Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Biológicas

Exact and heuristic approaches for Traveling Salesman Problems With Drones

Júlia Cária de Freitas

Ouro Preto, Brazil

October 2021

Júlia Cária de Freitas

Exact and heuristic approaches for Traveling Salesman Problems With Drones

Dissertation preseted to the *Universidade Federal de Ouro Preto* in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science

Advisor:
Prof. Ph.D. Puca Huachi Vaz Penna

Co-advisor:
Prof. Ph.D. Túlio Ângelo Machado Toffolo

Ouro Preto, Brazil

October 2021

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

F866e Freitas, Júlia Cária de.
Exact and heuristic approaches for traveling salesman problems with drones. [manuscrito] / Júlia Cária de Freitas. - 2021.
90 f.: il.: color., gráf., tab..

Orientador: Prof. Dr. Puca Huachi Vaz Penna.
Coorientador: Prof. Dr. Túlio Ângelo Machado Toffolo.
Dissertação (Mestrado Acadêmico). Universidade Federal de Ouro Preto. Departamento de Computação. Programa de Pós-Graduação em Ciência da Computação.
Área de Concentração: Ciência da Computação.

1. Programação linear. 2. Heurística. 3. Otimização combinatória. I. Penna, Puca Huachi Vaz . II. Toffolo, Túlio Ângelo Machado. III. Universidade Federal de Ouro Preto. IV. Título.

CDU 004.5

Bibliotecário(a) Responsável: Celina Brasil Luiz - CRB6-1589

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E BIOLOGICAS
DEPARTAMENTO DE COMPUTACAO
PROGRAMA DE POS-GRADUACAO EM CIENCIA DA COMPUTACAO

FOLHA DE APROVAÇÃO

Júlia Cária de Freitas

Exact and heuristic approaches for Traveling Salesman Problems With Drones

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Mestre em Ciência da Computação

Aprovada em 05 de outubro de 2021

Membros da banca

Prof. Dr. Puca Huachi Vaz Penna - Orientador - Universidade Federal de Ouro Preto
Prof. Dr. Túlio Ângelo Machado Toffolo - Co-orientador - Universidade Federal de Ouro Preto
Prof. Dr. Marcone Jamilson Freitas Souza - Universidade Federal de Ouro Preto
Prof. Dr. Anand Subramanian - Universidade Federal da Paraíba

Prof. Dr. Puca Huachi Vaz Penna, orientador do trabalho, aprovou a versão final e autorizou seu depósito no Repositório Institucional da UFOP em 04/11/2021

Documento assinado eletronicamente por **Puca Huachi Vaz Penna**, **COORDENADOR(A) DE CURSO DE PÓS-GRADUACÃO EM CIÊNCIA DA COMPUTAÇÃO**, em 05/11/2021, às 15:02, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0241040** e o código CRC **18571042**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35400-000
Telefone: - www.ufop.br

Abstract

The technological advances concerning drones have encouraged the market to consider drone applications in different areas including last mile delivery. However, limitations due to battery capacity, maximum weight, and legal regulations restrict the effective operational range of drones in many practical applications. To overcome the battery issue, hybrid operations involving one or more drones launching from a larger vehicle have emerged, in which the larger vehicle operates as a mobile depot and a recharging platform. In this dissertation, we describe a routing model that leverage the drone and truck working as a synchronized unit. The Flying Sidekick Traveling Salesman Problem (FSTSP) considers a delivery system composed by a truck and a drone. The drone launches from the truck with a single package to deliver to a customer. Each drone must return to the truck to recharge batteries, pick up another package, and launch again to a new customer location. This work proposes two novel Mixed Integer Programming (MIP) formulations and a heuristic approach to address the problem. The proposed MIP formulations yields better linear relaxation bounds than previously proposed formulations for all instances, and was capable of optimally solving several unsolved instances from the literature. We developed a hybrid heuristic based on the General Variable Neighborhood Search metaheuristic to tackle a generalization of the FSTSP called Multiple Traveling Salesman Problem with Drones, in which multiple trucks and drones are considered as part of the delivery system. The heuristic obtained high-quality solutions for large-size instances. The efficiency of the algorithm was evaluated on 410 benchmark instances from the literature, and over 80% of the best known solutions were improved.

Keywords: Unmanned Aerial Vehicle, Drones, Flying Sidekick Traveling Salesman Problem, Multiple Traveling Salesman Problem with Drones, Variable Neighborhood Search.

Resumo

Os avanços tecnológicos em relação a drones têm incentivado o mercado a desenvolver novas aplicações em diferentes áreas, incluindo a entrega de última milha. No entanto, as limitações devido à capacidade da bateria, peso máximo e regulamentos restringem a operação dos drones em muitas aplicações práticas. Com o intuito de superar a limitação da bateria em operações de entrega, surgiram operações híbridas envolvendo um ou mais drones que são lançados de um veículo maior. Este veículo funciona como um depósito móvel e uma plataforma de recarga. Nesta dissertação, descrevemos um modelo de roteamento que aproveita o trabalho sincronizado entre drone e o caminhão. O *Flying Sidekick Traveling Salesman Problem* (FSTSP) considera um sistema de entrega composto por um caminhão e um drone. O drone é lançado do caminhão com um único pacote para realizar a entrega ao cliente. O drone deve retornar ao caminhão para recarregar as baterias, pegar outro pacote e ser lançado novamente de um novo local. Este trabalho propõe duas formulações de Programação Inteira Mista (MIP) e uma abordagem heurística para tratar o problema. Foram propostas duas formulações que produzem melhores limites de relaxação linear do que as formulações já existentes na literatura para todas as instâncias. Além disso, desenvolvemos uma heurística híbrida com base na meta-heurística *General Variable Neighbourhood Search* para lidar com a variante FSTSP chamada *Multiple Traveling Salesman Problem with Drones*, na qual vários caminhões e drones são considerados como parte do sistema de entrega. A heurística desenvolvida obteve soluções de alta qualidade para instâncias grandes. A eficiência do algoritmo foi avaliada em 410 instâncias de benchmark da literatura, e mais de 80% das soluções conhecidas foram melhoradas.

Palavras-chave: Veículos Aéreos Não Tripulados, Drones, Problema do Caixeiro Viajante, Problema do Caixeiro Viajante Múltiplo com Drones, Variable Neighborhood Search.

Acknowledgements

I would like to express my deepest thanks to my parents, Ana Elisa and Gilson, for believing in me. They have always encouraged me and given me the strength to continue towards my goals. João Pedro, the love of my life, I am grateful for his kindness, always listening to my complaints, and celebrating my achievements.

Throughout the writing of this dissertation, I received great assistance. I would like to acknowledge my advisors, Prof. Ph.D. Puca H. V. Penna e Prof. Ph.D Túlio A. M. Toffolo, for their support and guidance over these years.

I would like to thank Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), and Universidade Federal de Ouro Preto (UFOP) for funding this project. I thank the Programa de Pós-graduação em Ciência da Computação (PPGCC/UFOP) for enabling this research by providing access to their research laboratories and computing infrastructure.

Agradecimentos

Gostaria de expressar os meus maiores agradecimentos aos meus pais, Ana Elisa e Gilson por acreditarem em mim. Eles que sempre me apoiaram e me deram força para continuar seguindo os meus objetivos. Ao João Pedro, grande amor da minha vida, agradeço o carinho, por aguentar minhas reclamações e por comemorar minhas conquistas.

Durante a elaboração desta dissertação, recebi grande ajuda. Gostaria de agradecer aos meus orientadores, Prof. Dr. Puca H. V. Penna e Prof. Dr. Túlio A. M. Toffolo, pelo apoio e orientação ao longo desses anos.

Gostaria de agradecer a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e Universidade Federal de Ouro Preto (UFOP) pelo financiamento deste projeto. Agradeço ao Programa de Pós-Graduação em Ciência da Computação (PPGCC/UFOP) por viabilizar esta pesquisa, fornecendo acesso aos laboratórios de pesquisa e às suas infraestruturas computacionais.

List of Figures

List of Tables

List of Algorithms

Glossary

BKS Best Known Solution.

CETSP Close-Enough Traveling Salesman Problem.

CFRS Cluster First Route Second.

CVRP Capacitated Vehicle Routing Problem.

DDP Drone Delivery Problem.

DSS Decision Support System.

DTSP Drone Traveling Salesman Problem.

FCURP Fuel Constrained UAV Routing Problem.

FSTSP Flying Sidekick Traveling Salesman Problem.

GRASP Greedy Randomized Adaptive Search Procedure.

GVNS General Variable Neighborhood Search.

IoT Internet of Things.

LIMA less is more approach.

MC-DDP Minimum Cost Drone Delivery Problem.

MDVRPTW Multiple-Depot VRP With Time Window.

mFSTSP Multiple Flying Sidekicks Traveling Salesman Problem.

MILP Mixed Integer Linear Programming.

MIP Mixed Integer Programing.

MT-DDP Minimum-Time Drone Delivery Problem.

MTDRP Multi-Trip Drone Routing Problem.

mTSP Multiple Traveling Salesman Problem.

mTSPD Multiple Traveling Salesman Problem with Drones.

MVDRP Multi-Visit Drone Routing Problem.

NN Nearest Neighborhood.

PDSTSP Parallel Drone Scheduling Traveling Salesman Problem.

PDSTSP Parallel Drone Scheduling Traveling Salesman Problem.

PDSTSP^{+DP} Parallel Drone Scheduling Traveling Salesman Problem Drop-Pickup.

PMS Parallel Machine Scheduling Problem.

RFCS Route First Cluster Second.

RHTA Receding Horizon Task Assignment.

RVND Randomized Variable Neighborhood Descent.

SDD Same Day Delivery.

SDDPHF Same Day Delivery Routing Problem with Heterogeneous Fleets.

TBRD Truck-Based Robot Delivery.

TSP Traveling Salesman Problem.

TSP-D Traveling Salesman Problem with Drones.

TSPLIB Traveling Salesman Problem Library.

UAV Unmanned Aerial Vehicle.

VND Variable Neighborhood Descent.

VNDS-SIH Variable Neighborhood Decomposition Search with Savings Insertion Heuristic.

VNS Variable Neighborhood Search.

VRP Vehicle Routing Problem.

VRPD Vehicle Routing Problem with Drones.

VRPTW Vehicle Routing Problem with Time Windows.

VRPTW Vehicle Routing Problem with Time Windows with Drones.

VRPWSC Vehicle Routing Problem With Synchronization Constraints.

WSR Wilcoxon Signed-Rank.

Contents

1 Introduction

Over the past few years, vehicles, commonly known as drones, have been studied to reduce logistics costs. From quick deliveries at rush hour to scanning an unreachable military base, drones are proving to be highly beneficial in places where a man cannot reach or cannot perform activities in a timely and efficient manner.

Some of the top uses drones offer to industries are increase work efficiency and productivity, decrease workload and production costs, improve accuracy, refine service, and resolve security issues on a vast scale (Battsengel et al., 2020).

The COVID-19 pandemic has been one of the most significant health crises in modern history. As consumers pivoted and adopted new habits, business leaders wondered whether these changes, which accelerated trends already in motion, would be fleeting or permanent. Online shopping for personal items was already a trend in 2017, when Amazon Prime reported shipping more than five billion items worldwide (Amazon, 2018). Since the beginning of the pandemic, consumers have switched to be even more digital and eco-friendly. People started buying groceries and not only meals from delivery apps. Daily visits to a store became a message to deliver an item.

Supply chains are under pressure to become far more resilient and agile under an operating model that is faster and more flexible and provides a dramatically shorter lead time from order to delivery. Therefore, companies need to continually create ingenious ways to shorten delivery times and satisfy customers' needs. Further, the process must be cost and resource-efficient to boost growth and support investment (Carr et al., 2021).

One way of improving the delivery experience is by using drones. Unmanned Aerial Vehicle (UAV) can both reduce delivery time and cheapen operations and management costs. Raffaello D'Andrea, who cofounded Kiva Systems (the warehouse robots used by Amazon), delivers a 2-kilogram package over 9.7 kilometers with the cost of 10 cents using a drone, while the traditional delivery truck costs between $2 to $8 per package to Amazon.

This work aims to tackle the last mile delivery problems to make it fast and efficient by combining truck and drone. To sum up and quickly give an outline of the problems with whom this dissertation is concerned, it all starts with Murray and Chu and their Flying Sidekick Traveling Salesman Problem (FSTSP). There the FSTSP is described as a variant of the classical Traveling Salesman Problem (TSP), where a drone launches from a truck, proceeds to deliver goods to a customer, and finally joins back to the truck in a third location. While the drone flies in an operation, the truck can travel to other customers.

In addition of the FSTSP, this work address the Multiple Traveling Salesman Problem with Drones (mTSPD). The mTSPD instead of considering a single truck and a single drone, employs a fleet of $|M|$ trucks and $|K|$ drones per truck to deliver goods to the customers. The combination of drone trips and truck deliveries makes it possible to cover all customers reducing the overall delivery time. The objective of both FSTSP and mTSPD is to minimize the total duration of the tour built with these vehicles.

The problems addressed in this dissertation are $\mathcal{NP}$-Hard and lie in the subject of distribution management. It is faced each day by thousands of companies and organizations engaged in delivering and collecting goods or people. Because of the problem nature, optimal solutions for a large set of instances cannot be obtained in deterministic polynomial time if $\mathcal{P} \neq \mathcal{NP}$. Therefore, this dissertation provides an optimal solution to instances up to 10 customers and a heuristic to study larger instances and how multiple trucks and multiple drones working in tandem can improve the delivery process.

1.1 Objectives

1.1.1 General Objectives

The primary objective of this work is the development of computational systems that allow greater efficiency in the decision-making of the delivery planning. For this purpose, we propose two mathematical formulations and an heuristic to tackle Traveling Salesman Problem with Drones (TSP-D).

1.1.2 Specific Objectives

The specific objectives to be achieved are as follows:

- Evaluate several models of optimization proposed in the literature, bringing them together to compare the features of each one.
- Introduce two mathematical formulations capable of finding optimal solutions to literature instance with better lower bound.
- Apply the General Variable Neighborhood Search (GVNS) metaheuristic to tackle the problem.
- Evaluate the heuristic proposed by testing it on available test problems and compare its results with literature methods.

1.2 Contributions

The contribution of this dissertation consists in developing computational methods to tackle TSP-D. For this purpose, we define the following contribution regarding the problems:

- We presented several models of optimization proposed in the literature, bringing them together to compare the features of each one.
- We implemented an exact formulation with an exponential number of variables to solve the FSTSP
- We implemented an exact compact formulation to solve the FSTSP
- We developed a heuristic employing the General Variable Neighborhood Search (GVNS) concept to tackle the mTSPD.
- We evaluated the proposed method by testing it on publicly available test problems and compare its results with literature methods.

The research presented in this dissertation has resulted in the following papers:

- J. C. Freitas, P. H. V. Penna, and H. Gambini. Truck and drone collaboratively delivery for green logistics in smart city. *L Brazilian Symposium on Operational Research*, 2018
- J. C. Freitas and P. H. V. Penna. A variable neighborhood search for flying sidekick traveling salesman problem. *International Transactions in Operational Research*, 27 (1):267–290, 2020. doi: 10.1111/itor.12671
- J. C. Freitas, P. H. V Penna, and T. A. M. Toffolo. Exact and heuristic approaches to drone delivery problems, 2021. **Submitted to a high impact-factor journal.**

1.3 Document organization

The remainder of the dissertation is organized as follows: Chapter 2 address an analysis of previous works on drones and drone deliveries. The literature is in its early stages, but it is growing. Chapter 3 formally describes the problem. Chapter 4 introduces two formulations for the FSTSP. Chapter 5 discusses the methodology used to approach the mFSTSP: Section 5.1 presents the data structures employed to represent a solution. Section 5.2 examines how to built an initial solution, followed by the main method GVNS in Section 5.3. The following sections discuss the implemented neighborhoods (Section 5.3.1), and finally how the solution is evaluated (Section 5.4). In Chapter 6, the reader

first can refer to the formulations results in Section 6.1. Then, the heuristic experimental results are presented. Section 6.2.1 covers the instance settings, Section 6.2.2 shows an instance analysis and, finally, Section 6.2.3 compares the heuristic results to the literature. Chapter 7 presents the conclusions and future work.

2 Literature Review

The literature review includes two sections: Section 2.1 concerns parcel delivery in Operations Research, and discuss works that truck and drone operate as synchronized units (Section 2.1.1), truck and drone performing independent tasks (Section 2.1.2), and drones working as a single unit (Section 2.1.3). Section 2.2 presents drone applications in different sectors such as surveillance and health care.

2.1 Delivery Applications

The Vehicle Routing Problem (VRP) definition states that M vehicles initially located at a depot deliver discrete quantities of goods to N customers. The objective is to minimize the overall transportation cost of serving the N customers by the M vehicles. The classical VRP solution is a set of routes that begin and end in the depot, satisfying the constraint that all the customers are visited only once. The transportation cost can be improved by reducing the total traveled distance and the number of required vehicles.

The majority of real-world problems are often much more complex than the classical VRP, the constraints are often augmented. For example, the vehicles may present capacity (Capacitated Vehicle Routing Problem (CVRP)) or time interval, in which each customer has to be visited (Vehicle Routing Problem with Time Windows (VRPTW)). In the last fifty years, many real-world problems have required extended formulation that resulted in the multiple depots VRP, periodic VRP, split delivery VRP, stochastic VRP, VRP with backhauls, VRP with pickup and delivering, and many others. We can find extensive research of the VRP in Toth and Vigo (2014).

Another VRP variant is the Multiple Traveling Salesman Problem (mTSP), which defines a set of routes for M salespeople who start from and return to a depot. The reader is referred to the work by Betkas Bektas (2006), which sums up the mTSP description and reports distinct methods for its solution—the author details applications context for the mTSP and its relationship with other problems. Further, exact and heuristic approaches to the mTSP are discussed to clarify their positive aspects and point their inefficient side.

An extensive body of literature exists on the mTSP and VRP. Neither of the problems is applied as-is to the examined problem. The mTSPD is a generalization of the mTSP, and a relaxation of VRP, with the capacity constraints removed. Therefore, all the formulations and solution approaches proposed for the VRP are also valid and applicable to the mTSPD. A solution can be achieved by assigning sufficiently large capacities to the vehicles and removing the usage of drones. The mTSPD can also be considered a

reduction to the well-known TSP when there are only a single truck and no drones, for which an extensive amount of research exists (e.g., Applegate et al. (2007); Bellmore and Nemhauser (1968); Held and Karp (1970); Laporte (1992)).

In Drexl (2012), a thorough review of some possibilities for these constraints undergoes an intensive study, as they represent a topic of growing interest in the field. In particular, the paper involves a threefold objective of classifying the different types of synchronization, discussing exact and heuristic solutions to the problem and identifying promising algorithmic solution approaches. A variant of the VRP associated with this study is Vehicle Routing Problem With Synchronization Constraints (VRPWSC). The primary influence of this VRPWSC variant in this dissertation is the constraints applied to assure drones launch and return to the truck/depot synchronization. According to Drexl, the type of synchronization in the mTSPD would be classified as "operation synchronization" since the drone launch would be the operation to facilitate the drone delivery at a different location, and the return to the truck would facilitate the use of the drone for further deliveries.

Hereafter, we discuss three different scenarios employing drones. Subsection 2.1.1 presents works that truck and drone delivery collaboratively and synchronized, i.e., at a given point of the delivery, a vehicle has to wait for the other to proceed. In Subsection 2.1.2, the vehicles perform independent tasks, i.e., the vehicles do not depend on the other to continue the delivery route. Finally, in 2.1.3, only drones are considered to perform deliveries.

2.1.1 Truck and Drone working as synchronized units

The first work we are aware of truck and drone performing synchronized delivery is Murray and Chu (2015). The Flying Sidekick Traveling Salesman Problem (FSTSP) describes a single truck – single drone scenario where the total time spent on the tour must be minimized. The drone launches the truck at a customer, then delivers a package, and finally travels back to the truck at another customer location. Meanwhile, the truck may visit one or more customers. The vehicles must be reunited at the return location before the drone runs out of battery. The authors proposed a MILP and a heuristic to tackle the FSTSP.

The thesis of Ponza (2016) is based on FSTSP formulation by Murray and Chu within slight changes in the original model regarding waiting and setup times. The author proposed a Simulated Annealing algorithm to find optimal or near-optimal solutions to instances with 10 to 200 seconds within 500 seconds. Ponza (2016) optimally solved all instances with up to 10 customers from the benchmark introduced by Murray and Chu (2015).

In Ha et al. (2018) they denominated the problem as Min-Cost TSP whose objective is to minimize the total operational cost, considering truck and drone transportation costs and waiting times. To minimize the waiting time, they employed a coefficient that multiplies the waiting time of the vehicle. A waiting time penalty is an interesting approach, especially for the truck, since waiting for the drone increases the total delivery time. They provided an exact approach and two heuristics – Greedy Randomized Adaptive Search Procedure (GRASP) and TSP-LS (Local Search). The approaches were tested in two instance sets. The first set introduced by Murray and Chu with up to 10 customers, where GRASP outperformed 95% of Murray and Chu results. Ha et al. (2018) introduced a new set of instances varying the number of customers $n = \{10, 50, 100\}$. The proposed GRASP performs better than the TSP-LS on all instances regarding solution quality. Concerning run time, GRASP runs much faster with an average run time of 0.1 seconds. They also compared their formulation and heuristics with both optimal TSP and FSTSP solution. The results showed that out of 8 instances, FSTSP outperformed their TSP-D 6 times with an average gap of 30%.

The Vehicle Routing Problem with Drones (VRPD) discussed in Schermer et al. (2018) considered a set of customers location and a fleet of homogeneous vehicles, each carrying the same amount of identical drones. In the VRPD, the objective is to minimize the time required to serve all customers using trucks and drones such that, by the end of the mission, all vehicles must be at the depot. A Variable Neighborhood Search (VNS) heuristic was developed and tested in the instances proposed by the authors based on TSPLIB. After a series of computational experiments, it is possible to notice that the relative improvement in the objective value becomes increasingly smaller after a particular number of drones.

A different formulation to the FSTSP is the Traveling Salesman Problem with Drones (TSP-D) introduced by Agatz et al. (2018). The major difference is that the authors assumed that the truck and the drone travel on the same road network, which has multiple advantages and drawbacks. The main advantages are that they develop heuristics with approximation guarantees and provide a bound on the maximum possible gains of the truck and drone coupling over a traditional truck-only system. With the ability to get a lower bound on the optimal solution, there is the double edge of tying the drone to the road network without taking full advantage of its speed and ability to take shortcuts via Euclidean distances. Moreover, another difference between the FSTSP and the TSP-D is that the latter allows the drone's launch and return to be the same point, whereas the FSTSP forced to be different customers.

In Marinelli et al. (2018) a drone can launch while en route, i.e., the truck may not be in a customer node at the moment of the drone take-off but traveling from one customer to another. The en-route operational is beneficial as it may reduce the drone

traveling time of a given trip. Consequently, drone coverage increases within a reduction in total traveling costs. Unfortunately, it is impossible to numerically argue the advantage of using en route operations as the work did not compare directly with instances from the literature.

In the work by Dayarian et al. (2020), they considered a dynamic scenario where new orders arrive all day long. While trucks perform deliveries, drones are dispatched from the depot to transfer a set of orders to a truck at a pre-determined meeting location and time. While receiving the orders, the truck route is re-optimized. Therefore the service time guaranteed of the orders on-board the truck is satisfied. The goal is to deliver as many orders as possible, respecting the guaranteed service time at a minimum cost.

Wang et al. (2017) introduced the VRPD as a generalization of the VRP. In the VRPD a fleet of vehicles, each vehicle equipped with a given number of drones, is tasked with delivering parcels to customers. The drones may launch from the depot or the truck at any customer location. Moreover, the drone must join the truck at a different customer location it was launched (or at the depot, concluding its tour). When a drone launches, it can visit precisely one customer before returning to the vehicle (or to the depot). The objective is to minimize mission time, i.e., the time required to visit all customers guaranteeing that the fleet has returned to the depot at the end of the mission. Wang et al. introduced several upper bounds on the amount of time that can be saved by employing drones compared to the classic VRP. The upper bounds are obtained by investigating the structure of optimal solutions and examining the relative velocity of drones compared to the vehicles as the number of drones per vehicle.

Poikonen et al. (2017) inspired by the work by Wang et al. studied the impact when integrating limited battery life, different distance metrics, and operational expenditures of deploying drones and vehicles in the objective function. They presented different perspectives of the problem, i.e., different distance metric for each vehicle has a significant impact on the solution cost, since the drone may be faster than the truck. They questioned which is the better resource to invest: more drones per truck or faster drones. Different points of view arise from this comparison. On the one hand, a substantial number of drones can serve more customers in parallel, while a higher drone speed has the advantage of serving more customers in sequence. On the other hand, if drone range or capacity is severely limited, a larger number of drones may be preferred. It is a complicated question that requires further study and depends on the necessities of each problem. Furthermore, Poikonen et al. proposed extending the VRPD model similar to the Close-Enough Traveling Salesman Problem (CETSP). The authors suggested the possibility of launching and recovering the drones from arbitrary locations instead of being restricted to customer locations.

A different formulation for the VRPD is the one introduced by Daknama and Kraus

(2017) where the drone can be launched and collected by a different truck. Therefore, each truck may carry many drones, including all drones at a unique truck. Although increasing the number of drones generally improves the solution, a decrease occurs in the marginal benefit of each additional drone. The employment of drones reduced by 10% the time required to deliver all packages.

Pugliese and Guerriero (2017) formulated a variant of the VRPTW where each vehicle is equipped with drones called Vehicle Routing Problem with Time Windows with Drones (VRPTW). In this formulation, a limited number of drones is preassigned to each truck, and the drones must return to the same truck after each delivery. The drone has a maximum waiting time to be retrieved by the truck. Also, the authors considered time window constraints. The objective function is to minimize the total cost deriving from the employment of trucks and drones. The work shows that drones are not a viable alternative for the delivery process compared to traditional trucks. However, aerial vehicles working as a complementary feature is a great strategy.

While in Dayarian et al. trucks performed the deliveries and drones worked as a mobile depot to the truck, Poikonen and Golden (2020) described the opposite behavior. The problem considered is the Multi-Visit Drone Routing Problem (MVDRP) where a single truck acts as a mobile depot and recharging station for a single drone. The drone may launch from the truck with one or more packages, deliver them to their respective locations, and return to the truck for recharging and additional pickup packages. Another problem approached by Poikonen and Golden was the k-MVDRP, in which a truck can carry d homogeneous drones at a time. The truck must travel immediately towards the retrieve location and does not stop in between nodes. Since a mechanical or other problem can occur to the drone, the truck must be available for any surprise.

Dell'Amico et al. (2020) approached the Parallel Drone Scheduling Traveling Salesman Problem (PDSTSP) proposing a MILP model and several matheuristics. The authors experimented with the algorithms on the benchmark instances introduced by Saleu et al. (2018) and Murray and Chu (2015). The computational study validates that the proposed algorithms produce competitive results in both efficiency and effectiveness, mainly on small and medium-size instances.

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

Freitas and Penna (2020) introduces new instances based on the TSPLIB and compares the HGVNS (General Variable Neighborhood Search) result with instances found in the literature (Agatz et al., 2018; Ponza, 2016). In this work, we complement the

heuristics by using a list based on the Tabu Search to avoid cycling in the neighborhoods. Here, we also propose a MILP to solve the FSTSP in the instances proposed in (Murray and Chu, 2015; Ponza, 2016).

The Multiple Flying Sidekicks Traveling Salesman Problem (mFSTSP) introduced by Chase and Ritwik (2020) considers a delivery truck operating in coordination with a fleet of drones. A drone is launched from the truck, travels to deliver a single package, then returns to the truck to be loaded again. They employed a three-phased (I. initial truck assignments, II.create drone routes, III.combining phase I and II) heuristic solution to approach the problem. The heuristic result analysis revealed that drones with high-speed and long-range offer more significant benefits in larger geographic regions, where customers are distributed over a larger area.

Table 1 – Labels employed in the Tables 2, 3, and 4 to describe the articles' characteristics.

L	Indicates the location a drone launches from a truck.
R	Indicates the location a drone rendezvous to a truck.
E	Indicates whether or not the work considers drone endurance.
ST	Indicates whether or not the work considers drone setup time in the launch and/or rendezvous.
RN	Road Network.
DT	Minimize Delivery Time.
C	Minimize Cost.
CnE	Minimize Cost and Energy.
DD	Minimize Delivery Distance.
AvgDTP	Minimize Average Delivery Time per package.
AvgLT	Minimize Average Lead Time.

Table 2 presents an overview of all the formulations with drones and trucks working as synchronized units in the literature. The column 'Problem' highlights the one or two models proposed in the respective paper or the type they belong to if no name is provided. The third and fourth columns regard the number of vehicles of each type allowed by that formulation, while the 'L $\neq$ R' column specifies if the formulation allows a sortie to have launch and return in the same node. 'Drone RN $\neq$ Truck RN' column describes whether the vehicles travel in the same road network. Columns 'E' and 'ST' respectively concern if the formulation considers drone endurance and setup time. The column 'Objective' states the objective function of each formulation. Since all analyzed works are minimization problems, we omit the word 'minimize' in the acronyms DT, C, CnE, and DD described in Table 1. Finally, column 'Others' sums up characteristics considered by the author that we found interesting in being reported.

Table 2 – Overview of previous strategies employing drone and truck as synchronized units in the literature.

Truck and Drone as synchronized working units										
Author	Year	Problem	# Truck	# Drone	L ≠ R	Drone RN ≠ Truck RN	E	ST	Objective	Others
Murray and Chu	2015	FSTSP	1	1	✓	✓	✓	✓	DT	
Ponza	2016	FSTSP	1	1	✓	✓	✓	✓	DT	
Ha et al.	2016	min-cost TSP-D TSP-D	1	1	✓	✓	✓	✓	C DT	
Agatz et al.	2015	TSP-D	1	1		✓			DT	
Bouman et al.	2017	TSP-D	1	1		✓			DT	
Marinelli et al.	2018	TSP-D	1	1	✓	✓	✓	✓	DC	
Dayarian et al.	2017	VRPDR	m	d	✓	✓	✓		DC	◇ drone carries mult. parcels
Wang et al.	2017	VRPD	m	d	✓	✓	✓	✓	DT	
Poikonen et al.	2017	VRPD	m	d	✓	✓	✓	✓	DT	
Daknama and Kraus	2017	VRPDR	m	d	✓	✓	✓	✓	AvgDTP	
Poikonen and Golden	2018	MVDRP k-MVDRP	1 1	1 d	✓	✓	✓	✓	DT	◇ drone carries mult. parcels
Schermer et al.	2018	VRPD	m	d	✓	✓	✓	✓	DT	
Pugliese and Guerriero	2017	VDRPTW	m	d	✓	✓	✓		DC	◇ time-window ◇ max. drone wait

2.1.2 Truck and Drone performing independent tasks

The work by Boysen et al. (2018) introduced the Truck-Based Robot Delivery (TBRD) problem. The TBRD considers a set of robots that must deliver parcels to a set of customers. A single truck loaded with drones and parcels travels to a drop-off point, launching the set of drones to accomplish delivery. After delivery, the drone proceeds to a depot, where the truck can pick it up and relaunch. The problem also considers time windows to attend to all customer's locations. The authors proposed two sets of instances, a small one that can be optimally solved by the exact approach proposed and the multi-start local search heuristic. For the large dataset, the gap to the bound is nearly 90% for the heuristic procedure, and when applied to the small dataset, the gap to the best-found solution, of which 80% are optimal, is about 11%. The exact approach had a time limit of 1800 seconds, while the heuristic's average CPU time was 48 seconds.

Liu et al. (2018) described the Drone Traveling Salesman Problem (DTSP) where a truck works as a hub to recharge and reload the drone. While the drone is delivering orders, the truck is waiting for the drone to return. The objective of the DTSP is to find the solution that saves energy and time with the shorter truck route. Computational experiments showed that the usage of more drones could reduce costs up to 29.75% when compared to TSP, while only one drone was delivering the parcels, the result was worse than TSP.

Murray and Chu (2015) introduced the Parallel Drone Scheduling Traveling Salesman Problem (PDSTSP) a merge of two classical operational research problems. The TSP sequences the customers assigned to the truck. The other problem is of scheduling the remaining customers to the fleet of drones, which is equivalent to the Parallel Machine Scheduling Problem (PMS) with minimum makespan objective. In the PDSTSP drone and trucks serve customers in parallel. While the truck visits customers along a TSP route, the drone flies back and forth to the depot to be recharged and reloaded. The PDSTSP arises from practical constraints, such as the limited payload or the flying range of drones. A heuristic and a MILP formulation were provided to solve the PDSTSP, and test problems were generated with either 10 or 20 customers. The MILP was able to find the optimal solutions for all of the 10-customer problems. However, 24% of the 20-customer instances terminated at the time limit.

In Saleu et al. (2018) multiple drones are available to perform deliveries. They provided a MILP formulation for the PDSTSP and an iterative two-step heuristic. The heuristic first step is to partition the customers between the vehicle and the fleet of drones. The second step manages to solve a TSP for the vehicle and a PMS problem for drones. In the PDSTSP single-drone version Saleu et al. found the best-known solutions obtained by Murray and Chu (2015). When the number of drones is greater than one, the heuristic could not obtain the optimal solutions. However, the gap was only up to 0.18%.

The Parallel Drone Scheduling Traveling Salesman Problem Drop-Pickup (PDSTSP^{+DP}) by Ham (2018) described drone and truck performing independent tasks. The vehicles execute drops (delivery packages) and pickups (collect packages in customer's location). The drone has a limited battery and capacity. Further, it launches and returns directly to the depot for each trip. During a trip, the drone can perform multiples drops and pickups respecting its battery. Since the vehicles are not synchronized, the truck leaves the depot and returns after delivering all packages. Furthermore, the PDSTSP^{+DP} considers time-windows, multi-depots, and multi-visits, i.e., a customer can be visited more than once (for example, for a drop and pick up).

Ulmer and Thomas (2018) assumed a dynamic scenario, in which new requests arrive along the day, resulting in the Same Day Delivery Routing Problem with Heterogeneous Fleets (SDDPHF). In the SDDPHF, the provider decides whether a customer can be served on the same day and which vehicle (truck or drone) performs the delivery. When the Same Day Delivery (SDD) is available, either a drone is loaded, or a truck picks up the package in the depot to deliver it within the delivery deadline. The SDDPHF seeks to maximize the expected number of customers served on the same day. The fleet of vehicles is composed of m trucks and d drones. For each delivery, it is required a service and loading time for both truck and drone. Further, the drone needs a setup and recharging time. They proposed a parametric policy function approximation approach to tackling the problem. The parameterized policy is a threshold of travel distance from the depot, which splits the service into two areas. Customers in the threshold range are served by truck, and the ones located further are served by drone. The two areas were decided by considering that trucks may be suitable in areas close to the depot because of the high customer density, while drones may benefit more remote areas with widely dispersed customers. The extensive computational experiment concluded that geographical districting is highly effective in increasing the expected number of same-day deliveries, and a combination of drone and vehicle fleets may significantly reduce routing.

The work by Fikar et al. (2016) has a different approach. While the papers mentioned before focus on delivering products brought by consumers, Fikar, Gronalt, and Hirsch described a Decision Support System (DSS) disaster relief distribution scenario, in which potentially roads or bridges are not traversable without special equipment or permissions and road inundations resulting from floods or road blockades as a result of earthquakes or mudslides. The DSS developed provides decision-makers with a tool to test different strategies during professional training in a risk-free environment, enabling what-if studies to be performed and to analyze worst-case scenarios to foster decision-making skills and to gain a better understanding of disaster relief distribution.

Table 3 – Overview of previous strategies employing drone and truck as independent units in the literature.

Truck and Drone performing independent tasks									
Author	Year	Problem	# Truck	# Drone	L ≠ R	Drone RD ≠ Truck RD	E	ST	Objective
Boysen et al.	2018	TBRD	1	d	✓		✓		DT
Liu et al.	2018	DTSP	m	d			✓		DT
Ham	2018	PDSTSP^{+DP}	m	d			✓	✓	DT
Murray and Chu	2015	PDSTSP	1	1	✓	✓	✓	✓	DT
Fikar et al.	2016	DSS	m	d	✓				AvgLT

2.1.3 Single drone operations

Dorling et al. (2017) approach a drone problem from the energy consumption point of view. The authors elaborated an energy consumption model to balance the drone weight and the energy stored in its battery since these two factors limit the vehicle flight time. Therefore, such a model helps optimize deliveries to compare the energy consumed by alternative routes. They proposed two approaches considering a multi-trip vehicle routing problem for drone delivery. The one called Minimum-Time Drone Delivery Problem (MT-DDP) seeks to minimize the delivery time to deliver all packages. The other minimizes the cost of making deliveries and is referred to as the Minimum Cost Drone Delivery Problem (MC-DDP). Both problems consider battery weight, payload weight, and energy consumption to implement a MILP and a string-based Simulated Annealing algorithm for solving the DDPs. The DDPs allow a drone to be reused, i.e., the aerial vehicles can return to the depot and be relaunched. By reusing drones, the cost of $112.7 per 60 minutes delivery time drops to $11.7. Note that reusing drones increases the maintenance, however as new drones are generally expensive, a reduced number of drones is more likely the less expensive option.

Cheng et al. (2018) defined the Multi-Trip Drone Routing Problem (MTDRP). The problem objective is to minimize transportation costs by considering a fleet of homogeneous drones with a maximum capacity to deliver parcels to the customers. Each customer is associated with a non-negative demand and a hard time window. The drones can perform several trips as long as they return to the ending depot before the time window.

Song et al. (2018) considered the Multiple-Depot VRP With Time Window (MD-VRPTW). The MDVRPTW goal is to maximize the weighted sum of two objectives, the total number of covered tasks, and the total traveling distance during delivery service. In the problem, depots are called stations where the drones start their service. After serving customers, the drone returns to a service station for recharging and reloading to

serve additional customers. Moreover, the formulation considers the time window and load capacity of the drones. The flight time of a drone critically depends on the number of loaded products. Therefore, they developed and applied a weight function to the drone flight time based on the number of loaded products to derive practical drone schedules that can be applied in real-world service situations. They developed a MILP and a Receding Horizon Task Assignment (RHTA) heuristic to test the instances, which are based on an island area. The RHTA derives optimal or near-optimal solutions for large-scale problems in a short time.

Sundar and Rathinam (2014) studied the Fuel Constrained UAV Routing Problem (FCURP) that given a set of targets and depots, a drone, initially stationed at one of the depots, must visit each target at least once. The objective is to minimize the drone travel cost, the total fuel consumed by the vehicle along the path.

Boone et al. (2015) considered a mTSP replacing the traditional delivery trucks for drones. None of the drone's restrictions are considered, such as battery and payload capacity. The objective is to minimize the longest route of a drone to find a solution to the mTSP, the set of cities are divided into clusters based on their location in space, typically using K-means clustering, and each drone is assigned to a cluster. Next, the TSP is solved for each cluster using the constructive heuristic 2-Opt to determine the shortest route through the set of cities.

Table 4 – Overview of previous strategies employing single drones operations in the literature.

Single drone operations									
Author	Year	Problem	# Drone	L	R	# visits per trip	E	ST	Objective
Dorling et al.	2016	MC-DDP MT-DDP	d	W	W	multiple	✓	✓	DC
Cheng et al.	2018	MTDRP	d	W	W	multiple	✓	✓	DCnE
Song et al.	2018	UAVRP	d	W	W	multiple	✓	✓	DD
Sundar and Rathinam	2014	FCURP	1	W	W	1	✓		DT
Boone et al.	2015	mTSP	d	W	W	1			AvgLT

2.2 Drone Applications

Although most drone applications involving health care are directly related to delivery, in this section, we approach the importance of drone delivery to health insurance, law enforcement, and agriculture instead of the logistic aspect. Drones offer various exciting

possibilities to the health care industry, possibilities that help save money and lives. As previously reported, drones can make it possible to deliver blood, vaccines, snakebite serum, and other medical supplies to rural areas and have the ability to reach victims who require immediate medical attention within minutes. They can transport medicine within hospital walls, courier blood between hospital buildings, and give elderly patients tools to support them as they age in place.

A partnership between Zipline, a Silicon Valley robotics company, and Rwanda's health ministry has delivered more than 5,500 units of blood over 2017, often in life-saving situations. Never before have patients in the country received blood so quickly and efficiently. When a doctor or medical staff at one of the clinics needs blood, they send a message or log on to Zipline's order site. Then, they are sent a confirmation message saying a Zip drone is on its way. The drone flies to the clinic at up to 60 mph. When it is within a minute of the destination, the doctor receives a text. The drone then drops the package, attached to a parachute, into a particular zone near the clinic before returning to base (McVeigh, 2018; Scott and Scott, 2017; Glauser, 2018).

The use of a drone in law enforcement is a vast area of research and technology development. Drones can help monitor perimeters, parking lots, prisons, college campuses, stadiums, and other outdoor venues. Employing a thermal imaging camera makes this possible at night as well. Security teams can inspect and monitor roofs and other high places from the ground with drones. Drones also can analyze images and use audio and video sensors to listen for gunfire and explosions. Dozens of drones were authorized by the Federal Aviation Administration (FAA) to respond to hurricanes Harvey, Maria, and Irma, to monitor levees and measure damage, allowing them to get an aerial view of the disaster and determine which areas to prioritize for assistance.

In agriculture, drone images are employed to estimate the degree and extent of plant utilization, monitor nutrients, moisture levels, and measure crop issues like disease, pest problems, weeds, and water-stress (Quilter and Anderson, 2001; Zhang and Kovacs, 2012). While the potential for drone employment in agriculture is significant, there are still several notable drawbacks to their progression beyond the niche market they occupy today. There are well-known technical problems for small drones such as engine power, short flight duration, difficulties in maintaining flight altitude, aircraft stability, and maneuverability in winds and turbulence. Most applications of drone technology rely on its ability to generate and deliver precise and accurate information to inform complementary activities like crop analysis and monitoring. Consequently, data quality is crucial and should be a core priority of drone use decisions. However, there are several known problems with these cameras, including limited optical quality, zoom lenses, fully automatic focusing, and, most notably for vegetation surveys, a lack of a near-infrared band (Probst et al., 2018).

These potential benefits and advantages do not come without their percentage of risks and challenges, particularly when it comes to the complexities of enterprise security. Some of these difficulties include: data crime, drones are just as exposed to hackers, and security vulnerabilities as any other Internet of Things (IoT) device. Another issue concerning drones is legislation. Global regulations are still in their infancy and have a long process to fit the companies' continuous desire to expand their applications. Drones are susceptible to loss of control and collisions, and this can happen from a system failure or frequency interference (Ludwig, 2018).

There are several new potential uses of drones, as we have seen in the previous sections, in public and private sectors, and agriculture, commerce, environment, and health. Given the growing interest and the ability for drones to be easily adapted for new areas of application, as the technology advances, these robots may become more robust, with the ability to accommodate heavier payloads and longer flight times. However, countries have to establish robust and stable regulations concerning drones to provide safety to the population. The work by Vacca and Onishi (2017) provides an excellent overview of implications and consequences for an ever-expanding application and misuse of drones.

3 Problem Definition

This chapter provides a definition of the Flying Sidekick Traveling Salesman Problem (FSTSP) and Multiple Traveling Salesman Problem with Drones (mTSPD), along with examples of solution to the problems.

3.1 Flying Sidekick Traveling Salesman Problem

The Flying Sidekick Traveling Salesman Problem (FSTSP) introduced by Murray and Chu (2015) can be described as follows. Let C represent the set of all customer parcels, such that $C = \{1, 2, \ldots, c\}$. Let $G = (V, A)$ be an undirected graph with $|V| = c + 1$ nodes. Each customer must receive exactly one delivery by either the delivery truck or the drone. Nodes 0 and $c + 1$ represent the depot, from which all vehicles must originate and complete the service. This convention accommodates the case in which the origin depot (0) and destination depot $(c + 1)$ have different physical locations.

For the FSTSP we consider a single drone and a single truck to perform all the deliveries. The two vehicles (drone and truck) may depart and return from the depot either in tandem or independently. While traveling in tandem, the drone is transported by the truck to conserve battery.

The truck acts as a mobile depot and recharging platform for the drone. A drone is capable of carrying one parcel at a time. The drone may be launched from the depot (node 0) or from the truck at any customer location. While the drone can be launched multiple times, it cannot be launched from the same location more than once. A drone can be retrieved at the depot or by the truck at a customer location, but it cannot be retrieved at the same customer location from which it was launched. The truck can visit multiple customers between the launch and retrieval of the drone. The truck must depart from node 0, and return to node $c + 1$. The set of nodes from which a vehicle may depart is represented by $N_0 = \{0, 1, ..., c\}$, while $N_+ = \{1, 2, ..., c + 1\}$ describes the set of nodes a vehicle may retrieve.

An important definition for the problem is operation. An operation consists of combined nodes, a launch node $(i \in N_0)$, return node $(j \in N_+)$, at most one node served by drone $(k \in C)$, and potentially several nodes served by truck $(\{v_i, \ldots, v_j\} \in C)$. In an operation, a drone departs from a truck at customer i, serves customer k, and meets up with the truck at the return node j $(i \neq j \neq k)$. We assume that when a drone launches, the delivery is successful. Before launch, a setup time s^L is required to change the drone's battery and loads the vehicle with a parcel. An additional setup time s^R is needed in

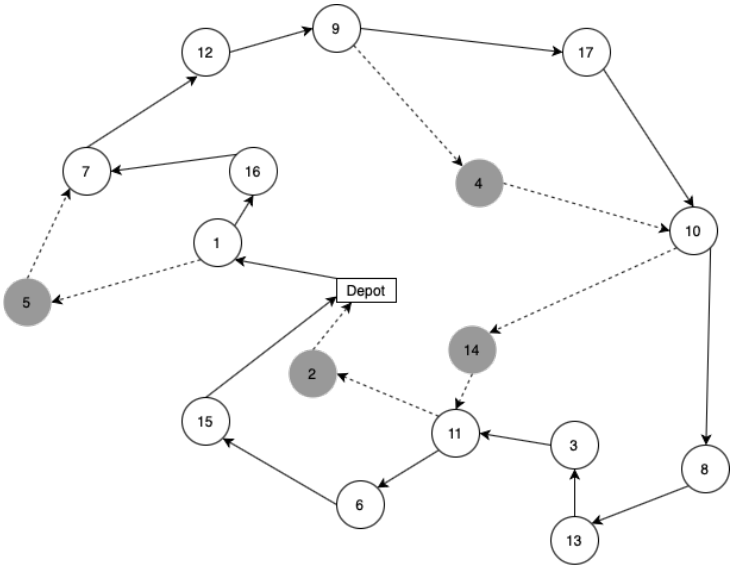


Figure 1 – Spacial FSTSP representation.

node j (return node) to recover the drone. The time to traverse edge (i, j) is defined as t_{ij}^C and t_{ij}^D for truck and drone, respectively. The truck can travel directly from the launch node i to the return node j or can visit any number of nodes in between, as long as the time travel does not exceeds the drone maximum flight range $\left(\sum_{l=i}^{j} t_{l,l+1}^C \leq e\right)$. The vehicle which arrives first at the return node must wait for the other. Further, the drone must not run out of energy before returning to the truck. The drone can carry and deliver only one parcel per operation, while no limit is imposed on the truck's capacity.

The parameters d_{ij}^C and d_{ij}^D define the distance required to travel from node $i \in N_0$ to node $j \in N_+$ by truck and drone, respectively. The vehicles do not necessarily follow the same distance metric. The truck is limited to the road network, while the drone can use a different network to travel between customers. Hence we may consider that $d_{ij}^D \geq d_{ij}^C : \forall i, j \in N$, without loss of generality.

Parameter α is the ratio of drone speed to truck speed, the truck velocity is 1 ($v^T = 1$) and the drone is α times the velocity of the truck ($v^D = \alpha \times v^T$). We assume the vehicles travel at constant speed, thus the time required to travel from customer i to customer j is $t_{ij}^C = \frac{d_{ij}^C}{v^T}$ to the truck and $t_{ij}^D = \frac{d_{ij}^D}{v^d}$ to the drone. Since we assume that $\alpha \geq 1$ and $d_{ij}^C \geq d_{ij}^D : \forall i, j \in N$, we may conclude that $t_{ij}^C \geq t_{ij}^D : \forall i, j \in N$.

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

In Figure 1, the gray nodes are the customers served by the drone. The continuous lines describe the truck route, while the dashed lines are the drone path. We can observe three different operations in Figure 1. Operations (1,5,7) and (9,4,10) in which the truck visits only one customer after launching the drone. Operation (10, 14, 11) that the truck

visits multiple customers during the drone delivery. Finally, in operation (11,2,0), the drone and the truck return to the depot independently.

3.2 Multiple Traveling Salesman Problem with Drones

The Multiple Traveling Salesman Problem with Drones (mTSPD) is a generalization of the FSTSP. While the FSTSP considers a single drone and truck, the mTSPD considers a fleet of trucks and drones to deliver goods to a predefined number of customers.

Let consider M a homogeneous fleet of trucks with infinite capacity and D a fleet of drones with a maximum flight range of e distance units per operation and a payload capacity of one unit. Each vehicle $m \in M$ is assigned with an equal number of D' drones. Each customer must receive exactly one delivery by either one of the delivery trucks from set M or by one of the drones denoted by the set D. Each truck $m \in M$ carries D' drones from set D. Drones must return to the same truck it was launched. Exchange of drones between trucks are not allowed.

All constrains approached in Section 3.1 to the FSTSP are valid for the mTSPD. Figure 2 illustrates a solution to the mTSPD. In this example, $M = M1, M2, M3, M4$, $D = 1, 2, 3, 4, 5, 6, 7, 8$, and $D' = 2$, i.e., we have a set of four trucks and two drones per truck performing the deliveries. The goal of mTSPD is to minimize completion time, i.e., the elapsed time from the first departure of a vehicle from the depot until the return of the last vehicle to the depot.

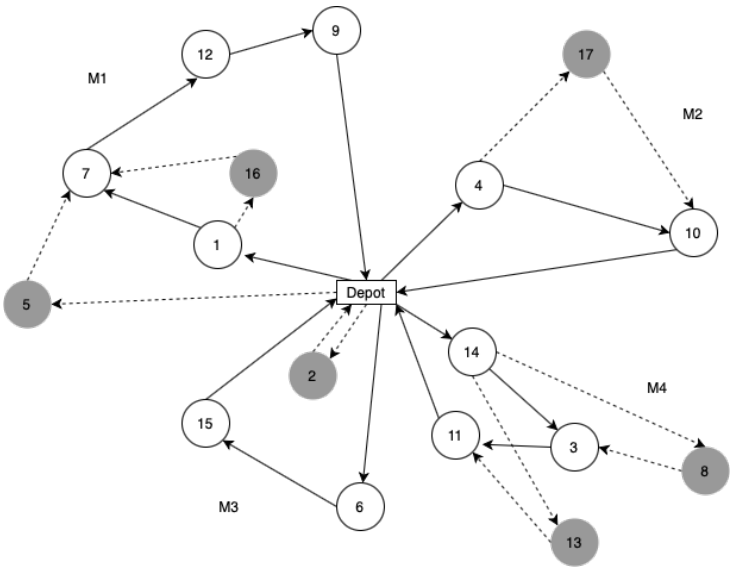


Figure 2 – Spacial mTSPD representation.

4 Mathematical formulations

In this chapter we present two mathematical formulations for the Flying Sidekick Traveling Salesman Problem (FSTSP). A compact model is presented in Section 4.1 while a model with an exponential number of variables is presented in Section 4.2. Table 5 presents the notation employed in the presentation of both formulations.

Table 5 – Sets and input data utilized within the formulation

V	vertex set including the depot and the n customers, $V = \{v_0, \ldots, v_n\}$
V'	vertex set excluding the depot, $V' = V \backslash \{v_0\}$
A	arc set
D	set of possible drone paths (i, k, j) formed by two arcs, (i, k) and (k, j) that respects the drone's maximum endurance
L	set of possible position for truck visits, $L = \{0, \ldots, n\}$
e	drone flight endurance time
s^L	setup time for launching the drone
s^R	setup time for returning the drone
$\tau_{i,j}$	time required by the truck to traverse arc (i, j)
$\tau^D_{i,k,j}$	time required by the drone to traverse arcs (i, k) and (k, j)
M	upper bound for the time required by the truck to visit all customers

4.1 Compact model

This section proposes a compact Mixed Integer Programming (MIP) formulation for the FSTSP. A paper is in processing to be published, this is a fragment of the text that can be found in Freitas et al. (2021).

As with the problem description in Chapter 3, a graph $G = (V, A)$ is considered to represent the problem. Note in Table 5 that L is the set of possible *position* for the truck to visit a customer. It is defined such that $L = \{0, \ldots, n\}$, where n is the number of customers. This set contains indices used to select the order in which the customers are visited. The customers' visiting order is crucial to synchronize the truck and the drone.

The formulation considers three variable sets:

t^ℓ : variable that defines the total travel time until position ℓ;

$x^\ell_{i,j}$: binary variable equal to 1 if the truck traverses arc (i, j) at position ℓ, and 0 otherwise;

$y_{i,k,j}^{\ell,\ell'}$: binary variable equal to 1 if the drone traverses arcs (i,k) and (k,j), launching from vertex i at position ℓ and returning to the truck in vertex j at position $\ell' > \ell$, and 0 otherwise.

At first sight, the number of variables may seem prohibitively significant. However, in practice, this number can be considerably reduced by filtering variable sets x and y to consider only feasible connections, meaning only $(i,j) \in A$ and $(i,k,j) \in D$ should be considered. Moreover, despite requiring more variables, the formulation here proposed yields much better linear relaxation lower bounds than the formulations proposed by Murray and Chu (2015) and Ponza (2016) for all instances considered (computational results are presented in Section 6.1).

The formulation is presented by Equations (4.1)–(4.15). To simplify the notation and reduce the constraints length, we assume $x_{i,j}^{\ell} = 0$ for all $(i,j) \notin A$ and, analogously, $y_{i,k,j}^{\ell,\ell'} = 0$ for all $(i,k,j) \notin D$ and all nonexistent positions pairs (ℓ,ℓ') with $\ell \geq \ell'$. Note that our implementation does not generate such variables.

$$min. \quad t^{n+1} \tag{4.1}$$

$$s.t. \quad \sum_{j \in V} x_{v_0,j}^{0} = \sum_{j \in V} \sum_{\ell \in L \setminus \{0\}} x_{j,v_0}^{\ell} = 1 \tag{4.2}$$

$$\sum_{j \in V} \sum_{\ell \in L} x_{i,j}^{\ell} = \sum_{j \in V} \sum_{\ell \in L} x_{j,i}^{\ell} \leq 1 \qquad \forall i \in V \tag{4.3}$$

$$\sum_{j \in V} x_{j,k}^{\ell-1} = \sum_{j \in V} x_{k,j}^{\ell} \qquad \forall k \in V', \ell \in L \setminus \{0\} \tag{4.4}$$

$$\sum_{(i,j) \in A} x_{i,j}^{\ell} \leq 1 \qquad \forall \ell \in L \tag{4.5}$$

$$\sum_{(i,k,j) \in D} \sum_{l=0}^{\ell} \sum_{l'=\ell+1}^{n} y_{i,k,j}^{l,l'} \leq 1 \qquad \forall \ell \in L \tag{4.6}$$

$$\sum_{j \in V} \sum_{\ell \in L} x_{k,j}^{\ell} + \sum_{i \in V} \sum_{j \in V} \sum_{\ell \in L} \sum_{\ell' \in L} y_{i,k,j}^{\ell,\ell'} = 1 \qquad \forall k \in V' \tag{4.7}$$

$$\sum_{k \in V'} \sum_{j \in V} \sum_{\ell' \in L} y_{i,k,j}^{\ell,\ell'} \leq \sum_{j \in V} x_{i,j}^{\ell} \qquad \forall i \in V, \ell \in L \tag{4.8}$$

$$\sum_{i \in V} \sum_{k \in V'} \sum_{\ell \in L} y_{i,k,j}^{\ell,\ell'} \leq \sum_{i \in V} x_{i,j}^{\ell'-1} \qquad \forall j \in V, \ell' \in L \tag{4.9}$$

$$t^{\ell'} - t^{\ell} \leq e + M \left(1 - \sum_{(i,k,j) \in D} y_{i,k,j}^{\ell,\ell'} \right) \qquad \forall \ell \in L \setminus \{0\}, \ell' \in L : \ell' > \ell \tag{4.10}$$

$$t^{\ell} \geq t^{\ell-1} + \sum_{(i,j) \in A} \tau_{i,j} x_{i,j}^{\ell'-1} + \sum_{\substack{(i,k,j) \in D, \\ \ell > 1}} \sum_{l'=\ell}^{n} s^{L} y_{i,k,j}^{\ell-1,l'} + \sum_{(i,k,j) \in D} \sum_{l=1}^{\ell-1} s^{R} y_{i,k,j}^{l,\ell}$$

$$\forall \ell \in L \setminus \{0\} \cup \{n+1\} \tag{4.11}$$

$$t^{\ell'} \geq t^{\ell} + \sum_{(i,k,j)\in D} \left(s^L + \tau^D_{i,k,j} + s^R\right) y^{\ell,\ell'}_{i,k,j} \quad \forall \ell' \in L\backslash\{0\}, \ell \in L : \ell < \ell' \tag{4.12}$$

$$t^0 = 0 \tag{4.13}$$

$$x^{\ell}_{i,j} \in \{0,1\} \qquad \forall (i,j) \in A, \ell \in L \tag{4.14}$$

$$y^{\ell,\ell'}_{i,k,j} \in \{0,1\} \qquad \forall (i,j,k) \in D, \ell \in L, \ell' \in L : \ell' > \ell \tag{4.15}$$

The objective function presented by Equation (4.1) minimizes the total time to visit all customers, given by the sum of the truck's traveling time and all required setup times to launch and collect the drone. Constraints (4.2) ensure the truck leaves the depot at the position zero and returns to it at the tour's end. Constraints (4.3) limit the number of truck visits to any customer to one. Constraints (4.4) are flow preservation constraints that force the truck to leave a customer at the next position of its visit. Constraints (4.5) limit the number of arcs traversed at each position to at most one. Constraints (4.6) prohibit launching the drone more than once in overlapping time windows (given by l and l') and therefore assert the drone is not launched when it is not with the truck. Note that l and l' cover all time windows, including position ℓ. Constraints (4.7) guarantee every customer is visited exactly once, either by truck or by drone. Constraints (4.8) and (4.9) synchronize the truck's position with the drone's launch and return, respectively. Constraints (4.10) certify that the drone's endurance is respected. Note that these constraints employ a '*Big M*', which disables the constraint whenever the drone is not launched. The value of M is set to an upper bound[1] on the time at which both the drone and the truck return to the depot. Also, the truck's travel time is not considered for endurance when the drone launches from the depot (when $\ell = 0$). Constraints (4.11) update the travel time until position ℓ considering the truck's route. Eventual setup times s^L and s^R of launching and returning the drone, respectively, are considered. Similarly, Constraints (4.12) ensure the travel time until position ℓ includes the time traveled by the drone and eventual setup times s^L and s^R. Therefore, time t^{ℓ} of any position $\ell > 0$ considers the travel time of both truck and drone, whichever is larger. Constraint (4.13) sets the total travel time at the first position to zero and, finally, Constraints (4.14) and (4.15) declare the binary nature of variables x and y.

4.2 Model with an exponential number of variables

In this section, we propose an alternative formulation for the FSTSP which considers all possible paths P that the truck can travel without the drone. For simplicity, let β_p be the truck travel time to traverse path p, and let $P_{i,j} \subseteq P$ be the subset of paths in

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

P which starts at vertex i and ends at vertex j. In total, the formulation considers four variable sets:

$x_{i,j}$: binary variable equal to 1 if the truck travels from vertex i to vertex j using some path $p \in P_{i,j}$, and 0 otherwise;

$y_{i,k,j}$: binary variable equal to 1 if the drone traverses arcs (i,k) and (k,j), launching from vertex i and returning to the truck in vertex j, and 0 otherwise;

$t_{i,j}$: variable that defines the total travel time from vertex i to j, considering the service time;

λ_p : binary variable equal to 1 if the truck traverses path $p \in P$, and 0 otherwise;

Figures 3 and 4 show two examples of λ variables. In Figure 3 one can see the meaning of λ_p with $p = (3,1,2,4)$, while Figure 4 shows the meaning of variable λ_p when p represents the path $(3,4)$.

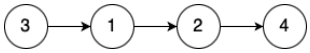


Figure 3 – Representation of variable $\lambda_{(3,1,2,4)}$.

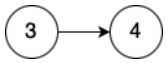


Figure 4 – Representation of variable $\lambda_{(3,4)}$.

Note that there may be a prohibitively large number of λ variables, since there may be too many paths the truck can perform to visit the customers without carrying the drone. To circumvent this issue, initially only paths consisting of one edge (or two vertices) are created. Paths containing additional vertices are created *on demand* by solving a pricing problem in a column generation scheme that will be discussed in Section 4.2.1.

Formulation (4.16)–(4.27) presents a mathematical model for the FSTSP which uses an exponential number of variables λ.

$$min. \quad \sum_{(i,j)\in A} t_{i,j} + \sum_{(i,k,j)\in D} (s_L + s_R) y_{i,k,j} \tag{4.16}$$

$$s.t. \quad \sum_{j\in V} x_{0,j} = \sum_{j\in V} x_{j,0} = 1 \tag{4.17}$$

$$\sum_{j\in V} x_{i,j} = \sum_{j\in V} x_{j,i} \qquad \forall i \in V \tag{4.18}$$

$$\sum_{k\in V'} y_{i,k,j} \leq x_{i,j} \qquad \forall (i,j) \in A \tag{4.19}$$

$$\sum_{i\in S}\sum_{j\in S} x_{i,j} \leq |S| - 1 \qquad \forall S \subseteq V', |S| \geq 2 \tag{4.20}$$

$$\sum_{k \in C^D} t^D_{i,k,j} y_{i,k,j} \leq t_{i,j} \qquad \forall (i,j) \in A \tag{4.21}$$

$$\sum_{p \in P_{i,j}} \beta_p \lambda_p \leq t_{i,j} \qquad \forall (i,j) \in A \tag{4.22}$$

$$\sum_{i \in V} \sum_{j \in V} \sum_{p \in P_{i,j} | k \in p} \lambda_p + \sum_{i \in V} \sum_{j \in V} y_{i,k,j} = 1 \qquad \forall k \in V' \tag{4.23}$$

$$\sum_{p \in P_{i,j}} \lambda_p = x_{i,j} \qquad \forall (i,j) \in A \tag{4.24}$$

$$x_{i,j} \in \{0,1\} \qquad \forall (i,j) \in A \tag{4.25}$$

$$y_{i,k,j} \in \{0,1\} \qquad \forall (i,k,j) \in D \tag{4.26}$$

$$\lambda_p \in \{0,1\} \qquad \forall p \in P \tag{4.27}$$

The objective function presented by Equation (4.16) minimizes the total time to visit all customers, given by the sum of the truck's travel time and the setup times s^L and s^R of launching and returning the drone. Constraint (4.17) ensures that the truck leaves the depot and returns only once. Constraints (4.18) are flow preservation constraints that force the truck to leave a customer right after of its visit. Constraints (4.19) synchronize the truck's position with the drone's launch and return, i.e., the drone only launches and returns to vertices visited by the truck. Constraints (4.20) ensure the non existence of sub-routes. Constraints (4.22) and (4.21) update route time based on the truck and the drone travel time, respectively. Constraints (4.23) guarantee every customer is visited exactly once, either by truck or drone. Constraints (4.24) set λ_p the values of $x_{i,j}$ and, finally, Constraints (4.14) to (4.27) declare the binary nature of variables.

4.2.1 Solving the formulation

Hereafter, we present the column generation scheme employed to solve the linear relaxation of formulation (4.16)–(4.27).

First the integrality constraints (4.25), (4.26) and (4.27) are relaxed, and only λ variables which consider paths $p \in P$ containing exactly two vertices are generated. This results in a feasible *restricted master* problem. Afterwards, a classic column generation algorithm is executed (Leventhal et al., 1973). The restricted master problem is solved and, next, one or more paths $p \in P$ which result in variables λ with negative reduced costs are generated. These variables are then added to the restricted master problem. The procedure repeats until no variable with negative reduced cost can be obtained. Once the linear relaxation is solved, the resulting solution may be fractional. In this case, a branch-and-price algorithm is employed which branches on variables x and y. Note that branching on these variables considerably simplify the *branch-and-price* algorithm (Savelsbergh, 1997), as it only requires removing certain paths from the problem.

The *pricing* problem

The pricing problem consists in obtaining a path $p \in P$ which yields a variable λ with negative reduced cost considering the current solution of the restricted master problem. To formally present the pricing problem as an integer programming formulation, the following notation will be used:

$z_{i,j}$: binary variable equal to 1 if the truck traverses arc (i, j), and 0 otherwise;

$w_{i,j}$: binary variable equal to 1 if the truck path starts at vertex i and ends at vertex j, and 0 otherwise;

$\beta_{i,j}$: variable that defines the truck travel time to go from vertex i to vertex j, considering the service time;

$\delta_{i,j}$: shadow price associated with Constraint (4.22) for pair (i, j), which computes the truck travel time between vertices i and j;

γ_k : shadow price associated with Constraints (4.23) for customer k, which is responsible for selecting either truck or drone to perform the delivery for the customer;

$\mu_{i,j}$: shadow price associated with Constraint (4.24) for pair (i, j).

Formulation (4.28)–(4.38) presents the pricing problem.

$$min. \quad -\sum_{(i,j)\in A} \delta_{i,j}\beta_{i,j} - \sum_{i\in V'}\sum_{j\in V} \gamma_i z_{i,j} - \sum_{(i,j)\in A} \mu_{i,j} w_{i,j} \tag{4.28}$$

$$s.t. \quad \sum_{(i,j)\in A} t^C_{i,j} z_{i,j} \leq e + s_L + s_R \tag{4.29}$$

$$\sum_{(i,j)\in A} w_{i,j} = 1 \tag{4.30}$$

$$\sum_{i\in V} z_{i,j} - \sum_{i\in V} z_{j,i} = \sum_{i\in V} w_{i,j} - \sum_{i\in V} w_{j,i} \quad \forall j \in V \tag{4.31}$$

$$\beta_{i,j} \geq \sum_{(k,l)\in A} t^C_{k,l} z_{k,l} - M(1 - w_{i,j}) \quad \forall (i,j) \in A \tag{4.32}$$

$$\beta_{i,j} \leq M w_{i,j} \quad \forall (i,j) \in A \tag{4.33}$$

$$\sum_{j\in V} z_{0,j} \leq \sum_{j\in V} w_{0,j} \tag{4.34}$$

$$\sum_{j\in V} z_{j,0} \leq \sum_{j\in V} w_{j,0} \tag{4.35}$$

$$\sum_{i\in S}\sum_{j\in S} z_{i,j} \leq |S| - 1 \quad \forall S \subseteq V', |S| \geq 2 \tag{4.36}$$

$$z_{i,j} \in \{0, 1\} \quad \forall (i,j) \in A \tag{4.37}$$

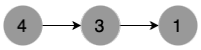
$$w_{i,j}, \in \{0,1\} \qquad \forall (i,j) \in A \tag{4.38}$$

The objective function (4.1) minimizes the travel time of the truck considering the shadow prices $\delta_{i,j}$, γ_i and $\mu_{i,j}$. Constraint (4.29) guarantees the drone's endurance is respected. Constraint (4.30) ensures creation of only one truck path starting at vertex i and ending at vertex j. Constraints (4.31) are flow preservation constraints. Constraints (4.32) and (4.33) set $\beta_{i,j}$ the truck path time. Note that these constraints employ a 'Big M', which disables the constraint whenever the truck does not start its path at vertex i and ends at j. Constraints (4.34) and (4.35) ensure that the deposit (vertex 0) only appears as the first or last vertex of a path. Constraints (4.36) assure that there is no sub-route in the generated path. Finally, Constraints (4.37) and (4.38) state the binary nature of z and w variables, respectively.

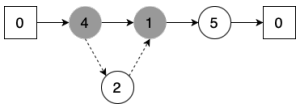
We employ a simple heuristic to quickly obtain solutions for the pricing problem, and formulation (4.28)–(4.38) is used only when the heuristic cannot find feasible columns with negative reduced cost. This heuristic considers a weighted graph $G = (V, A)$, and aims at finding a path starting at a vertex $k \in V$ and ending at a vertex $\ell \in V$ which results in a column with negative reduced cost. To be feasible, such a path must not exceed the drone endurance. Moreover, note that a pair of source and sink vertices (k, ℓ) must be selected. Therefore, the heuristic iterates over all pairs $(k, \ell) \in A$, beginning by those with largest $\mu_{k,\ell}$ values. At each iteration, the weight of edges $(i, j) \in A$ are updated to $\omega_{i,j} = -\gamma_i - t^C_{i,j}\delta_{k,\ell}$ and a shortest path problem is solved considering vertex k as the source and vertex ℓ as the sink. If a feasible solution with cost smaller than $\mu_{k,\ell}$ is obtained, it is returned. The loop is performed until such a solution is obtained or all vertex pairs $(k, \ell) \in A$ are evaluated.

Restricted master problem solution construction

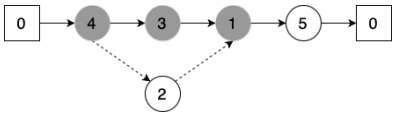
To demonstrate the solution construction, we present an example. Let's assume that Formulation (4.28)–(4.38) resulted in the creation of variable $\lambda_{(4,3,1)}$, illustrated by Figure 5a. The restricted master problem is then updated with the addition of this new variable. After solving the problem, a new solution to the FSTSP considering variables x is obtained, shown by Figure 5b. Note that vertex 3 is missing from the solution. That is expected since vertex 3 is not considered by variables x, being instead part of the path of variable $\lambda_{(4,3,1)}$. Therefore, the paths given by variables λ must be considered to create the complete solution, given by Figure 5c.



(a) Generated $\lambda_{(4,3,1)}$



(b) Solution considering only variables x and y



(c) Solution considering variables λ and y

Figure 5 – FSTSP problem solution

5 Heuristic Method

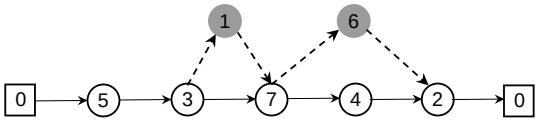
We start this chapter by explaining the data structures adopted in the heuristic implementation (Section 5.1). In Section 5.2 the initial solution is fully detailed. The implementation of the General Variable Neighborhood Search is reported in Section 5.3 along with the description of the neighborhoods. Finally, in Section 5.4 the solution evaluation process is outlined in the possible ways it can be performed.

5.1 Data Structure

Employing the correct data structure is crucial to have a fast and efficient algorithm that can run a significant number of iterations in a reasonable amount of time. Being able to have efficient processing is related to the algorithm time complexity, and the employment of the most appropriate representation may impact saving time in the long run.

At first, the evident approach is an array representing the truck tour and a triple of values representing the drone trip (*launch*, *visit*, *return*). However, the *launch* value would constitute redundant information already stored in the truck solution. Therefore, inspired in the data structure employed by Ponza (2016), the truck's route is represented by array t and the drone's by d^V, d^R, d^I. All arrays have the same size. Array d^V stores the customer visited only by the drone. Array d^R indicates the returning position of the drone. Finally, array d^I represents the truck customers between the launch and return of a drone.

To better understand the solution representation, observe Figure 6 that illustrates the mTSPD. Figure 6a represents the delivery route and Figure 6b is the delivery route represented as the arrays solution. There are two operations in the route. We will analyze the second operation, which is composed of customers $\langle 7, 4, 2, 6 \rangle$.



(a) Route representation of single truck carrying single drone.

	0	1	2	3	4	5	6
t	0	5	3	7	4	2	0
d^V	0	0	1	6	0	0	0
d^R	0	0	3	5	0	0	0
d^I	0	0	0	0	3	0	0

(b) Data structure employed to represent the mTSPD.

Figure 6 – mTSPD solution representation.

- Customer 7 - index 3: Location where a drone is launched from a truck.
- Customer 4 - index 4: Only customer between a drone launch and return
- Customer 2 - index 5: Location where a drone return to a truck.
- Customer 6: Customer visited by drone.

The drone was launched from customer 7, which can be represented in array t by $index = 3$. Array d^V position 3 is then filled with the correspondent customer visited by the drone, in this case, customer 6. In the same position ($index = 3$), d^R stores index 5, indicating the drone return index, i.e, the drone operation ends at node 2 $index = 5$. Finally, array d^I describes the nodes visited by the truck alone. For every position $3 < index < 5$, array d^I is filled with index 3, to indicate the nodes visited by the truck while the drone was on a trip. The values in the array indicate where the operation began, i.e., where the drone was launched.

5.2 Initial Solution

The initial solution to the mTSPD is obtained with a multi-start procedure that executes two different approaches, the Route First Cluster Second (RFCS) (Beasley, 1983) and the Cluster First Route Second (CFRS) (Garside and Laili, 2019), and returns the best solution. The CFRS heuristic is an approach to vehicle routing problems in which problem decomposition is used to tackle the fact that the VRP is $\mathcal{NP}$-Hard. The problem is decomposed into subproblems where each is a TSP. The challenge is to build groups such that the route distances sum is minimized in each cluster (Fisher and Jaikumar,

1981). The RFCS approach is ideally applied to problems with an unrestricted number of vehicles with capacity constraint (Beasley, 1983), as new trucks only will be allocated if there is no space left in the trucks already in use. However, even with a fixed number of vehicles is simple to partition a tour into a set of vehicle routes near-optimally.

The multi-start reported in Algorithm 1 requires two arguments: $maxIter$ and δ. Parameter $maxIter$ indicates the maximum number of iterations the method can perform. In the classical VRP, the vehicles can visit a restricted number of customers per tour because of their limited capacity. Capacity constraints can be combined with well-known shortest path algorithms as Bellman (1958) and Dijkstra (1959). The mTSPD addressed in this work neglect the trucks' capacity, making the split algorithm not straightforward to apply in the mTSPD. Therefore, to balance the number of customers per vehicle, the truck with the highest number of customers and the smallest may have a maximum difference δ.

The algorithm begins with a loop (line 2) that selects the method to be performed (line 3). Followed by a switch statement that checks which approach was chosen, executes it, and the solution is saved in S (lines 5 – 10). If the best solution S^* is improved, it is updated (lines 11 and 12). Finally, the best solution obtained is returned once the main loop finishes (line 14).

Algorithm 1: Multi-Start Initial Solution

```
1  initialSolution (maxIter, δ)
2      for i ← 0 to maxIter do
3          approach ← randomly choose between RFCS and CFRS;
4          nCustomerPerTruck ← divideCustomers(δ);
5          switch approach do
6              case RFCS do
7                  S ← RFCS(nCustomerPerTruck);
8              case CFRS do
9                  S ← CFRS(nCustomerPerTruck);
10         end
11         if f(S) < f(S*) then
12             S* ← S;
13     end
14     return S*;
```

In what follows, we detail how the Route First Cluster Second and Cluster First Route Second algorithms were implemented. Beasley (1983) proposed the Route First Cluster Second approach. The method first creates a giant tour from the depot visiting all customers and back to the depot, i.e., a TSP tour around all customers, including the depot. Then, the tour is portioned into several routes, one for each truck. Algorithm 2 describes the Route First Cluster Second method. The first step of the algorithm is to calculate the TSP tour (line 2). Afterward, a loop starts at line 5 to fulfill the trucks with

the number of customers stored in parameter $nCustomerPerTruck$. In each iteration, a customer $tour_c$ is added to vehicle t and index c is incremented in one unit to go to the next customer in the tour (lines 7 and 8). At the end of the algorithm, the route is returned (line 11).

Algorithm 2: Route First Cluster Second

```
1  RFCS (nCustomerPerTruck)
2      tour ← calcTSP();
3      c ← 0;
4      s ← ∅;
5      foreach t ∈ nTruck do
6          while c ≤ nCustomerPerTruck_t do
7              s_t ← s_t ∪ tour_c ;
8              c ← c + 1;
9          end
10     end
11     return s;
```

When applying the Cluster First Route Second approach, the original problem is decomposed into smaller subproblems by first building groups with the customers. The customers in each group are routed. The routing is the well-known TSP. In general, petal-shaped routes are a common geometric feature in VRP solutions (Ryan et al., 1993). Optimal solutions exhibit a geometric structure for many problems, although finding the optimal solution is still $\mathcal{NP}$-hard. Clustering methods that encourage such features can be useful to find good solutions for practical-sized problems. In this work, we applied the Sweep Algorithm introduced by Gillett and Miller (1974).

The Sweep algorithm employs polar coordinates and consists of two stages. The first stage is called split, where clusters are built by rotating a ray centered at the depot, as shown in Figure 7. The circles represent the customers, and the straight lines are the sweep hand that moves anti-clockwise. Each group joins the closest nodes to the depot, i.e., the nodes with the smallest angle until it satisfies a certain number of customers per cluster. The process finishes when all nodes are included in a cluster.

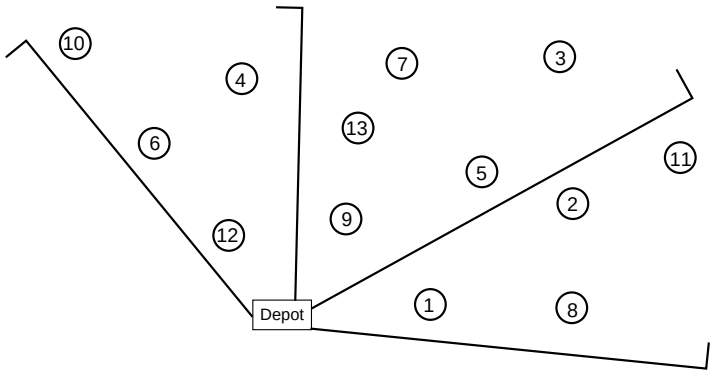


Figure 7 – Clustering Process.

The second stage of the Sweep Algorithm is to generate the routes, i.e., link all nodes in every cluster, starting and ending at the depot. The result of this process is a cycle connecting all nodes in the cluster. At this stage, we need an algorithm to tackle the TSP to build the route. We applied the Nearest Neighborhood (NN) algorithm in each cluster to create the route, which is later described in this section. The Nearest NeighborhoodNN is possibly the most straightforward TSP heuristic; it can be coded to have time complexity $O(n^2)$ and generates solutions often 25% from optimality (Nilsson, 2003).

Algorithm 3 illustrates the process. The function receives as a parameter a single array, $nCustomerPerTruck$, that indicates the number of customers in the clusters. As a result, the algorithm returns the customers of each cluster (line 18). In line 2 variable t indicates which truck is being processed. In line 4, one calculates the angle of each

customer concerning the depot and then sorted in increasing order.

Algorithm 3: Cluster First Route Second

```
1  CFRS (nCustomerPerTruck)
2      t ← 0;
3      i ← 0;
4      customers ← sort(calcAngles());
5      while i < n and t < nTrucks do
6          loadCustomers ← 0;
7          cluster ← ∅;
8          while loadCustomers ≤ nCustomerPerTruck_t do
9              clusters ← cluster ∪ customers_i;
10             loadCustomers ← loadCustomers+1;
11             i ← i+1;
12         end
13         if (ConcordeSolution(cluster_t)) then
14             NearestNeighbor(cluster_t);
15         end
16         t ← t+1;
17     end
18     return clusters;
```

The main loop is executed until all customers are explored (line 5 – 17). The variable $loadCustomer$, initialized in line 6, indicates how many customers are in cluster t at each point of the algorithm. In each iteration of the inner loop (line 8 – 12) a customer from $customers$ is added to the current cluster (line 9), until $loadCustomer$ reaches $nCustomerPerTruck_t$. After a cluster is complete, the set of customers t is passed to Concorde (Applegate et al., 1996) find a valid tour. If Concorde cannot find a solution in 5 seconds, than, the Nearest Neighborhood algorithm is performed to built the route (line 15).

In what follows, we give a thorough description of obtaining the customers' angles and building a truck tour. Algorithm 4 shows how to calculate the angles for each customer coordinate. First, the reference point (depot) and angle are set (lines 2 to 4). Next, a loop (line 5) is responsible for determine the angle of each customer. Since the depot is our reference, we need to calculate the difference between the Cartesian coordinates v_i and the depot (lines 6 and 7). The angle is determined by $\tan^{-1}(y/x)$. Since the customers can be in any quadrant, the angle value may be summed up with 180 or 360. If both x and y are greater than or equal to 0, the point is in the first quadrant, and nothing has to be done (line 8). If x is less than 0 and y is greater (less) than 0, then the point is in the second (third) quadrant and 180 has to be added to the arctangent value (lines 10 and 12). Lastly, if x is greater than 0 and y is less than 0, the point is in the fourth quadrant. Therefore, 360 is summed to the arctangent value (line 14). At the end of each iteration, the tuple (angle, index) is added to the set $customers$.

Algorithm 4: Calculate customers cartesian coordinates

```
1  calcAngles ()
2      xDepot ← v_0.x;
3      yDepot ← v_0.y;
4      customers ← ∅;
5      for i ← 1 to n do
6          x ← v_i.x - xDepot;
7          y ← v_i.y - yDepot;
8          if x ≥ 0 and y ≥ 0 then
9              θ ← arctan(y/x);                          /* first quadrant */
10         else if x < 0 and y > 0 then
11             θ ← arctan(y/x) + 180;                    /* second quadrant */
12         else if x < 0 and y < 0 then
13             θ ← arctan(y/x) + 180;                    /* third quadrant */
14         else if x > 0 and y < 0 then
15             θ ← arctan(y/x) + 360;                    /* forth quadrant */
16         customers ← customers ∪ (θ, i);
17     end
18     return customers;
```

The route generation works as reported in Algorithm 5. For each cluster built in the first stage, the Nearest Neighborhood algorithm is performed to find a TSP tour. The first node selected is the depot (v_0) (line 2). Afterward, a loop starts visiting all nodes until there are no remaining ones. In each iteration, a node ($lastVisited$) is inserted into the solution (line 4), then it is immediately removed from the array *cluster* to prevent duplicate nodes from appearing in the solution (line 5). Later, the nearest unvisited node from the array *nearestNode* is selected and added to the solution in the next iteration (line 6).

Algorithm 5: Nearest Neighborhood

```
1  NearestNeighborhood (cluster)
2      nearestNode ← 0;
3      while cluster ≠ empty do
4          S ← S ∪ {nearestNode};
5          cluster ← cluster \ {nearestNode};
6          nearestNode ← findNearest(cluster, nearestNode);
7      end
8      return S;
```

5.3 General Variable Neighborhood Search

VNS is a flexible framework for building heuristics to approximate solutions of optimization problems. It was introduced by Mladenović and Hansen (1997), and its main

idea is to systematically change neighborhood structures during the search for an optimal (or near-optimal) solution. According to the authors, the foundations of VNS are based on the following properties: (i) a local optimum relative to one neighborhood structure is not necessarily a local optimum for another neighborhood structure; (ii) a global optimum is a local optimum concerning all neighborhood structures; (iii) for many problems, empirical evidence shows that all local optima are relatively close to each other.

The Variable Neighborhood Descent (VND), proposed by Mladenović and Hansen (1997), is a local search that explores the search space through systematic changes of neighborhood structures in a deterministic way. The process iterates over each neighborhood while improvements are found. Only strictly better solutions are accepted after each neighborhood search. According to Penna et al. (2013) the use of a randomized order led, on average, to better results when compared to the deterministic method. Several papers (Souza et al., 2010; Subramanian et al., 2010; Kramer et al., 2016; Penna et al., 2017) are applying the randomized VND method to explore the neighborhoods instead of a deterministic ordering. With the potential presented by Randomized Variable Neighborhood Descent (RVND) in the previous works, we decided to employ the RVND as the local search procedure. Algorithm 6 shows the proposed VND, which differs from the classical one only by Line 2 that randomly sort the neighborhood list.

Algorithm 6: Randomized Variable Neighborhood Descent

```
1  VND (S)
2      N ← Shuffle Neighborhood List;
3      S* ← S;
4      p ← 1;
5      while p ≤ | N | do
6          S' ← Find the best neighbor s' ∈ N^(k)(s);
7          if f(S') < f(S*) then
8              S* ← S';
9              p ← 1;
10         end
11         else
12             p ← p + 1
13         end
14     end
15     return S*;
```

The General Variable Neighborhood Search (GVNS) described in this work is the VNS using VND as local search. The choice concerning the use of GVNS comes from the recently less is more approach (LIMA) proposed in Mladenović et al. (2016). LIMA 's main idea is to find the minimum number of search components when solving a specific optimization problem, i.e., the goal is to make a heuristic as simple as possible. But at the same time, more effective and efficient than the current state-of-the-art heuristic. The

pseudo-code of the proposed GVNS heuristic is given in Algorithm 7. The whole process is repeated until the imposed time limit of t_{max} seconds is reached (the outer loop that starts from line 3). Besides t_{max}, GVNS has p_{max} and $\rho*$ parameters, which the former defines the maximum number of iterations (see neighborhood loop that starts from line 4) and the latter is the maximum perturbation level. The choice of t_{max}, p_{max} and $\rho*$ values will be described later in Section 6.

The GVNS is presented in Algorithm 7. It begins by setting the initial solution (line 2), the shaking level ρ, and neighborhood count p to 1 (line 2). The main loop (line 3) first calls the shake procedure (line 5) followed by the local search VND (line 6). If the solution produced by VND is better than the current best solution, the best solution is updated, and the shaking level and the neighborhood count are reset to 1 (lines 7 – 11). Otherwise, the current solution is reset to the best solution (line 13). Next, the perturbation level ρ and the neighborhood count are increased (lines 15 and 14) and the loop repeated. Note that the given maximum perturbation level $\rho*$ is never exceeded: ρ is reset to 1 after $\rho*$ is reached. Once the time limit is reached, the best solution produced is returned (line 19).

Algorithm 7: GVNS heuristic to address the mTSPD

```
1  GVNS (S, t_max, p_max, ρ*)
2      S* ← S; ρ ← 1; p ← 1;
3      repeat
4          while p < p_max do
5              S ← Shake(S, ρ);
6              S' ← LS(S);
7              if f(S') < S* then
8                  S* ← S';
9                  p ← 1;
10                 ρ ← 1;
11             end
12             else
13                 S ← S*;
14                 p ← p + 1;
15                 ρ ← (ρ mod ρ*) + 1;
16             end
17         end
18     until t < t_max;
19     return S*;
```

The shaking procedure is employed to avoid local optima traps generated by the local search procedure. GVNS employs the Shake(S, ρ), presented in Algorithm 8. The shaking procedure has two formal parameters: solution S, and shake intensity ρ. The parameter ρ determines the number of iterations performed within the shaking procedure. At each iteration a random move is selected (line 3) and applied to the current solution

(line 4). After the moves are applied, the solution is returned inline 6.

Algorithm 8: Shake

```
1 Shake (S, ρ)
2     for i ← 0 to ρ do
3         N ← selected neighborhood structure;
4         S ← random neighbor N(S);
5     end
6     return S;
```

5.3.1 Neighborhood Structures

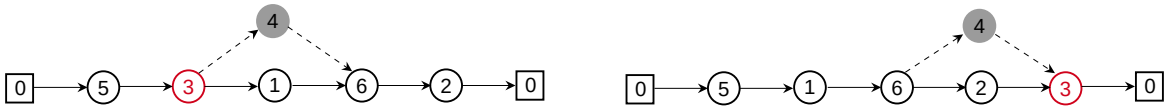
To explore the solution space of the mTSPD, we define seven neighborhood structures: Reinsertion, Swap, Shift(1,0), Shift(1,1), Relocate truck to drone, Swap truck drone, Remove operation. All these structures are used in our GVNS algorithm and are described and detailed as follows.

5.3.1.1 Reinsertion

The move is called reinsertion and is relatively complex, allowing us to explore the neighborhood overall. There are three classifications for the customers in a generic solution: mixed, drone-only, and truck-only. Drone-only customers are those visited only by drones. Mixed customers are those where the drone launches or returns. Finally, truck-only customers are customers served by the truck. The reinsertion move removes customer t_i^a, i.e, the node in position i served by truck t^a, and reinserts it in position j at the same truck a If position j is under the drone operation d_k, the truck path between launch and return increases with the insertion of a new customer. Therefore the endurance constraint may be violated as the drone could have to wait hovering for too long for the truck's arrival in the return node.

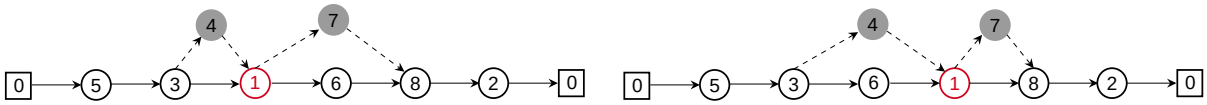
If customer t_i^a launches a drone and $j > i$, then the drone trip may be reversed, as shown in Figure 8. Customer 6, which was previously a launch node, becomes the return node, and customer 3 is the new return of this operation. Therefore, the trip $\langle 3, 4, 6\rangle$ becomes $\langle 6, 4, 3\rangle$. Another important matter to notice in this example is that the reversed drone trip increases the truck path and changes the intermediary customers, which implies that the feasibility constraint may be violated and the move may not be performed.

Considering Figure 8b a different scenario emerges as position i is both return and launch. In this situation, the move can only be performed if position j is between the first launch (after customer 3) and the last return (before customer 8). Otherwise, the operation is infeasible, as the reversed trip would result in a drone trip starting before the end of a previous one.



(a) Solution before *reinsertion move*. (b) Solution after *reinsertion move*.

Figure 8 – Example of the intra route neighborhood reinsertion.

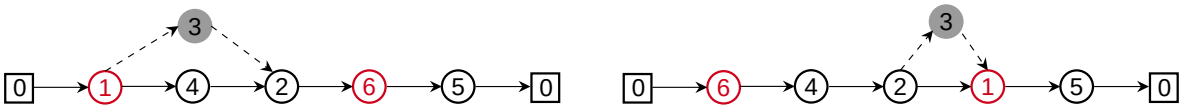


(a) Solution before relocate move. (b) Solution after relocate move.

Figure 9 – Possible route modifications involving mixed customers.

5.3.1.2 Swap

In the swap move (Osman, 1993) a customer t_i^a swaps position with the customer t_j^a. This move can also results in reversing of a drone trip, as we can observe in Figure 10. The move is completed as long as the swap solution is feasible, i.e., the endurance constraints were not violated, and a drone operation does not overlap an already existing operation.



(a) Solution before swap intra route move. (b) Solution after swap intra route move.

Figure 10 – Example of swap intra route move.

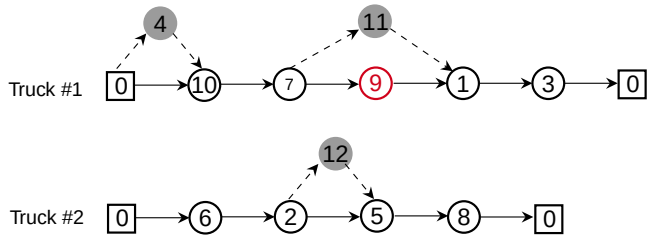
5.3.1.3 Shift(1,0)

This is a inter route move, which customer t_i^a is relocated to position j at truck t_b. The validation in the destiny truck t_b is relatively simple. Since customer t_i^a may be relocated to a position j under an operation, we have to guarantee that the endurance constraint is not violated with the increase of the truck delivery time. If it is the case that the insertion of customer t_i^a made the truck travel time longer than the endurance, then the move is not completed.

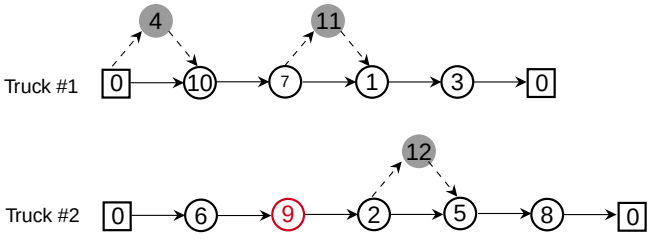
In contrast with truck t_b, when considering truck t_a several scenarios arise. If t_i^a is not in a drone operation the move is fairly straightforward, as can be seen in Figure 11b, where customer 9 from *Truck #1* is moved to position 3 in *Truck #2*.

However, if t_i^a is a launch or return node, a different situation occurs. When an operation starts at i, a backward search $(i-1,\ldots,0)$ begins at node $i-1$ and continues until it finds a return node or the depot. The search stops at the first return node because

an operation must start after the ending of a previous one. If the search does not find a node to be the new launch, i.e., the travel time exceeds the endurance for each new possible launch, then the move is not complete. In Figure 11a this would mean that if customer 7 is removed from *Truck #1*, then a backward search would start at node 10, and also end at node 10 since it is a return node. If the trip $\langle 10, 1, 3\rangle$ was not feasible, the move would not be completed. The same procedure is performed when t_i^a is a return node; however, instead of a backward search, it is a forward search $(i, i+1, \ldots, 0)$. Customer 10 is removed from its position, then starts a forward search at node 7 and ends at the same node 7 as a launch node. As before, the move would be completed as long as the new trip is feasible.



(a) Before the relocate move.



(b) After the relocate move.

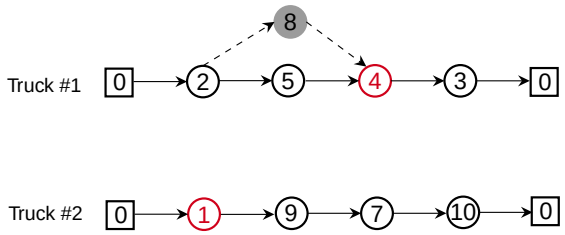
Figure 11 – Example of a solution before and after a relocate inter route move.

5.3.1.4 Swap(1,1)

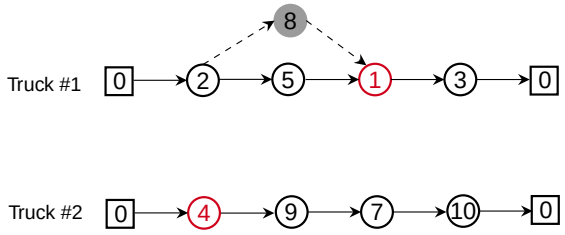
In this move customer t_i^a swaps position with customer t_j^b. This move is relatively easy to implement. A node can become part of an operation or may cease being in an operation, as shown in Figure 12, where node 4 was a return node in *Truck #1*, with the swap in *Truck #2* it is not part of an operation. On the other hand, customer 1 became the return node of a drone trip.

5.3.1.5 Relocate truck to drone

The next move is an inter route move between truck and drone, meaning that both vehicles are involved. Customer t_i^a is relocated to drone d_a^k, where d_a^k is drone k carried by truck a. This move removes customer t_i^a and inserts it into drone d_a^k, creating



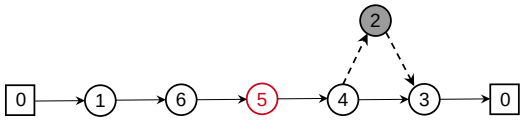
(a) Before the swap move.



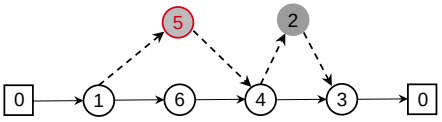
(b) Customer from Truck #1 and #2 exchange position.

Figure 12 – Example of a solution before and after a swap inter route move.

a new operation. As can be seen in Figure 13a, customer 5 was served by the truck, and after the move in Figure 13b is served by the drone, and it was removed from the truck route. In order to create an operation, triples composed of three distinct nodes, launch, visit and return, are investigated until the solution quality is improved and the endurance constraints are satisfied. Considering Figure 13b the inequalities $\tau^D_{1,6,4} \leq e$ and $\sum_{i=1}^{3} \tau^T_{i,i+1} \leq e$ must be assured to complete the move.



(a) Solution representation before move.

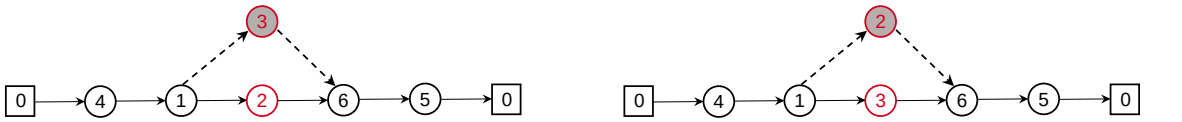


(b) Solution representation with customer 5 being attended by drone.

Figure 13 – Move where a truck customer becomes a drone customer.

5.3.1.6 Swap truck drone

This move swaps a truck customer t^a_i with a drone customer d^a_{kj}, meaning that the customer in position j carried by the drone d_k in truck a is now served by truck a and customer t^a_i is served by drone d^a_k. The move is complete as long as the endurance constraints are still satisfied.



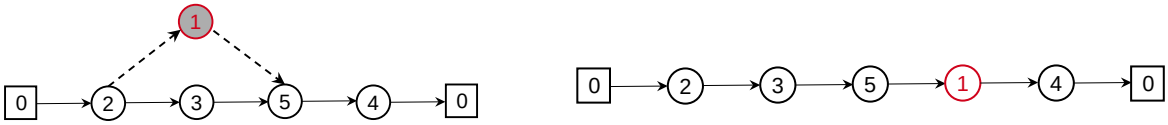
(a) Solution before swap move.

(b) Route where customers 2 and 3 exchange positions.

Figure 14 – Swap move where truck and drone exchange customers.

5.3.1.7 Remove operation

The *remove operation* move is the opposite of *Relocate truck to drone* move. This move removes an operation, i.e., a drone customer is relocated back to the truck route. A random drone customer d^a_{ki} is removed from the drone route and re-inserted in a random position j in truck a. Note that customer d^a_{ki} is re-inserted in the same truck that carries the drone. This move is only used in the shaking procedure, as it may increase the solution value.



(a) Original route with one drone operation.

(b) Route without drone operation.

Figure 15 – Move where a drone operation is removed.

5.4 Solution evaluation

The last topic to discuss concerning the heuristics is the solution evaluation. There are two alternatives to evaluate a solution, a complete evaluation from the starting depot to the end depot and a delta evaluation to find the solution cost after a move is performed.

The complete evaluation method is straightforward to code and provides consistent results. However, it is by its nature redundant since it computes components of the solution that stay untouched after a move. This characteristic makes the evaluation presents a complexity of $O(n^2)$ in the best and worst-case scenarios. On the other hand, the delta evaluation is $O(1)$ in the best case and $O(n^2)$ only in the worst-case scenario.

Figure 16 shows a segment of a vehicle route. Let assume that the number of drones $D = 0$, the vehicle k arrives at the depot after finishing its tour. Since the vehicle has not to wait at any node, the time a^k_{n+1} is equal to the time required by the vehicle to serve all customers. Now, consider $D = 1$, and a drone operation will be placed in the route, such that the arrival time in Figure 17 is reduced. Furthermore, assume that we know the arrival times at each node, $a_1, \ldots, a_6$, respectively.

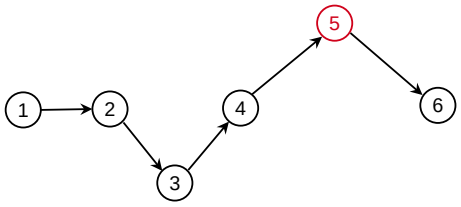


Figure 16 – A segment of route.

Next, consider that we want to add the drone operation $\langle 1, 5, 6\rangle$. As shown in Figure 17, the drone is launched from node 1, continues its flight to serve node 5, and then will retrieve to the truck at node 6. Considering that $a^{t,1}$ and $a^{d,1}$ are the earliest arrival time of the truck and the drone, the arrival time a'_6 at node 6, where both vehicles have reached, can be calculated as follows:

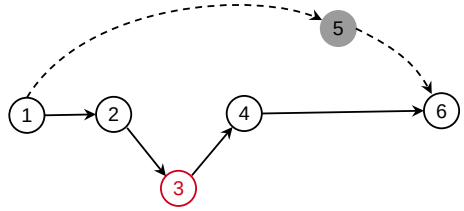


Figure 17 – A segment of route with a single drone operation $\langle 1, 5, 6\rangle$.

$$a_6^{d,1} = a_1 + \tau'_{1,5} + \tau'_{5,6} \tag{5.1}$$
$$a_6^{t,1} = a_4 - \tau_{4,5} - \tau_{5,6} + \tau_{4,6} \tag{5.2}$$
$$a'_6 = \max\{a_6^{d,1}, a_6^{t,1}\} \tag{5.3}$$

Now, consider $D = 2$, i.e., we have one more drone available. Given the arrangement of the nodes, the operation $\langle 2, 3, 4\rangle$ seems a reasonable choice for new drone operation, as shown in Figure 18. Furthermore, assume that we know the updated earliest arrival times at each node $a'_1, \ldots, a'_6$ of the graph in Figure 17. The earliest arrival time a''_6, at which the vehicle and the drones have reached the node 6, can be calculated as follows (where $a^{d,1}$ and $a^{d,2}$ are the earliest arrival time of the first and second drone and $a^{t,2}$ is the earliest arrival time of the vehicle at node $i : \forall i \in N$):

$$a_6^{d,1} = a'_1 + \tau'_{1,5} + \tau'_{5,6} \tag{5.4}$$
$$a_4^{t,2} = a'_4 - \tau_{2,3} - \tau_{3,4} + \tau_{2,4} \tag{5.5}$$
$$a_4^{d,2} = a'_2 + \tau'_{2,3} + \tau'_{3,4} \tag{5.6}$$

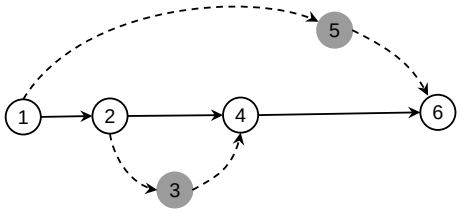


Figure 18 – A segment of route with two drone operations $\langle 1, 5, 6\rangle$ and $\langle 2, 3, 4\rangle$

$$a_4'' = \max\{a_4^{d,2}, a_4^{t,1}\} \tag{5.7}$$
$$a_6^{t,2} = a_4'' + \tau_{4,6} \tag{5.8}$$
$$a_6'' = \max\{a_6^{d,1}, a_6^{t,2}\} \tag{5.9}$$
$$\tag{5.10}$$

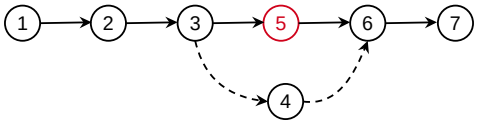
Adding the operation $\langle 2, 3, 4\rangle$ will improve the objective function only if the condition described in Eq. (5.11) holds.

$$a_6'' < a_6' = \max\{a_6^{d,1}, a_6^{t,1}\} \leq \max\{a_6^{d,1}, a_6^{t,1}, a_6^{t,2}\} \tag{5.11}$$

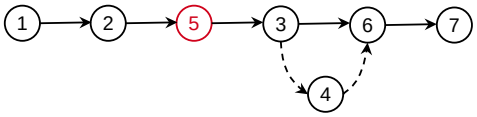
In other words, if the sequence of vertices 1, 5, 6, visited by the first drone in Figure 18, is the longest path through the directed graph that connects vertices 1 and 6 (i.e., $a_6^{d,1} > a_6^{t,1}$, then the earliest arrival time to reach node 6 in Figure 18 is determined by the drone as $a_6' = a_6^{d,1} = max\{a_6^{d,1}, a_6^{t,1}\}$. Consequently, if $a_6' = a_6^{d,1} = max\{a_6^{d,1}, a_6^{t,1}\}$, we will not be able to improve the objective function by reducing the arrival time of the truck at node 6 (i.e., find a drone operation such that $a_6^{t,2} < a_6^{t,1}$, because the earliest arrival time a_6'' will always be determined by $a_6^{d,1}$. Further, if $a_6' = a_6^{t,1} = max\{a_6^{d,1}, a_6^{t,1}\}$ then the only way to improve the objective value is if the arrival time of the vehicle is reduced by adding an additional drone operation.

Now, consider the relocate move where a node is removed and re-inserted in another position in the truck's tour. As shown in Figure 19 node 5 was under a drone operation, and was moved to another position. Considering $a^{t,1}$ and $a^{d,1}$ the earliest arrival time of the truck and the drone, and we know the arrival times at each node $a1, \ldots, a7$. The new arrival time a_7' at node 7 can be calculated as showed in Equation (5.12).

$$a_6^{d,1} = a_3 + \tau_{3,4}' + \tau_{4,6}' \tag{5.12}$$
$$a_6^{t,1} = a_6 - \tau_{2,3} - \tau_{3,5} - \tau_{5,6} + \tau_{2,5} + \tau_{5,3} + \tau_{3,6} \tag{5.13}$$
$$a_7' = a_7 - a_6 + \max\{a_6^{d,1}, a_6^{t,1}\} \tag{5.14}$$



(a) Segment of route in which customer 5 is about to change position.

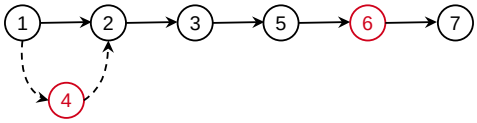


(b) Segment of route after relocation.

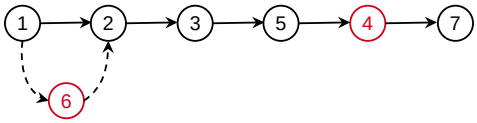
Figure 19 – Relocation of customer 5 in the truck route.

To obtain the arrival time of the truck at node 6 we need to subtract the removed edges $\langle(2,3),(3,5),(5,6)\rangle$ and sum up the newly created edges $\langle(2,5),(5,3),(3,6)\rangle$. The relocation move will be completed as long as $a'_7 < a_7$. The same procedure is done to obtain the updated solution value to the swap move.

Figure 20a is a representation of the *swap truck and drone* move, where customer 6 served by the truck swaps position with customer 4 served by the drone. Equation (5.15) describes how to evaluate the new solution of Figure 20b.



(a) Segment of route.



(b) Truck customer 6 and drone customer 4 after changing position.

Figure 20 – Representation of a swap customers between the vehicles.

$$a_2^{d,1} = a_1 + \tau'_{1,6} + \tau'_{6,2} \tag{5.15}$$

$$a_2^{t,1} = a_1 + \tau_{1,2} \tag{5.16}$$

$$a'_2 = max\{a_2^{d,1}, a_2^{t,1}\} \tag{5.17}$$

$$a'_6 = a_7 - \tau_{5,6} - \tau_{6,7} + \tau_{5,4} + \tau_{4,7} \tag{5.18}$$

$$a'_7 = a_7 - a_2 + a'_2 - a_6 + a'_6 \tag{5.19}$$

6 Computational Experiments

We implemented the GVNS approach, described in Chapter 5, in C++ (g++ 5.3.1). The experiment was done by running the GVNS ten times for every instance. The formulations were implemented using Python 3 and Python-MIP (Toffolo and Gambini, 2021). We used Gurobi Optimizer to solve the models and Pypy 3.10 to run the formulations experiments. All experiments were executed on an Intel®Core™ i7 Processor 3.6 GHz with 16 GB of RAM running Ubuntu Linux 16.04.

The remainder of this chapter is organized as follows: Section 6.1 presents the results obtained with the exact approaches. Section 6.2 discuss the heuristic experiments. Subsection 6.2.1 describes the instances settings. employed for the heuristic experiments.

6.1 Exact approaches

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

1. in the problem described by Murray and Chu (2015), the truck's travel time between the drone's launch and return can be longer than the drone endurance; the drone can therefore run out of battery while waiting for the truck;
2. Murray and Chu (2015) do not consider the setup time for launching the drone as part of the drone's flying time, and it does not count for the total completion time or the battery's endurance, even when the drone leaves from the depot.

Both formulations can be adapted to obtain results comparable with those by Murray and Chu (2015). In Formulation (4.1)–(4.15), it is necessary to remove Constraints (4.10) and altering Constraints (4.11) and (4.12). In Formulation (4.16)–(4.27), we have to disconsider the s^L from setup time in the object function (4.16), and remove Constraints 4.29 from pricing Formulation (4.28)-(4.38).

The formulation given by (4.1)–(4.15) has a total of $\mathcal{O}(|V|^5)$ variables. However, as aforementioned, the number of variables generated is proportional to the sizes of sets A and D (see Section 4.1), which are generally much smaller in practice than $|V|^2$ and $|V|^3$, respectively. For compactness, in all tables presented in this section we will refer to Murray and Chu (2015) as M&C (2015), Model 1 is the compact formulation presented in Section 4.1 and Model 2 refers to the formulation solved by branch-and-price from Section 4.2. Table 6 presents the average number of generated variables (#Vars) and

Table 6 – Average number of variables and constraints per instance-set and endurance value

Instance set	#nodes	e	Model 1		Model 2		Ponza (2016)		M&C (2015)	
			#Vars	#Constrs	#Vars	#Constrs	#Vars	#Constrs	#Vars	#Constrs
Ponza (2016)	5	1440	658	158	88	105	116	68	-	-
Ponza (2016)	6	1440	1023	212	126	157	144	547	-	-
Ponza (2016)	7	1440	1893	274	160	252	196	790	-	-
Ponza (2016)	8	1440	4411	344	284	450	281	1125	-	-
Ponza (2016)	9	1440	4161	422	280	694	295	1439	-	-
Ponza (2016)	10	1440	9208	508	374	1279	422	1941	-	-
M&C (2015)	10	1200	31758	453	746	1377	-	-	867	2840
M&C (2015)	10	2400	41273	453	974	1377	-	-	867	2840

constraints (#Constrs) for the instances considered. The number of variables regarding Model 2 is the total after the branch-and-price. Note how Model 1 dimensions depend heavily upon the endurance of the drone (e). This is expected since a smaller endurance enables reducing set D's size. It is also noteworthy that even being large the number of variables within the Model 1 is not prohibitive for small instances. When we compare the number of variables of the two models, Model 2 results in a much smaller number of variables in Murray and Chu (2015) instances, which demonstrates that Model 2 does not make the use of memory a bottleneck, unlikely Model 1 as the number of variables can easily escalate.

Model 2 spent 96% of its time solving the pricing problem which indicates that more effort must be applied in the pricing problem resolution to achieve results faster. For solving the shortest path problem during the pricing heuristic, we employed Bellman-Ford algorithm (Bellman, 1958; Ford and Fulkerson, 2015) from NetworkX (2004-2021) library. Even though the heuristic provided good results, its running time showed that it is not competitive. Therefore, to improve the pricing time and the overall results a possibility is to implement a heuristic instead of using one from a library.

Tables 7 and 8 present the results obtained by the modified formulation considering the instances from Murray and Chu (2015) with $e = 20$ and $e = 40$, respectively. Column LB^0 presents the value of the linear relaxation, column *Sol.* presents the solution value, rows with ⊛ indicates that the optimal solution was proven, and column Time reports the total execution runtime in seconds. Note that a ⊛ is included next to column *Sol.* whenever the solution is proven optimal using the indicated formulation. Note also that a runtime limit of 1800 seconds was imposed and that the formulation proposed by Murray and Chu (2015) resulted in value zero for the linear relaxation (LB^0) for all instances. Model 1 resulted in proven optimal solutions for all instances addressed in Tables 7 and 8. Model 1 outperformed Model 2 regarding execution time as it presented an average runtime of 90 seconds, while Model 2 required 130 seconds on average. This results are quite remarkable since with the formulation proposed by Murray and Chu (2015) the solver was incapable

of providing any proven optimal solution within the runtime limit.

Table 7 – Formulation results for Murray and Chu (2015) instances with $e = 20$

Instance	M&C (2015)			Model 1			Model 2		
	LB^0	Sol.	Time	LB^0	Sol.	Time	LB^0	Sol.	Time
20140810T123437v1	0.00	56.47	1800.15	38.78	⊛ 56.47	10.73	49.58	⊛ 56.47	291.42
20140810T123437v2	0.00	53.21	1800.15	36.05	⊛ 53.21	10.07	45.71	⊛ 53.21	188.90
20140810T123437v3	0.00	53.69	1800.18	37.48	⊛ 53.69	10.71	45.05	⊛ 53.69	129.77
20140810T123437v4	0.00	67.46	1800.15	51.77	⊛ 67.46	7.30	57.95	⊛ 67.46	104.42
20140810T123437v5	0.00	50.55	1800.22	30.67	⊛ 50.55	455.30	38.52	⊛ 50.55	96.88
20140810T123437v6	0.00	47.60	1800.23	27.69	⊛ 47.31	330.01	34.28	⊛ 47.31	83.63
20140810T123437v7	0.00	51.89	1800.24	30.85	⊛ 48.58	69.80	35.82	⊛ 48.58	36.94
20140810T123437v8	0.00	64.69	1800.23	43.60	⊛ 61.38	56.82	48.07	⊛ 61.38	35.21
20140810T123437v9	0.00	45.98	1800.25	30.62	⊛ 42.42	92.00	36.03	⊛ 42.42	91.93
20140810T123437v10	0.00	43.09	1800.28	27.60	⊛ 41.73	100.72	31.57	⊛ 41.73	67.93
20140810T123437v11	0.00	48.21	1800.25	30.81	⊛ 42.90	19.26	33.67	⊛ 42.90	33.87
20140810T123437v12	0.00	61.57	1800.27	43.54	⊛ 55.70	28.37	45.58	⊛ 55.70	79.41
20140810T123440v1	0.00	49.43	1800.15	28.23	⊛ 49.43	26.08	41.36	⊛ 49.43	244.98
20140810T123440v2	0.00	50.71	1800.15	28.22	⊛ 50.71	20.48	39.27	⊛ 50.71	153.67
20140810T123440v3	0.00	56.10	1800.17	35.00	⊛ 56.10	21.50	45.30	⊛ 56.10	179.03
20140810T123440v4	0.00	69.90	1800.14	49.00	⊛ 69.90	19.42	59.28	⊛ 69.90	189.12
20140810T123440v5	0.00	45.36	1800.22	28.17	⊛ 43.53	44.28	34.98	⊛ 43.53	76.48
20140810T123440v6	0.00	44.08	1800.22	27.93	⊛ 43.95	40.88	34.11	⊛ 43.95	78.78
20140810T123440v7	0.00	51.92	1800.22	34.94	⊛ 49.42	43.36	39.45	⊛ 49.42	72.44
20140810T123440v8	0.00	65.62	1800.22	47.74	⊛ 62.22	39.82	52.63	⊛ 62.22	68.41
20140810T123440v9	0.00	44.25	1800.27	28.15	⊛ 42.53	62.33	32.60	⊛ 42.53	142.24
20140810T123440v10	0.00	43.08	1800.27	27.81	⊛ 43.08	60.98	32.20	⊛ 43.08	102.15
20140810T123440v11	0.00	49.20	1800.27	34.93	⊛ 49.20	35.41	37.45	⊛ 49.20	82.23
20140810T123440v12	0.00	62.00	1800.27	47.73	⊛ 62.00	54.01	49.90	⊛ 62.00	138.63
20140810T123443v1	0.00	69.59	1800.16	54.27	⊛ 69.59	4.07	57.31	⊛ 69.59	149.84
20140810T123443v2	0.00	72.15	1800.14	58.45	⊛ 72.15	4.92	63.63	⊛ 72.15	212.61
20140810T123443v3	0.00	77.34	1800.13	65.44	⊛ 77.34	1.90	67.04	⊛ 77.34	141.87
20140810T123443v4	0.00	90.14	1800.16	78.59	⊛ 90.14	3.26	79.84	⊛ 90.14	288.88
20140810T123443v5	0.00	63.25	1800.22	33.55	⊛ 53.05	32.20	48.72	⊛ 53.05	50.42
20140810T123443v6	0.00	64.70	1800.24	36.81	⊛ 55.21	61.71	51.77	⊛ 55.21	61.79
20140810T123443v7	0.00	67.77	1800.21	51.37	⊛ 64.41	34.90	57.78	⊛ 64.41	172.97
20140810T123443v8	0.00	83.70	1800.20	64.35	⊛ 77.21	32.95	74.95	⊛ 77.21	167.49
20140810T123443v9	0.00	59.32	1800.23	30.92	⊛ 45.93	170.79	40.49	⊛ 45.93	98.32
20140810T123443v10	0.00	61.24	1800.23	36.29	⊛ 46.93	32.20	42.61	⊛ 46.93	62.86
20140810T123443v11	0.00	67.43	1800.23	49.09	⊛ 56.40	19.65	52.91	⊛ 56.40	78.85
20140810T123443v12	0.00	83.70	1800.22	61.89	⊛ 69.20	9.25	66.34	⊛ 69.20	193.30

Table 8 – Formulation results for Murray and Chu (2015) instances with $e = 40$

Instance	M&C (2015)			Model 1			Model 2		
	LB^0	Sol.	Time	LB^0	Sol.	Time	LB^0	Sol.	Time
20140810T123437v1	0.00	52.10	1800.15	31.93	⊛ 50.57	666.66	43.17	⊛ 50.57	487.32
20140810T123437v2	0.00	47.31	1800.15	27.82	⊛ 47.31	345.97	38.71	⊛ 47.31	210.99
20140810T123437v3	0.00	53.69	1800.18	30.89	⊛ 53.69	381.99	39.49	⊛ 53.69	140.33
20140810T123437v4	0.00	66.49	1800.15	43.66	⊛ 66.49	471.39	51.4	⊛ 66.49	380.96
20140810T123437v5	0.00	45.84	1800.22	30.67	⊛ 44.84	308.67	38.28	⊛ 44.84	262.35
20140810T123437v6	0.00	47.60	1800.23	27.68	⊛ 43.60	275.60	33.86	⊛ 43.60	356.53
20140810T123437v7	0.00	46.62	1800.24	30.84	⊛ 46.62	250.79	35.13	⊛ 46.62	210.18
20140810T123437v8	0.00	59.78	1800.23	43.58	⊛ 59.42	260.01	46.42	⊛ 59.42	260.01
20140810T123437v9	0.00	42.42	1800.25	30.62	⊛ 42.42	102.37	35.91	⊛ 42.42	180.06
20140810T123437v10	0.00	41.73	1800.28	27.60	⊛ 41.73	108.91	31.3	⊛ 41.73	100.26
20140810T123437v11	0.00	42.90	1800.25	30.81	⊛ 42.90	83.97	33.13	⊛ 42.90	66.69
20140810T123437v12	0.00	55.70	1800.27	43.54	⊛ 55.70	63.07	44.38	⊛ 55.70	77.13
20140810T123440v1	0.00	48.72	1800.15	28.22	⊛ 46.89	329.34	38.12	⊛ 46.89	411.03
20140810T123440v2	0.00	46.42	1800.15	28.20	⊛ 46.42	78.81	36.59	⊛ 46.42	101.78
20140810T123440v3	0.00	53.93	1800.17	34.99	⊛ 53.93	303.25	42.47	⊛ 53.93	114.73
20140810T123440v4	0.00	68.40	1800.14	47.76	⊛ 68.40	335.51	54.58	⊛ 68.40	210.79
20140810T123440v5	0.00	46.59	1800.22	28.17	⊛ 43.53	100.54	34.69	⊛ 43.53	177.74
20140810T123440v6	0.00	44.08	1800.22	27.93	⊛ 43.81	49.68	33.82	⊛ 43.81	153.96
20140810T123440v7	0.00	49.20	1800.22	34.94	⊛ 49.20	37.78	38.86	⊛ 49.20	75.13
20140810T123440v8	0.00	62.27	1800.22	47.74	⊛ 62.00	57.42	51.15	⊛ 62.00	163.61
20140810T123440v9	0.00	44.25	1800.27	28.15	⊛ 42.53	60.81	32.59	⊛ 42.53	98.32
20140810T123440v10	0.00	43.08	1800.27	27.81	⊛ 43.08	82.18	32.19	⊛ 43.08	155.80
20140810T123440v11	0.00	49.20	1800.27	34.93	⊛ 49.20	51.01	37.42	⊛ 49.20	133.91
20140810T123440v12	0.00	62.00	1800.27	47.73	⊛ 62.00	52.78	49.88	⊛ 62.00	140.03
20140810T123443v1	0.00	57.25	1800.16	31.29	⊛ 55.49	680.10	53.73	⊛ 55.49	484.21
20140810T123443v2	0.00	58.05	1800.14	36.29	⊛ 58.05	621.87	55.48	⊛ 58.05	1800.00
20140810T123443v3	0.00	69.17	1800.13	49.20	⊛ 68.43	290.85	62.01	⊛ 68.43	274.07
20140810T123443v4	0.00	82.70	1800.16	62.13	⊛ 82.70	324.36	74.54	⊛ 82.70	210.27
20140810T123443v5	0.00	53.45	1800.22	30.92	⊛ 51.93	1800.77	42.75	⊛ 51.93	1216.07
20140810T123443v6	0.00	52.33	1800.24	36.29	⊛ 52.33	266.32	45.24	⊛ 52.33	826.26
20140810T123443v7	0.00	60.74	1800.21	49.09	⊛ 60.74	59.62	55.25	⊛ 60.74	155.95
20140810T123443v8	0.00	74.69	1800.20	61.89	⊛ 72.97	86.36	67.26	⊛ 72.97	98.11
20140810T123443v9	0.00	47.25	1800.23	30.92	⊛ 45.93	201.58	39.36	⊛ 45.93	127.03
20140810T123443v10	0.00	48.87	1800.23	36.29	⊛ 46.93	39.26	42.15	⊛ 46.93	97.68
20140810T123443v11	0.00	56.40	1800.23	49.09	⊛ 56.40	10.68	52.71	⊛ 56.40	99.04
20140810T123443v12	0.00	69.20	1800.22	61.89	⊛ 69.20	22.40	65.15	⊛ 69.20	67.33

Table 9 presents the results obtained by Model 1 and Model 2 without alterations considering Ponza (2016)'s instances containing from 5 to 10 customers. No formulation was capable of solving larger instances with 50, 100, 150, and 200 customers in a feasible amount of time. For these instances, the formulations presented by Ponza (2016) did not obtain any feasible solution, and the formulation we propose could not be executed due to memory or time limitations. It is thus by no means a coincidence that these large instances have only been addressed with heuristic approaches so far. Therefore, we developed a heuristic to tackle the large instances found in the literature.

Table 9 – Formulation results for Ponza (2016) instances

Instance	Ponza (2016)		Model 1			Model 2		
	Sol.	Time	LB^0	Sol.	Time	LB^0	Sol.	Time
Instance_005.1	4456.83	0.13	3851.22	⊛ 4456.83	0.38	3502.85	⊛ 4456.83	6.13
Instance_005.2	3507.07	0.12	1984.71	⊛ 3507.07	0.07	1810.48	⊛ 3507.07	53.11
Instance_005.3	3275.69	0.14	2979.03	⊛ 3275.69	0.12	3015.17	⊛ 3275.69	61.48
Instance_005.4	5312.47	0.09	3423.66	⊛ 5312.47	0.07	2835.27	⊛ 5312.47	25.98
Instance_005.5	5510.17	0.10	5021.23	⊛ 5510.17	0.05	5317.51	⊛ 5510.17	2.53
Instance_006.1	7080.94	0.25	6064.16	⊛ 7080.94	0.08	6786.80	⊛ 7080.94	6.01
Instance_006.2	6147.96	0.32	5713.98	⊛ 6147.96	0.23	5954.92	⊛ 6147.96	14.66
Instance_006.3	6835.16	0.23	5878.56	⊛ 6835.16	0.08	6786.80	⊛ 6835.16	216.39
Instance_006.4	4402.08	0.32	3424.12	⊛ 4402.08	0.41	3745.90	⊛ 4402.08	216.69
Instance_006.5	5392.08	0.38	4031.53	⊛ 5392.08	0.34	4335.11	⊛ 5392.08	231.16
Instance_007.1	5533.85	3.31	3606.98	⊛ 5533.85	0.48	4913.88	⊛ 5533.85	9.26
Instance_007.2	5342.68	1.79	3258.57	⊛ 5342.68	0.96	5138.58	⊛ 5342.68	9.33
Instance_007.3	7725.89	1.07	6293.13	⊛ 7725.89	0.21	7295.36	⊛ 7725.89	6.50
Instance_007.4	7610.38	1.39	6284.05	⊛ 7610.38	0.16	7243.10	⊛ 7610.38	53.94
Instance_007.5	7010.99	2.10	6211.52	⊛ 7010.99	0.27	6512.04	⊛ 7010.99	6.70
Instance_008.1	6709.02	5.90	4764.75	⊛ 6709.02	1.26	6093.02	⊛ 6709.02	31.63
Instance_008.2	6587.18	10.08	4916.63	⊛ 6587.18	2.06	5762.91	⊛ 6587.18	345.31
Instance_008.3	5780.12	14.68	4133.14	⊛ 5780.12	3.00	4959.20	⊛ 5780.12	201.91
Instance_008.4	6505.12	8.91	3694.07	⊛ 6505.12	1.76	5703.22	⊛ 6505.12	341.65
Instance_008.5	5953.51	15.72	4748.36	⊛ 5953.51	2.48	5467.27	⊛ 5953.51	312.32
Instance_009.1	7338.77	189.38	5773.50	⊛ 7338.77	2.95	6273.84	⊛ 7338.77	298.98
Instance_009.2	6204.63	129.12	4073.60	⊛ 6204.63	3.30	5026.09	⊛ 6204.63	199.41
Instance_009.3	7698.14	87.45	3995.14	⊛ 7698.14	5.16	5463.21	⊛ 7698.14	176.32
Instance_009.4	6817.72	79.71	4281.48	⊛ 6817.72	3.69	4430.43	⊛ 6817.72	101.11
Instance_009.5	7802.67	115.02	5253.94	⊛ 7802.67	4.85	6681.09	⊛ 7802.67	96.32
Instance_010.1	5986.71	1800.15	4502.21	⊛ 5986.71	50.82	4898.25	⊛ 5986.71	297.42
Instance_010.2	6394.39	1800.15	5141.80	⊛ 6394.39	15.81	5648.46	⊛ 6394.39	368.12
Instance_010.3	6310.60	1800.15	3204.10	⊛ 6310.60	21.94	4504.35	⊛ 6310.60	243.99
Instance_010.4	8377.92	752.87	7186.84	⊛ 8377.92	3.58	7362.23	⊛ 8377.92	211.18
Instance_010.5	8934.41	1800.15	5662.50	⊛ 8934.41	10.56	7231.74	⊛ 8934.41	259.74

The formulation proposed by Ponza (2016) obtained linear relaxation lower bounds of value zero for all instances, and so we omitted column LB^0. The formulations previously proposed in the literature are outperformed by the ones we propose, which were capable of producing significantly better lower bounds and, by consequence, proven optimal solutions for all small instances. The runtimes presented by Model 2 are not as competitive as the ones obtained with Model 1. None of our models solved the problem in the root node, however Model 2 obtained a better liner relaxation for 95% of the instances compared to Model 1.

6.2 Heuristic approach

This section is dedicated to our computational experiments for the heuristic approach and their numerical results. More precisely, Section 6.2.1 discuss instance settings, Section 6.2.2 makes an instance analysis and Section 6.2.3 presets a comparison of the GVNS to Schermer et al. (2018) results.

6.2.1 Instance Setting

In Algorithm 2, the computation of the TSP tour can be done in two manners, using a Mixed Integer Programing (MIP) solver or the Nearest Neighborhood heuristic. The MIP solver chosen is the TSP Solver Concorde (Applegate et al., 1996). Concorde is a state of art solver for the TSP; however, the time needed for it to solve a given instance can vary considerably with the number and distribution of vertices. Considering the possible time issue, we determined a maximum time of 5 seconds to Concorde solve the TSP for a given instance. When the limit time is reached, Nearest Neighborhood is executed to obtain the TSP solution. Another parameter in the initial solution is δ, which determines the maximum difference in the number of customers between the truck with more customers and the one with less customers. The variable δ determines where the giant tour is going to be split. Then all configurations are tested considering $t_i = floor(n/2) \pm \{0 \dots \delta\}$. For example, considering $c = 51$ and $M = 2$, $t_0 = |25 \pm \{0 \dots \delta\}|$ and $t_1 = |26 \pm \{\delta \dots 0\}|$.

The Algorithms 1 and 7 need some parameters to be defined for its execution. For the experiments presented in this chapter, these parameters were determined employing the Irace tool (López-Ibáñez et al., 2016), a package coded in R language. Irace is an offline method of automatic configuration of optimization algorithms. Given a set of problem instances and possible values for the parameters, Irace determines an appropriate combination of values for the parameters. Table 10 presents the ranges of values that Irace considered when defining each of the parameters.

Table 10 – Algorithms' parameters.

Parameter	Value
$maxIter$	{10, 20, 30, 40, 50}
t_{max} (seconds)	{20, 50, 100, 300, 600, 1200, 2400}
p_{max}	{20, 50, 100, 200, 300, 500, 800, 1000, 1300, 1500, 1800, 2000}
ρ	{1, 2, 3, 4, 5}
δ	{2, 5, 7, 9, 10, 15, 20, 30}

The parameters defined in the experiments and their respective values are:

- $maxIter = 10$
- $t_{max} = 600$

- $p_{max} = 1800$
- $\rho = 5$
- $\delta = 9$

Two of the five parameters defined by Irace, *maxIter* and ρ, were the limit value. Hence, we could have rerun Irace with smaller options in *maxIter* case and larger values for parameter ρ.

We used five instances from Traveling Salesman Problem Library (TSPLIB) (Reinelt, 1997), *berlin52, ch150, kroA200, rd400* and *att532*. The instances have a range between 52 and 532 vertices, the first vertex in each instance was used as a distribution center (depot), and the remaining ones were customers' locations to be visited by the vehicles. In order to test how the use of drones can impact the cost, we employed a different set of parameters to analyze the solution behavior. The set of parameters and their values are summarized in Table 11. In total, we evaluated the proposed GVNS in 410 literature instances, 81 for each instance varying the aforementioned parameters.

Table 11 – Instances' parameters.

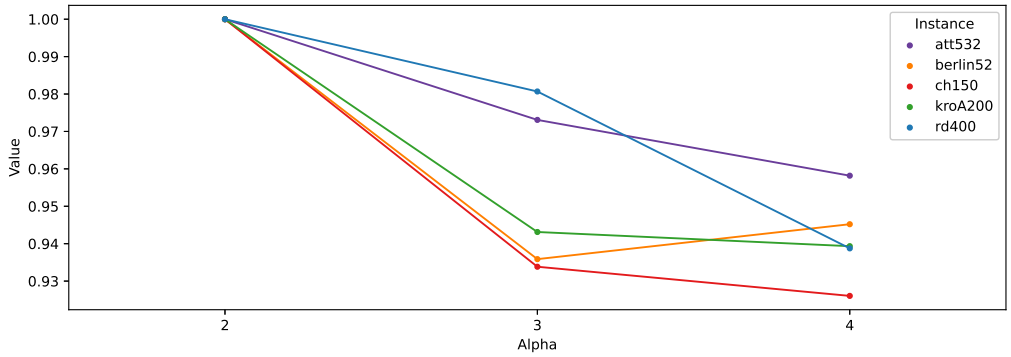
Parameter	Value
$\|M\|$	{1, 2, 3}
$\|D\|$	{1, 2, 3}
α	{2, 3, 4}
β	{0.1, 0.25, 0.5}

Parameter $|M|$ indicates the number of vehicles, $M = 1 \vee 1, 2 \vee 1, 2, 3$, i.e., we have either one, two or three trucks. Parameter $|D|$ indicates the number of drones per truck α determines the relative speed of the drones, and it can assume values {2, 3, 4}, e.g., $\alpha = 2$ indicates that the drones are twice as fast as the truck. Parameter β is an auxiliary variable to set drone endurance, and it can have the following values {0.1, 0.25, 0.5}. The endurance ε is set as $\varepsilon = \beta \cdot \max(D(G(V, E))$ where $\max(D(G(V, E))$ is the maximum value in the distance matrix of the graph.

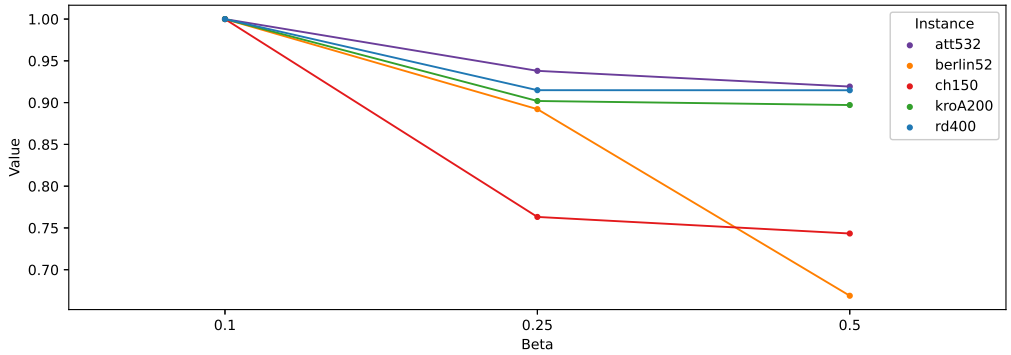
6.2.2 Instance analysis

Examining Figure 21a when α is increased from 2 to 3, a noticeable improvement is visible in most cases. However, increasing the value of α from 3 to 4 does not significantly influence the objective value. This makes us believe that after the drone speed reaches a certain threshold, the vehicles' final arrival time may always be determined by the trucks, which means that speeding up the drones will not result in any further improvements. Different behavior is noted when fixing parameter α, and alternating the value of β. We can observe in Figure 21b an interesting behavior. We can observe a considerable improvement

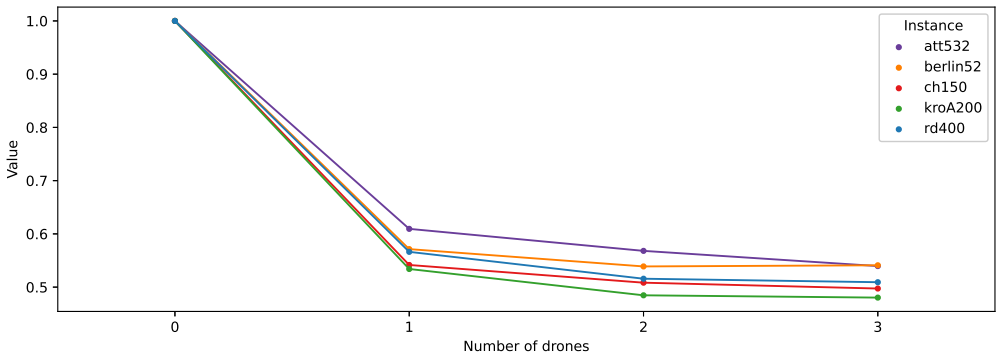
in the objective function, e.g., *berlin52* and *ch150*. However, in the majority of the cases, the objective value shows little sensitivity concerning a change in β.



(a) Fix α parameter.



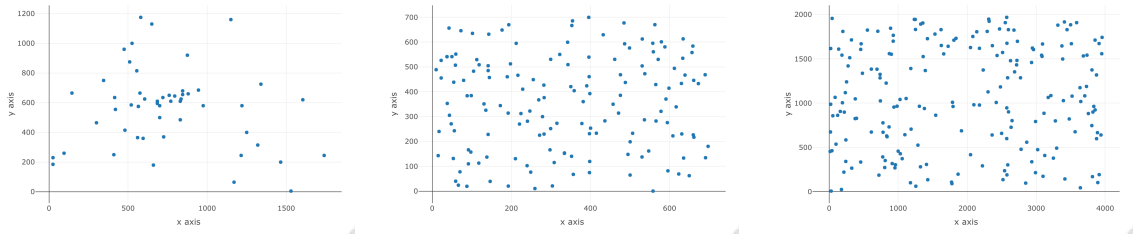
(b) Fix β parameter.



(c) Fix number of drones per truck.

Figure 21 – Fixing parameters.

Now, let us observe Figure 21c that concerns the number of drones per truck. A significant reduction in the completion time occurs, about 35% when adding a single drone to perform the deliveries in synchronization with the truck. When adding the second and third drone, it is possible to achieve another 10% reduction in the delivery time. However, the relative change in the objective value becomes increasingly smaller — a scenario that indicates that it becomes gradually challenging to employ additional drones efficiently.



(a) Instance *berlin52*. (b) Instance *ch150*. (c) Instance *kroA200*.

Figure 22 – Plot of the instances coordinates.

When we put together all information about the number of drones, speed, endurance, and geographic position (Figure 22) of the customers, we can see that when customers are uniformly distributed, then when the endurance is already sufficiently large as in *ch150* and *kroA200*, it may not be possible to improve the objective value by further increasing the endurance. The customers in *berlin52* are more spread in the map area, thus increasing the endurance enables the drone to serve farther customers.

6.2.3 Comparison against the state-of-the-art

Hereafter, the results found by GVNS are compared to the solution values of Variable Neighborhood Decomposition Search with Savings Insertion Heuristic (VNDS-SIH) (Schermer et al., 2018). In the total 400 combinations, 80 of each instance set, varying α, β, K, and M, were tested considering the instances *berlin52, ch150, kroA200, rd400, att532*.

A summary of the average results considering ten runs of the GVNS are presented in Table 12. The complete table with the results can be found in Appendix A. The first column is the best solution in the literature (BKS), followed by the best solution (S^*) found by GVNS. In the third (S) and fourth (S_0) columns are the average of solutions found considering ten executions and the average value of the initial solution. (gap_{S^*,S_0}) is the percentage distance between S_0 and S^*, and ($gap_{S*,BKS}$) is the percentage distance between S^* and BKS. A negative gap indicates an improvement in the solution quality. Considering two value solutions s^1 and s^2, the gap_{s^1,s^2}, is calculated as reported by Equation (6.1).

$$gap = \frac{s^1 - s^2}{s^2 \times 100} \tag{6.1}$$

As we can observe in Table 12, the set of instances derived from *berlin52* achieved the best average result, improving the average solution in 3.93%. Even though the average result of Schermer et al. in instance *kroA200* was 0.1% better, GVNS had the best performance in this set improving 71 of 80 instances, while in *berlin52* 56 instances were improved. The set *ch150* had the worst performance concerning average solution value, presenting a gap of 2.53% compared to Schermer et al.'s algorithm. In terms of number of

Table 12 – Average GVNS results.

Instance	BKS	S_0	S^*	S	gap_{S_0,S^*}	gap_{BKS,S^*}
berlin52	4025.82	4797.04	3867.70	4018.31	-19.37	-3.93
ch100	2757.60	4285.70	2827.42	3028.47	-34.03	2.53
kroA200	13093.65	16800.59	13106.82	13336.25	-21.99	0.10
rd400	7048.70	8311.25	6937.49	7163.97	-16.53	-1.58
att532	43942.39	55490.27	44024.43	44213.18	-20.66	0.19

solutions improved, *ch150* only stays behind *att532*. The former improved 45 instances, and the latter 37 instances. The set *rd400* improved 50 instances achieving a gap of 1.58%.

Now, to provide a statistical base to our previous analysis, we will present the result of a few tests. First, we applied the Shapiro-Wilk Shapiro and Wilk (1965) normality test in each set of instances to the GVNS and VNDS-SIH results. The null hypothesis stating that the method results could be modeled according to a normal distribution was rejected in all sets. The values of the Shapiro-Wilk test can be found in Table 13.

Table 13 – Values of Shapiro-Wilk test

	VNDS-SIH		GVNS	
Instance	W	p-value	W	p-value
berlin52	0.89174	4.937e-06	0.89966	1.053e-05
ch150	0.86796	6.008e-07	0.88551	2.778e-06
kroA200	0.88214	2.05e-06	0.88687	3.145e-06
rd400	0.8575	2.549e-07	0.84319	2.369e-08
att532	0.8478	1.189e-07	0.86051	3.25e-07

Afterwards, we applied the Wilcoxon Signed-Rank (WSR) (Rey and Neuhäuser, 2011; Wilcoxon, 1945) test, which is a non-parametric statistical hypothesis test to compare two paired samples. WSR was employed to verify if exits significant difference between the results. Observing the results of *berlin52*, WSR test showed that there is significant difference between the GVNS and VNDS-SIH presenting the values (V = 928, p-value = 2.84e-05). The same behavior can be observed in instance *kroA200* with the results (V = 243, p-value = 1.265e-11). Therefore, we can confirm that GVNS is better than VNDS-SIH for those instances since p-value < 0.05. However, for the other three sets there is not significant difference between the results *ch150* (V = 2282, p-value = 0.9983), *rd400* (V = 1397, p-value = 0.1078), *att532* (V = 1706, p-value = 0.5857), then we cannot affirm nothing about GVNS behavior in relation to VNDS-SIH.

7 Conclusions and open perspectives

In this work, we have studied Traveling Salesman Problems with Drones, which consist of a combination of a truck and drone in a last-mile delivery setting. Our main contributions were proposing solution approaches able to improve the results of a relatively new problem in the literature.

We introduced two exact approaches to the Flying Sidekick Traveling Salesman Problem (FSTSP) that were capable of finding optimal solutions to all 102 instances proposed by Ponza (2016) and Murray and Chu (2015). The compact formulation most significant contribution is to solve the problem in an extremely low time. The runtime for Ponza (2016)'s instance presented an average of 5 seconds and Murray and Chu (2015)'s instances when $e = 20$, the runtime was of 57 seconds and $e = 40$ it was 257 seconds. The formulation with an exponential number of variables presented a higher execution time. On the other hand, its linear relaxation was the best one compared to the other formulations and its total number of generated variables is considerably smaller.

We addressed another problem, the Multiple Traveling Salesman Problem with Drones (mTSPD). We proposed a fast heuristic based on General Variable Neighborhood Search (GVNS) and an extensive computational study that considers the influence of employing multiple vehicles, multiple drones, and several different drone parameters such as the relative velocity and flight endurance on the objective value in a given instance. Our computational study shows that meaningful savings in terms of time are possible with the combination of trucks and drones in last-mile delivery.

A new modality of parcel distribution is rising from the increasing development of drones and the effort of companies to perform deliveries faster at a reduced cost. Thus, this work has plenty to contribute demonstrating that collaborative work of truck and drone can drastically decrease delivery times.

There are many possibilities for extending this work. With respect to the branch-and-price formulation, one could investigate better ways to solve the pricing problem and explore others branch approaches in order to address large instances of the problem. For the mTSPD, it would be interesting to incorporate additional constraints to the problem, such as capacity constraints and time-windows. Further, an interesting possibility it is to study a robust approach for delivery with drones considering weather changing and others uncertainties.

Bibliography

N. Agatz, P. Bouman, and M. Schmidt. Optimization approaches for the traveling salesman problem with drone. *Transportation Science*, 52(4):965–981, 2018. doi: 10.1287/trsc.2017.0791.

Amazon. Amazon's Best of Prime 2017 Reveals the Year's Biggest Trends —More than 5 Billion Items Shipped with Prime in 2017 | Business Wire, 2018. URL https://www.businesswire.com/news/home/20180102005390/en/.

D. Applegate, E. R. Bixby, V. Chvátal, and W. J. Cook. Concorde home, 1996. URL http://www.math.uwaterloo.ca/tsp/concorde.html.

D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, Princeton, NJ, USA, 2007. ISBN 0691129932, 9780691129938.

G. Battsengel, S. Geetha, and J. Jeon. Analysis of Technological Trends and Technological Portfolio of Unmanned Aerial Vehicle. *Journal of Open Innovation: Technology, Market, and Complexity*, 6(3):1–14, July 2020. doi: 10.3390/joitmc6030048.

J.E. Beasley. Route first—Cluster second methods for vehicle routing. *Omega*, 11(4):403–408, jan 1983. ISSN 0305-0483. doi: 10.1016/0305-0483(83)90033-6.

T. Bektas. The multiple traveling salesman problem: An overview of formulations and solution procedures. *Omega*, 34(3):209–219, 2006. ISSN 03050483. doi: 10.1016/j.omega.2004.10.004.

R. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90, apr 1958. ISSN 0033-569X. doi: 10.1090/qam/102435.

M. Bellmore and G. L. Nemhauser. The Traveling Salesman Problem: A Survey. *Operations Research*, 16(3):538–558, jun 1968. ISSN 0030-364X. doi: 10.1287/opre.16.3.538.

N. Boone, A. Sathyan, and K. Cohen. Enhanced approaches to solving the multiple traveling salesman problem. *American Institute of Aeronautics and Astronautics*, 2015. doi: 10.2514/6.2015-0889.

P. Bouman, N. Agatz, and M. Schmidt. Dynamic programming approaches for the traveling salesman problem with drone. *SSRN Electronic Journal*, sep 2017. ISSN 1556-5068. doi: 10.2139/ssrn.3035323.

N. Boysen, S. Schwerdfeger, and F. Weidinger. Scheduling last-mile deliveries with truck-based autonomous robots. *European Journal of Operational Research*, 271(3):1085–1099, 2018. ISSN 03772217. doi: 10.1016/j.ejor.2018.05.058.

T. Carr, E. Dumitrescu, K. Hutzler, R. Kumar, A. Martin, C. Mazuera, and C. Johnson. Can your supply chain deliver what your consumer wants?, 2021. URL https://www.mckinsey.com/business-functions/operations/our-insights/can-your-supply-chain-deliver-what-your-consumer-wants.

M. C. Chase and R. Ritwik. The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones. *Transportation Research Part C: Emerging Technologies*, 110:368–398, 2020. ISSN 0968-090X. doi: 10.1016/j.trc.2019.11.003.

C. Cheng, Y. Adulyasak, and L.M. Rousseau. Formulations and Exact Algorithms for Drone Routing Problem. Technical Report July, Centre interuniversitaire de recherche sur les reseaux d'entreprise, la logistique et le transport (CIRRELT), 2018.

R. Daknama and E. Kraus. Vehicle routing with drones. *ArXiv*, abs/1705.06431, 2017.

I. Dayarian, M. Savelsbergh, and J. Clarke. Same-day delivery with drone resupply. *Transp. Sci.*, 54:229–249, 2020.

M. Dell'Amico, R. Montemanni, and S. Novellani. Matheuristic algorithms for the parallel drone scheduling traveling salesman problem. *Annals of Operations Research*, 289: 211–226, 2020. ISSN 15729338. doi: 10.1007/s10479-020-03562-3.

E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, dec 1959. ISSN 0029-599X. doi: 10.1007/BF01386390.

K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski. Vehicle routing problems for drone delivery. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(1): 70–85, 2017. doi: 10.1109/TSMC.2016.2582745.

M. Drexl. Synchronization in vehicle routing—a survey of vrps with multiple synchronization constraints. *Transportation Science*, 46(3):297–316, August 2012. ISSN 1526-5447. doi: 10.1287/trsc.1110.0400.

C. Fikar, M. Gronalt, and P. Hirsch. A decision support system for coordinated disaster relief distribution. *Expert Systems with Applications*, 57:104–116, 2016. ISSN 09574174. doi: 10.1016/j.eswa.2016.03.039.

M. L. Fisher and R. Jaikumar. A generalized assignment heuristic for vehicle routing. *Networks*, 11(2):109–124, 1981. ISSN 00283045. doi: 10.1002/net.3230110205.

L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 2015. ISBN 9781400875184. doi: 10.1515/9781400875184.

J. C. Freitas and P. H. V. Penna. A variable neighborhood search for flying sidekick traveling salesman problem. *International Transactions in Operational Research*, 27(1): 267–290, 2020. doi: 10.1111/itor.12671.

J. C. Freitas, P. H. V. Penna, and H. Gambini. Truck and drone collaboratively delivery for green logistics in smart city. *L Brazilian Symposium on Operational Research*, 2018.

J. C. Freitas, P. H. V Penna, and T. A. M. Toffolo. Exact and heuristic approaches to drone delivery problems, 2021.

A. Garside and N. Laili. A cluster-first route-second heuristic approach to solve the multi-trip periodic vehicle routing problem. *Jurnal Teknik Industri*, 20:68, 08 2019. doi: 10.22219/JTIUMM.Vol20.No2.68-77.

B. E. Gillett and L. R. Miller. A heuristic algorithm for the vehicle-dispatch problem. *Operations Research*, 22(2):340–349, 1974. doi: 10.1287/opre.22.2.340.

W. Glauser. Blood-delivering drones saving lives in Africa and maybe soon in Canada. *CMAJ : Canadian Medical Association journal = journal de l'Association medicale canadienne*, 190(3):E88–E89, jan 2018. ISSN 1488-2329. doi: 10.1503/cmaj.109-5541.

P. L. Gonzalez-R, D. Canca, J. L. Andrade-Pineda, M. Calle, and J. M. Leon-Blanco. Truck-drone team logistics: A heuristic approach to multi-drop route planning. *Transportation Research Part C: Emerging Technologies*, 114:657–680, 2020. ISSN 0968-090X. doi: 10.1016/j.trc.2020.02.030.

Q. M. Ha, Y. Deville, Q. D. Pham, and M. H. Hà. On the min-cost traveling salesman problem with drone. *Transportation Research Part C: Emerging Technologies*, 86: 597–621, 2018. ISSN 0968-090X. doi: 10.1016/j.trc.2017.11.015.

A. M. Ham. Integrated scheduling of m-truck, m-drone, and m-depot constrained by time-window, drop-pickup, and m-visit using constraint programming. *Transportation Research Part C: Emerging Technologies*, 91(July 2017):1–14, 2018. ISSN 0968090X. doi: 10.1016/j.trc.2018.03.025.

M. Held and R. M. Karp. The Traveling-Salesman Problem and Minimum Spanning Trees. *Operations Research*, 18(6):1138–1162, dec 1970. ISSN 0030-364X. doi: 10.1287/opre.18.6.1138.

R. H. F. R. Kramer, A. Subramanian, and P. H. V. Penna. Problema de roteamento de veículos assimétrico com frota heterogênea limitada: um estudo de caso em uma indústria de bebidas. *Gestão & Produção*, 23(1):165–176, 2016. ISSN 0104-530X. doi: 10.1590/0104-530X1442-14.

G. Laporte. The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2):231–247, jun 1992. ISSN 03772217. doi: 10.1016/0377-2217(92)90138-Y.

T. Leventhal, G. Nemhauser, and L. Trotter. A column generation algorithm for optimal traffic assignment. *Transportation Science*, 7(2):168–176, 1973. doi: 10.1287/trsc.7.2.168.

J. Liu, Z. Guan, J. Shang, and X. Xie. Application of drone in solving last mile parcel delivery. *Journal of Systems Science and Information*, 6:302–319, 09 2018. doi: 10.21078/JSSI-2018-302-18.

S. Ludwig. Drones: A Security Tool, Threat and Challenge | 2018-03-09 | Security Magazine, 2018. URL https://www.securitymagazine.com/articles/88803-drones-a-security-tool-threat-and-challenge.

M. López-Ibáñez, J. Dubois-Lacoste, L. P. Cáceres, M. Birattari, and T. Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016. ISSN 2214-7160. doi: 10.1016/j.orp.2016.09.002.

M. Marinelli, L. Caggiani, M. Ottomanelli, and M. Dell'Orco. En route truck–drone parcel delivery for optimal vehicle routing strategies. *IET Intelligent Transport Systems*, 12(4):253–261, 2018. ISSN 1751-956X. doi: 10.1049/iet-its.2017.0227.

K. McVeigh. Uber for blood: how rwandan delivery robots are saving lives, 2018. URL https://www.theguardian.com/global-development/2018/jan/02/rwanda-scheme-saving-blood-drone.

N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, nov 1997. ISSN 0305-0548. doi: 10.1016/S0305-0548(97)00031-2.

N. Mladenović, R. Todosijević, and D. Urošević. Less is more: Basic variable neighborhood search for minimum differential dispersion problem. *Information Sciences*, 326:160–171, jan 2016. ISSN 0020-0255. doi: 10.1016/J.INS.2015.07.044.

C. C. Murray and A. G. Chu. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies*, 54:86–109, may 2015. ISSN 0968-090X. doi: 10.1016/J.TRC.2015.03.005.

NetworkX. Network analysis in python, 2004-2021. URL https://networkx.org/documentation/stable/reference/algorithms/shortest_paths.html.

C. Nilsson. Heuristics for the traveling salesman problem, 2003.

I.H. Osman. Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problems. *Annals of Operations Research*, 41:421–452, 1993.

P. H. V. Penna, A. Subramanian, and L. S. Ochi. An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics*, 19(2):201–232, 2013. ISSN 1572-9397. doi: 10.1007/s10732-011-9186-y.

P. H. V. Penna, A. Subramanian, L. S. Ochi, T. Vidal, and C. Prins. A hybrid heuristic for a broad class of vehicle routing problems with heterogeneous fleet. *Annals of Operations Research*, Nov 2017. ISSN 1572-9338. doi: 10.1007/s10479-017-2642-9.

S. Poikonen and B. Golden. Multi-visit drone routing problem. *Computers & Operations Research*, 113:104802, 2020. ISSN 0305-0548. doi: 10.1016/j.cor.2019.104802.

S. Poikonen, X. Wang, and B. Golden. The vehicle routing problem with drones: Extended models and connections. *Networks*, 70(1):34–43, aug 2017. ISSN 00283045. doi: 10.1002/net.21746.

A. Ponza. *Optimization of Drone-Assisted Parcel Delivery*. PhD thesis, Universita Degli Studi Di Padova, 2016. URL https://goo.gl/4frcz5.

L. Probst, B. Pedersen, and L. Dakkak-Arnoux. Digital Transformation Monitor Drones in agriculture, 2018. URL https://ec.europa.eu/growth/tools-databases/dem/monitor/sites/default/files/Drones{_}vf.pdf.

L. Di Puglia Pugliese and F. Guerriero. Last-mile deliveries by using drones and classical vehicles. In Antonio Sforza and Claudio Sterle, editors, *Optimization and Decision Science: Methodologies and Applications*, pages 557–565. Springer International Publishing, 2017. ISBN 978-3-319-67308-0.

M. C. Quilter and V. Anderson. A proposed method for determining shrub utilization using (la/ls) imagery. *Journal of Range Management*, 54:378–381, 2001.

G. Reinelt. TSPLIB, 1997. URL http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html.

D. Rey and M. Neuhäuser. *Wilcoxon-Signed-Rank Test*, pages 1658–1659. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-04898-2. doi: 10.1007/978-3-642-04898-2_616.

D. M. Ryan, C. Hjorring, and F. Glover. Extensions of the petal method for vehicle routeing. *Journal of The Operational Research Society - J OPER RES SOC*, 44:289–296, 1993. doi: 10.1038/sj/jors/0440309.

R. G. M. Saleu, L. Deroussi, D. Feillet, N. Grangeon, and A. Quilliot. An iterative two-step heuristic for the parallel drone scheduling traveling salesman problem. *Networks*, jul 2018. ISSN 00283045. doi: 10.1002/net.21846.

M. Savelsbergh. A branch-and-price algorithm for the generalized assignment problem. *Operations Research*, 45(6):831–841, 1997. doi: 10.1287/opre.45.6.831.

D. Schermer, M. Moeini, and O. Wendt. A variable neighborhood search algorithm for solving the vehicle routing problem with drones. 2018.

J. E. Scott and C. Scott. Drone delivery models for healthcare. In *HICSS*, 2017.

S. S. Shapiro and M. B. Wilk. An Analysis of Variance Test for Normality (Complete Samples). *Biometrika*, 52(3/4):591, dec 1965. ISSN 00063444. doi: 10.2307/2333709.

B. D. Song, K. Park, and J. Kim. Persistent UAV delivery logistics: MILP formulation and efficient heuristic. *Computers and Industrial Engineering*, 120(January):418–428, 2018. ISSN 03608352. doi: 10.1016/j.cie.2018.05.013.

M. J. F. Souza, I. M. Coelho, S. Ribas, H. G. Santos, and L. H. C. Merschmann. A hybrid heuristic algorithm for the open-pit-mining operational planning problem. *European Journal of Operational Research*, 207(2):1041–1051, 2010. ISSN 03772217. doi: 10.1016/j.ejor.2010.05.031.

A. Subramanian, L. M. A. Drummond, C. Bentes, L. S. Ochi, and R. Farias. A parallel heuristic for the Vehicle Routing Problem with Simultaneous Pickup and Delivery. *Computers and Operations Research*, 37(11):1899–1911, 2010. ISSN 03050548. doi: 10.1016/j.cor.2009.10.011.

K. Sundar and S. Rathinam. Algorithms for routing an unmanned aerial vehicle in the presence of refueling depots. *IEEE Transactions on Automation Science and Engineering*, 11(1):287–294, 2014. ISSN 15455955. doi: 10.1109/TASE.2013.2279544.

T. A. M Toffolo and H. Gambini. Python-mip, 2021. URL https://python-mip.com.

P. Toth and D. Vigo. *Vehicle Routing: Problems, Methods, and Applications, Second Edition*. Number 18 in MOS-SIAM Series on Optimization. SIAM, 2014. ISBN 9781611973587.

M. W. Ulmer and B. W. Thomas. Same-day delivery with heterogeneous fleets of drones and vehicles. *Networks*, 72(4):475–505, 2018. doi: 10.1002/net.21855.

A. Vacca and H. Onishi. Drones: military weapons, surveillance or mapping tools for environmental monitoring? The need for legal framework is required. *Transportation Research Procedia*, 25:51–62, jan 2017. doi: 10.1016/J.TRPRO.2017.05.209.

X. Wang, S. Poikonen, and B. Golden. The vehicle routing problem with drones: several worst-case results. *Optimization Letters*, 11(4):679–697, 2017. ISSN 18624480. doi: 10.1007/s11590-016-1035-3.

F. Wilcoxon. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80, dec 1945. ISSN 00994987. doi: 10.2307/3001968.

C. Zhang and J. M. Kovacs. The application of small unmanned aerial systems for precision agriculture: a review. *Precision Agriculture*, 13(6):693–712, dec 2012. ISSN 1385-2256. doi: 10.1007/s11119-012-9274-5.

APPENDIX A – Appendix

Table 14 presents the results obtained with GVNS. The first column, 'Instance', indicates the instance name, followed by the parameters '#truck', '# drone', α, and β, respectively. Succeeding is the best solution in the literature (BKS), followed by the best solution ($S*$) found by GVNS. In the third (S) and fourth (S_0) columns are the average of solutions found considering ten executions and the average value of the initial solution. (gap_{S*,S_0}) is the percentage distance between S_0 and $S*$, and ($gap_{S*,BKS}$) is the percentage distance between $S*$ and BKS. A negative gap indicates an improvement in the solution quality. Note that when $\#truck = 1$ and $\#drone = 1$ we are dealing with the FSTSP.

Table 14 – Complete comparison between Schermer (2018) and GVNS.

Instance	# truck	# drone	α	β	BKS	S_0	S^*	S	gap_{S_0,S^*}	gap_{BKS,S^*}
berlin52	1	1	2	0,1	7396,45	7544,37	7126,17	7393,39	-5,54	-3,65
berlin52	1	1	2	0,25	6572,30	7544,37	6335,07	6573,98	-16,03	-3,61
berlin52	1	1	2	0,50	5329,15	7544,37	5377,57	6038,25	-28,72	0,91
berlin52	1	1	3	0,1	7354,01	7544,37	7038,29	7249,25	-6,71	-4,29
berlin52	1	1	3	0,25	6238,94	7544,37	5927,60	6182,96	-21,43	-4,99
berlin52	1	1	3	0,50	4765,06	7544,37	4769,51	5281,48	-36,78	0,09
berlin52	1	1	4	0,1	7351,69	7544,37	7126,14	7362,97	-5,54	-3,07
berlin52	1	1	4	0,25	6223,51	7544,37	6150,63	6482,78	-18,47	-1,17
berlin52	1	1	4	0,50	4716,44	7544,37	4693,38	4928,48	-37,79	-0,49
berlin52	1	2	2	0,1	7337,80	7544,37	7241,79	7393,85	-4,01	-1,31
berlin52	1	2	2	0,25	6273,67	7544,37	6157,12	6482,48	-18,39	-1,86
berlin52	1	2	2	0,50	4960,29	7544,37	4733,25	5983,89	-37,26	-4,58
berlin52	1	2	3	0,1	7335,49	7544,37	6983,41	7294,46	-7,44	-4,80
berlin52	1	2	3	0,25	5910,98	7544,37	5558,87	5982,43	-26,32	-5,96
berlin52	1	2	3	0,50	4109,14	7544,37	3952,04	4184,74	-47,62	-3,82
berlin52	1	2	4	0,1	7373,30	7544,37	6982,59	7094,27	-7,45	-5,30
berlin52	1	2	4	0,25	6569,22	7544,37	6401,72	6538,25	-15,15	-2,55
berlin52	1	2	4	0,50	4200,97	7544,37	3978,71	4284,49	-47,26	-5,29
berlin52	1	3	2	0,1	7408,02	7544,37	7208,65	7533,94	-4,45	-2,69
berlin52	1	3	2	0,25	6304,54	7544,37	6181,50	6736,49	-18,06	-1,95
berlin52	1	3	2	0,50	4730,33	7544,37	4773,26	5363,95	-36,73	0,91
berlin52	1	3	3	0,1	7323,14	7544,37	6997,85	7362,84	-7,24	-4,44
berlin52	1	3	3	0,25	6011,30	7544,37	5994,81	6937,74	-20,54	-0,27
berlin52	1	3	3	0,50	4097,56	7544,37	3867,70	4762,83	-48,73	-5,61
berlin52	1	3	4	0,1	7307,71	7544,37	7031,83	7528,28	-6,79	-3,78
berlin52	1	3	4	0,25	6127,05	7544,37	5930,75	6284,78	-21,39	-3,20
berlin52	1	3	4	0,50	3812,04	7544,37	3666,64	3935,83	-51,40	-3,81
berlin52	2	1	2	0,1	4308,49	4797,04	4614,51	4783,78	-3,81	7,10

Table 14 – Complete comparison between Schermer (2018) and GVNS.

Instance	# truck	# drone	α	β	BKS	S_0	S^*	S	gap_{S_0,S^*}	gap_{BKS,S^*}
berlin52	2	1	2	0,25	4193,66	4797,04	4099,46	4284,98	-14,54	-2,25
berlin52	2	1	2	0,50	3417,64	4797,04	3399,99	4284,83	-29,12	-0,52
berlin52	2	1	3	0,1	4451,15	4797,04	4274,18	4762,13	-10,90	-3,98
berlin52	2	1	3	0,25	4303,63	4797,04	4486,29	4762,84	-6,48	4,24
berlin52	2	1	3	0,50	2968,02	4797,04	3093,11	3652,12	-35,52	4,21
berlin52	2	1	4	0,1	4451,15	4797,04	4236,02	4982,04	-11,70	-4,83
berlin52	2	1	4	0,25	4303,63	4797,04	4163,03	4492,14	-13,22	-3,27
berlin52	2	1	4	0,50	2968,02	4797,04	3197,66	3376,97	-33,34	7,74
berlin52	2	2	2	0,1	4407,87	4797,04	4723,32	5193,24	-1,54	7,16
berlin52	2	2	2	0,25	3507,30	4797,04	3381,85	3592,27	-29,50	-3,58
berlin52	2	2	2	0,50	3176,93	4797,04	3056,53	3282,25	-36,28	-3,79
berlin52	2	2	3	0,1	4318,65	4797,04	4156,65	4492,27	-13,35	-3,75
berlin52	2	2	3	0,25	4025,82	4797,04	3914,01	4018,31	-18,41	-2,78
berlin52	2	2	3	0,50	2855,40	4797,04	2655,42	2871,32	-44,64	-7,00
berlin52	2	2	4	0,1	4171,57	4797,04	4099,38	4482,37	-14,54	-1,73
berlin52	2	2	4	0,25	4185,26	4797,04	3993,56	4392,38	-16,75	-4,58
berlin52	2	2	4	0,50	2791,35	4797,04	2688,63	2873,14	-43,95	-3,68
berlin52	2	3	2	0,1	4425,53	4797,04	4653,29	4782,27	-3,00	5,15
berlin52	2	3	2	0,25	4173,78	4797,04	3980,93	3998,48	-17,01	-4,62
berlin52	2	3	2	0,50	2915,02	4797,04	3083,20	3183,25	-35,73	5,77
berlin52	2	3	3	0,1	4332,78	4797,04	4110,43	4284,21	-14,31	-5,13
berlin52	2	3	3	0,25	3705,61	4797,04	3674,60	3982,38	-23,40	-0,84
berlin52	2	3	3	0,50	2950,36	4797,04	3091,28	3284,28	-35,56	4,78
berlin52	2	3	4	0,1	4156,11	4797,04	4129,85	4482,48	-13,91	-0,63
berlin52	2	3	4	0,25	3811,61	4797,04	3805,04	3981,83	-20,68	-0,17
berlin52	2	3	4	0,50	2950,36	4797,04	3107,56	3284,42	-35,22	5,33
berlin52	3	1	2	0,1	3267,03	3485,87	3143,84	3478,01	-9,81	-3,77
berlin52	3	1	2	0,25	3242,72	3485,87	3210,73	3382,24	-7,89	-0,99
berlin52	3	1	2	0,50	2847,38	3485,87	2804,30	2898,14	-19,55	-1,51
berlin52	3	1	3	0,1	3120,21	3485,87	2976,05	3098,34	-14,63	-4,62
berlin52	3	1	3	0,25	3209,14	3485,87	3203,23	3374,23	-8,11	-0,18
berlin52	3	1	3	0,50	2473,46	3485,87	2423,77	2863,14	-30,47	-2,01
berlin52	3	1	4	0,1	3228,33	3485,87	3340,70	3482,24	-4,16	3,48
berlin52	3	1	4	0,25	2974,04	3485,87	2918,82	3018,32	-16,27	-1,86
berlin52	3	1	4	0,50	2142,73	3485,87	2259,89	2472,49	-35,17	5,47
berlin52	3	2	2	0,1	3311,17	3485,87	3432,54	3482,15	-1,53	3,67
berlin52	3	2	2	0,25	2883,20	3485,87	2905,09	3183,48	-16,66	0,76
berlin52	3	2	2	0,50	2284,75	3485,87	2416,42	2501,48	-30,68	5,76
berlin52	3	2	3	0,1	3109,98	3485,87	3014,73	3183,27	-13,52	-3,06
berlin52	3	2	3	0,25	3045,05	3485,87	2965,58	3087,24	-14,93	-2,61
berlin52	3	2	3	0,50	2061,48	3485,87	2092,47	2274,38	-39,97	1,50
berlin52	3	2	4	0,1	3060,72	3485,87	3181,77	3382,46	-8,72	3,95
berlin52	3	2	4	0,25	2964,76	3485,87	2994,58	3274,24	-14,09	1,01

Table 14 – Complete comparison between Schermer (2018) and GVNS.

Instance	# truck	# drone	α	β	BKS	S_0	S^*	S	gap_{S_0,S^*}	gap_{BKS,S^*}
berlin52	3	2	4	0,50	2083,87	3485,87	2103,78	2273,21	-39,65	0,96
berlin52	3	3	2	0,1	3284,94	3485,87	3244,46	3402,27	-6,93	-1,23
berlin52	3	3	2	0,25	2984,28	3485,87	3114,94	3372,29	-10,64	4,38
berlin52	3	3	2	0,50	2258,20	3485,87	2348,70	2572,82	-32,62	4,01
berlin52	3	3	3	0,1	3166,59	3485,87	3036,33	3272,93	-12,90	-4,11
berlin52	3	3	3	0,25	2984,28	3485,87	2860,90	3019,33	-17,93	-4,13
berlin52	3	3	3	0,50	2095,07	3485,87	2142,26	2297,33	-38,54	2,25
berlin52	3	3	4	0,1	3348,91	3485,87	3234,23	3382,25	-7,22	-3,42
berlin52	3	3	4	0,25	2984,28	3485,87	2866,36	2918,33	-17,77	-3,95
berlin52	3	3	4	0,50	1957,53	3485,87	2067,82	2109,33	-40,68	5,63
ch150	1	1	2	0,1	6225,31	6532,28	5703,93	5972,27	-12,68	-8,38
ch150	1	1	2	0,25	5117,54	6532,28	5056,93	5182,25	-22,59	-1,18
ch150	1	1	2	0,50	5120,89	6532,28	5164,81	5282,24	-20,93	0,86
ch150	1	1	3	0,1	6440,97	6532,28	5909,43	6093,47	-9,53	-8,25
ch150	1	1	3	0,25	4675,51	6532,28	4693,85	4772,14	-28,14	0,39
ch150	1	1	3	0,50	4677,52	6532,28	4771,61	4872,98	-26,95	2,01
ch150	1	1	4	0,1	6105,42	6532,28	5881,30	6032,48	-9,97	-3,67
ch150	1	1	4	0,25	4575,72	6532,28	4587,25	4692,13	-29,78	0,25
ch150	1	1	4	0,50	4582,42	6532,28	4593,75	4692,13	-29,68	0,25
ch150	1	2	2	0,1	6161,01	6532,28	5808,86	5872,43	-11,07	-5,72
ch150	1	2	2	0,25	4885,81	6532,28	4816,70	5049,38	-26,26	-1,41
ch150	1	2	2	0,50	4640,68	6532,28	4517,06	4698,33	-30,85	-2,66
ch150	1	2	3	0,1	6195,17	6532,28	5650,68	5772,72	-13,50	-8,79
ch150	1	2	3	0,25	4033,22	6532,28	3958,63	4093,36	-39,40	-1,85
ch150	1	2	3	0,50	4012,46	6532,28	3995,05	4142,25	-38,84	-0,43
ch150	1	2	4	0,1	6267,50	6532,28	5715,02	5819,32	-12,51	-8,81
ch150	1	2	4	0,25	4060,68	6532,28	4065,59	4184,28	-37,76	0,12
ch150	1	2	4	0,50	3881,19	6532,28	3722,80	4091,38	-43,01	-4,08
ch150	1	3	2	0,1	6322,42	6532,28	5927,42	6293,27	-9,26	-6,25
ch150	1	3	2	0,25	4272,99	6532,28	4100,40	4282,92	-37,23	-4,04
ch150	1	3	2	0,50	4259,60	6532,28	4169,22	4283,92	-36,18	-2,12
ch150	1	3	3	0,1	6154,98	6532,28	5775,85	5928,38	-11,58	-6,16
ch150	1	3	3	0,25	4192,62	6532,28	3923,54	4293,18	-39,94	-6,42
ch150	1	3	3	0,50	4038,58	6532,28	3940,77	4172,83	-39,67	-2,42
ch150	1	3	4	0,1	6248,75	6532,28	5880,93	6082,87	-9,97	-5,89
ch150	1	3	4	0,25	3938,12	6532,28	3620,82	3883,28	-44,57	-8,06
ch150	1	3	4	0,50	3877,84	6532,28	3761,66	3928,24	-42,41	-3,00
ch150	2	1	2	0,1	3739,90	4285,70	3751,30	3982,18	-12,47	0,30
ch150	2	1	2	0,25	3171,28	4285,70	3193,97	3382,28	-25,47	0,72
ch150	2	1	2	0,50	3177,42	4285,70	3194,66	3381,98	-25,46	0,54
ch150	2	1	3	0,1	3691,59	4285,70	3719,07	3918,24	-13,22	0,74
ch150	2	1	3	0,25	2657,88	4285,70	2737,07	2915,25	-36,13	2,98
ch150	2	1	3	0,50	2672,45	4285,70	2500,88	2698,35	-41,65	-6,42

Table 14 – Complete comparison between Schermer (2018) and GVNS.

Instance	# truck	# drone	α	β	BKS	S_0	S^*	S	gap_{S_0,S^*}	gap_{BKS,S^*}
ch150	2	1	4	0,1	3691,59	4285,70	3419,92	3672,24	-20,20	-7,36
ch150	2	1	4	0,25	2657,88	4285,70	2507,92	2872,83	-41,48	-5,64
ch150	2	1	4	0,50	2672,45	4285,70	2534,45	2872,25	-40,86	-5,16
ch150	2	2	2	0,1	3697,72	4285,70	3789,71	3972,25	-11,57	2,49
ch150	2	2	2	0,25	2693,92	4285,70	2778,61	2921,47	-35,17	3,14
ch150	2	2	2	0,50	2594,62	4285,70	2648,56	2873,70	-38,20	2,08
ch150	2	2	3	0,1	3606,47	4285,70	3427,86	3652,25	-20,02	-4,95
ch150	2	2	3	0,25	2692,77	4285,70	2725,17	3018,32	-36,41	1,20
ch150	2	2	3	0,50	2514,86	4285,70	2419,24	2651,39	-43,55	-3,80
ch150	2	2	4	0,1	3561,99	4285,70	3686,93	3872,28	-13,97	3,51
ch150	2	2	4	0,25	2629,51	4285,70	2815,98	3081,38	-34,29	7,09
ch150	2	2	4	0,50	2634,11	4285,70	2527,88	2762,18	-41,02	-4,03
ch150	2	3	2	0,1	3592,66	4285,70	3619,89	3871,01	-15,54	0,76
ch150	2	3	2	0,25	2856,49	4285,70	2944,30	3102,76	-31,30	3,07
ch150	2	3	2	0,50	2695,46	4285,70	2888,47	3028,47	-32,60	7,16
ch150	2	3	3	0,1	3703,86	4285,70	3714,30	3871,35	-13,33	0,28
ch150	2	3	3	0,25	2553,59	4285,70	2481,58	2591,37	-42,10	-2,82
ch150	2	3	3	0,50	2484,57	4285,70	2227,49	2593,81	-48,03	-10,35
ch150	2	3	4	0,1	3585,00	4285,70	3457,70	3692,25	-19,32	-3,55
ch150	2	3	4	0,25	2561,26	4285,70	2608,63	2698,10	-39,13	1,85
ch150	2	3	4	0,50	2576,60	4285,70	2506,44	2601,72	-41,52	-2,72
ch150	3	1	2	0,1	2736,90	3047,88	2827,42	2902,48	-7,23	3,31
ch150	3	1	2	0,25	2356,52	3047,88	2366,26	2514,38	-22,36	0,41
ch150	3	1	2	0,50	2254,11	3047,88	2182,68	2381,39	-28,39	-3,17
ch150	3	1	3	0,1	2559,13	3047,88	2692,13	2798,38	-11,67	5,20
ch150	3	1	3	0,25	2249,14	3047,88	2197,13	2291,47	-27,91	-2,31
ch150	3	1	3	0,50	2188,41	3047,88	2282,48	2482,47	-25,11	4,30
ch150	3	1	4	0,1	2734,69	3047,88	2825,75	3018,33	-7,29	3,33
ch150	3	1	4	0,25	2168,26	3047,88	2145,80	2282,82	-29,60	-1,04
ch150	3	1	4	0,50	2169,64	3047,88	2080,79	2301,47	-31,73	-4,10
ch150	3	2	2	0,1	2757,60	3047,88	2680,43	2718,33	-12,06	-2,80
ch150	3	2	2	0,25	2169,64	3047,88	2242,98	2391,44	-26,41	3,38
ch150	3	2	2	0,50	2042,94	3047,88	2046,09	2103,44	-32,87	0,15
ch150	3	2	3	0,1	2746,01	3047,88	2682,71	2713,85	-11,98	-2,31
ch150	3	2	3	0,25	2188,41	3047,88	2023,72	2293,15	-33,60	-7,53
ch150	3	2	3	0,50	1992,15	3047,88	2086,11	2097,64	-31,56	4,72
ch150	3	2	4	0,1	2572,93	3047,88	2594,30	2663,25	-14,88	0,83
ch150	3	2	4	0,25	2123,54	3047,88	2149,37	2291,46	-29,48	1,22
ch150	3	2	4	0,50	1997,12	3047,88	2079,94	2197,36	-31,76	4,15
ch150	3	3	2	0,1	2785,20	3047,88	2682,71	2763,37	-11,98	-3,68
ch150	3	3	2	0,25	2158,60	3047,88	2107,48	2474,43	-30,85	-2,37
ch150	3	3	2	0,50	2111,68	3047,88	2212,05	2298,44	-27,42	4,75
ch150	3	3	3	0,1	2680,31	3047,88	2582,71	2683,43	-15,26	-3,64

Table 14 – Complete comparison between Schermer (2018) and GVNS.

Instance	# truck	# drone	α	β	BKS	S_0	S^*	S	gap_{S_0,S^*}	gap_{BKS,S^*}
ch150	3	3	3	0,25	2114,44	3047,88	2082,71	2193,75	-31,67	-1,50
ch150	3	3	3	0,50	1923,97	3047,88	1907,91	2018,26	-37,40	-0,83
ch150	3	3	4	0,1	2649,95	3047,88	2468,01	2582,46	-19,03	-6,87
ch150	3	3	4	0,25	2136,52	3047,88	2158,72	2284,43	-29,17	1,04
ch150	3	3	4	0,50	1965,38	3047,88	1968,56	2282,46	-35,41	0,16
kroA200	1	1	2	0.1	26232.18	29369.41	24407.30	24598.49	-16.90	-6.96
kroA200	1	1	2	0.25	24953.87	29369.41	23130.20	23915.38	-21.24	-7.31
kroA200	1	1	2	0.50	24332.64	29369.41	23523.20	23981.48	-19.91	-3.33
kroA200	1	1	3	0.1	24532.75	29369.41	22843.10	23084.97	-22.22	-6.89
kroA200	1	1	3	0.25	23762.18	29369.41	22791.50	22981.03	-22.40	-4.08
kroA200	1	1	3	0.50	23176.78	29369.41	22892.10	23284.47	-22.05	-1.23
kroA200	1	1	4	0.1	24461.06	29369.41	22359.10	23815.24	-23.87	-8.59
kroA200	1	1	4	0.25	23445.59	29369.41	23636.20	24014.84	-19.52	0.81
kroA200	1	1	4	0.50	22489.84	29369.41	22389.84	23814.43	-23.76	-0.44
kroA200	1	2	2	0.1	23317.16	29369.41	22397.80	22981.94	-23.74	-3.94
kroA200	1	2	2	0.25	21982.10	29369.41	21449.50	21983.01	-26.97	-2.42
kroA200	1	2	2	0.50	20240.86	29369.41	19990.20	20984.41	-31.94	-1.24
kroA200	1	2	3	0.1	21121.94	29369.41	20188.00	21349.28	-31.26	-4.42
kroA200	1	2	3	0.25	18547.40	29369.41	18309.30	20194.48	-37.66	-1.28
kroA200	1	2	3	0.50	18714.66	29369.41	18210.90	18472.83	-37.99	-2.69
kroA200	1	2	4	0.1	22221.04	29369.41	20019.30	2132.43	-31.84	-9.91
kroA200	1	2	4	0.25	20488.76	29369.41	19211.30	19598.27	-34.59	-6.23
kroA200	1	2	4	0.50	19353.81	29369.41	18938.10	19284.92	-35.52	-2.15
kroA200	1	3	2	0.1	23765.16	29369.41	22568.60	23841.33	-23.16	-5.03
kroA200	1	3	2	0.25	22280.77	29369.41	20738.50	21846.47	-29.39	-6.92
kroA200	1	3	2	0.50	22122.48	29369.41	21017.20	22974.43	-28.44	-5.00
kroA200	1	3	3	0.1	20700.81	29369.41	19945.50	21352.64	-32.09	-3.65
kroA200	1	3	3	0.25	20094.51	29369.41	19454.80	20184.63	-33.76	-3.18
kroA200	1	3	3	0.50	18813.22	29369.41	18214.20	19218.25	-37.98	-3.18
kroA200	1	3	4	0.1	20578.36	29369.41	20336.40	21841.43	-30.76	-1.18
kroA200	1	3	4	0.25	18726.60	29369.41	18820.00	20184.92	-35.92	0.50
kroA200	1	3	4	0.50	18278.60	29369.41	18352.30	19381.85	-37.51	0.40
kroA200	2	1	2	0.1	15135.47	16800.59	15068.80	15831.91	-10.31	-0.44
kroA200	2	1	2	0.25	12872.68	16800.59	12280.70	12415.29	-26.90	-4.60
kroA200	2	1	2	0.50	14241.32	16800.59	14162.50	14981.23	-15.70	-0.55
kroA200	2	1	3	0.1	14758.62	16800.59	13867.42	14921.49	-17.46	-6.04
kroA200	2	1	3	0.25	13093.65	16800.59	12749.84	13525.28	-24.11	-2.63
kroA200	2	1	3	0.50	13449.94	16800.59	13274.37	13982.82	-20.99	-1.31
kroA200	2	1	4	0.1	14758.62	16800.59	14364.25	14425.25	-14.50	-2.67
kroA200	2	1	4	0.25	13093.65	16800.59	13145.34	13272.43	-21.76	0.39
kroA200	2	1	4	0.50	13449.94	16800.59	13189.08	13674.28	-21.50	-1.94
kroA200	2	2	2	0.1	14578.76	16800.59	13983.33	14324.36	-16.77	-4.08
kroA200	2	2	2	0.25	13002.87	16800.59	13106.82	13336.25	-21.99	0.80

Table 14 – Complete comparison between Schermer (2018) and GVNS.

Instance	# truck	# drone	α	β	BKS	S_0	S^*	S	gap_{S_0,S^*}	gap_{BKS,S^*}
kroA200	2	2	2	0.50	12896.67	16800.59	12132.94	12974.84	-27.78	-5.92
kroA200	2	2	3	0.1	14544.51	16800.59	13897.32	14274.87	-17.28	-4.45
kroA200	2	2	3	0.25	12161.82	16800.59	12023.32	12763.91	-28.44	-1.14
kroA200	2	2	3	0.50	12456.44	16800.59	12023.32	12574.25	-28.44	-3.48
kroA200	2	2	4	0.1	14813.44	16800.59	15257.43	15981.48	-9.19	3.00
kroA200	2	2	4	0.25	11916.87	16800.59	11532.38	12211.28	-31.36	-3.23
kroA200	2	2	4	0.50	12480.42	16800.59	12193.33	12732.35	-27.42	-2.30
kroA200	2	3	2	0.1	14645.57	16800.59	14073.39	15121.32	-16.23	-3.91
kroA200	2	3	2	0.25	12709.96	16800.59	12874.73	13382.48	-23.37	1.30
kroA200	2	3	2	0.50	12093.30	16800.59	11522.39	11739.38	-31.42	-4.72
kroA200	2	3	3	0.1	14491.40	16800.59	14764.43	14902.75	-12.12	1.88
kroA200	2	3	3	0.25	12196.08	16800.59	12083.85	12184.74	-28.07	-0.92
kroA200	2	3	3	0.50	11922.01	16800.59	12000.00	12455.12	-28.57	0.65
kroA200	2	3	4	0.1	14148.82	16800.59	13729.74	14028.36	-18.28	-2.96
kroA200	2	3	4	0.25	11887.75	16800.59	11374.25	11631.38	-32.30	-4.32
kroA200	2	3	4	0.50	10825.73	16800.59	11093.33	11873.84	-33.97	2.47
kroA200	3	1	2	0.1	11971.70	13219.04	11732.33	12024.48	-11.25	-2.00
kroA200	3	1	2	0.25	9951.45	13219.04	9834.83	10824.67	-25.60	-1.17
kroA200	3	1	2	0.50	11039.57	13219.04	10739.43	11763.24	-18.76	-2.72
kroA200	3	1	3	0.1	11207.92	13219.04	11138.33	11973.35	-15.74	-0.62
kroA200	3	1	3	0.25	10144.57	13219.04	9938.33	10872.62	-24.82	-2.03
kroA200	3	1	3	0.50	9712.54	13219.04	9658.33	10018.78	-26.94	-0.56
kroA200	3	1	4	0.1	11210.40	13219.04	10803.33	11342.02	-18.27	-3.63
kroA200	3	1	4	0.25	9615.98	13219.04	9473.83	9873.45	-28.33	-1.48
kroA200	3	1	4	0.50	10336.44	13219.04	10093.93	10134.25	-23.64	-2.35
kroA200	3	2	2	0.1	11548.34	13219.04	11183.39	11462.23	-15.40	-3.16
kroA200	3	2	2	0.25	10450.33	13219.04	9938.33	10928.14	-24.82	-4.90
kroA200	3	2	2	0.50	9831.38	13219.04	9627.33	9873.73	-27.17	-2.08
kroA200	3	2	3	0.1	11070.51	13219.04	10760.33	11083.58	-18.60	-2.80
kroA200	3	2	3	0.25	9770.72	13219.04	9387.44	9892.48	-28.99	-3.92
kroA200	3	2	3	0.50	9458.77	13219.04	9137.38	9641.63	-30.88	-3.40
kroA200	3	2	4	0.1	11439.41	13219.04	11032.49	11736.24	-16.54	-3.56
kroA200	3	2	4	0.25	9434.01	13219.04	9138.49	9573.27	-30.87	-3.13
kroA200	3	2	4	0.50	9550.38	13219.04	9232.72	9472.18	-30.16	-3.33
kroA200	3	3	2	0.1	11821.92	13219.04	11634.94	11843.25	-11.98	-1.58
kroA200	3	3	2	0.25	9556.56	13219.04	9317.33	9873.85	-29.52	-2.50
kroA200	3	3	2	0.50	9965.07	13219.04	9942.38	11024.47	-24.79	-0.23
kroA200	3	3	3	0.1	11116.32	13219.04	11538.33	12974.47	-12.71	3.80
kroA200	3	3	3	0.25	9073.79	13219.04	8793.33	9128.93	-33.48	-3.09
kroA200	3	3	3	0.50	9160.44	13219.04	9038.76	9734.92	-31.62	-1.33
kroA200	3	3	4	0.1	10881.11	13219.04	10773.84	11448.13	-18.50	-0.99
kroA200	3	3	4	0.25	8826.21	13219.04	8492.37	9003.49	-35.76	-3.78
kroA200	3	3	4	0.50	8665.28	13219.04	8379.28	8521.33	-36.61	-3.30

Table 14 – Complete comparison between Schermer (2018) and GVNS.

Instance	# truck	# drone	α	β	BKS	S_0	S^*	S	gap_{S_0,S^*}	gap_{BKS,S^*}
rd400	1	1	2	0.1	13718.12	15280.36	14532.97	14782.39	-4.89	5.94
rd400	1	1	2	0.25	13276.86	15280.36	14037.83	14134.84	-8.13	5.73
rd400	1	1	2	0.50	13381.07	15280.36	14283.43	14372.43	-6.52	6.74
rd400	1	1	3	0.1	13322.46	15280.36	13974.86	14082.48	-8.54	4.90
rd400	1	1	3	0.25	12684.18	15280.36	13387.78	13752.02	-12.39	5.55
rd400	1	1	3	0.50	12951.21	15280.36	13184.74	13762.76	-13.71	1.80
rd400	1	1	4	0.1	13924.91	15280.36	13903.74	14283.53	-9.01	-0.15
rd400	1	1	4	0.25	12078.46	15280.36	12345.47	12982.24	-19.21	2.21
rd400	1	1	4	0.50	12677.66	15280.36	12283.83	12381.35	-19.61	-3.11
rd400	1	2	2	0.1	13226.39	15280.36	13425.83	13941.58	-12.14	1.51
rd400	1	2	2	0.25	12042.64	15280.36	12974.37	13192.72	-15.09	7.74
rd400	1	2	2	0.50	12352.01	15280.36	12184.73	12772.58	-20.26	-1.35
rd400	1	2	3	0.1	12578.34	15280.36	12983.79	13182.45	-15.03	3.22
rd400	1	2	3	0.25	11173.15	15280.36	12038.38	12182.87	-21.22	7.74
rd400	1	2	3	0.50	11855.39	15280.36	12183.48	12742.47	-20.27	2.77
rd400	1	2	4	0.1	13714.87	15280.36	12974.87	13083.13	-15.09	-5.40
rd400	1	2	4	0.25	11060.80	15280.36	11183.48	11414.27	-26.81	1.11
rd400	1	2	4	0.50	11438.55	15280.36	10735.48	11381.45	-29.74	-6.15
rd400	1	3	2	0.1	13042.39	15280.36	12847.34	13428.83	-15.92	-1.50
rd400	1	3	2	0.25	11414.13	15280.36	11084.84	11982.31	-27.46	-2.88
rd400	1	3	2	0.50	11788.63	15280.36	12834.84	13082.92	-16.00	8.87
rd400	1	3	3	0.1	13351.76	15280.36	13984.48	14092.57	-8.48	4.74
rd400	1	3	3	0.25	10925.65	15280.36	11837.38	12081.24	-22.53	8.34
rd400	1	3	3	0.50	11414.13	15280.36	12845.47	13031.33	-15.93	12.54
rd400	1	3	4	0.1	13530.87	15280.36	12843.48	13352.33	-15.95	-5.08
rd400	1	3	4	0.25	11202.46	15280.36	10832.48	11138.32	-29.11	-3.30
rd400	1	3	4	0.50	11528.11	15280.36	11084.33	11562.47	-27.46	-3.85
rd400	2	1	2	0.1	7951.70	8311.25	7394.44	7428.67	-11.03	-7.01
rd400	2	1	2	0.25	7495.31	8311.25	7193.44	7352.83	-13.45	-4.03
rd400	2	1	2	0.50	7741.74	8311.25	7493.40	7681.37	-9.84	-3.21
rd400	2	1	3	0.1	7705.27	8311.25	7479.93	7662.73	-10.00	-2.92
rd400	2	1	3	0.25	7048.70	8311.25	7098.38	7264.27	-14.59	0.70
rd400	2	1	3	0.50	7057.60	8311.25	7164.84	7274.18	-13.79	1.52
rd400	2	1	4	0.1	7705.27	8311.25	7424.08	7524.27	-10.67	-3.65
rd400	2	1	4	0.25	7048.70	8311.25	7197.35	7318.74	-13.40	2.11
rd400	2	1	4	0.50	7057.60	8311.25	7027.44	7163.97	-15.45	-0.43
rd400	2	2	2	0.1	7728.40	8311.25	7034.82	7381.28	-15.36	-8.97
rd400	2	2	2	0.25	6627.01	8311.25	6973.89	7124.28	-16.09	5.23
rd400	2	2	2	0.50	6970.42	8311.25	6493.44	6772.23	-21.87	-6.84
rd400	2	2	3	0.1	7566.48	8311.25	7034.83	7321.85	-15.36	-7.03
rd400	2	2	3	0.25	6688.40	8311.25	6297.44	6481.37	-24.23	-5.85
rd400	2	2	3	0.50	6378.80	8311.25	6194.84	6379.34	-25.46	-2.88
rd400	2	2	4	0.1	7267.56	8311.25	6937.49	7181.47	-16.53	-4.54

Table 14 – Complete comparison between Schermer (2018) and GVNS.

Instance	# truck	# drone	α	β	BKS	S_0	S^*	S	gap_{S_0,S^*}	gap_{BKS,S^*}
rd400	2	2	4	0.25	6194.64	8311.25	6398.49	6521.38	-23.01	3.29
rd400	2	2	4	0.50	6565.62	8311.25	6353.75	6523.91	-23.55	-3.23
rd400	2	3	2	0.1	7321.83	8311.25	6873.84	7018.37	-17.29	-6.12
rd400	2	3	2	0.25	6654.59	8311.25	6389.84	6572.74	-23.12	-3.98
rd400	2	3	2	0.50	6592.31	8311.25	6198.95	6883.63	-25.41	-5.97
rd400	2	3	3	0.1	7508.65	8311.25	6834.84	7572.62	-17.76	-8.97
rd400	2	3	3	0.25	6076.32	8311.25	6184.29	6821.83	-25.59	1.78
rd400	2	3	3	0.50	6547.83	8311.25	6128.84	6723.41	-26.26	-6.40
rd400	2	3	4	0.1	7330.72	8311.25	6982.84	7391.14	-15.98	-4.75
rd400	2	3	4	0.25	6280.94	8311.25	6385.25	6724.26	-23.17	1.66
rd400	2	3	4	0.50	6441.07	8311.25	6425.84	6921.03	-22.69	-0.24
rd400	3	1	2	0.1	5593.67	5703.85	5439.48	5662.13	-4.63	-2.76
rd400	3	1	2	0.25	5213.89	5703.85	5263.84	5501.28	-7.71	0.96
rd400	3	1	2	0.50	5143.77	5703.85	5284.84	5482.19	-7.35	2.74
rd400	3	1	3	0.1	5341.73	5703.85	5132.74	5381.87	-10.01	-3.91
rd400	3	1	3	0.25	4926.58	5703.85	5143.84	5312.58	-9.82	4.41
rd400	3	1	3	0.50	5113.36	5703.85	5274.89	5493.77	-7.52	3.16
rd400	3	1	4	0.1	5395.10	5703.85	5198.38	5381.36	-8.86	-3.65
rd400	3	1	4	0.25	4693.87	5703.85	4873.89	4991.48	-14.55	3.84
rd400	3	1	4	0.50	4382.97	5703.85	4598.84	4782.23	-19.37	4.93
rd400	3	2	2	0.1	5282.78	5703.85	5093.83	5482.17	-10.69	-3.58
rd400	3	2	2	0.25	4971.88	5703.85	4873.84	5193.28	-14.55	-1.97
rd400	3	2	2	0.50	4431.38	5703.85	4387.35	4581.38	-23.08	-0.99
rd400	3	2	3	0.1	5146.87	5703.85	5027.38	5291.93	-11.86	-2.32
rd400	3	2	3	0.25	4749.72	5703.85	4893.84	5019.28	-14.20	3.03
rd400	3	2	3	0.50	4785.71	5703.85	4572.84	4729.37	-19.83	-4.45
rd400	3	2	4	0.1	5293.33	5703.85	5118.24	5331.87	-10.27	-3.31
rd400	3	2	4	0.25	4500.88	5703.85	4239.48	4483.13	-25.67	-5.81
rd400	3	2	4	0.50	4393.52	5703.85	4184.84	4291.84	-26.63	-4.75
rd400	3	3	2	0.1	5454.67	5703.85	5226.43	5482.19	-8.37	-4.18
rd400	3	3	2	0.25	4623.13	5703.85	4534.91	4661.83	-20.49	-1.91
rd400	3	3	2	0.50	4592.10	5703.85	4519.23	4982.17	-20.77	-1.59
rd400	3	3	3	0.1	5299.53	5703.85	5173.37	5313.73	-9.30	-2.38
rd400	3	3	3	0.25	4399.73	5703.85	4298.42	4313.83	-24.64	-2.30
rd400	3	3	3	0.50	4474.19	5703.85	4175.74	4282.74	-26.79	-6.67
rd400	3	3	4	0.1	5287.12	5703.85	5131.98	5384.28	-10.03	-2.93
rd400	3	3	4	0.25	4120.48	5703.85	4297.24	4312.38	-24.66	4.29
rd400	3	3	4	0.50	4585.89	5703.85	4139.47	4482.10	-27.43	-9.73
att532	1	1	2	0.1	81335.90	86742.77	83292.38	83982.47	-3.98	2.41
att532	1	1	2	0.25	79694.17	86742.77	81372.89	81582.21	-6.19	2.11
att532	1	1	2	0.50	79046.86	86742.77	80982.42	81935.32	-6.64	2.45
att532	1	1	3	0.1	79947.46	86742.77	81834.37	82984.14	-5.66	2.36
att532	1	1	3	0.25	76785.97	86742.77	78732.73	80183.81	-9.23	2.54

Table 14 – Complete comparison between Schermer (2018) and GVNS.

Instance	# truck	# drone	α	β	BKS	S_0	S^*	S	gap_{S_0,S^*}	gap_{BKS,S^*}
att532	1	1	3	0.50	77724.10	86742.77	75324.73	75982.52	-13.16	-3.09
att532	1	1	4	0.1	76448.24	86742.77	77194.87	77928.47	-11.01	0.98
att532	1	1	4	0.25	75904.12	86742.77	74239.34	75145.26	-14.41	-2.19
att532	1	1	4	0.50	79365.82	86742.77	75824.02	76018.37	-12.59	-4.46
att532	1	2	2	0.1	79816.13	86742.77	78762.43	79238.41	-9.20	-1.32
att532	1	2	2	0.25	75369.39	86742.77	72844.84	74361.74	-16.02	-3.35
att532	1	2	2	0.50	71888.92	86742.77	72274.74	73193.47	-16.68	0.54
att532	1	2	3	0.1	75885.36	86742.77	74293.69	74872.21	-14.35	-2.10
att532	1	2	3	0.25	69046.39	86742.77	71235.49	71744.56	-17.88	3.17
att532	1	2	3	0.50	70369.15	86742.77	70173.47	70827.84	-19.10	-0.28
att532	1	2	4	0.1	76101.13	86742.77	74842.17	75103.24	-13.72	-1.65
att532	1	2	4	0.25	67883.11	86742.77	68734.37	69721.37	-20.76	1.25
att532	1	2	4	0.50	68511.66	86742.77	67367.34	68018.38	-22.34	-1.67
att532	1	3	2	0.1	74956.61	86742.77	73428.43	73829.95	-15.35	-2.04
att532	1	3	2	0.25	67545.38	86742.77	66384.26	67183.40	-23.47	-1.72
att532	1	3	2	0.50	66513.44	86742.77	65372.87	69271.92	-24.64	-1.71
att532	1	3	3	0.1	74299.92	86742.77	71743.48	74714.27	-17.29	-3.44
att532	1	3	3	0.25	67170.13	86742.77	68632.44	70183.75	-20.88	2.18
att532	1	3	3	0.50	63980.49	86742.77	64833.48	67621.82	-25.26	1.33
att532	1	3	4	0.1	75613.30	86742.77	71874.37	73613.47	-17.14	-4.94
att532	1	3	4	0.25	65200.06	86742.77	66371.28	68271.74	-23.48	1.80
att532	1	3	4	0.50	65293.87	86742.77	64827.35	66173.86	-25.26	-0.71
att532	2	1	2	0.1	46808.20	55490.27	46973.37	48172.84	-15.35	0.35
att532	2	1	2	0.25	46245.65	55490.27	47382.43	49632.48	-14.61	2.46
att532	2	1	2	0.50	47078.86	55490.27	46242.48	47731.25	-16.67	-1.78
att532	2	1	3	0.1	47105.39	55490.27	47413.84	47872.71	-14.55	0.65
att532	2	1	3	0.25	44934.81	55490.27	43948.49	44213.18	-20.80	-2.20
att532	2	1	3	0.50	43623.97	55490.27	41439.48	42845.35	-25.32	-5.01
att532	2	1	4	0.1	45332.84	55490.27	46389.35	46872.42	-16.40	2.33
att532	2	1	4	0.25	44897.66	55490.27	45725.48	46029.34	-17.60	1.84
att532	2	1	4	0.50	44685.38	55490.27	44893.42	45128.65	-19.10	0.47
att532	2	2	2	0.1	47407.90	55490.27	46932.84	47238.15	-15.42	-1.00
att532	2	2	2	0.25	44749.06	55490.27	44924.84	45824.47	-19.04	0.39
att532	2	2	2	0.50	40678.55	55490.27	42948.00	43134.22	-22.60	5.58
att532	2	2	3	0.1	45025.03	55490.27	46298.89	47241.22	-16.56	2.83
att532	2	2	3	0.25	40200.92	55490.27	40838.43	41834.75	-26.40	1.59
att532	2	2	3	0.50	40970.44	55490.27	41042.37	41372.18	-26.04	0.18
att532	2	2	4	0.1	45253.23	55490.27	45988.48	46837.47	-17.12	1.62
att532	2	2	4	0.25	42891.59	55490.27	41843.40	42841.38	-24.59	-2.44
att532	2	2	4	0.50	41888.56	55490.27	39743.78	42841.12	-28.38	-5.12
att532	2	3	2	0.1	44260.81	55490.27	45013.44	46528.37	-18.88	1.70
att532	2	3	2	0.25	42084.92	55490.27	41843.78	42315.66	-24.59	-0.57
att532	2	3	2	0.50	40758.16	55490.27	38743.48	39414.76	-30.18	-4.94

Table 14 – Complete comparison between Schermer (2018) and GVNS.

Instance	# truck	# drone	α	β	BKS	S_0	S^*	S	gap_{S_0,S^*}	gap_{BKS,S^*}
att532	2	3	3	0.1	43040.19	55490.27	44725.85	44873.41	-19.40	3.92
att532	2	3	3	0.25	39219.11	55490.27	40284.49	40762.78	-27.40	2.72
att532	2	3	3	0.50	40439.74	55490.27	39732.75	40183.64	-28.40	-1.75
att532	2	3	4	0.1	43942.39	55490.27	44024.43	44183.75	-20.66	0.19
att532	2	3	4	0.25	40545.88	55490.27	39276.43	40183.74	-29.22	-3.13
att532	2	3	4	0.50	41925.71	55490.27	40642.97	41732.58	-26.76	-3.06
att532	3	1	2	0.1	34899.89	48618.90	34187.84	34731.42	-29.68	-2.04
att532	3	1	2	0.25	33483.23	48618.90	34742.49	35732.84	-28.54	3.76
att532	3	1	2	0.50	33703.51	48618.90	35863.89	37611.55	-26.23	6.41
att532	3	1	3	0.1	33775.68	48618.90	34764.94	34872.63	-28.50	2.93
att532	3	1	3	0.25	34599.84	48618.90	35732.78	35987.14	-26.50	3.27
att532	3	1	3	0.50	31512.06	48618.90	33842.14	34762.16	-30.39	7.39
att532	3	1	4	0.1	34835.32	48618.90	35832.94	36071.74	-26.30	2.86
att532	3	1	4	0.25	30657.51	48618.90	31872.54	32027.24	-34.44	3.96
att532	3	1	4	0.50	32017.20	48618.90	31283.98	31965.33	-35.65	-2.29
att532	3	2	2	0.1	33851.64	48618.90	34837.94	34992.65	-28.34	2.91
att532	3	2	2	0.25	33259.15	48618.90	32874.25	33183.47	-32.38	-1.16
att532	3	2	2	0.50	31648.79	48618.90	31252.37	31764.35	-35.72	-1.25
att532	3	2	3	0.1	33525.01	48618.90	34847.82	35018.38	-28.32	3.95
att532	3	2	3	0.25	30927.17	48618.90	31837.94	32014.48	-34.52	2.94
att532	3	2	3	0.50	29369.99	48618.90	29874.84	31028.42	-38.55	1.72
att532	3	2	4	0.1	32594.50	48618.90	33219.39	33651.65	-31.67	1.92
att532	3	2	4	0.25	30596.74	48618.90	30843.90	31468.37	-36.56	0.81
att532	3	2	4	0.50	30729.67	48618.90	29834.82	31846.26	-38.64	-2.91
att532	3	3	2	0.1	33726.30	48618.90	32874.51	33081.27	-32.38	-2.53
att532	3	3	2	0.25	28295.15	48618.90	28743.82	29018.38	-40.88	1.59
att532	3	3	2	0.50	31523.46	48618.90	30824.93	30972.28	-36.60	-2.22
att532	3	3	3	0.1	33080.64	48618.90	32984.78	33083.74	-32.16	-0.29
att532	3	3	3	0.25	30384.06	48618.90	29842.11	30284.38	-38.62	-1.78
att532	3	3	3	0.50	29130.71	48618.90	28743.89	28981.47	-40.88	-1.33
att532	3	3	4	0.1	33080.64	48618.90	32874.08	32973.38	-32.38	-0.62
att532	3	3	4	0.25	29852.34	48618.90	29143.84	29982.74	-40.06	-2.37
att532	3	3	4	0.50	30232.14	48618.90	29834.22	30827.26	-38.64	-1.32