the drone only if the customer $i \in N$ or $j \in N$ is served by truck, and this enforces the drone to serve only one customer in each flight, or equivalently, it imposes that a drone leg must consist of exactly two drone arcs. Constraints (5.12) and (5.13) set the minimum and maximum drone travel times for each drone leg $i \dashrightarrow j \dashrightarrow k$, with $i \in N_0$, $j \in N$ and $k \in N_0'$. Constraints (5.14) link the variables z_{ijk} with the variables y_j^D. Constraints (5.15) imposes that a drone leg can contain the arc $(i, j) \in A$ only if the arc (i, j) is traversed by the drone. Constraints (5.16) - (5.19) model the range of the decision variables.

The correctness of constraints (5.12) and (5.13) is explained below. First observe that, because of constraints (5.15), it is always true that

$$\sum_{j \in N} z_{ijk} \leq 1 \quad \forall\, i \in N_0, k \in N_0'. \tag{5.20}$$

Therefore, $\forall\, i \in N_0, k \in N_0'$ we have $\sum_{j \in N} z_{ijk} \in \{0, 1\}$. When there are no drone legs that start from i and ends in k ($\sum_{j \in N} z_{ijk} = 0$) we have

$$a_i \leq a_k + M \tag{5.21}$$

that is equivalent to an inactive constraint because the value of the parameter M in the right-hand side is defined as a very large number greater than any possible arrival time (see Section 5.3 for more details).
Instead, when there is a drone leg that starts from i and ends in k, it means that for some $\bar{j} \in N$ $z_{i\bar{j}k} = 1$ and $\sum_j z_{ijk} = z_{i\bar{j}k} = 1$. Therefore, we have

$$a_i + t_{i\bar{j}k} \leq a_k \tag{5.22}$$

that is the desired constraint when the drone leg $i \dashrightarrow \bar{j} \dashrightarrow k$ is traversed by the drone. The same reasoning can be used to verify the correctness of constraint (5.13).

The advantage of using constraints (5.12) and (5.13) instead of the more

intuitive constraints

$$a_i + t^D_{ijk} \leq a_k + M(1 - z_{ijk}) \quad \forall\, i \in N_0, j \in N, k \in N'_0 \tag{5.23}$$

$$a_i + T^D_{ijk} \geq a_k - M(1 - z_{ijk}) \quad \forall\, i \in N_0, j \in N, k \in N'_0 \tag{5.24}$$

is that, constraints (5.12) and (5.13) are $\Theta(n^2)$ whereas, constraints (5.23) and (5.24) are $\Theta(n^3)$. It can also be proven that constraints (5.12) and (5.13) are tighter than constraints (5.23) and (5.24).

5.2 Valid Inequalities

The computational time required to solve the formulation (5.1) - (5.19) can be significantly reduced by adding the following constraints:

$$a_{0'} \geq \sum_{(i,j)\in A} t^T_{ij} x^T_{ij} \tag{5.25}$$

$$a_{0'} \geq \sum_{i\in N_0} \sum_{j\in N} \sum_{k\in N'_0} t^D_{ijk} z_{ijk} \tag{5.26}$$

that act as lower bounds for the overall tour duration $a_{0'}$. In particular, the constraint (5.25) specifies that the overall completion time cannot be lower of the sum of the truck travel times of the arcs traversed by the truck.
Analogously, constraint (5.25) specifies that the overall completion time cannot be lower than the sum of the minimum drone travel times of the drone legs traversed by the drone.

Another useful constraint to speed-up the computation is the following constraint.

$$\sum_{i\in N} y^D_i \leq \lceil N/2 \rceil \tag{5.27}$$

Constraint (5.27) specifies that there cannot be more than $\lceil N/2 \rceil$ drone customers since at most there can be one drone customer every two nodes.

5.3 Big-M Value

Constraints (5.9), (5.10), (5.12), (5.13), are also called big-M constraints because, by exploiting an auxiliary parameter M (typically vary large, hence, big-M), they implement a logical implication depending on the value that a specific binary

decision variable assumes. In this setting, the M parameter is used to disable a constraint if a binary decision variable is not triggered. In our case, we want to "activate" constraints (5.9) only when x_{ij}^T is set to one, constraints (5.10) only when x_{ij}^D is set to one, and constraints (5.12), (5.13), only when z_{ijk} is set to one. For the sake of simplicity we define a unique M value valid for all four big-M constraints in the formulation. Alternatively, a specific M value could be defined for each big-M constraint. To provide the tightest possible formulation, we are interested in finding the smallest M value that correctly satisfies the constraints (5.9), (5.10), (5.11), (5.12), even when the decision variables $x_{ij}^T, x_{ij}^D, z_{ijk}$ are set to zero. You can observe that constraints (5.13) requires the largest M value among the four big-M constraints in the formulation.
In constraint (5.13), the value of M must be defined such that

$$M \geq a_i - a_j \quad \forall\, i \in N_0, j \in N_0' \tag{5.28}$$

that is equivalent to

$$M \geq \max_{\substack{i \in N_0 \\ j \in N_0'}} \{a_i - a_j\} = a_{0'} - a_0 = a_{0'} \tag{5.29}$$

Therefore, we can define the value of the parameter M as the cost of the optimal TSP solution that corresponds to the FSTSP, since, a TSP solution corresponds to a truck-only tour for the FSTSP-VDS. Alternatively, if the optimal solution for the correspondent TSP is not available, the M parameter can be defined as an upper bound of the optimal TSP solution cost. A possibility is to order the arcs in ascending order with respect to their truck traveling cost, and define the M parameter as the sum of the truck costs of the first (N+1) arcs. Notice that, by defining the M parameter with these values, we actually cut some feasible solutions, however, since we know that the optimal TSP solution corresponds to a feasible FSTSP-VDS solution, and such solution is not cut off by this procedure, then, the proposed M values are both valid.

5.4 Cutting Plane Approach

To find the quantities t_{ijk}^D and T_{ijk}^D it is required to solve the convex optimization problems (4.4) - (4.7) and (4.9) - (4.13), and this must be repeated for each drone leg in the problem. Since, it could not be always possible to compute the exact minimum and maximum feasible time for all drone legs in the problem, we propose a cutting plane approach based on the compact formulation introduced in (5.1) that assumes that the minimum and maximum drone leg travel times t_{ijk}^D and

T^D_{ijk} are not known in advance, but it is only known a lower and an upper bound for both t^D_{ijk} and T^D_{ijk}. Let denote these quantities $l_{ijk}, u_{ijk}, L_{ijk}, U_{ijk}$, such that $l_{ijk} \leq t^D_{ijk} \leq u_{ijk}$ and $L_{ijk} \leq T^D_{ijk} \leq U_{ijk}$. Since, t^D_{ijk} is the minimum feasible drone travel time to traverse the drone leg $i \dashrightarrow j \dashrightarrow k$, l_{ijk} and u_{ijk} are respectively an infeasible and a feasible travel time. Analogously, since, T^D_{ijk} is the maximum feasible drone travel time to traverse the drone leg $i \dashrightarrow j \dashrightarrow k$, L_{ijk} and U_{ijk} are respectively a feasible and an infeasible travel time.
Finally, it can be observed that:

$$[u_{ijk}, L_{ijk}] \subseteq [t^D_{ijk}, T^D_{ijk}] \subseteq [l_{ijk}, U_{ijk}] \tag{5.30}$$

Where $[u_{ijk}, L_{ijk}]$ represents the known feasible interval for the drone leg $i \dashrightarrow j \dashrightarrow k$.

Given these assumptions, the proposed cutting plane algorithm can be described as follows:

1. Solve the MILP (5.1) - (5.19) replacing constraints (5.12) and (5.13) with the relaxed constraints (5.31) and (5.32).

$$a_i + \sum_{j \in N} (l_{ijk} + M) z_{ijk} \leq a_k + M \qquad i \in N_0, k \in N'_0 \tag{5.31}$$

$$a_i + \sum_{j \in N} (U_{ijk} - M) z_{ijk} \geq a_k - M \qquad i \in N_0, k \in N'_0 \tag{5.32}$$

2. If all drone legs of the returned solution are feasible, the solution is optimal, and therefore, the algorithm stops. Otherwise, for each drone leg $i \dashrightarrow j \dashrightarrow k$ of the returned solution whose travel time is not in the known feasible range $[u_{ijk}, L_{ijk}]$, proceed as follows.
   If the travel time is less than u_{ijk}, refine the interval $[l_{ijk}, u_{ijk}]$ and add the following cut in the main LP.

$$a_i + l_{ijk} \leq a_k + M(1 - z_{ijk}) \tag{5.33}$$

   If instead, the travel time exceed L_{ijk}, refine the interval $[L_{ijk}, U_{ijk}]$ and add the following cut in the main LP.

$$a_i + U_{ijk} \geq a_k + M(1 - z_{ijk}) \tag{5.34}$$

3. Resume the MILP and go to 2.

Chapter 6

Genetic Algorithm

When exact algorithms are not sufficiently fast, the alternative is to rely on heuristic algorithms that aim at providing near-optimal solutions in much shorter times by sacrificing the optimality guarantee of exact algorithms.
In the last decades, a new family of algorithms called metaheuristic algorithms has emerged, that, by combining basic heuristic methods in higher level frameworks aim at exploring efficiently and effectively a solution space. The term metaheuristic is the combination of two words of Greek derivation, *meta* and *heuristic*. Heuristic derives from the verb heuriskein (ευρισκειν) which means "to find", while the prefix meta derives from the word μετα, which means "beyond, after". A popular definition of metaheuristic, introduced by Osman and Laporte [12] in 1996, is the following:

> *A metaheuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions.*

Example of classical metaheuristic algorithms are genetic algorithms [8], tabu search [21], simulated annealing [19], variable neighborhood search [26], and ant colony [2] optimization methods. Very often, metaheuristic techniques are based on concepts borrowed from biological evolution, swarm behavior or from the laws of physics. One of the central elements of metaheuristic methods is the balance between the *intensification* and *diversification* phases. During the intensification phase different local search procedures are applied with the aim of improving the current best solution. On the contrary, during the diversification phase, new regions of the solution space are explored with the aim to escape from local minimum regions.
In this thesis, we propose a genetic algorithm as metaheuristic approach for the

FSTSP-VDS. Genetic algorithms are population-based metaheuristic that are inspired by the Darwin's theory of evolution. In the nineteenth century Charles Darwin proposed the "theory of evolution by natural selection", often simply called *theory of evolution*. In this theory, Darwin proposes that biological evolution occurs because of a phenomenon called natural selection, that is, organisms with traits favorable to the environments in which they live, are better equipped to survive, reproduce, and therefore, to transmit these helpful traits to the next generation. On the other hand, those organisms that are less equipped, have less probability to survive and then give birth to a lower expected number of children. Hence, it is like applying a pressure on the population based on the fitness of the individuals. The general scheme of a genetic algorithm is shown in Algorithm 1.
To devise our genetic algorithm, we took inspiration from the work of Vidal et al. [25], a genetic algorithm that has proved to be extremely effective for different classes of VRPs. From now on, we refer to the term *fitness* to indicate the cost associated to an individual, and to the term *offspring* to indicate the new individuals generated during the procedure.

Algorithm 1: Genetic Algorithm(η, γ, P_m, *epochs*)

Data:
η: population size
γ: number of offspring at each iteration
P_m: mutation probability
epochs: max number of iterations

```
Generate a random population of η individuals;
Evaluate the (biased) fitness of all individuals in the population;
while iterations < epochs do
    for i = 1 to γ do
        Select two parent individuals P1 and P2 according to their fitness;
        Generate offspring C from P1 and P2 (crossover);
        With probability P_m apply mutations to offspring C;
        Educate offspring C (local search procedures);
    Evaluate the (biased) fitness of all individuals in the population;
    Replace γ individuals in the population with the new offspring;
return the best individual;
```

6.1 Solution Representation

We decided to represent an individual as a permutation of the sequence $(0, 1, 2, ..., N, 0')$ with the first and last element fixed respectively to 0 and $0'$.
Each individual then represents all the solutions that satisfy the following property:

Property 1. *$\forall\, i, j \in N_0 \cup N_0'$, if i and j are visited by the same vehicle and i is visited before j, then, i must occur before j in the chromosome sequence.*

An example of this encoding is illustrated in Figure 6.1, in which are shown all the 12 solutions associated with the chromosome sequence $(0, 1, 2, 3, 0')$. Note that, since the relationship of precedence between customers visited by the same vehicle is determined by the chromosome sequence, there is a bijection between the set of operations and the set of drone legs, that is, an operation is uniquely determined by a drone leg, and of course, vice versa. The advantage of such encoding relies on the fact that, as we will show in Section 6.2, we can retrieve the best solution among all the solutions represented by a sequence that satisfy Property 1 in time $\Theta(N^3)$. In the following, when we refer to the solutions of a (chromosome) sequence, we refer to the solutions that satisfy Property 1 with respect to such sequence.
A lower bound on the number of solutions represented by a chromosome is given by the following theorem.

Theorem 2. *The number of solutions represented by a sequence seq of length n that satisfy property 1 is $\Omega(2^n)$.*

Proof. First, observe that a sequence of length 6 encodes at least 32 unique solutions. This number can be derived by counting how many new solutions are generated by adding an extra node at the beginning or at the end of the 12 solutions represented by a sequence of length 5. In general, a FSTSP-VDS solution can be seen as a concatenation of multiple partial FSTSP-VDS solutions such that the last node of a solution is the starting node of the successive one. Let assume for the sake of simplicity that $n-1$ is a multiple of 5. Let $z_i := seq[i*5, ..., i*5+5], i \in [0, (n-1)/5 - 1]$. Then, each z_i is a subsequences of length 6 of the sequence *seq* and therefore, it encodes at least 32 partial FSTSP-VDS solutions that begin from node $seq[i*5]$ and terminate at node $seq[i*5+5]$. Since the concatenation of partial solutions represented respectively by the sequences $z_0, z_1, ...z_{(n-1-5)/5}$ produce FSTSP-VDS solutions, the number of solutions that satisfy property 1 encoded by a sequence *seq* of length n are at least $2^{5\frac{n-1}{5}} = 2^{n-1} = \Omega(2^n)$.
If $n-1$ is not a multiple of 5, by applying the same reasoning, we obtain that the minimum number of solutions is $2^{5(\lfloor \frac{n-1}{5} \rfloor)} + 1 > 2^{5(\frac{n-1}{5}-1)} + 1 = 2^n 2^{-6} + 1 = \Omega(2^n)$. ■

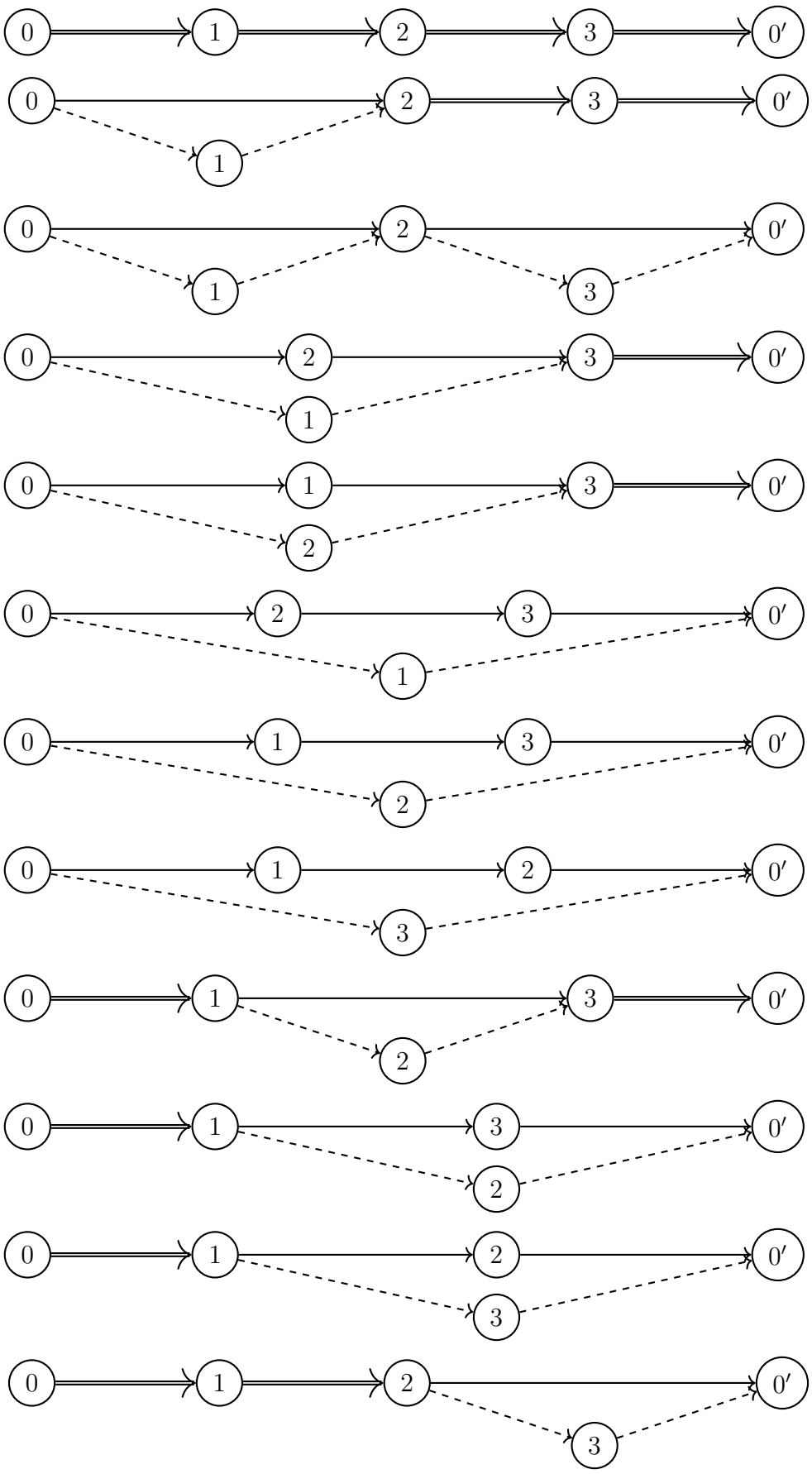

Figure 6.1: All the 12 solutions encoded by the sequence $(0, 1, 2, 3, 0')$.

6.2 Method to retrieve the best solution from an individual

In this section, we describe a dynamic programming algorithm, named *find_best_solution*, that, given a chromosome sequence, it returns the solution with the shortest completion time among all the solutions represented by the chromosome. The pseudo code of *find_best_solution* can be found at Algorithm 2. The algorithm takes in input a sequence seq of n integer numbers, and a value μ, that corresponds to the overall time, in seconds, required to launch and retrieve the drone from the truck during an operation. The algorithm implements a dynamic programming approach to evaluate all the solutions associated with the sequence seq, while reusing some past computation. The algorithm can be described as follows. Let T an array of arrival times such that, $T[i], i \in [0, n-1]$, represents the shortest arrival time at node $seq[i]$ when $seq[i]$ is visited by both the truck and the drone. At the beginning, $T[0...n-1]$ is initialized as the arrival times related to the truck-only solution. After that, starting from $i = 0$ to $n-2$, for each feasible drone leg $seq[i] \dashrightarrow seq[j] \dashrightarrow seq[k]\ \forall\ i < j < k$, it is computed the time required to perform the correspondent operation t_{op}, and if $T[i] + t_{op} + \mu < T[k]$, $T[k]$ is substituted with the improved arrival time just found. When the algorithm terminates, $T[n-1]$ corresponds to the completion time of the best solution among all the ones associated with the sequence seq.
Let $m(i)$ denote the minimum time required to go from node $seq[0]$ to node $seq[i]$ under the condition that node $m(i)$ get visited by the both truck and the drone. Then, it is easy to see that $m(i)$ satisfies the following recurrence equation:

$$m(i) = \begin{cases} 0 & \text{if } i = 0 \\ t^T_{seq[0]seq[1]} & \text{if } i = 1 \\ \min\Big\{ m(i-1) + t^T_{seq[i-1]seq[i]},\ \min_{a<b<i}\Big\{ m(a) + t^{seq}_{op}(a,b,i) \Big\} \Big\} & \text{if } i > 1 \end{cases}$$

where $t^{seq}_{op}(a,b,i)$ is the time required by the operation that involves the drone leg $seq[a] \dashrightarrow seq[b] \dashrightarrow seq[i]$, including launch and retrieval times.

Theorem 3. *When Algorithm 2 terminates, it returns the minimum time of all the solutions represented by the sequence seq.*

Proof. Algorithm 2 evaluates $m(i)$ bottom-up by storing $m(i)$ in $T[i]$. Note that, since the algorithm evaluate $T[i]$ as i increases from 0 to $n-1$, at iteration i all sub-problems referenced by $m(i)$ have been already computed. Therefore, at the

end the algorithm returns $T[n-1] = m(i)$ which corresponds to the minimum time of all the solutions represented by the sequence seq. ■

For each $i \in [0, n-2]$, the number of iterations performed in the inner loop is exactly $(n-2-i-1+1)(n-1-i-2+1) = (n-2-i)^2$. Therefore, the total number of iterations of the nested loop is $\sum_{i=0}^{n-2}(n-2-i)^2 = (n-2)^3/3 + (n-2)^2/2 + (n-2)/6 = \Theta(n^3)$. Where in the last equality we used the formula for the sum of the first $(n-2)$ squares. Therefore, since, the initialization is $\Theta(n)$, and in each iteration of the nested loop there is a constant number of instructions, the time complexity of Algorithm 2 is $\Theta(n^3)$.

To retrieve the truck and drone routes of the best solution found, we make use of three auxiliary arrays, $move\_type$, $drone\_cust$ and $takeoff\_node$.
The value $move\_type[k]$ encodes the type of move associated with the time stored in $T[k]$ that can be either a combined arc ($move\_type = 0$) or an operation ($move\_type = 1$). If $move\_type[k] = 0$, then $T[k]$ is equal to $T[k-1] + t^T_{seq[k-1]seq[k]}$. If instead, $move\_type[k] = 1$, it means that $T[k]$ has been obtained by an operation that starts from $seq[takeoff\_node[k]]$, whose drone customer is $seq[drone\_cust[k]]$, and ends in $seq[k]$. When $move\_type[k] = 0$, the values in $seq[takeoff\_node[k]]$ and $seq[drone\_cust[k]]$ have no meaning.

The execution time of Algorithm 2 can be improved by observing that when the takeoff node $seq[i]$ and the drone customer $seq[j]$ are fixed, the truck travel time $truck\_leg\_time$ required by the operation associated with the drone leg $seq[i] \dashrightarrow seq[j] \dashrightarrow seq[k]$ increases monotonically with respect to k. Then, if $truck\_leg\_time$ becomes larger than $\max_k\{T^D_{seq[i]seq[j]seq[k]}\}$, there cannot be feasible operations for larger values of k, and therefore, we can terminate the inner loop prematurely.

6.2.1 Linear Time Approximated Method

In this section we propose an approximated variant of Algorithm 2 that allows to reduce the time complexity to $\Theta(n)$ where n is the sequence length. The idea to couple the truck with a drone is to reduce the overall completion time by allowing the drone to serve customers in parallel with the truck. The main constraint is that the drone can only perform one serving per each flight. Therefore, when the truck leg of an operation consists of multiple arcs, the maximum number of customers served by the drone decreases, and this reduces the parallelism between the truck and the drone. The maximum parallelism between the truck and the drone is obtained in those solutions in which the truck legs do not consist of

Algorithm 2: find_best_solution(seq, n, μ)

Data:
seq: chromosome sequence
n: chromosome length
μ: sum of launch and retrieval times

Let $T[0...n-1]$ a new array; //arrival times
Let $truck\_only[0...n-1]$ a new array; //truck-only arrival times
//Auxiliary arrays to backtrack the best solution
Let $move\_type[0...n-1]$ a new array; //0:combined arc|1:operation
Let $drone\_cust[0...n-1]$ a new array;
Let $takeoff\_node[0...n-1]$ a new array;
//Initialization
$T[0] \leftarrow 0.0$;
$truck\_only[0] \leftarrow T[0]$;
for $i = 0$ **to** $n-2$ **do**
    $T[i+1] \leftarrow T[i] + t^T_{seq[i]seq[j]}$;
    $truck\_only[i+1] \leftarrow T[i+1]$;
    $move\_type[i+1] \leftarrow 0$; //combined arc
//Dynamic Programming
for $i = 0$ **to** $n-2$ **do**
    if $T[i] + t^T_{seq[i]seq[i+1]} < T[i+1]$ **then**
        $T[i+1] \leftarrow T[i] + t^T_{seq[i]seq[i+1]}$;
        $move\_type[i+1] \leftarrow 0$; //combined arc
    for $j = i+1$ **to** $n-2$ **do**
        $truck\_leg\_time \leftarrow truck\_only[j+1] - truck\_only[i] - t^T_{seq[j-1]seq[j]} - t^T_{seq[j]seq[j+1]} + t^T_{seq[j-1]seq[j+1]}$;
        for $k = j+1$ **to** $n-1$ **do**
            if $k > j+1$ **then**
                $truck\_leg\_time \leftarrow truck\_leg\_time + t^T_{seq[k-1]seq[k]}$;
            if $truck\_leg\_time > T^D_{seq[i]seq[j]seq[k]}$ **then**
                continue; //infeasible operation
            $t_{op} \leftarrow t^D_{seq[i]seq[j]seq[k]}$; //time required by the operation
            if $truck\_leg\_time > t^D_{seq[i]seq[j]seq[k]}$ **then**
                $t_{op} \leftarrow truck\_leg\_time$;
            if $T[i] + t_{op} + \mu < T[k]$ **then**
                $T[k] \leftarrow T[i] + t_{op} + \mu$;
                //store operation i --> j --> k
                $move\_type[k] \leftarrow 1$; //operation
                $drone\_cust[k] \leftarrow j$; //set drone customer index
                $takeoff\_node[k] \leftarrow i$; //set takeoff node index
return $T[n-1], move\_type, drone\_cust, takeoff\_node$;

Algorithm 3: retrieve_truck_drone_routes(seq, n, $move\_type$, $drone\_cust$, $takeoff\_node$)

Data:
seq: chromosome sequence
n: chromosome length
$move\_type$: array of move types
$drone\_cust$: array of drone customers indices
$takeoff\_node$: array of takeoff node indices

Let $truck\_route[0...n-1]$ a new array;
Let $drone\_route[0...n-1]$ a new array;
`//Initialization`
$truck\_idx \leftarrow n-1$;
$drone\_idx \leftarrow n-1$;
$truck\_route[n-1] \leftarrow seq[n-1]$;
$drone\_route[n-1] \leftarrow seq[n-1]$;
for $k = n-1$ **to** 1 **do**
    if $move\_type[k] == 0$ **then**
        $truck\_route[truck\_idx] \leftarrow seq[k-1]$;
        $drone\_route[drone\_idx] \leftarrow seq[k-1]$;
        $truck\_idx \leftarrow truck\_idx - 1$;
        $drone\_idx \leftarrow drone\_idx - 1$;
    else
        $drone\_seq[drone\_idx] \leftarrow seq[drone\_cust[k]]$;
        $drone\_seq[drone\_idx] \leftarrow seq[takeoff\_node[k]]$;
        $drone\_idx \leftarrow drone\_idx - 2$;
        $i \leftarrow k-1$;
        while $i > takeoff\_node[k]$ **do**
            if $i \neq drone\_cust[k]$ **then**
                $truck\_seq[truck\_idx] \leftarrow seq[i]$;
                $truck\_idx \leftarrow truck\_idx - 1$;
            $i \leftarrow i-1$;
        $truck\_seq[truck\_idx] \leftarrow seq[takeoff\_node[k]]$;
        $truck\_idx \leftarrow truck\_idx - 1$;
        $k \leftarrow takeoff\_node[k] + 1$;
return $truck\_seq[truck\_idx+1...n-1], drone\_seq[drone\_idx+1...n-1]$;

multiple arcs. Even if the best solution may not be the one with the maximum parallelism, solutions with low parallelism are typically associated with a high completion time, since they tend to be structurally more similar to TSP solutions, in which the truck is not assisted by a drone.
We then propose a simple modification of Algorithm 2 that allows to reduce the time complexity to $\Theta(n)$ at the price of renouncing to explore solutions with low parallelism. The modification consist of imposing a maximum constant offset (e.g., 5) between the drone customer index j and the takeoff node index i, and between the landing node index k and i. In this way, for each $i \in [0, n-1]$, a constant number of instruction are executed, and the time complexity of the algorithm becomes $\Theta(n)$.

6.3 Evaluation of Individuals

A typical problem of genetic algorithms is *premature convergence*, that happens when the best individual in the population is far from optimality and the algorithm does not improve anymore because all individuals in the population resemble to each other in terms of genetic material. Therefore, one key element of genetic algorithms is maintaining a diversified population along the entire evolution process with the aim to avoid a premature convergence.
In population-based metaheuristic, a value representing the "quality" of the individual with respect to the entire population is assigned to each individual, and usually it is expressed as a function of the individual fitness and one or more penalty terms. This value is therefore called *biased fitness*, and it is used to guide the "natural selection" during the evolution of the population. Inspired by the work of Vidal et al. [25], we define the *diversity contribution* $\Delta(P)$ for an individual P, as the average distance to its n_{close} neighbors in the population, represented by the set $N_{close}(P)$. The distance $\delta(P_1, P_2)$ between two individuals P_1 and P_2 is defined as follows. Let $truck\_succ(P)$ and $drone\_succ(P)$ the arrays of successors of, respectively, the truck and the drone routes of the best solution among all the ones encoded by the individual P, such that, $truck\_succ(P)[i]$ is the next node visited after i in the the truck route, and $drone\_succ(P)[i]$ is the next node visited after i in the the drone route. Then, the distance $\delta(P_1, P_2)$ between two individuals P_1 and P_2 is defined as the Hamming distance between the array of successors $truck\_succ(P_1)$ and $truck\_succ(P_2)$, plus the Hamming distance between the array of successors $drone\_succ(P_1)$ and $drone\_succ(P_2)$. Analytically, the distance $\delta(P_1, P_2)$ is computed according to Equation 6.1, where $\mathbb{1}(x)$ is the indicator function, that returns 1 if the condition x is satisfied, and 0 otherwise. We assume that, if node i is not visited by the truck or by the drone, then $truck\_succ(P)[i] = -1$

and $drone\_succ(P)[i] = -1$ respectively.

$$\begin{aligned}\delta(P_1, P_2) = &\sum_{i=0}^{N} \mathbb{1}(truck\_succ(P_1)[i] \neq truck\_succ(P_2)[i]) \\ &+ \sum_{i=0}^{N} \mathbb{1}(drone\_succ(P_1)[i] \neq drone\_succ(P_2)[i])\end{aligned} \tag{6.1}$$

Therefore, the diversity contribution $\Delta(P)$ of an individual P is computed according to Equation 6.2.

$$\Delta(P) = \sum_{P_i \in N_{close}(P)} \delta(P, P_i) \tag{6.2}$$

Let $fit(P)$ and $dc(P)$ be the positions of the individual P in the fitness ranking and in the diversity contribution ranking in ascending and descending order respectively, with respect to the entire population.
Then, we define the *biased fitness* $BF(P)$ of an individual P as follows:

$$BF(P) = fit(P) + \lambda\, dc(P) \tag{6.3}$$

where $\lambda \in (0, 1]$ is a trade-off parameter used to control the relative importance of the diversity contribution over the fitness contribution of an individual. The *biased fitness* is thus a measure that balance the evolutionary pressure based on the fitness of the individuals (elitism) with the genetic diversity of the population, by penalizing individuals very "similar" to their neighbors and individuals with a low fitness.

6.4 Parent Selection and Crossover

An important concept in genetics is *genetic recombination*, in which the genetic material of different organisms is exchanged to generate offspring with combinations of traits that differ from those found in both parents. This process, also known as *crossing over* (or crossover), works by breaking and rejoining chromosome segments. In genetic algorithms, the biological crossover operation is simulated to generate a new individual by combining the genetic information of two parent individuals. Various types of crossover have been proposed in literature. In this thesis, we focus on the single point crossover. The idea of the single point crossover, is to define a random crossover point c such that the resulting offspring

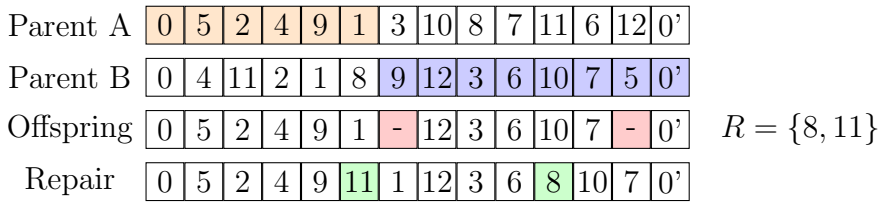

Figure 6.2: Example of a single point crossover with a crossover point $c = 6$ between two individuals with a chromosome length $n = 14$. The resulting offspring is the concatenation of the first $c = 6$ elements of parent A with the last $n - c = 8$ elements of parent B. During the copy of the elements of parent B, values that are already present in the offspring sequence are discarded (values 9 and 5), in order to avoid duplicates. After the combine phase, the values $R = \{8, 11\}$ are not present in the offspring sequence because they reside in the last $n - c$ elements of parent A, and in the first c elements of parent B. During the repair phase, the left over values are added in the offspring chromosome.

is the concatenation of the first c elements of the first parent, with the last $n - c$ elements of the second parent, where n is the length of the two chromosomes. An example of single point crossover can be found in Figure 6.2. However, with the single point crossover, the elements that reside in the last $n - c$ elements of the first parent and in the first c elements of the second parent are not copied in the offspring sequence, and duplicates can occur. Therefore, after the copying phase, a *repair* phase is applied in order to have a valid chromosome at the end of the crossover operation. The repair phase consists of removing the duplicates and adding the left over elements in the offspring sequence. In this work we adopted the following greedy repair phase. The left over elements are added in the offspring sequence one by one in the position that minimize the cost associated to the sequence. The crossover algorithm employed in this thesis can be found in Algorithm 4. Finally, parents are selected using a binary tournament procedure, that twice randomly (with uniform probability) select two individuals from the population, and returns the one with the best fitness.

Algorithm 4: crossover($parentA$, $parentB$, n, μ)

```
Data:
  parentA: parent A chromosome sequence
  parentB: parent B chromosome sequence
  n: chromosomes length
  μ: sum of launch and retrieval times
Let ind an empty sequence;                          //offspring chromosome
Let c a random number between 1 and n − 1;
//Add the first c elements of parentA
for i = 0 to c − 1 do
    ind.add(parentA[i]);
//Add the last n − c elements of parentB
for i = c to n − 1 do
    if parentB[i] ∉ new_ind then
        ind.add(parentB[i]);
//Repair:  add the remaining elements
Let R the list of the remaining elements;
foreach v ∈ R do
    min_cost ← +∞;
    min_seq ← NULL;
    for i = 1 to ind.len − 2 do
        //Add the value v in position i and compute the cost
          associated to this sequence
        seq ← (ind[0, ··· , i − 1], v, ind[i, ··· , ind.len − 1]);
        cost ← find_best_solution(seq, ind.len + 1, μ);
        if cost < min_cost then
            min_seq ← seq;
            min_cost ← cost;
    ind ← min_seq;
    R ← R − v;
return ind, min_cost;
```

6.5 Education

When an offspring is generated, it undergoes an *Education* phase, which consists of different local search procedures applied to the best solution associated with the individual with the aim of improving the solution cost. Inspired by the work of Vidal et al. [25] we did not devise any mutation operator. The local search procedures devised for the education phase do the following operations.

- **Optimize truck legs**: for each truck leg with at most 3 truck customers, optimize the order of the truck customers. For the truck legs with more than 3 truck customers, apply 2-opt moves with the greatest improvement, until no improving moves are found.
- **Reverse operations**: for each operation, check if the solution cost decreases by reversing the operation, and if it is the case, apply the inversion.
- **Swap customers**: for each operation, check if the solution cost can be improved by swapping the takeoff node or the rendezvous node with one of their neighbors. If so, apply the most improving substitution.

These local search procedures are applied one after the other until no improvement is found.

6.6 Population Management

In the last sections we have presented a crossover operator that describe how two individuals combine together to generate an offspring, an education operator that describe the local search procedures an offspring undergoes when it is created, and how parents are selected. The population management operator complements these operators with the aim of propagating good and promising solutions while enhancing the diversity in the population. Our population management mechanism consists of three components: *initialization*, *diversification* and *survivor selection*. In the *initialization* phase, η individuals are randomly generated by assigning to each of them a random chromosome sequence.
The *diversification* phase is activated when the genetic algorithm is not able to improve the best solution for It_{max} consecutive iterations. The aim of this stage is to add new genetic material in the population in order to escape from the local minimum in which the algorithm is currently stuck. In this phase, some individuals are replaced with new random generated individuals. In particular, the $\eta/10$ individuals with the best biased fitness are replaced with probability 0.25, the $3\eta/4$ individuals with the worst biased fitness are replaced with probability 1, while the

remaining individuals are replaced with probability 0.75.

As anticipated in Section 6.3, one of the main challenges in population-based metaheuristics is to avoid premature convergence. This problem is particularly exacerbated, when, as in our case, the repair phase in the crossover operator and the education operator are greedy procedures. The result, is a general tendency to favor individuals with a high fitness at the expense of the diversity of the population. For this purpose, the *biased fitness* measure has been defined with the aim of preventing premature convergence by discarding individuals with a low diversity contribution. Then, our *survivor selection* policy consists of replacing after each iteration the γ individuals with the worst biased fitness in the population with the offspring just generated.

Other than that, at every iteration we check if there are *clones* in the population, that is, if there exist two individuals that share the same chromosome. If two clones are found, we replace one of the two individual with a new randomly generated individual or with a new offspring, with equal probability.

Chapter 7

Computational Results

In this section, we discuss the computational results achieved by the exact solution method described in Chapter 5 and by the metaheuristic solution method described in Chapter 6. In section 7.1 we describe the test instances used in the computational experiments. In section 7.2 we compare the results achieved by the proposed exact solution method and by the genetic algorithm with those achieved by Raj and Murray (2020) [7] on single-drone FSTSP-VDS instances. In section 7.3 we compare the results achieved by the proposed genetic algorithm with those achieved by the branch-and-price algorithm proposed by Roberti and Ruthmair (2020) [18] on the TSP-D instances.

Both the exact solution method and the genetic algorithm have been implemented in C. All experiments are conducted on the *Blade* computing cluster of the University of Padova Department of Information Engineering, using a configuration with 32 Intel Xeon threads and 128 GB of ram. The MILP formulation of the exact solution method was solved with CPLEX 12.10. We use the default parameters of CPLEX, except for CPXPARAM_Parallel (set to CPX_PARALLEL_OPPORTUNISTIC), CPX_PARAM_EPINT (set to 0 to decrease integrality tolerance), CPX_PARAM_EPRHS (set to 1e-9 to decrease feasibility tolerance) and CPX_PARAM_EPGAP (set to 1e-5 to decrease relative MIP gap tolerance). A time limit of one hour is imposed to solve each instance. All computing times reported in this section are in seconds. Each instance is solved 10 times, both with the exact solution method and with the genetic algorithm, and the average values are used to make comparisons between different methods. When the exact solution method is used, each one of the 10 runs are solved by setting a different value to the CPLEX parameter CPX_PARAM_RANDOMSEED.

The genetic algorithm parameters used in the experiments can be found in Table 7.1. To compute the best solution of an individual, it was used the approximated version of Algorithm 2 with a maximum offset between the takeoff node and the rendezvous node equal to 5 (see Section 6.2.1 for more details). The

NLopt nonlinear-optimization library [15] is used in preprocessing to solve the convex optimization problems (4.4) - (4.7) and (4.9) - (4.13) in order to retrieve the minimum and maximum drone leg travel times.
In the following, we denote with t_{avg} the average solution times over 10 runs (including preprocessing), with z_{avg} the average solution costs over 10 runs, and with z_{best} the best solution costs over 10 runs. When the genetic algorithm is tested, we denote with τ_{avg} the average convergence time obtained over 10 runs, measured as the time elapsed until the last improvement.
In the following experiments, the exact solution method and the genetic algorithm proposed are denoted respectively as MILP and GA. The results obtained on FSTSP-VDS instances are compared with the ones obtained by the constructive heuristic (hereafter called CH) of Raj and Murray (2020) [7], available at github.com/optimatorlab/mFSTSP-VDS. The results obtained on TSP-D instances are compared with the ones obtained by the branch-and-price method (hereafter called BP) of Roberti and Ruthmair (2021) [18], available at mario.ruthmair.at/?page_id=226.

Table 7.1: Genetic algorithm parameters.

Parameter		Value
η	Population size	200
γ	Number of offspring in a generation	40
n_{close}	Number of close individuals considered for distance evaluation	5
λ	Biased fitness trade-off parameter	0.9
It_{max}	Max #iterations without improvement	400
epochs	Number of iterations	2000

7.1 Test instances

7.1.1 FSTSP-VDS instances

The FSTSP-VDS instances used in these experiments are the ones used by Raj and Murray (2020)[7], available at github.com/optimatorlab/mFSTSP-VDS. The test set consists of two sets with customer locations in the Seattle and Washington area. The first set consists of 10 problems for each of four levels of the number of customers (10, 25, 50, and 100) generated on a service area of 918.0 km^2, for a total of 40 problem instances. The second set consists of 10 problems, each one with 50 customers, for each of four levels of service area (57.4, 229.5, 516.4,

and 918.0 km^2), for a total of 40 problems. In all instances, customer locations are generated from a uniform distribution on the road network within the service area. For each level of customers and for each level of service area, 5 instances featured a centrally-located depot and 5 instances contained a depot at the periphery.
The drone battery capacity is equal to $B = 500$ kJ. The minimum and maximum drone speeds are set respectively to 0.1 m/s and 40.0 m/s. The takeoff and landing drone speeds are constants and set respectively to 10 m/s and 5 m/s. The cruise altitude h is set to 50 meters. The maximum drone payload capacity is 5 lbs, customers associated with a parcel weight that exceed this capacity must be served by the truck. Truck and drone delivery times σ_T and σ_D are set to 30 and 60 seconds respectively. Drone launch and retrieval times, denoted respectively with s_L and s_R, are set to 60 and 30 seconds respectively. The distance d between two locations are computed using the Haversine distance formula as follows:

$$d = 2R \arcsin\left(\sqrt{\sin^2\left(\frac{\theta_2 - \theta_1}{2}\right) + \cos(\theta_1)\cos(\theta_2)\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right)}\right) \tag{7.1}$$

where θ_1, φ_1 represent the latitude and longitude of the first location, θ_2, φ_2 the latitude and longitude of the second location, and $R = 6378100.0$ denotes the radius of the earth in meters. The Liu et al. [11] model is used to characterize the drone power consumption with respect to speed and payload weight. The Liu et al. [11] model parameters are defined in chapter 4.
By comparing our solution costs with the ones of Raj and Murray (2020) [7], we found a discrepancy in the drone traveling times. In particular, by looking at their code (available at `github.com/optimatorlab/mFSTSP-VDS`), we assessed that a constant time of 0.5 seconds required to rotate the drone is added every time the drone is launched. Even if this parameter is not described in their work [7], we included it in order to produce comparable results.

7.1.2 TSP-D instances

The TSP-D instances used in these experiments are the ones used by Roberti and Ruthmair (2021) [18], originally generated by Poikonen et al. [17] and available at `mario.ruthmair.at/?page_id=226`. In particular, the test set consists of a total of 100 instances with 9, 19, 29 and 39 customers, 25 for each problem size. The instances were created by randomly locating the depot and the customers on a 50-by-50 grid using a uniform distribution on both axis. Given two locations i and j and the corresponding $x - y$ coordinates $(x_i,\ y_i)$ and $(x_j,\ y_j)$, truck travel times are computed according to the Manhattan (taxicab) metric, i.e., $t_{ij}^T = \lfloor |x_i - x_j| + |y_i - j_j| \rfloor$, drone travel times are instead computed according

to the Euclidean distance metric at a speed that is α times faster the truck, i.e., $t_{ij}^D = \lfloor \sqrt{(x_i - x_j)^2 + (y_i - j_j)^2}/\alpha \rfloor$. Each instance is solved with $\alpha = 1, 2, 3$, for a total of 300 instances.

7.2 Computational results on single drone FSTSP-VDS instances

In this section we compare the performances of the exact method (MILP) and of the genetic algorithm (GA) we proposed in Chapter 5 and Chapter 6, against the state-of-the-art heuristic of Raj and Murray (2020) [7] (CH). Due to the problem complexity, the MILP approach was tested only on FSTSP-VDS instances with 10 and 25 customers. Table 7.2 summarizes the results achieved by the three methods on the FSTSP-VDS instances for different instance sizes. The column Opt indicates the number of instances solved to optimality. The column ΔTSP indicate the average cost difference with respect to the optimal TSP solution costs. Finally, column ΔCH denotes the average percentage solution cost difference of the GA approach with respect to the CH approach.

Table 7.2: Computational results on the FSTS-VDS.

		MILP			CH			GA				
#cust	#instances	Opt	t_{avg}	%ΔTSP	Opt	t	%ΔTSP	Opt	t_{avg}	τ_{avg}	%ΔTSP	%ΔCH
10	10	10	1.65	-25.26	1	0.20	-20.31	10	53.76	0.80	-25.26	-6.10
25	10	1	3606.07	-24.04	0	2.83	-20.86	1	134.41	25.62	-25.80	-6.23
50	50	-	-	-	-	28.09	-19.70	-	302.80	150.71	-24.00	-5.39
100	10	-	-	-	-	454.70	-18.69	-	832.18	693.39	-24.51	-7.15
Overall	80	11			1		-19.78	11			-24.45	-5.81

Table 7.2 shows that all the FSTSP-VDS instances with 10 customers and one instance with 25 customers are solved to optimality using the MILP approach. Each one of these 11 instances are solved to optimality also by the GA approach, whereas, the CH approach is only able to solve to optimality one instance. The average improvement over the TSP optimal solutions costs is 19.78 % for the CH approach, and 24.45 % for the GA approach. Table A.1 and Table A.2 show the results obtained by the three methods on instances with 10 and 25 customers respectively. The columns %GAP$_{avg}$ and %GAP$_{min}$ indicate the average and minimum CPLEX gap over 10 runs. For all the FSTSP-VDS instances with 10 and 25 customers, the GA approach obtained the same solution cost at every run.
Even if only one instance with 25 customers is solved to optimality by the MILP

approach, the average CPLEX gap %GAP$_{avg}$ is 12.13 %, allowing the MILP approach to obtain an average improvement of 24.04 % over the optimal TSP solution costs, against the 19.78 % average improvement of the CH approach. Table A.3 and Table A.4 show the results obtained by the GA and the CH approaches on instances with 50 and 100 customers respectively. Even if the CH approach is about an order of magnitude faster than the GA approach on instances with up to 50 customers, and about two times faster on instances with 100 customers, in term of solution costs, the GA approach is much better than the CH approach. In fact, all solution costs of the CH approach have been improved by the GA approach, with an average improvement of 5.81%, that grows up to to 7.15% on instances with 100 customers. Finally, it interesting to note how the average improvement over the optimal TSP solutions costs is close to 25% for all instance sizes tested.

7.2.1 Sensitivity analysis on optimal drone leg speeds

Computing the exact optimal drone speeds related to the minimum and maximum drone travel times for each possible drone leg can be quite demanding when dealing with large instances. Therefore, as alternative, we proposed a grid-search method that, for each drone leg, computes the optimal drone speeds over a discretization of the drone speed domain with a discretization step δ m/s.
To understand the solution cost deterioration of this approach, we tested the genetic algorithm using the same methodology of section 7.2 but with a discretization step $\delta = 0.5$ m/s on the FSTSP-VDS instances with 10 customers. We refer to this version of the genetic algorithm with discrete optimal speeds, as GA$_\delta$. The results are summarized in Table 7.3. As expected, GA obtained better results than GA$_\delta$, but only for 6 out of 10 instances, and in average, the difference is only 0.08%. Therefore, this preliminary sensitivity analysis suggests that, when computing the exact minimum and maximum drone travel times is not practical, a valid alternative is to discretize the drone speed domain, and computing the optimal drone leg speeds (associated with the minimum and maximum drone leg travel times) using simpler and less demanding approaches.

Table 7.3: Comparison between continuous optimal and discrete optimal min/max drone travel times on FSTSP-VDS instances with 10 customers. In this setting, the discretization step of the drone speed domain is equal to $\delta = 0.5$ m/s.

	GA_δ				GA				
Instance	z_{avg}	z_{best}	t_{avg}	τ_{avg}	z_{avg}	z_{best}	t_{avg}	τ_{avg}	$\%\Delta z_{avg}$
20191230T145624016194	4758.73	4758.73	55.60	0.33	**4754.38**	**4754.38**	53.93	0.81	-0.09
20191230T145645377021	4694.12	4694.12	55.84	0.42	**4687.37**	**4687.37**	54.08	0.64	-0.14
20191230T145706863992	**4339.44**	**4339.44**	57.00	0.28	**4339.44**	**4339.44**	54.83	0.48	0.00
20191230T145728368390	5298.52	5298.52	55.03	0.68	**5293.51**	**5293.51**	52.86	0.97	-0.09
20191230T145749863540	**5370.37**	**5370.37**	56.55	0.38	**5370.37**	**5370.37**	54.27	0.51	0.00
20191230T145854314056	**5996.43**	**5996.43**	56.63	0.26	**5996.43**	**5996.43**	54.37	0.44	0.00
20191230T145916460302	3922.90	3922.90	54.54	0.33	**3916.94**	**3916.94**	52.85	0.61	-0.15
20191230T145938067895	6267.06	6267.06	55.22	0.35	**6255.81**	**6255.81**	53.57	0.47	-0.18
20191230T145959409904	4393.96	4393.96	54.73	0.27	**4389.31**	**4389.31**	52.95	0.36	-0.11
20191230T150020711011	**4890.98**	**4890.98**	55.42	1.29	**4890.98**	**4890.98**	53.94	2.68	0.00
Average									-0.08

7.3 Computational results on TSP-D instances

Table 7.4 reports the results achieved on the TSP-D by the BP approach and the GA approach. The table summarizes the results for each instance size (#cust) and for each value of $\alpha = 1, 2, 3$.
Column #instances reports the number of instances evaluated (i.e., 25). The column OPT reports the number of instances solved to optimality. Since the only known optimal solution costs are the ones of the BP approach, the number of optimal solutions found by the GA approach cannot be greater than the number of optimal solutions found by the BP approach. Tables A.5, A.6, A.7 and A.8 report the detailed results achieved by the BP approach and the GA approach on TSP-D instances with, respectively, 9, 19, 29 and 39 customers.

Table 7.4: Computational results on the TSP-D

			BP		GA		
#cust	α	#instances	t	OPT	t_{avg}	τ_{avg}	OPT
	1	25	0.3	25	47.8	1.6	25
9	2	25	0.3	25	38.9	0.1	25
	3	25	0.3	25	38.3	0.2	25
	1	25	47.4	25	111.3	1.2	14
19	2	25	20.7	25	90.5	2.6	24
	3	25	17.6	25	85.9	4.3	25
	1	25	1209.0	19	198.8	4.3	7
29	2	25	554.3	24	149.7	11.3	21
	3	25	537.0	23	132.6	16.8	19
	1	25	3219.7	1	261.1	19.8	0
39	2	25	2604.4	3	193.3	18.2	2
	3	25	1414.2	7	189.2	36.8	7
All		300		227			194

Table 7.4 shows that GA can solve all 75 instances with 9 customers, 63 out of 75 instances with 19 customers, 47 out of 66 instances with 29 customers and 9 out of 10 instances with 39 customers, that is, 194 out of 227 instances (85.5%). Table 7.4 shows also that the GA approach is much faster than the BP approach. In particular, for instances with 39 customers, the average convergence time τ_{avg} is two orders of magnitude smaller than the average computing time of the BP approach. Table 7.5 reports the percentage difference of the solutions costs obtained by the GA approach relative to the BP approach. For a meaningful comparison, we only considered the instances solved to optimality by the BP. Table 7.5 shows that, the solutions obtained with the GA approach are on average only 0.30% higher than the optimal solutions found by the BP approach.

Table 7.5: Percentage differences of the GA approach solution times relative to the BP approach.

	α			
#cust	1	2	3	avg
9	0.07	0.00	0.00	0.02
19	1.34	0.03	0.01	0.46
29	1.19	0.08	0.27	0.47
39	1.35	0.19	0.04	0.20
avg	0.85	0.04	0.08	0.30

It is also interesting to note that for $\alpha = 1$ (the drone is slower), GA obtains much worse results even for relatively small instances. From a preliminary analysis we found that this deterioration is caused by a too small value assigned to the maximum offset parameter of the approximated version of Algorithm 2, that determines the maximum length of an operation, and that was set to 5. In fact, when $\alpha = 1$ the drone is slower, and this imply longer operations on average.

Chapter 8

Conclusions and Future Work

In this thesis, we studied the *Flying Sidekick Traveling Salesman Problem with Variable Drone Speeds* (FSTSP-VDS), a variant of the FSTSP in which the drone power consumption is modeled as a non-linear function with respect to speed and parcel weight, and in which the drone can fly from one location to another at different speeds. We proposed a compact MILP formulation and a genetic algorithm. Both solutions are based on the idea of computing in advance the minimum and maximum feasible drone leg travel times in order to avoid to handle drone speeds as decision variables. The resulting MILP formulation allows to easily solve to optimality FSTSP-VDS instances with 10 customers, generally in few seconds or even less. We have also been able to solve to optimality one instance with 25 customers, but solving these kind of instances require a considerable amount of computing resources. For this purpose, we devised a genetic algorithm to tackle larger FSTSP-VDS instances. The core idea of our heuristic solution relies on an encoding scheme that allows to represent $\Omega(2^n)$ solutions with a single sequence (chromosome), and on an efficient procedure that in time $\Theta(n^3)$ retrieves the best solution among all the ones represented by the sequence. We also presented an approximated variant of this retrieval method that runs in time $\Theta(n)$ instead of $\Theta(n^3)$ by limiting the maximum operation length, which was used for the experiments.

The proposed genetic algorithm allows to obtain a 6% average improvement over the current state of the art FSTSP-VDS heuristic. We also show that our genetic algorithm is the first heuristic that allows to find known optimal solutions of TSP-D instances with up to 39 customers. In particular, our genetic algorithm was tested on 227 TSP-D instances with known optimal solutions and with a number of customers ranging from 9 to 39, obtaining an average optimality gap of only 0.3% in a fraction of the time required by the state-of-the-art exact methods. We also presented a cutting plane approach based on the proposed MILP formulation that works without requiring the exact minimum and maximum drone travel

times in advance, refining these quantities only when needed. However, the above method was not tested in this work, and could be interesting to compare it with the MILP approach. The experiments carried out on TSP-D instances show a loss of performance when our genetic algorithm is tested with the slowest drone speed configuration ($\alpha = 1$). From a preliminary analysis we found that this degradation was caused by setting the maximum operation length equal to 5, a value that turned out to be too small. In fact, if the drone speed is low, the operations will tend to be longer. Therefore, it would be interesting to conduct a sensitivity analysis to understand how much the maximum operation length parameter affects the performances under different scenarios.

Since computing optimal speeds could be too demanding when dealing with large instances for real-world application, one could asks how much performance loss you get by applying a discretization of the drone speed domain. A preliminary analysis have been conducted in our thesis, showing that a discretization step of 0.5 m/s causes a loss inferior of 0.1%, but a more general sensitivity analysis should be carried out.

Another topic of interest concerns the study and integration of a robust approach in the genetic algorithm devised during the this thesis. The idea is to have a population of feasible individuals and a population of infeasible ones, such that, an individual is feasible if it satisfies the robustness requirements, and infeasible otherwise. The aim of this approach is to explore the boundary of the feasible region of the solution space, and therefore, it could be particularly suited in this context. A similar approach has been devised for the deterministic Vehicle Routing Problem with Time Windows (VRP-D) by Vidal et al. [25], whereas, to the best of our knowledge, this simple approach has not yet been explored in the context of robust optimization and could provide a valid robust approach for larger instances of the FSTSP-VDS.

Appendix A

Extended Results

Table A.1: Extended results on the FSTS-VDS instances with 10 customers.

		MILP						CH			GA				
Instance	TSP	z_{avg}	z_{best}	%GAP$_{avg}$	%GAP$_{min}$	t_{avg}	%ΔTSP	z	t	%ΔTSP	z_{avg}	z_{best}	t_{avg}	τ_{avg}	%ΔTSP
20191230T145624016194	6851.99	**4754.38**	**4754.38**	0.00	0.00	0.60	-30.61	5189.09	0.15	-24.27	**4754.38**	**4754.38**	53.93	0.81	-30.61
20191230T145645377021	6096.01	**4687.37**	**4687.37**	0.00	0.00	4.27	-23.11	5058.22	0.30	-17.02	**4687.37**	**4687.37**	54.08	0.64	-23.11
20191230T145706863992	5917.73	**4339.44**	**4339.44**	0.00	0.00	0.83	-26.67	4383.55	0.22	-25.93	**4339.44**	**4339.44**	54.83	0.48	-26.67
20191230T145728368390	7242.69	**5293.51**	**5293.51**	0.00	0.00	4.25	-26.91	5968.15	0.19	-17.60	**5293.51**	**5293.51**	52.86	0.97	-26.91
20191230T145749863540	6410.57	**5370.37**	**5370.37**	0.00	0.00	1.01	-16.23	5604.95	0.20	-12.57	**5370.37**	**5370.37**	54.27	0.51	-16.23
20191230T145854314056	7552.52	**5996.43**	**5996.43**	0.00	0.00	0.89	-20.60	6593.85	0.26	-12.69	**5996.43**	**5996.43**	54.37	0.44	-20.60
20191230T145916460302	5493.20	**3916.94**	**3916.94**	0.00	0.00	1.89	-28.69	4186.19	0.23	-23.79	**3916.94**	**3916.94**	52.85	0.61	-28.69
20191230T145938067895	8125.98	**6255.81**	**6255.81**	0.00	0.00	0.88	-23.01	6880.43	0.18	-15.33	**6255.81**	**6255.81**	53.57	0.47	-23.01
20191230T145959409904	6821.98	**4389.31**	**4389.31**	0.00	0.00	0.84	-35.66	4584.39	0.16	-32.80	**4389.31**	**4389.31**	52.95	0.36	-35.66
20191230T150020711011	6202.56	**4890.98**	**4890.98**	0.00	0.00	1.11	-21.15	**4890.98**	0.14	-21.15	**4890.98**	**4890.98**	53.94	2.68	-21.15
Average				0.00	0.00	1.65	-25.26		0.20	-20.31			53.76	0.80	-25.26

Table A.2: Extended results on the FSTS-VDS instances with 25 customers.

		Exact method						CH			GA				
Instance	TSP	z_{avg}	z_{best}	%GAP$_{avg}$	%GAP$_{min}$	t_{avg}	%ΔTSP	z	t	%ΔTSP	z_{avg}	z_{best}	t_{avg}	τ_{avg}	%ΔTSP
20191230T150126383669	11167.17	9005.35	8803.54	10.42	8.66	3610.30	-19.36	9468.35	2.17	-15.21	**8774.42**	**8774.42**	137.01	57.11	-21.43
20191230T150218531563	10390.13	8096.16	7952.84	-11.42	7.36	3609.41	-22.08	8313.86	2.87	-19.98	**7938.85**	**7938.85**	126.15	45.65	-23.59
20191230T150311834153	11081.93	8296.01	**8267.76**	9.20	7.19	3609.98	-25.14	8633.73	2.36	-22.09	**8267.76**	**8267.76**	131.19	40.37	-25.39
20191230T150403491108	12751.29	8703.58	8549.45	11.79	9.66	3612.34	-31.74	9211.20	2.22	-27.76	**8527.94**	**8527.94**	129.17	11.87	-33.12
20191230T150455119746	10737.63	7665.77	**7540.49**	8.42	6.43	3612.47	-28.61	8614.61	4.35	-19.77	**7540.49**	**7540.49**	149.72	15.49	-29.78
20191230T150732856144	10911.88	8851.22	8571.03	16.97	12.77	3612.28	-18.88	8962.70	2.73	-17.86	**8485.77**	**8485.77**	140.03	22.08	-22.23
20191230T150825180963	11201.48	7538.06	7429.48	15.75	13.69	3611.82	-32.70	7834.20	3.28	-30.06	**7412.20**	**7412.20**	139.72	14.37	-33.83
20191230T150917303441	10751.94	8609.39	8410.58	18.96	16.41	3613.00	-19.93	8921.36	3.68	-17.03	**8248.81**	**8248.81**	124.20	12.33	-23.28
20191230T151009384583	10830.13	8784.30	**8762.32**	0.83	0.00	3556.17	-18.89	8944.79	1.90	-17.41	**8762.32**	**8762.32**	137.87	27.72	-19.09
20191230T151101326987	10973.56	8439.17	8408.46	17.53	15.74	3612.93	-23.10	8618.62	2.71	-21.46	**8096.49**	**8096.49**	128.99	9.18	-26.22
Average				12.13	9.79	3606.07	-24.04		2.83	-20.86			134.41	25.62	**-25.80**

Table A.3: Extended results on the FSTS-VDS instances with 50 customers.

		CH			GA				
Instance	TSP	z	t	%ΔTSP	z_{avg}	z_{best}	t_{avg}	τ_{avg}	%ΔTSP
20191118T122820306867	5837.51	5110.51	7.81	-12.45	5102.28	**5097.62**	356.00	117.95	-12.59
20191118T122949122725	5609.43	5173.91	36.08	-7.76	4868.72	**4864.31**	337.21	187.90	-13.20
20191118T123118221785	5704.43	4911.42	33.48	-13.90	4881.47	**4870.75**	352.03	153.59	-14.43
20191118T123246160894	5864.41	5077.31	34.63	-13.42	5048.48	**5047.97**	361.95	145.42	-13.91
20191118T123534617288	5907.47	5231.22	33.73	-11.45	5122.50	**5116.02**	362.26	240.56	-13.29
20191118T123824123933	6055.90	5526.22	25.04	-8.75	5443.72	**5435.16**	342.41	228.24	-10.11
20191118T123955510273	6152.89	5118.24	34.36	-16.82	5041.33	**5037.97**	322.06	172.40	-18.07
20191118T124124624402	6316.06	5495.97	26.69	-12.98	5447.75	**5439.70**	320.85	107.19	-13.75
20191118T124254325293	8223.48	6587.52	26.26	-19.89	6467.38	**6456.69**	312.10	179.08	-21.35
20191118T124427752580	8592.41	6604.07	36.62	-23.14	6318.03	**6306.87**	301.57	138.32	-26.47
20191118T124843195400	9537.48	6881.29	33.34	-27.85	6544.83	**6502.85**	295.97	175.18	-31.38
20191118T125015912865	8899.09	6830.77	29.88	-23.24	6603.57	**6576.79**	290.56	135.51	-25.80
20191118T125147711719	8588.44	6907.53	26.81	-19.57	6732.61	**6732.61**	358.88	155.41	-21.61
20191118T125441538212	9133.16	7512.22	34.55	-17.75	7103.27	**7076.87**	286.79	145.71	-22.23
20191118T125614299548	9366.60	7358.56	43.46	-21.44	6779.14	**6687.14**	341.46	188.39	-27.62
20191118T125747188478	14531.78	10643.55	24.34	-26.76	**9307.43**	**9307.43**	268.53	125.87	-35.95
20191118T125923266294	13289.57	10784.55	19.53	-18.85	**10387.50**	**10387.50**	269.29	119.29	-21.84
20191118T130058846733	12897.98	11163.37	30.46	-13.45	10666.97	**10602.74**	277.94	183.52	-17.30
20191118T130234948300	12625.44	10872.93	33.63	-13.88	9877.63	**9847.68**	269.01	149.23	-21.76
20191118T130410224030	13706.59	10775.24	20.00	-21.39	10278.63	**10217.28**	272.16	168.66	-25.01
20191118T130545047476	12662.39	9623.64	22.38	-24.00	9399.71	**9337.92**	283.60	150.27	-25.77
20191118T130850731760	12734.93	10486.45	34.78	-17.66	9383.57	**9324.94**	266.18	92.03	-26.32
20191118T131151763979	11894.31	10632.02	32.99	-10.61	9414.56	**9387.22**	256.24	102.70	-20.85
20191118T150546225143	15253.31	12867.87	28.69	-15.64	11151.18	**11109.32**	256.19	98.17	-26.89
20191118T150723196947	15958.80	12891.56	20.74	-19.22	12344.74	**12323.72**	340.93	252.61	-22.65
20191118T151307990474	15115.17	11877.41	23.69	-21.42	10804.19	**10790.33**	312.35	211.70	-28.52
20191118T151445730078	16934.47	12224.73	21.46	-27.81	11438.43	**11421.88**	306.89	113.60	-32.45
20191118T151744822171	15738.06	12411.80	23.20	-21.14	11895.01	**11865.46**	287.18	151.23	-24.42
20191118T151923741580	15223.77	12093.90	21.98	-20.56	11216.66	**11168.75**	290.86	124.03	-26.32
20191118T152913966760	6680.58	5323.70	29.70	-20.31	5203.64	**5193.42**	276.17	166.80	-22.11
20191118T153332147450	6305.60	5122.03	29.03	-18.77	5085.86	**5080.79**	369.44	111.15	-19.34
20191118T153929356548	9352.30	7218.63	28.66	-22.81	6794.03	**6758.28**	285.00	190.19	-27.35
20191118T154101395017	9257.12	7007.66	27.19	-24.30	6816.71	**6782.98**	313.23	147.84	-26.36
20191118T154538429291	9236.86	7061.48	34.10	-23.55	6833.79	**6824.16**	323.05	112.87	-26.02
20191118T155800923519	13460.15	10986.22	17.45	-18.38	10593.27	**10472.86**	297.51	176.53	-21.30
20191118T155936386731	14037.68	10687.31	31.86	-23.87	9613.68	**9596.48**	311.61	176.49	-31.52
20191118T160426829317	15453.26	11697.51	18.58	-24.30	11224.94	**11156.01**	304.99	147.42	-27.36
20191118T160726433090	16460.07	12968.73	28.63	-21.21	12181.81	**12164.38**	282.35	217.31	-25.99
20191118T160903638340	15657.85	12043.60	28.40	-23.08	11143.44	**11129.11**	302.85	137.51	-28.83
20191118T161041628422	16554.56	13001.24	25.51	-21.46	12173.16	**12145.29**	286.47	105.49	-26.47
20191230T151658283335	16563.57	12145.97	26.04	-26.67	11390.26	**11367.70**	292.03	156.66	-31.23
20191230T151843966978	17002.65	13830.03	26.05	-18.66	11973.30	**11947.37**	340.12	114.31	-29.58
20191230T152029684793	15477.33	13212.46	27.30	-14.63	12138.82	**12043.87**	274.32	155.23	-21.57
20191230T152216413583	14470.88	11634.35	19.68	-19.60	11017.45	**10996.94**	270.82	102.26	-23.86
20191230T152402488108	14885.45	12239.45	23.64	-17.78	11606.48	**11590.11**	274.54	161.11	-22.03
20191230T152549310182	15004.47	11824.93	62.79	-21.19	10830.29	**10759.97**	281.54	121.62	-27.82
20191230T152737169828	14670.67	11609.56	26.17	-20.87	11074.00	**11072.46**	279.79	126.60	-24.52
20191230T153406366529	16392.40	11865.23	19.08	-27.62	**11253.33**	**11253.33**	277.55	155.69	-31.35
20191230T153549168847	16376.98	12193.44	30.83	-25.55	11419.56	**11394.65**	314.02	116.17	-30.27
20191230T163954653903	16391.12	11896.50	23.40	-27.42	10947.43	**10946.37**	283.21	122.35	-33.21
Average				-19.70					**-24.00**

Table A.4: Extended results on the FSTS-VDS instances with 100 customers.

		CH			GA				
Instance	TSP	z	t	%ΔTSP	z_{avg}	z_{best}	t_{avg}	τ_{avg}	%ΔTSP
20191230T153733732593	24216.39	19742.67	585.57	-18.47	17778.94	**17536.99**	851.78	723.19	-26.58
20191230T154541035990	22984.40	18493.66	448.74	-19.54	17392.82	**17296.59**	846.62	769.35	-24.33
20191230T155255485975	23253.00	18747.73	380.01	-19.38	17990.98	**17859.40**	838.18	705.09	-22.63
20191230T160316717797	23420.32	19300.25	427.26	-17.59	17745.07	**17640.08**	840.32	650.58	-24.23
20191230T161653345275	23570.09	19408.47	532.63	-17.66	17501.17	**17386.07**	902.60	800.20	-25.75
20191230T162058887701	23836.02	19378.78	368.33	-18.70	18062.90	**17925.70**	869.33	682.36	-24.22
20191230T162504128235	23067.20	18218.28	518.52	-21.02	17233.59	**17105.98**	779.81	650.39	-25.29
20191230T164411574945	22553.21	18863.16	384.89	-16.36	17980.04	**17884.25**	796.40	646.61	-20.28
20191230T164814918251	24520.73	19462.68	439.15	-20.63	17772.70	**17635.53**	790.82	655.35	-27.52
20191230T165217106753	23934.78	19741.76	461.94	-17.52	18131.90	**18036.76**	805.93	650.76	-24.24
Average				-18.69					**-24.51**

Table A.5: Extended results on the TSP-D with 9 customers.

			BP		GA				
Instance	TSP	α	z	t	z_{avg}	z_{best}	t_{avg}	τ_{avg}	%ΔBP
		1	149	0.18	149.00	149	49.69	0.13	0.00
10-1	177	2	141	0.97	141.00	141	40.07	0.13	0.00
		3	113	0.38	113.00	113	37.48	0.08	0.00
		1	138	0.67	138.00	138	47.43	0.09	0.00
10-2	175	2	111	0.21	111.00	111	37.85	0.19	0.00
		3	109	0.55	109.00	109	38.58	0.13	0.00
		1	151	0.53	151.00	151	48.03	0.07	0.00
10-3	179	2	110	0.1	110.00	110	37.69	0.06	0.00
		3	98	0.11	98.00	98	41.01	0.06	0.00
		1	138	0.2	138.00	138	47.29	0.10	0.00
10-4	178	2	113	0.25	113.00	113	36.61	0.12	0.00
		3	94	0.22	94.00	94	38.45	0.11	0.00
		1	136	0.2	136.20	136	48.14	13.47	0.15
10-5	156	2	114	0.35	114.00	114	39.41	0.18	0.00
		3	85	0.18	85.00	85	39.05	0.12	0.00
		1	140	0.25	140.00	140	47.55	0.08	0.00
10-6	180	2	118	0.66	118.00	118	38.23	0.16	0.00
		3	113	0.39	113.00	113	38.89	0.07	0.00
		1	109	0.17	110.20	109	46.91	3.23	1.10
10-7	159	2	87	0.19	87.00	87	36.54	0.09	0.00
		3	79	0.19	79.00	79	38.79	0.24	0.00
		1	101	0.32	101.00	101	45.86	0.06	0.00
10-8	139	2	80	0.3	80.00	80	38.54	0.10	0.00
		3	66	0.17	66.00	66	39.40	0.12	0.00
		1	152	0.13	152.00	152	48.60	0.10	0.00
10-9	189	2	142	0.21	142.00	142	39.43	0.17	0.00
		3	139	0.34	139.00	139	39.87	0.12	0.00
		1	118	0.27	118.00	118	45.97	0.11	0.00
10-10	177	2	85	0.64	85.00	85	37.51	0.12	0.00
		3	67	0.38	67.00	67	35.39	0.12	0.00

Continued on next page

Table A.5 – continued from previous page

			BP		GA				
Instance	TSP	α	z	t	z_{avg}	z_{best}	t_{avg}	τ_{avg}	%ΔBP
		1	142	0.31	142.00	142	48.40	0.10	0.00
10-11	172	2	106	0.13	106.00	106	39.58	0.11	0.00
		3	81	0.17	81.00	81	36.79	0.11	0.00
		1	138	0.22	138.00	138	46.21	0.09	0.00
10-12	188	2	107	0.19	107.00	107	37.72	0.13	0.00
		3	74	0.2	74.00	74	37.26	0.13	0.00
		1	114	0.16	114.00	114	48.30	7.14	0.00
10-13	175	2	90	0.29	90.00	90	41.58	0.11	0.00
		3	82	0.56	82.00	82	38.84	0.82	0.00
		1	115	0.24	115.60	115	48.34	15.01	0.52
10-14	163	2	90	0.19	90.00	90	40.86	0.16	0.00
		3	87	0.21	87.00	87	37.92	0.11	0.00
		1	145	0.24	145.00	145	47.48	0.07	0.00
10-15	199	2	121	0.21	121.00	121	39.06	0.11	0.00
		3	97	0.39	97.00	97	36.33	0.12	0.00
		1	136	0.48	136.00	136	47.12	0.09	0.00
10-16	174	2	103	0.19	103.00	103	38.13	0.08	0.00
		3	84	0.26	84.00	84	36.20	0.09	0.00
		1	141	0.14	141.00	141	46.67	0.09	0.00
10-17	201	2	108	0.6	108.00	108	40.11	0.12	0.00
		3	92	0.41	92.00	92	37.58	0.21	0.00
		1	127	0.36	127.00	127	47.85	0.08	0.00
10-18	166	2	101	0.16	101.00	101	39.49	0.19	0.00
		3	100	0.3	100.00	100	37.53	0.22	0.00
		1	139	0.2	139.00	139	48.76	0.09	0.00
10-19	177	2	129	0.34	129.00	129	40.20	0.17	0.00
		3	118	0.53	118.00	118	38.19	0.09	0.00
		1	128	0.16	128.00	128	49.20	0.09	0.00
10-20	155	2	114	0.3	114.00	114	38.82	0.08	0.00
		3	105	0.13	105.00	105	37.14	0.13	0.00
		1	96	0.62	96.00	96	48.23	0.08	0.00
10-21	133	2	71	0.19	71.00	71	39.24	0.14	0.00
		3	67	0.19	67.00	67	37.69	0.08	0.00
		1	121	0.69	121.00	121	47.30	0.10	0.00
10-22	153	2	86	0.24	86.00	86	38.68	0.11	0.00
		3	67	0.22	67.00	67	37.19	0.15	0.00
		1	165	0.5	165.00	165	50.10	0.09	0.00
10-23	182	2	129	0.19	129.00	129	38.36	0.06	0.00
		3	121	0.53	121.00	121	40.28	0.09	0.00
		1	126	0.69	126.00	126	46.46	0.08	0.00
10-24	168	2	94	0.29	94.00	94	37.94	0.25	0.00
		3	81	0.19	81.00	81	40.01	0.18	0.00
		1	136	0.2	136.00	136	49.05	0.10	0.00
10-25	169	2	131	0.77	131.00	131	39.81	0.08	0.00
		3	125	0.19	125.00	125	41.15	0.10	0.00
Average				0.32			41.65	0.64	0.02

Table A.6: Extended results on the TSP-D with 19 customers.

			BP		GA				
Instance	TSP	α	z	t	z_{avg}	z_{best}	t_{avg}	τ_{avg}	%ΔBP
		1	188	165.21	189.00	189	112.94	1.40	0.53
20-1	239	2	153	20.63	153.00	153	90.60	1.27	0.00
		3	131	3.71	131.00	131	88.57	2.42	0.00
		1	180	14.17	180.00	180	113.10	0.94	0.00
20-2	240	2	151	22.79	151.00	151	90.87	3.18	0.00
		3	130	2.35	130.00	130	81.37	3.31	0.00
		1	196	20.75	206.00	206	111.12	1.27	5.10
20-3	257	2	169	16.51	169.00	169	92.80	1.86	0.00
		3	154	14.13	154.00	154	85.25	2.54	0.00
		1	179	50.74	179.00	179	110.75	1.14	0.00
20-4	236	2	140	12.61	140.00	140	91.50	1.59	0.00
		3	125	26.8	125.00	125	82.98	5.47	0.00
		1	164	12.33	164.00	164	111.27	1.88	0.00
20-5	224	2	138	32.41	138.00	138	90.86	1.91	0.00
		3	129	53.91	129.20	129	88.09	33.21	0.16
		1	161	26.58	161.00	161	108.92	0.98	0.00
20-6	222	2	121	2.81	121.00	121	90.93	2.13	0.00
		3	110	17.7	110.00	110	84.93	2.53	0.00
		1	179	64.5	184.00	184	109.15	0.92	2.79
20-7	226	2	148	42.54	148.00	148	91.54	1.91	0.00
		3	123	10.11	123.00	123	87.23	4.12	0.00
		1	180	56.74	187.00	187	110.82	1.02	3.89
20-8	224	2	149	17.81	149.00	149	91.37	0.90	0.00
		3	131	2.59	131.00	131	82.44	2.51	0.00
		1	153	5.87	159.00	159	111.25	1.34	3.92
20-9	196	2	135	24.74	136.00	136	95.15	1.53	0.74
		3	130	51.35	130.00	130	91.32	1.70	0.00
		1	173	8.07	173.00	173	108.21	0.92	0.00
20-10	220	2	144	6.96	144.00	144	87.53	1.57	0.00
		3	130	1.85	130.00	130	89.80	2.83	0.00
		1	164	4.47	169.00	169	110.47	1.07	3.05
20-11	212	2	150	16.32	150.00	150	89.58	1.65	0.00
		3	148	29.76	148.00	148	91.48	1.75	0.00
		1	169	50.52	170.00	170	108.65	1.52	0.59
20-12	235	2	133	32.36	133.00	133	88.38	2.62	0.00
		3	123	36.75	123.00	123	90.36	2.07	0.00
		1	171	242.3	171.00	171	112.01	1.33	0.00
20-13	218	2	151	62.11	151.00	151	89.86	6.86	0.00
		3	129	32.89	129.00	129	90.15	6.81	0.00
		1	164	8.31	164.00	164	113.02	1.20	0.00
20-14	198	2	145	20.71	145.00	145	91.33	8.45	0.00
		3	134	8.67	134.00	134	85.15	2.30	0.00
		1	183	69.59	184.00	184	111.83	1.13	0.55
20-15	260	2	139	6.71	139.00	139	88.11	1.09	0.00
		3	121	5.17	121.00	121	81.19	3.73	0.00
		1	165	7.7	165.00	165	111.59	0.83	0.00
20-16	225	2	129	8.44	129.00	129	87.81	1.48	0.00
		3	118	3.6	118.00	118	85.82	5.41	0.00
		1	150	110.64	153.00	153	110.63	0.84	2.00
20-17	198	2	108	3.21	108.00	108	89.86	1.44	0.00
		3	101	4.32	101.00	101	84.95	1.91	0.00

Continued on next page

Table A.6 – continued from previous page

			BP		GA				
Instance	TSP	α	z	t	z_{avg}	z_{best}	t_{avg}	τ_{avg}	%ΔBP
		1	177	36.62	177.00	177	113.14	1.24	0.00
20-18	233	2	151	79.37	151.00	151	93.88	11.07	0.00
		3	135	7.07	135.00	135	82.06	4.06	0.00
		1	162	14.71	162.00	162	112.64	0.99	0.00
20-19	222	2	122	1.91	122.00	122	89.62	1.43	0.00
		3	114	13.49	114.00	114	82.85	1.39	0.00
		1	180	92.59	180.00	180	114.15	1.05	0.00
20-20	222	2	154	30.16	154.00	154	90.80	1.74	0.00
		3	148	55.68	148.00	148	86.78	1.58	0.00
		1	164	7.57	164.00	164	111.54	0.65	0.00
20-21	210	2	137	1.74	137.00	137	89.54	1.56	0.00
		3	129	8.75	129.00	129	85.91	1.75	0.00
		1	174	38.45	180.00	180	110.61	1.71	3.45
20-22	241	2	136	28.28	136.00	136	91.02	1.61	0.00
		3	114	13.92	114.00	114	86.08	4.81	0.00
		1	172	12.33	185.00	185	115.20	1.04	7.56
20-23	237	2	134	8.11	134.00	134	89.92	1.39	0.00
		3	123	10.61	123.00	123	83.52	1.70	0.00
		1	171	29.29	171.00	171	108.58	1.55	0.00
20-24	246	2	125	3.54	125.00	125	89.20	2.91	0.00
		3	111	7.59	111.00	111	84.28	3.86	0.00
		1	178	35.47	178.00	178	110.10	0.97	0.00
20-25	239	2	141	14.53	141.00	141	89.87	2.26	0.00
		3	122	15.99	122.00	122	84.15	3.28	0.00
Average				28.55			95.87	2.69	0.46

Table A.7: Extended results on the TSP-D with 29 customers.

			BP		GA				
Instance	TSP	α	z	t	z_{avg}	z_{best}	t_{avg}	τ_{avg}	%ΔBP
		1	-	-	-	205	198.45	5.02	-
30-1	257	2	164	469.12	164.00	164	155.45	11.52	0.00
		3	139	30.51	140.00	140	128.70	11.28	0.72
		1	209	3226.99	209.00	209	202.59	3.83	0.00
30-2	286	2	160	558.38	160.00	160	147.71	9.22	0.00
		3	142	543.07	142.50	142	127.50	21.01	0.35
		1	183	327.91	183.00	183	203.56	4.43	0.00
30-3	259	2	153	267.35	153.00	153	143.51	4.28	0.00
		3	143	253.01	143.00	143	135.64	26.32	0.00
		1	195	870.8	201.00	201	196.53	4.54	3.08
30-4	275	2	150	514.61	151.00	151	163.38	7.59	0.67
		3	129	1348.43	129.00	129	138.70	12.23	0.00
		1	186	347.47	188.00	188	202.42	3.99	1.08
30-5	246	2	161	659.56	162.00	162	162.48	11.64	0.62
		3	144	26.94	144.00	144	145.74	4.65	0.00
		1	186	225.18	189.00	189	193.44	2.67	1.61
30-6	263	2	156	434.72	156.00	156	147.46	4.27	0.00
		3	146	578.81	146.60	146	132.75	33.69	0.41
		1	182	1701.8	182.00	182	198.01	4.50	0.00
30-7	252	2	147	170.38	147.00	147	145.29	15.34	0.00
		3	131	317.04	131.00	131	130.24	12.30	0.00
		1	195	733.1	197.00	197	196.12	4.20	1.03
30-8	259	2	164	432.68	164.00	164	155.64	8.25	0.00
		3	155	744.19	155.00	155	131.64	6.04	0.00
		1	195	638.37	195.00	195	194.70	5.63	0.00
30-9	278	2	157	489.34	157.00	157	160.86	4.42	0.00
		3	143	618.7	143.00	143	123.63	15.53	0.00
		1	195	1143.05	198.00	198	198.64	4.50	1.54
30-10	268	2	159	1358.23	159.00	159	158.33	6.16	0.00
		3	139	143.52	139.00	139	138.26	5.18	0.00
		1	200	813.88	202	202	194.72	5.73	1.00
30-11	273	2	162	654.06	162.00	162	132.66	8.97	0.00
		3	144	1611.14	144.00	144	136.22	36.11	0.00
		1	210	988.27	210.00	210	197.11	5.90	0.00
30-12	287	2	160	51.76	160.00	160	148.68	5.20	0.00
		3	150	228.25	150.80	150	137.45	24.29	0.53
		1	200	449.49	201.00	201	200.68	3.98	0.50
30-13	275	2	157	124.14	157.00	157	142.01	19.52	0.00
		3	147	1025.19	147.00	147	125.67	10.93	0.00
		1	188	2371.3	190.00	190	198.79	4.31	1.06
30-14	238	2	160	906.98	160.00	160	155.15	20.33	0.00
		3	148	372.68	149.00	149	142.50	10.33	0.68
		1	185	1403.83	189.00	189	204.90	2.96	2.16
30-15	237	2	144	123.7	144.00	144	154.94	3.12	0.00
		3	137	824.39	137.00	137	136.96	10.73	0.00
		1	196	1072.32	196.00	196	199.78	4.09	0.00
30-16	237	2	156	378.4	156.00	156	145.18	6.72	0.00
		3	143	445.01	146.00	146	125.26	6.83	2.10
		1	196	1447.34	201.00	201	192.23	3.64	2.55
30-17	264	2	159	331.57	159.00	159	153.48	7.34	0.00
		3	147	217.76	148.00	147	138.37	38.20	0.68

Continued on next page

Table A.7 – continued from previous page

			BP		GA				
Instance	TSP	α	z	t	z_{avg}	z_{best}	t_{avg}	τ_{avg}	%ΔBP
		1	-	-	206.00	206	196.21	7.19	-
30-18	273	2	160	230.23	160.00	160	134.44	18.96	0.00
		3	136	304.84	136.00	136	136.53	11.60	0.00
		1	-	-	196.00	196	204.24	5.73	-
30-19	253	2	165	1223.66	165.00	165	145.66	9.33	0.00
		3	155	956.2	155.00	155	131.19	46.81	0.00
		1	180	797.61	189.00	189	202.72	4.96	5.00
30-20	249	2	147	1235.24	148.00	148	160.81	8.46	0.68
		3	-	-	138.00	138	140.62	6.43	-
		1	198	1657.66	202.00	202	205.98	5.02	2.02
30-21	248	2	164	2080.03	164.10	164	147.05	47.07	0.06
		3	142	467.45	143.00	143	125.33	6.53	0.70
		1	207	2754.53	207.00	207	193.96	3.47	0.00
30-22	268	2	173	271.54	173.00	173	123.92	8.37	0.00
		3	162	882.23	162.00	162	124.74	10.11	0.00
		1	-	-	191.00	191	198.70	5.81	0.00
30-23	253	2	-	-	151.00	151	149.50	9.13	-
		3	-	-	137.60	137	129.01	27.57	-
		1	-	-	183.00	183	200.24	8.28	-
30-24	247	2	146	276.59	146.00	146	155.66	17.78	0.00
		3	132	288.58	132.00	132	127.63	9.15	0.00
		1	-	-	224.00	224	212.04	8.91	-
30-25	268	2	180	61.65	180.00	180	151.82	6.09	0.00
		3	164	123.15	164.00	164	128.00	17.17	0.00
Average				736.76			157.83	11.20	0.47

Table A.8: Extended results on the TSP-D with 39 customers

			BP		GA				
Instance	TSP	α	z	t	z_{avg}	z_{best}	t_{avg}	τ_{avg}	%ΔBP
		1	-	-	226.00	226	265.33	14.94	-
40-1	303	2	-	-	172.00	172	194.12	18.62	-
		3	-	-	154.30	154	190.32	46.45	-
		1	-	-	219.00	219	291.17	7.60	-
40-2	291	2	173	2967.32	174.00	174	181.99	14.72	0.58
		3	152	819.28	152.00	152	190.80	16.22	0.00
		1	-	-	238.00	238	225.04	12.35	-
40-3	293	2	-	-	192.00	192	204.97	75.58	-
		3	-	-	178.60	178	173.38	54.06	-
		1	-	-	232.00	232	275.14	12.38	-
40-4	311	2	-	-	193.20	193	178.49	60.21	-
		3	167	2033	167.00	167	190.88	30.66	0.00
		1	-	-	237.10	237	271.28	16.49	-
40-5	317	2	-	-	188.00	188	193.04	17.20	-
		3	-	-	159.00	159	194.74	17.04	-
		1	-	-	186.30	186	244.08	43.67	-
40-6	262	2	-	-	149.00	149	218.70	33.33	-
		3	-	-	140.00	140	203.26	16.16	-
		1	-	-	209.00	209	245.33	12.06	3.47
40-7	301	2	-	-	162.10	162	181.04	30.22	-
		3	-	-	144.50	144	183.63	29.47	-
		1	-	-	239.00	239	315.87	151.29	-
40-8	310	2	-	-	187.20	187	193.33	67.21	-
		3	-	-	168.40	168	203.31	77.20	-
		1	223	3219.67	226.00	226	276.86	27.57	1.35
40-9	311	2	-	-	183.00	183	196.67	52.54	-
		3	-	-	161.70	161	199.80	96.22	-
		1	-	-	215.00	215	250.70	9.61	-
40-10	285	2	-	-	177.00	177	193.40	40.26	-
		3	-	-	166.50	166	187.45	72.42	-
		1	-	-	221.00	221	269.02	9.05	-
40-11	289	2	175	3197.27	175.00	175	209.90	10.99	0.00
		3	159	1267.23	159.00	159	189.15	43.41	0.00
		1	-	-	210.00	210	263.22	7.58	-
40-12	280	2	-	-	168.20	168	227.51	40.16	-
		3	147	1715.02	147.00	147	202.78	30.44	0.00
		1	-	-	227.00	227	257.42	13.74	-
40-13	305	2	-	-	179.00	179	196.44	26.37	-
		3	-	-	163.40	161	190.53	53.23	-
		1	-	-	233.00	233	273.81	16.28	-
40-14	313	2	-	-	177.20	177	213.43	20.45	-
		3	-	-	159.00	159	191.07	25.98	-
		1	-	-	229.00	229	237.05	12.60	-
40-15	304	2	-	-	174.00	174	188.83	14.53	-
		3	-	-	161.90	161	206.78	17.41	-
		1	-	-	217.00	217	274.40	65.08	-
40-16	294	2	-	-	171.00	171	223.41	61.63	-
		3	149	1075.51	149.00	149	178.80	22.39	0.00
		1	-	-	230.00	230	291.63	13.30	-
40-17	322	2	-	-	180.00	180	171.12	16.49	-
		3	-	-	162.00	162	182.56	52.86	-

Continued on next page

Table A.8 – continued from previous page

			BP		GA				
Instance	TSP	α	z	t	z_{avg}	z_{best}	t_{avg}	τ_{avg}	%ΔBP
		1	-	-	235.00	235	262.58	9.78	-
40-18	313	2	-	-	185.00	185	172.41	16.55	-
		3	-	-	166.00	166	173.93	17.51	-
		1	-	-	196.00	196	268.35	8.96	-
40-19	269	2	-	-	158.00	158	183.43	23.25	-
		3	-	-	140.90	140	186.91	29.59	-
		1	-	-	219.00	219	275.87	16.94	-
40-20	282	2	-	-	180.00	180	191.31	19.04	-
		3	160	790.17	160.00	160	182.15	39.84	0.00
		1	-	-	227.00	227	294.61	9.14	-
40-21	301	2	169	1648.75	169.00	169	187.99	28.83	0.00
		3	154	2199.45	154.40	154	189.82	74.55	0.26
		1	-	-	247.00	247	262.20	21.92	-
40-22	308	2	-	-	201.10	200	191.63	54.42	-
		3	-	-	178.20	178	192.60	48.53	-
		1	-	-	245.00	245	271.02	8.24	-
40-23	324	2	-	-	193.00	193	209.67	48.75	-
		3	-	-	173.00	173	188.67	31.87	-
		1	-	-	206.00	206	289.84	10.09	-
40-24	274	2	-	-	171.00	171	174.83	8.70	-
		3	-	-	158.60	158	194.19	69.39	-
		1	-	-	233.00	233	314.28	57.07	-
40-25	309	2	-	-	180.20	180	189.88	49.14	-
		3	-	-	153.70	153	196.51	97.59	-
Average				2044.39			202.21	29.31	0.47

Bibliography

[1] James Vincent and Chaim Gartenberg. *Here's Amazon's new transforming Prime Air delivery drone*. https://www.theverge.com/2019/6/5/18654044/amazon-prime-air-delivery-drone-new-design-safety-transforming-flight-video. June 2019.

[2] A. Colorni, Marco Dorigo, and Vittorio Maniezzo. "Distributed Optimization by Ant Colonies". In: *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*. Ed. by F. J. Varela and P. Bourgine. Cambridge, MA: MIT Press, 1992, pp. 134–142.

[3] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. DOI: 10.1017/CBO9780511804441.

[4] William Cook. *Concorde*. Version 03.12.19. Dec. 19, 2003. URL: https://www.math.uwaterloo.ca/tsp/concorde/index.html.

[5] Okan Dukkanci, Bahar Y. Kara, and Tolga Bekta0.0296s. "Minimizing energy and cost in range-limited drone deliveries with speed optimization". In: *Transportation Research Part C: Emerging Technologies* 125 (2021), p. 102985. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2021.102985. URL: https://www.sciencedirect.com/science/article/pii/S0968090X21000206.

[6] Chase C. Murray and Amanda G. Chu. "The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery". In: *Transportation Research Part C: Emerging Technologies* 54 (2015), pp. 86–109. DOI: https://doi.org/10.1016/j.trc.2015.03.005.

[7] Ritwik Raj and Chase Murray. "The multiple flying sidekicks traveling salesman problem with variable drone speeds". In: *Transportation Research Part C: Emerging Technologies* 120 (2020), p. 102813. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2020.102813. URL: https://www.sciencedirect.com/science/article/pii/S0968090X2030718X.

[8] Alex S Fraser. "Simulation of genetic systems by automatic digital computers ii. effects of linkage on rates of advance under selection". In: *Australian Journal of Biological Sciences* 10.4 (1957), pp. 492–500.

[9] Minh Ha et al. "A hybrid genetic algorithm for the traveling salesman problem with drone". In: *Journal of Heuristics* 26 (Apr. 2020). DOI: 10.1007/s10732-019-09431-y.

[10] David L. Applegate et al. "Certification of an optimal TSP tour through 85,900 cities". In: *Operations Research Letters* 37.1 (2009), pp. 11–15. ISSN: 0167-6377. DOI: https://doi.org/10.1016/j.orl.2008.09.006. URL: https://www.sciencedirect.com/science/article/pii/S0167637708001132.

[11] Zhilong Liu, Raja Sengupta, and Alex Kurzhanskiy. "A power consumption model for multi-rotor small unmanned aircraft systems". In: (2017), pp. 310–315. DOI: 10.1109/ICUAS.2017.7991310.

[12] Ibrahim Osman and Gilbert Laporte. "Metaheuristics: A Bibliography". In: *Annals of Operational Research* 63 (Oct. 1996), pp. 513–628. DOI: 10.1007/BF02125421.

[13] Chase C. Murray and Ritwik Raj. "The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones". In: *Transportation Research Part C: Emerging Technologies* 110 (2020), pp. 368–398. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2019.11.003. URL: https://www.sciencedirect.com/science/article/pii/S0968090X19302505.

[14] Alan H. Guth. "Time Since the Beginning". In: (2003). DOI: 10.48550/ARXIV.ASTRO-PH/0301199. URL: https://arxiv.org/abs/astro-ph/0301199.

[15] Steven G. Johnson. *NLopt*. Version 2.7.1. Dec. 3, 2021. URL: http://github.com/stevengj/nlopt.

[16] Ritwik Raj et al. "A branch-and-price approach for the parallel drone scheduling vehicle routing problem". In: *SSRN Electronic Journal* (Jan. 2021). DOI: 10.2139/ssrn.3879710.

[17] Stefan Poikonen and Bruce Golden. "Multi-visit drone routing problem". In: *Computers and Operations Research* 113 (2020), p. 104802. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2019.104802. URL: https://www.sciencedirect.com/science/article/pii/S0305054819302448.

[18] Roberto Roberti and Mario Ruthmair. "Exact Methods for the Traveling Salesman Problem with Drone". In: *Transportation Science* 55(2).10 (2021), pp. 315–335. DOI: https://doi.org/10.1287/trsc.2020.1017.

[19] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. "Optimization by Simulated Annealing". In: *Science* 220.4598 (1983), pp. 671–680. DOI: 10.1126/science.220.4598.671. eprint: https://www.science.org/doi/pdf/10.1126/science.220.4598.671. URL: https://www.science.org/doi/abs/10.1126/science.220.4598.671.

[20] Joshuah K. Stolaroff et al. "Energy use and life cycle greenhouse gas emissions of drones for commercial package delivery". In: *Nature Communications* 9.1 (Feb. 2018). DOI: 10.1038/s41467-017-02411-5.

[21] Fred Glover. "Future paths for integer programming and links to artificial intelligence". In: *Computers and Operations Research* 13.5 (1986). Applications of Integer Programming, pp. 533–549. ISSN: 0305-0548. DOI: https://doi.org/10.1016/0305-0548(86)90048-1. URL: https://www.sciencedirect.com/science/article/pii/0305054886900481.

[22] Felix Tamke and Udo Buscher. "The vehicle routing problem with drones and drone speed selection". In: (Nov. 2021). DOI: https://doi.org/10.48550/arxiv.2111.13050.

[23] David L. Applegate et al. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006. URL: http://www.jstor.org/stable/j.ctt7s8xg.

[24] Rachel Premack. *UPS just beat out Amazon, FedEx, and Uber to make America's first revenue-generating drone delivery.* https://www.businessinsider.com/ups-first-revenue-generating-drone-delivery-with-matternet-2019-3?r=US&IR=T. Mar. 2019.

[25] Thibaut Vidal et al. "A Hybrid Genetic Algorithm for Multidepot and Periodic Vehicle Routing Problems". In: *Operations Research* 60.3 (2012), pp. 611–624. DOI: 10.1287/opre.1120.1048. URL: https://doi.org/10.1287/opre.1120.1048.

[26] N. Mladenović and P. Hansen. "Variable neighborhood search". In: *Computers and Operations Research* 24.11 (1997), pp. 1097–1100. ISSN: 0305-0548. DOI: https://doi.org/10.1016/S0305-0548(97)00031-2. URL: https://www.sciencedirect.com/science/article/pii/S0305054897000312.

[27] Yong Zeng, Jie Xu, and Rui Zhang. "Energy Minimization for Wireless Communication With Rotary-Wing UAV". In: *IEEE Transactions on Wireless Communications* 18.4 (2019), pp. 2329–2345. DOI: 10.1109/TWC.2019.2902559.