General Variable Neighborhood Search for the Drone-Assisted Vehicle Routing Problem with Robot Stations

André Craveiro Morim

Master's Dissertation

Supervisor at FEUP: Prof. Pedro Amorim
Supervisor at the University of Twente: Prof. Martijn Mes

Mestrado Integrado em Engenharia e Gestão Industrial

2020-07-24

Abstract

The goal of the present study is to formally define the 'Drone-Assisted Vehicle Routing Problem with Robot Stations' (VRPD-RS), a problem that combines the delivery concept of trucks operating in tandem with drones and satellite stations that store robots to minimize the delivery completion time, i.e., makespan, or the operational cost. Due to the computational complexity of the problem and also to allow addressing instances of practical size and relevance, an heuristic solution approach tailored to the VRPD-RS is presented.

The VRPD-RS was modelled as a mixed integer linear program (MILP) and numerical experiments were performed using a state-of-the-art solver to understand the impact of adding trucks, drones and robot stations to a delivery system. Additionally, an algorithm based on the General Variable Neighborhood Search (GVNS) metaheuristic framework, that applies five local search operators created specifically for the problem, was developed. In order to validate the solution approach, the solutions returned by the algorithm were compared against the results achieved by the solver. Finally, the algorithm was used to solve problems marked by road network congestion to assess how drones and robots can help mitigate its impact on the makespan.

The numerical results evidence that the VRPD-RS is computationally demanding as for the minimization of makespan and operational cost, the solver failed to reach optimal solutions within one hour of execution for instances with 10 and 15 customers, respectively. The results also confirm that including robot stations leads to makespan and operational cost reductions. For the makespan objective, the results show that additional trucks are more effective in reducing the delivery time than additional drones, but contribute to increased operational costs. More drones, however, beyond improving makespan, result in cost reductions. Additionally, using multiple trucks and drones did not impact the operational cost obtained using one truck and one drone, but allowed to reduce the corresponding makespan. The proposed GVNS heuristic yielded an average gap of -2.8% to solver solutions within an average of 9 minutes for the makespan objective. In terms of cost minimization, an average gap of 2.5% was achieved for approximately 5 minutes of running time, on average. The analysis of congested scenarios confirmed that, compared to truck-only systems, not only do drones and robots help improve the makespan and associated operational cost but also increase system robustness against makespan increases caused by congestion.

This dissertation contributes to the scientific literature concerned with studying the impact of including autonomous vehicles in distribution systems. The computational experiments prove that a delivery concept supported by trucks, drones and robots has a positive impact in terms of minimal delivery time and operational cost. Moreover, an efficient solution approach capable of improving minimum makespan VRPD-RS solutions found by a state-of-the-art solver and reaching near-optimal minimum operational cost VRPD-RS solutions is presented. The GVNS metaheuristic has the potential to be applied in subsequent studies to further demonstrate the usefulness of the concept modelled by the VRPD-RS, e.g., inclusion of time-windows or optimization of a combined cost-makespan objective.

Resumo

O presente estudo procura definir formalmente o 'Drone-Assisted Vehicle Routing Problem with Robot Stations' (VRPD-RS), um problema que combina o conceito de transporte realizado por camiões em cooperação com *drones* e estações-satélite que armazenam robôs para realizar entregas no menor tempo ou ao menor custo possível. Devido à complexidade computacional do problema e com o objetivo de resolver problemas de dimensão e relevância prática, este estudo propõe uma heurística especializada na resolução do VRPD-RS.

O VRPD-RS foi definido segundo um modelo de programação linear inteira mista (PLIM) e aplicado num *solver* a fim de analisar o impacto da adição de camiões, *drones* e robôs a um sistema de distribuição. Adicionalmente, foi implementado um algoritmo baseado na meta-heurística General Variable Neighborhood Search (GVNS), que aplica cinco operadores de pesquisa local especializados no VRPD-RS. A fim de validar o algoritmo, os seus resultados foram comparados com os do *solver*. Finalmente, o algoritmo foi aplicado em cenários com congestionamento, para perceber de que forma *drones* e robôs atenuam o seu impacto nos tempos de entrega.

Os resultados numéricos evidenciam que o VRPD-RS é um problema computacionalmente exigente uma vez que o *solver* foi incapaz de obter soluções ótimas dentro de uma hora de execução para problemas com 10 e 15 clientes para a minimização do tempo de entrega e custo, respetivamente. Os resultados confirmam também que as estações-satélite contribuem para reduções dos tempos de entrega e custo operacional. No que diz respeito à redução do tempo de entrega, a adição de camiões é mais eficaz do que a adição de *drones*, mas aumenta o custo operacional. Por outro lado, adicionar *drones* permite reduzir simultaneamente o tempo de entrega e custo. Relativamente às experiências ligadas à minimização do custo, o uso de diversos camiões e *drones* não afetou o custo obtido com um cenário com apenas um camião e um *drone*, mas levou a uma redução dos tempos de entrega. Em média, a heurística GVNS foi capaz de obter soluções 2.8% melhores que as soluções do *solver* num tempo médio de execução de 9 minutos para a minimização do tempo de entrega. Em termos da minimização do custo, a heurística ofereceu resultados 2.5% piores que o *solver* em 5 minutos, em média. As experiências realizadas em cenários com congestionamento permitiram verificar que, comparativamente com sistemas baseados em camiões, *drones* e robôs não só melhoram os tempos de entrega e custo como conferem robustez contra aumentos dos tempos de entrega causados por tais cenários.

Esta dissertação contribui para a literatura científica ligada ao estudo do impacto de veículos autónomos em sistemas de distribuição. As experiências realizadas provam que um conceito de entregas baseado em camiões, *drones* e robôs tem um impacto positivo em termos da redução dos tempos de entrega e custo operacional. Adicionalmente, este trabalho propõe um algoritmo eficiente capaz de melhorar algumas das soluções para o objetivo temporal e de atingir soluções próximas das soluções de custo mínimo obtidas pelo *solver*. A heurística baseada em GVNS tem, portanto, potencial para ser aplicada em estudos dedicados à análise do conceito de transporte modelado pelo VRPD-RS em cenários práticos, e.g., otimização de operações sujeitas a janelas de entrega ou de problemas em que o custo e o tempo são condensados num único objetivo.

Acknowledgements

First, I would like to thank my supervisors at the University of Twente, Prof. Martijn Mes and Prof. Eduardo Lalla-Ruiz, for giving me the opportunity to work alongside them, for their support and for daring me to aim high.

I would also like to thank my supervisor at FEUP, Prof. Pedro Amorim, not only for the guidance provided throughout this study, but also for his enthusiasm and capability of transmitting knowledge. Indeed, he helped me become even more motivated about my degree and supply chain management, in particular.

To all the Professors I had the opportunity to learn with during my degree, I would like to thank for their valuable lessons and for stressing the importance of seeking excellence. Parts of the present study and of the engineer I have become were shaped by them.

I would also like to thank my friends that walked this journey with me for the past five years. For all the goofy moments we shared as well as the ones when it was needed to get down to business. I am very lucky to have met such good people that are extremely competent and manage to stay humble regardless.

Finally, yet importantly, I would like to take a moment to appreciate my family. To Sérgio and Rosa, my parents, I would like to thank for their unconditional support and for always believing in me. To my sister, Sofia, I would like to thank her for being my companion and for always making me laugh. After finishing my degree, I am sure we will put aside our careers and form a band. I would also like to thank my grandmother, Conceição, since she helped raising me when I was little and would always listen to me with the proudest expression on her face.

"Sometimes, struggles are exactly what we need in our life. If we were to go through our life without any obstacles, we would be crippled. We would not be as strong as what we could have been. Give every opportunity a chance, leave no room for regrets."

Friedrich Nietzsche

Contents

Acronyms and Symbols

ALNS	Adaptive Large Neighborhood Search
B2C	Business-to-Consumer
DC	Distribution Center
FSTSP	Flying Sidekick Traveling Salesman Problem
GRASP	Greedy Randomized Adaptive Search Procedure
GVNS	General Variable Neighborhood Search
HGA	Hybrid Genetic Algorithm
LMD	Last-mile Deliveries
LML	Last-mile Logistics
mFSTSP	Multiple Flying Sidekicks Traveling Salesman Problem
MILP	Mixed Integer Linear Program (MILP)
PDSTSP	Parallel Drone Scheduling Traveling Salesman Problem
PMS	Parallel Machine Scheduling
RVND	Randomized Variable Neighborhood Search
SA	Simulated Annealing
TTRP	Truck and Trailer Routing Problem
TSDSLP	Traveling Salesman Drone Station Location Problem
TSP	Traveling Salesman Problem
TSP-D	Traveling Salesman Problem with Drone
TSP-DS	Traveling Salesman Problem with a Drone Station
TSP-D-RS	Drone-Assisted Traveling Salesman Problem with Robot Stations
TSP-mD	Traveling Salesman Problem with Multiple Drones
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
VND	Variable Neighborhood Descent
VNS	Variable Neighborhood Search
VRP	Vehicle Routing Problem
VRPD	Vehicle Routing Problem with Drones
VRPD-RS	Drone-Assisted Vehicle Routing Problem with Robot Stations

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays, business-to-consumer (B2C) e-commerce plays an important role for several businesses, regardless of the industry they belong to. The adoption of this B2C initiative has been increasing rapidly throughout the years and the global e-commerce market is forecasted to reach $4,206B in sales in 2020, potentially representing 16.1% of total retail sales (Lipsman, 2019). B2C e-commerce represents an important asset for companies, since it consists in an additional distribution channel. However, handling this business process increases complexity in managing logistics activities. In fact, online customers put pressure on supply chain performance as they require a high service level, both in terms of meeting promised time-windows and offering reduced order lead times, while usually assuming no willingness to pay for such conditions (Mangiaracina et al., 2019).

Last-mile deliveries (LMD) or last-mile logistics (LML) refer to "the last stretch of a business-to-consumer parcel delivery service" (Lim et al., 2018) and are seen by many scholars as the most crucial step in logistic processes (Mangiaracina et al., 2019). Using an activity-based costing analysis, Vanelslander et al. (2013) were able to demonstrate that LMD often contribute to half of the total supply chain cost. The inefficient and costly nature of LMD originates in the fact that, normally, these involve the fulfillment of several individual small orders in locations with a significant level of customer dispersion. With the development of e-commerce, this condition is further accentuated as more customers are buying products to be delivered at home. Additionally, items sold are becoming more heterogeneous, which leads to more complexity on the planning of deliveries that become less-standardized. On top of these challenges, since the last-mile delivery is the contact point between the company and the client, it directly impacts customer experience. Thus, planning LML requires incorporating as much flexibility and convenience as possible to increase customer satisfaction (Macioszek, 2018). In fact, flexibility, alongside delivery speed and precision, i.e., width of the offered time slots, are determining drivers of delivery service and influence customers' time slot selection and willingness to pay (Amorim et al., 2020).

The volume of academic literature dedicated to analyzing innovative solutions for B2C last-mile deliveries has seen an increase in recent years. In fact, practitioners and academics are transitioning from focusing on the optimization of traditional delivery modes, which have been inten-

sively studied and where there is a limited gap for improvement, to explore disruptive solutions capable of handling LML more efficiently. Examples of innovative concepts include parcel lockers, crowdsourcing logistics, dynamic pricing with respect to the desired delivery time-window, mapping customer behaviour to derive probability of the customer being home, as well as the application of alternative vehicles such as drones and robots (Mangiaracina et al., 2019).

Aerial drones or unmanned aerial vehicles (UAVs) have been emerging as a valuable technology in applications for agriculture, infrastructure inspection, entertainment and media, security and delivery applications. Transport or delivery applications comprise package deliveries to sparse rural regions and the first- and last-mile logistics within urban areas, as well as express deliveries of medical supplies or parts inside a manufacturing facility (Otto et al., 2018). In terms of last-mile logistics, there have been several cases of companies presenting parcel delivery services based on drones. In 2013, Amazon CEO Jeff Bezos announced that his company would be launching a drone delivery service named 'Prime Air', which would perform parcel deliveries from distribution centers directly to customers (Murray and Chu, 2015). Recently, Amazon executive Jeff Wilke announced that 'Prime Air' is close to being commercially deployed with drones capable of flying up to 15 miles (approximately 24.14 km) and deliver packages under 5 pounds (2.27 kg) within 30 minutes (D'Onfro, 2019). Google set out to deploy food and medicine deliveries with drones in 2012 under a project named 'Wing', whose first successful delivery occurred in 2014 (X Development LLC, 2019). Chinese e-commerce retailer JD.com started developing its drone system in 2015 and has been performing deliveries using this technology since 2016 (JD.com, 2016). Additionally, traditional logistics service providers such as DHL, UPS and FedEx have been testing the use of UAVs to perform their operations (Bryan, 2014; Wing Medium, 2019; Burns, 2017). Robots or unmanned ground vehicles (UGVs) have also been regarded as a solution to reduce the impact of transportation in urban settings in terms of traffic and pollution. Compared to drones, there are less cases of applications concerning delivery robots and the range of specialized manufacturers is limited and composed of the following: Starship Technologies, Dispatch and Marble. Starship Technologies has already tested the delivery of groceries and medicine (Sonneberg et al., 2019) and, in 2016, celebrated a strategic partnership with Mercedes-Benz Vans to introduce a combined robot-truck system (Boysen et al., 2018b).

The widespread interest in the deployment of drones for LML stems from their attractive technological characteristics that have the potential to allow companies to perform deliveries faster and in a more cost-efficient manner (Agatz et al., 2016). One evident advantage relates to the reduction of labor costs, since a human pilot is not required. Additionally, pilotless technology reduces the weight of the aircraft, which ultimately decreases energy consumption. Drones are usually battery-powered, which in comparison to truck deliveries, leads to less CO_2 emissions and a lower cost per mile to operate (Otto et al., 2018). Another advantage is that drones are faster than trucks and can fly in 3D space, which allows them not to depend on the road network and avoid congestion, a common phenomenon in urban last-mile deliveries (Agatz et al., 2016).

However, there are several limitations associated with the application of drones for commercial purposes. Due to technological specifications, such as energy storing capacity and the size and

configuration of the aircraft, drones' carrying capacity is limited. In fact, payloads carried by a drone normally do not weigh more than 3 kg and drones usually deliver one package per *sortie* (Otto et al., 2018). This happens because drone energy consumption increases linearly with the applied payload, size of the drone and the weight of its battery or fuel carried (Dorling et al., 2017). Furthermore, drones have limited flight range as a consequence of being powered by a battery unit of limited capacity or a combustion engine consuming a finite amount of fuel. The endurance of a drone may also be affected by several factors including flying altitude, since its propellers have to rotate faster at higher altitudes where air density is lower in order to generate enough thrust, or meteorological conditions, such as the wind. Consequently, to perform deliveries with drones, a human operator is required to perform battery swaps or for refueling, as well as loading parcels to the drone, which partially offsets the benefits of not requiring a driver. In what concerns regulations, the development of drone-based delivery systems may be slowed down as its application raises both safety and privacy issues. In order to guarantee order in airspace traffic and protect people on the ground, the Federal Aviation Administration in the US establishes that drones should be operated within the visual line of sight of an exclusive human pilot. Furthermore, Amazon proposed a division of airspace into two zones of different allowed speeds located beneath a no-fly buffer zone that separates them from piloted aircrafts, while NASA recommended defining air corridors for UAVs. Even though these measures intend to promote the deployment of drones, they might overshadow some of the benefits provided by them (Otto et al., 2018).

One possible approach to overcome the limitations inherent to UAVs consists in applying them in an hybrid truck-drone delivery system. AMP Electric Vehicles and the University of Cincinnati partnered to build the 'HorseFly', a delivery drone designed to be mounted on top of electric trucks to work in tandem with them (Robinette, 2014). In this system, while the truck performs a route that serves a given subset of customers, the drone repeatedly retrieves a parcel from the truck, delivers it to its destination and returns to recharge its battery and to load another package for a further delivery (Agatz et al., 2016). This configuration increases the coverage of drones since the truck continuously positions customers within drone flight range. Additionally, it utilizes the truck's relatively larger loading capacity more effectively, as this vehicle acts both as a delivery resource and a mobile depot (Murray and Chu, 2015), and decreases its total travelled distance. Such delivery system has the potential to reduce the total time required to serve all customers due to the parallelization of tasks, but leads to a challenging planning problem consisting of both assignment decisions, i.e., attributing a delivery mode to each customer, and routing decisions, i.e., defining the sequence of customers for each vehicle (Agatz et al., 2016). This hybrid delivery system was first proposed as a variant of the traditional Traveling Salesman Problem (TSP) by Murray and Chu (2015) as the 'Flying Sidekick Traveling Salesman Problem' (FSTSP). Agatz et al. (2016) proposed the 'Traveling Salesman Problem with Drone' (TSP-D), similar to the FSTSP but with the possibility of launching and retrieving the drone in the same location.

Amazon's direct warehouse-to-customer delivery service was also studied by Murray and Chu (2015) in a problem they proposed named 'Parallel Drone Scheduling TSP' (PDSTSP). In this setting, a fleet of drones is available at the distribution center (DC), together with a single truck.

The truck performs its TSP route while the drones act independently to serve parcels from the DC to customer locations. As opposed to the FSTSP and TSP-D, the PDSTSP requires no synchronization between the truck and the drone. Given that drones possess a limited flight range, this scenario would only make sense in a situation where the depot is located within a dense customer cluster. However, this condition may not hold and companies could be forced to relocate or build warehouses in close proximity to highly populated areas to take advantage of economies of scale, incurring on large investment costs (Murray and Chu, 2015). To overcome this, Amazon proposed using streetlights as drone docking stations. These stations would charge drones, store them and could act as a depot from where the last-mile delivery could be carried on, either by traditional delivery means, drones or UGVs (Mogg, 2016). Based on this concept, Kim and Moon (2019) proposed the 'TSP with a drone station' (TSP-DS), a similar problem to the PDSTSP where the fleet of drones is positioned at a remote station, which is activated for delivery when the truck visits it as part of its TSP route. After activation, the drones depart from the station to perform their assigned delivery and return to the station to be either stored or recharged and resupplied.

Recently, Schermer et al. (2020b) proposed a novel problem consisting in a combination of the TSP-D and the TSP-DS, the 'Drone-Assisted Traveling Salesman Problem with Robot Stations' (TSP-D-RS). The TSP-D-RS considers a truck-drone tandem that departs from the depot to perform deliveries with the possibility of activating stations that store robots to perform unattended deliveries. The objective of this problem consists in the minimization of the makespan, i.e., the total time required to serve all customers and for each vehicle to return to its initial location, or in the minimization of operational costs. The addition of robots might be advantageous because, even though they are a slower delivery mode, they can travel along pedestrian areas and sidewalks which might help them access customer locations otherwise unreachable by drones or trucks. Furthermore, compared to drones, UGVs have a longer endurance and are capable of carrying heavier parcels (Moeini and Salewski, 2020). Another perspective with practical relevance is that the problem could be applied to a situation where robots represent the customers and the stations depict parcel lockers. Parcel lockers are boxes located in public spaces used by logistics service providers or retailers to deliver parcels to customers who need to reach the lockers to retrieve their order (Mangiaracina et al. (2019)). In that case, the objective would need to contemplate the minimization of customer walking distance. After reviewing the literature, it was possible to conclude that, for the TSP-D-RS, there is only a Mixed Integer Linear Programming (MILP) formulation proposed by Schermer et al. (2020b) that offers reasonably acceptable results within relatively short computational times for small- to medium-sized instances with up to 20 customers.

The present study will address the TSP-D-RS and propose the 'Drone-Assisted Vehicle Routing Problem with Robot Stations' (VRPD-RS), as well as a solution approach capable of addressing both problems. The VRPD-RS generalizes the TSP-D-RS to consider a fleet of homogeneous trucks, each carrying an homogeneous fleet of drones, besides the available robot stations. The ultimate goal of this project is to produce good quality solutions for larger instances within reasonable computational times. Additionally, an analysis of the performance in scenarios where congestion is present will be carried out to assess how deploying drones and robots in delivery op-

erations can mitigate its impact on the delivery completion time. With this approach, this project seeks to motivate practitioners to incorporate autonomous delivery resources in their operations as a mean to increase efficiency and business sustainability.

1.1 Dissertation Structure

The remainder of this dissertation is organized as follows. A literature review on routing problems involving drones and robots, as well as the respective solution approaches proposed so far, is presented in Chapter 2. Chapter 3 presents a formal definition of the VRPD-RS along with its assumptions and a MILP formulation. The proposed heuristic for the VRPD-RS is described in Chapter 4. Chapter 5 includes a set of computational experiments to explore the features of the VRPD-RS, evaluate the performance of the proposed algorithm and to analyze how drones and robots help mitigate the impact of congestion on delivery completion times. Finally, Chapter 6 establishes some concluding remarks and directions for future research.

Chapter 2

Literature Review

The present chapter aims to provide an introduction to the theoretical concepts related to the VRPD-RS. Initially, this review will frame the problem proposed in this dissertation within the vast range of Vehicle Routing Problem (VRP) categories. Additionally, problems in the literature that consider drones and robots as delivery resources, and that precede the VRPD-RS, will be analyzed. For each problem, existing solution approaches and relevant conclusions will be discussed.

2.1 The Vehicle Routing Problem

The VRP addresses the operational decisions of routing delivery resources to serve customers in a company's supply chain (Sharma et al., 2018). Dantzig and Ramser (1959) introduced the problem in 1959 and, since then, it has been extensively studied and several variants have been proposed. The vast literature on the VRP stems from its economic relevance and scientific interest. On the one hand, the design of efficient routes is of crucial importance for transportation companies, which operate in an industry marked by limited profit margins. On the other hand, the VRP generalizes the well-known NP-hard TSP, thus offering desirable characteristics to conduct combinatorial optimization studies (Vidal et al., 2019). In fact, most of the solution approaches presented for the VRP and its variants have been in the field of heuristics (Sharma et al., 2018).

According to Vidal et al. (2019), real applications motivate extensions to the VRP through three main lines: (1) considering emerging objectives; (2) integrating routing decisions with other strategic or tactical decisions; and (3) capturing the characteristics of modern supply chains. As mentioned in the previous chapter, the VRPD-RS results from a trend in the scientific community to study drone- and robot-based routing problems initiated by Murray and Chu (2015) and Agatz et al. (2016) with the introduction of the FSTSP and TSP-D, respectively. These problems were later extended to include several trucks and drones in a problem designated as the 'Vehicle Routing Problem with Drones' (VRPD) introduced by Wang et al. (2017). The VRPD-RS emerges as a combination of the VRPD and remote stations, a concept modelled by the TSP-DS. The resulting problem extends the original VRP according to the lines indicated previously as it considers a cost objective, like the majority of VRP literature does, as well as a makespan objective, which is a

metric of service quality usually employed in drone-based routing problems. Beyond routing decisions, the VRPD-RS considers aspects of facility location problems, since a set of robot stations is available to be activated for delivery. Additionally, the VRPD-RS models the utilization of a novel hybrid truck-drone delivery concept, already tested by UPS in 2017 (UPS, 2017).

One class of problems that shares similarities with the VRPD-RS is the vehicle routing problem with multiple synchronization constraints, which considers dependencies between vehicles in terms of spatial, temporal and load requirements (Murray and Chu, 2015). In a survey conducted by Drexl (2012), this VRP category is defined as being "a vehicle routing problem where more than one vehicle may or must be used to fulfill a task", which is a valid statement for the VRPD-RS as well. Among the types of synchronization included in the survey, it is possible to classify the VRPD-RS as being marked by movement synchronization *en route*, since the operations of the drone fleet need to be aligned with its respective truck route. Additionally, load synchronization is present since all parcels carried by the truck will be materialized in truck, drone or robot deliveries. Ultimately, these vehicle synchronization aspects will lead to operation synchronization, as the tasks of each vehicle will be temporally constrained by the activity of other vehicles.

The VRPD-RS also presents features that resemble the class of two-echelon routing problems. More specifically, the operation of the robot stations in the VRPD-RS is closely related the 'Two-Echelon Location Routing Problem', which also considers the selection of the best set of available satellite facility locations (Cuda et al., 2015; Gonzalez-Feliu, 2011). However, in the VRPD-RS, the robots have a unitary carrying capacity, which leads to the existence of multiple roundtrips beginning at the stations, instead of a second level route. Due to this last feature, some authors consider the robot subproblem as possessing aspects of Parallel Machine Scheduling Problems (PMS) with precedence constraints (Kim and Moon, 2019).

Another class of two-echelon routing problems, the 'Truck and Trailer Routing Problem' (TTRP), is similar to the VRPD-RS regarding the operation of truck-drone tandems. In a TTRP, transportation is performed using a set of trucks and trailers with the objective of minimizing total routing costs, while assigning a compatible vehicle to each customer. This last restriction occurs because some customers must be served by a truck alone, whereas others can be supplied by a complete vehicle, i.e., truck coupled with a trailer. To address the first group of customers, the truck should first decouple from the trailer. This leads to a first-level route, travelled by the complete vehicle, and a second-level route, performed by the truck that starts and ends at the same node where the decoupling operation took place. In the VRPD-RS, there is also a first-level route, travelled by the truck, and secondary routes, performed by the drones that begin and end in customer nodes from the first route.

2.2 Routing Problems with Drones

This section will focus primarily on drone routing studies whose origin is connected to the problems proposed by Murray and Chu (2015). Extensions of the classical routing problems TSP and VRP, as well as problems involving scheduling decisions, will be analyzed. Additionally,

other drone routing problems with different assumptions from those introduced by Murray and Chu (2015) with respect to the operational characteristics of drones will be exposed, since their conclusions provide valuable insights for the present study.

2.2.1 Drone Routing as an extension of the TSP

2.2.1.1 Flying Sidekick Travelling Salesman Problem

Murray and Chu (2015) were the first to introduce a variant of the TSP based on the hybrid truck-drone system, designated as the FSTSP. Together with the problem definition, the authors provided a MILP formulation and a simple route and re-assign heuristic. Initially, the heuristic builds a TSP truck route addressing all customers and, progressively, tests serving eligible customers via drone or at a different point in the truck's route, selecting the moves that offer the maximum time savings. Ponza (2016) proposed changes to the original model to ensure consistency in terms of time variables and presented a heuristic solution approach based on Simulated Annealing (SA). An alternative solution approach was presented by de Freitas and Penna (2018) in the form of a Randomized Variable Neighborhood Descent (RVND) algorithm, which starts by computing the optimal TSP route and converting truck deliveries into drone deliveries. Afterwards, the solution is improved using a Variable Neighborhood Descent (VND) heuristic whose neighborhood order is randomized at each iteration. In another work, de Freitas and Penna (2020) explored the RVND as the local search procedure in a Hybrid General Variable Neighborhood Search (GVNS) algorithm, where the same neighborhood list is used in the shaking and local search phases. Using this approach, an average improvement of 9% over the results of Ponza (2016) was achieved.

Jeong et al. (2019) extended the FSTSP to consider the effect of parcel weight on drone energy consumption and restricted flying areas. To solve this problem and demonstrate that practical issues may limit the efficiency of the FSTSP concept, a mathematical model and an evolutionary-based heuristic were developed. Murray and Raj (2020) proposed the 'Multiple Flying Sidekicks Traveling Salesman Problem' (mFSTSP), an extension of the FSTSP that models the collaboration of a truck with a fleet of heterogeneous drones. The mFSTSP also considers a more realistic view than the original model by testing different endurance models. The problem was defined as a MILP and a three-phased heuristic was presented. Computational tests demonstrated that the addition of UAVs to the fleet has diminishing marginal returns.

2.2.1.2 Traveling Salesman Problem with Drone

Agatz et al. (2016) introduced the TSP-D to model the innovative delivery concept of a truck collaborating with a drone (see Figure 2.1). The problem resembles the FSTSP but admits that a drone can perform roundtrips, i.e., be launched and retrieved at the same location, while the FSTSP expresses that only multi-leg operations, i.e., where the launch and retrieval nodes are distinct, should take place. An integer program using the concept of operation instead of a formulation based on arcs and vertices was proposed, as well as several route-first, cluster-second heuristics based on local search and dynamic programming. The operation-based integer programming formulation

allowed to solve optimally instances with a higher number of customers when compared to the formulation of Murray and Chu (2015). Additionally, computational experiments evidenced that starting with the optimal TSP tour before performing drone assignments leads to better results when compared to using approximate approaches for the TSP.

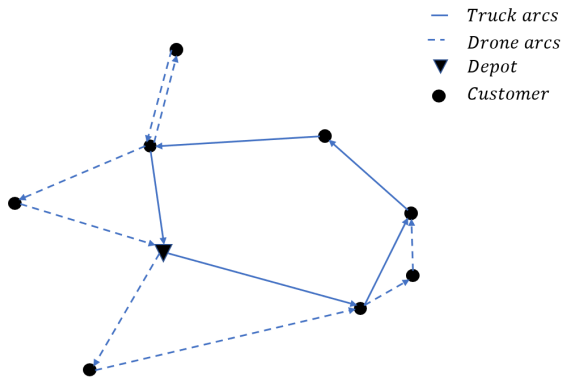

Figure 2.1: Illustration of the Traveling Salesman Problem with Drone.

Ha et al. (2015) studied a TSP-D allowing only multi-leg operations and proposed two heuristics using opposing concepts: route first, cluster second and cluster first, route second. For both heuristics, the cluster step of the algorithm is performed by solving a mixed integer program that chooses the set of drone operations that maximizes a given profit function. The authors concluded that clustering first and building a TSP tour for the selected operations offers better results. Es Yurek and Ozmutlu (2018) adopted a similar approach and considered an optimization-based iterative heuristic that decomposes the problem into two stages: (1) determination of the assignment of customers to each vehicle and the truck route and (2) application of a MILP formulation to determine the set of launch and retrieval nodes that minimizes truck waiting time. Bouman et al. (2018) developed an exact approach based on dynamic programming capable of solving large TSP-D instances by limiting the number of locations a truck can visit while the drone performs an operation. Although this restriction may prevent the algorithm from finding optimal solutions, computational experiments proved that it helps decreasing computational times and has a reduced impact on solution quality. Poikonen et al. (2019) proposed a branch-and-bound approach as well as a set of heuristics based on it. Among the algorithms presented, a divide-and-conquer heuristic that applies the branch-and-bound method in partitions of a TSP route proved to offer good results when compared to the approach presented by Agatz et al. (2016).

In a study performed by Tang et al. (2019), it was proven that the TSP-D is strongly NP-hard, even for the restricted case where drone operations need to be scheduled for a given truck route, and provided a constraint programming formulation. Compared to the approach of Bouman et al. (2018), their formulation was capable of scaling up to bigger problems more efficiently, whereas the dynamic programming-based approach solved small instances faster. Additionally, their formulation offered better results than the branch-and-bound-based heuristics of Poikonen et al. (2019) for small instances, but was outperformed in large instances. El-Adle et al. (2019) also proposed an alternative formulation as a mixed integer program and a set of pre-processing

techniques and bound tightening strategies, which allowed to optimally solve instances with up to 24 customers.

A variant of the TSP-D contemplating the minimization of operational costs was introduced by Ha et al. (2018), where, beyond the total transportation cost, a component related to the opportunity cost of having the vehicles wait for each other at retrieval nodes is also considered. To address the problem, a MILP formulation and a Greedy Randomized Adaptive Search Procedure (GRASP) were proposed, and the latter was compared with an adaptation of the local search algorithm of Murray and Chu (2015) for the cost minimization variant. Numerical results evidenced that, although the local search algorithm is more efficient, the GRASP outperforms it in terms of solution quality for instances with up to 50 customers. An hybrid genetic algorithm (HGA) approach was developed by Ha et al. (2019) where a chromosome assumes the form of a giant TSP tour and, at each generation, is subject to a split procedure to be converted into a TSP-D solution and educated through local search improvement heuristics, before being restored into a TSP tour again. Additionally, the authors promote the exploration of infeasible solutions with respect to drone endurance constraints. Comparing with the GRASP developed by Ha et al. (2018), HGA dominated in terms of solution quality for the two distinct TSP-D objectives, while GRASP proved to be a more efficient algorithm. Additionally, the approach improved some existing best-known solutions found by de Freitas and Penna (2020). While the majority of studies addresses either the minimization of the completion time or the operational costs, Wang et al. (2019) studied the TSP-D in a bi-objective problem considering the trade-off between both metrics.

A TSP-D variant including *en route* operations, i.e., drone launches and retrievals performed along route arcs, was proposed by Marinelli et al. (2018). Despite being a less studied problem, this variant leads to an increase in drone coverage, since the travel time needed to reach certain customers may be reduced by finding optimal launch and retrieval positions along arcs instead of restricting them to be customer nodes. To study the impact of this assumption, a greedy heuristic based on the GRASP proposed by Ha et al. (2018) was developed and the results evidenced reductions in terms of drone travel times and, consequently, energy consumption savings.

Using continuous approximation techniques, Carlsson and Song (2018) proved that the efficiency of the truck-drone tandem is proportional to the square root of the speeds of the truck and the drone. However, the problem analysed was the 'horsefly routing problem', which could be described as a TSP-D with *en route* operations where the truck does not perform direct deliveries. By applying continuous approximation techniques as well, Campbell et al. (2017) were able to demonstrate that the relative operating costs per mile, relative stop costs of trucks and drones and customer density affect the benefits generated by the system. Additionally, their analysis corroborated the finding of Murray and Raj (2020) that the addition of multiple drones per truck offers marginally decreasing savings. Phan et al. (2018) tested such scenario as the 'Traveling Salesman Problem with Multiple Drones' (TSP-mD) and developed an Adaptive Large Neighborhood Search (ALNS) heuristic, as well as an adaptation of the GRASP of Ha et al. (2018). While the ALNS heuristic found better results for the TSP-mD, it was beaten by the GRASP for the TSP-D.

The extension of the TSP-D to include multiple trucks, the 'Multiple Traveling Salesman

Problem with Drones', was introduced by Kitjacharoenchai et al. (2019). Through computational experiments, it was proved that this model leads to lower completion times when compared to single truck, multiple trucks and single truck-drone tandem scenarios.

2.2.2 Drone Routing as an extension of the VRP

2.2.2.1 Vehicle Routing Problem with Drones

Introduced by Wang et al. (2017), the VRPD considers the operation of a homogeneous truck fleet where each vehicle carries a set of drones to address customers whose demand is of one parcel. A worst-case analysis is presented by Wang et al. (2017) to determine the maximum savings provided by the application of drones, which depend on the number of drones per truck and the relative speed of drones compared to trucks. Poikonen et al. (2017) extended the worst-case analysis to consider scenarios where trucks and drones follow different distance metrics and where drones have limited battery life, as well as to study potential economic savings.

A formal definition of the VRPD is provided by Schermer et al. (2019b) as a MILP including a set of valid inequalities to improve solver performance. Additionally, it is presented a matheuristic algorithm that starts by solving a traditional VRP and progresses to solve the problem of assigning customers to drones and choosing the launch and retrieval locations through mathematical programming. This approach differs from the one presented by Es Yurek and Ozmutlu (2018), where customer assignments are already given before applying an exact optimization technique.

Schermer et al. (2018) presented two algorithms that apply a route first, cluster second method. The Two-Phase Heuristic focuses on improving the VRP solution before inserting drone operations and optimizing them, whereas the Single-Phase Heuristic improves truck and drone routes simultaneously. The Two-Phase Heuristic algorithm performed better, which lead the authors to claim that VRPD problems can be solved more effectively starting from good VRP solutions. Sacramento et al. (2019) presented an ALNS heuristic considering a single drone per truck, forbidding roundtrips and concerning the minimization of cost under a constraint on the maximum makespan.

The variant including *en route* operations was introduced by Schermer et al. (2019a). While Marinelli et al. (2018) had considered that launch and retrieval operations could occur in any point along an arc, the VRPD with *en route* operations considers only a set of discrete points along the arcs. Through the formulation of a MILP and an hybrid Variable Neighborhood Search (VNS) and Tabu Search algorithm, problem instances were solved effectively and it was proven that *en route* operations increase drone utilization and reduce the makespan.

A variant of the VRPD considering time-windows was introduced by Di Puglia Pugliese and Guerriero (2017) as a MILP. The authors concluded that deploying drones is not economically viable when these possess the same variable transportation cost as trucks. However, when considering the negative externalities related to the operation of trucks, e.g., congestion, noise or pollution, or the optimization of service quality metrics, the use of drones becomes justifiable.

Daknama and Kraus (2017) studied a similar problem to the VRPD where drones are not allocated to a specific truck. Additionally, drones require the time between two consecutive deliveries performed by the truck to recharge, which constitutes a more realistic assumption than the instantaneous setup normally considered.

2.2.3 Drone Routing with elements of Scheduling Problems

2.2.3.1 Parallel Drone Scheduling Traveling Salesman Problem

Amazon's last-mile delivery concept of drone-based direct warehouse-to-consumer operations was modeled as well by Murray and Chu (2015) in a problem named the 'Parallel Drone Scheduling TSP'. A PDSTSP solution divides the customers into a subset assigned to a single truck and another assigned to a fleet of identical drones based at the depot, as well as the truck route and the sequence of drone operations. Therefore, the PDSTSP can be seen as an hybrid TSP and a PMS problem with a minimal makespan objective. Compared to the FSTSP, the PDSTSP requires no movement synchronization between drones and the truck and is more appropriate for scenarios with a high customer density near the depot and within drone flight range. Along with a MILP formulation, heuristics based on local search operators that adjust the partition of customers in order to balance the drones' and the truck's workload were proposed.

An iterative two-step heuristic that alternates between the assignment and routing stages of the problem was proposed by Mbiadou Saleu et al. (2018). Given a PDSTSP solution, the algorithm repeatedly performs a coding step to transform the solution into a customer sequence and, subsequently, applies dynamic programming techniques to divide the sequence into a truck tour and drone operations, which are then optimized. A single-start version of this solution approach was able to reach the same level of solution quality within shorter computational times than the approach of Murray and Chu (2015). In addition, a multi-start configuration offered much more reduced optimality gaps with a moderate increase in runtimes.

A dynamic version of the PDSTSP was studied by Ulmer and Thomas (2018), in which orders are received dynamically throughout a working day. When a new request arrives, decisions on whether or not to address the order as a same-day delivery, vehicle assignment and routing need to be made. The solution approach proposed is based on an approximate dynamic programming technique called parametric policy function approximation, which allows to solve the vehicle assignment subproblem heuristically. While drone routing problems typically involve the minimization of makespan, this variant maximizes the expected number of same-day deliveries.

2.2.3.2 Traveling Salesman Problem with a Drone Station

Acknowledging the limitations of the PDSTSP related to the requirement that a significant proportion of customers should be located in the vicinity of the depot, Kim and Moon (2019) proposed the TSP-DS, in which the drone fleet is placed at a remote drone station. The TSP-DS shares similarities with the PDSTSP but includes the particularity that a station is activated to perform deliveries after being visited by the truck (see Figure 2.2). Kim and Moon (2019) formulated the

TSP-DS as mixed integer program, whose objective is to minimize the last delivery time. The problem assumes that the location of the station is given *a priori*, i.e., focuses on operational routing decisions. Using a decomposition of the TSP-DS into a TSP and a PMS problem, two solution approaches were presented and experimentally tested. Their results revealed that truck-drone systems modelled by the PDSTSP and the TSP-DS are more effective than a traditional truck-based system. Furthermore, the TSP-DS proved to be more useful than the PDSTSP in scenarios where the majority of customers is located far away from the depot.

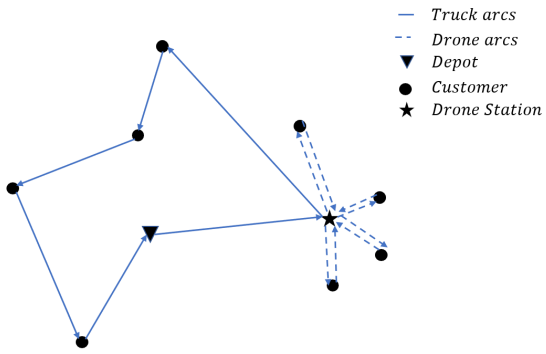

Figure 2.2: Illustration of the Traveling Salesman Problem with a Drone Station.

2.2.3.3 Traveling Salesman Drone Station Location Problem

Schermer et al. (2020c) introduced the 'Traveling Salesman Drone Station Location Problem' (TSDSLP), which generalizes the TSP-DS to include a set of potential drone station locations. Therefore, beyond truck routing and drone scheduling decisions, the problem also considers features of facility location problems. The TSDSLP seeks to minimize the makespan, similarly to the TSP-DS, or to minimize the operational cost, under a restriction on the maximum number of used stations. By comparing a setting where several drone stations are available with a traditional truck-based system, Schermer et al. (2020c) concluded that, while the FSTSP and TSP-D prefer drones with faster speed over drones with a higher flight range, in delivery systems with drone stations, a large operational range and similar speed to the truck already provides significant savings.

Ferrandez et al. (2016) studied a different problem but with some assumptions related to operational aspects that bring it closer to the TSDSLP. The problem admits that the truck moves along a TSP route and, when a launch location is reached, drones proceed to perform roundtrips to nearby customers. The proposed solution approach consisted in determining the set of launch locations using the centroids of the k-means clustering algorithm and solving the TSP route using a genetic algorithm. Therefore, this problem can be seen as a TSDSLP where the potential station sites are given by cluster centroids and where the truck needs wait for all drone deliveries to be performed before restarting its route. Chang and Lee (2018) extended the aforementioned solution approach by allowing launch locations to shift from their respective centroid with the objective of optimizing completion time. The magnitude of the shift of a given launch location along an axis that passes through the depot and the centroid was determined by solving a nonlinear programming model.

2.2.4 Other Variants

The problems and respective solution approaches reviewed hereunder consider either hybrid truck-drone or drone-only systems where drones have the possibility of performing more than one delivery per *sortie*. This assumption is less realistic than that of the TSP-D or VRPD, since drone flight ranges are reduced and its technological features typically do not allow the transportation of more than one parcel (Agatz et al., 2016).

Dorling et al. (2017) studied a drone-only VRP where multiple trips to the depot are allowed and an experimentally validated energy consumption model is applied. This model states that energy consumption varies linearly with the applied payload and battery weight. To solve this VRP variant, a MILP and a SA approach were proposed. The computational experiments evidenced that optimizing battery weight and reusing drones in multi-trips has positive economical outcomes if the cost of consumed energy and drone purchasing costs are considered.

Savuran and Karakaya (2016) analysed the cooperation between a truck moving in a predefined straight line path, viewed as a mobile depot, and a drone, the delivery vehicle. The problem is treated as a VRP with the objective of maximizing the number of visited targets by the drone within its flight endurance limit. The solution approach proposed to address the problem was an adapted version of the Genetic Algorithm. Boysen et al. (2018a) studied a problem where the truck route is known as well but in this case it is given as a sequence of customers, upon which drone operations should be scheduled. In their setting, however, only one delivery can be performed per *sortie* and multiple drones may be available on top of the truck. An interesting feature of this problem is that drone travel times are considered to be asymmetric to differentiate the speed between a loaded or unloaded flight, as well as the effect of opposing winds. Furthermore, this problem allows imposing that only roundtrips should occur, which has relevance on practical terms, as legislation in some countries requires continuous monitoring of drone flights (Otto et al., 2018).

Wang and Sheu (2019) defined a problem where docking hubs are used to assist the joint efforts of trucks and drones. These facilities act as a supporting unit to land and prepare drones for subsequent operations, as well as a place to store backup drones. If a drone has the necessary range to perform its multi-visit trip, it will depart from the station independently of a truck and complete its route. Otherwise, it will travel with a truck that visits the docking hub to be launched later on. For this problem, a mixed integer program and a branch-and-price algorithm were proposed.

A problem in which the truck assumes a purely supportive role was proposed by Agárdi et al. (2020). The problem assumes that the truck performs a route through a set of locations from where the drone will be launched to perform deliveries and, later on, be retrieved. To address the objective of minimizing the total travelled distance, a genetic algorithm was developed.

Kitjacharoenchai et al. (2020) formulated a two echelon VRP where trucks act both as primary vehicles and satellite depots and drones constitute the secondary vehicles. The proposed problem was defined as a mixed integer program that takes into account trucks' and drones' capacities to minimize the delivery completion time. Since the exact formulation was only capable of solving instances with less than 10 customers, a constructive heuristic and metaheuristic based on

Large Neighborhood Search were developed. The authors argue that, since the problem presents many constraints related to synchronization and capacity, a metaheuristic based on successive partial destructions might be more desirable than standard local search operators. Using small-sized customer instances, an average optimality gap of around 1.3% was achieved within lower computational times in comparison to the exact formulation. For larger instances, the heuristic was evaluated against benchmark capacitated VRP solutions and, even though computational times grew exponentially with the number of nodes, the algorithm computed solutions with lower objective values, thus proving the usefulness of hybrid truck-drone delivery systems.

The 'Multi-Visit Drone Routing Problem', proposed by Poikonen and Golden (2020), considers a delivery system similar to the TSP-mD where drones can perform several deliveries before rejoining the truck, drone energy consumption depends on the payload and *en-route* operations are allowed. Another assumption taken, that reduces computational requirements but represents a drawback of this problem, is that additional drones cannot be launched while others are in the middle of their *sortie*. Thus, launch and retrieval operations are forced to be concentrated in the same locations. Computational experiments evidenced that both drone speed and drone range increases have a positive effect on makespan, but drone speed has a more expressive impact.

Following the rising interest and adoption of on-demand meal delivery services, Liu (2019) studied a dynamic VRP using drones to perform meal pickups and deliveries. To address the problem, a mixed integer program was proposed, considering discrete time instants and treating space continually to allow a dynamic input of arbitrary pickup and delivery locations.

2.3 Routing Problems with Drones and Robots

The studies reviewed in this section will initially be focused on robot routing problems that involve delivery settings and that include cooperation with other transportation modes. Such restriction is reasonable since the role of robot routing studies on the proposition of the TSP-D-RS and, consequently, the VRPD-RS, is negligible, given that they are supported on both TSP-D and TSP-DS studies. Moreover, problems that specifically combine drones and robots, which is the case of the present study, will be discussed.

Sonneberg et al. (2019) introduced a location routing problem where the best subset of stations must be selected in order to address customer requests within the agreed delivery time-windows, while minimizing operational costs. The stations house a fleet of range-constrained robots capable of delivering to more than one customer per route. The problem admits that a customer may order more than one package and be served by one or more robots, i.e., split deliveries are allowed. A MILP was proposed and, by performing a sensitivity analysis, it was demonstrated that the number of compartments is a crucial factor to dictate the cost-efficiency of robot-based deliveries. In fact, more compartments allow more customers to be served in a single route, thus leading to less returns to the stations. However, among the three main delivery robot manufacturers, only one currently produces multi-compartment devices (Sonneberg et al., 2019).

A problem that partially resembles the TSP-D-RS and VRPD-RS is proposed by Boysen et al. (2018b). Here, instead of carrying drones, a truck carries robots that are deployed throughout its route to address a single customer and return to a given robot depot in the delivery area. Besides launching robots, a truck can deliver packages to these robot depots so that the remaining customers can be addressed. The objective of the introduced problem is to minimize the weighted number of late deliveries instead of the makespan. The authors chose this metric because makespan is more applicable in a drone-system setting where deliveries can be performed in an unattended manner, whereas robots require that the customer unlocks their compartment. For this problem a mixed integer program was proposed, along with an algorithm that schedules robot operations for a given truck route, as well as a multi-start local search procedure. In this study, it was proven that the proposed concept results in a better performance than having a scenario where the truck waits to retrieve the previously launched robots.

The problem proposed in the present study constitutes a generalization of the 'Drone-Assisted Traveling Salesman Problem with Robot Stations', introduced by Schermer et al. (2020b). The TSP-D-RS models the operation of a delivery system composed of a truck and a drone working synchronously, as well as a set of satellite stations that store robots and that may be activated to perform deliveries (see Figure 2.3). For the TSP-D-RS, a MILP formulation was proposed, considering both makespan and operational cost minimization objectives. Through numerical studies, it was proven that the makespan objective was more computationally demanding, with the MILP solver already failing to reach optimality within the defined running time for instances with 15 customers. The cost objective, however, could be solved optimally within reasonable computational times for instances with up to 20 customers. Additionally, the authors verified that optimizing the makespan is aligned with reductions in the operational cost, while optimizing the cost might lead to significant increases in the makespan. The reason for the latter, which is also the reason behind a better performance of the solver for cost minimization problems, is related to the fact that the cost definition does not take into account the waiting times of the vehicles. Therefore, better solutions addressing the cost objective will push deliveries to be performed by the cheapest delivery modes with respect to variable transportation costs, i.e., the drone and robots, and will minimize the distance covered by the truck. In fact, the collaboration of the truck and the drone in this scenario resumes mostly to the occurrence of roundtrips, i.e., the truck remains stationary while the drones perform back and forth trips to address nearby customers, which leads to large truck waiting times and, consequently, increased makespans (Schermer et al., 2020b). The makespan objective, however, is marked by the presence of multi-leg trips to increase the parallelization of tasks, which is more computationally demanding due to synchronization constraints.

Moeini and Salewski (2020) introduced the 'Truck-Drone-ATV Routing Problem', which describes the operation of a truck that performs a supportive role for the autonomous vehicles it carries on-board, drones and robots. The problem admits the existence of predefined grid points, which may be traversed by the truck at most once and where the launch and retrieval operations of vehicles may occur. Additionally, a differentiation between light and heavy parcels is established to highlight the benefit of considering two distinctive autonomous delivery modes, where

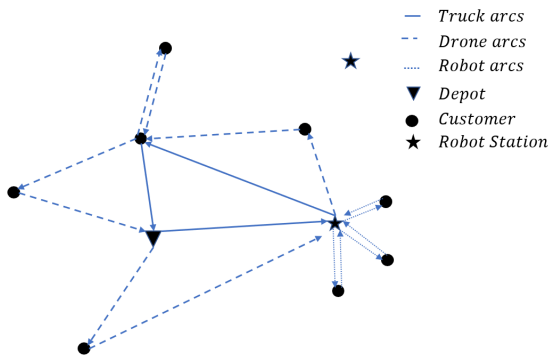

Figure 2.3: Illustration of the Drone-Assisted Traveling Salesman Problem with Robot Stations.

the robots are more energy efficient and able to carry higher payloads than drones (Moeini and Salewski, 2020). To address this problem, an algorithm based on the Lin-Kernighan heuristic, to determine the truck TSP-route through grid points, as well as a genetic algorithm, to schedule the operations of the autonomous vehicles, was presented.

Chapter 3

Problem Definition

In this chapter, the VRPD-RS will be presented. Initially, a description of the delivery system modeled by the VRPD-RS and the assumptions related to its mode of operation will be provided. Subsequently, the problem will be formally defined as a mixed integer linear program and the changes relative to the TSP-D-RS will be discussed.

3.1 The Drone-Assisted VRP with Robot Stations

The VRPD-RS can be regarded as an extension of the TSP-D-RS, proposed by Schermer et al. (2020b), to include multiple trucks and drones. Additionally, one could also state that the problem combines the VRPD and the TSP-DS introduced by Wang et al. (2017) and Kim and Moon (2019), respectively. Indeed, the VRPD-RS considers the operation of several trucks working in close collaboration with drones, as well as a fleet of delivery robots stored in available satellite stations. The objective of the VRPD-RS may be to utilize these resources with the purpose of minimizing the makespan, i.e., the latest arrival of every vehicle to its initial position, or the operational cost, defined as a function of the total distance travelled by the vehicles. A visual representation of the system depicted by the VRPD-RS is given in Figure 3.1.

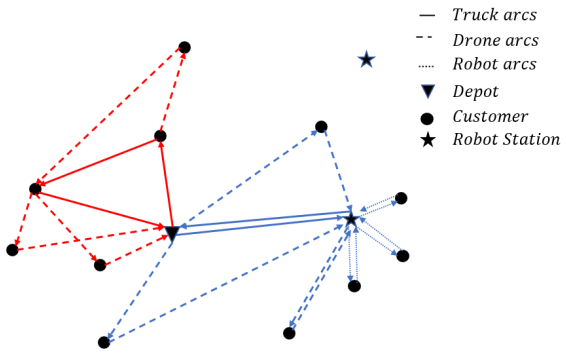

Figure 3.1: Illustration of the Drone-Assisted Vehicle Routing Problem with Robot Stations.

Since the VRPD-RS is closely related to the TSP-D-RS, a significant share of the assumptions of both problems is identical. However, the extension to consider a fleet of both trucks and drones requires specifying additional operational aspects. The assumptions included in the VRPD-RS are the following:

- A fleet of $|F| \in \mathbb{Z}^+$ homogeneous trucks is available, where each truck carries a fleet of $|D| \in \mathbb{Z}_0^+$ homogeneous drones;
- Each truck has a carrying capacity of $L \in \mathbb{Z}^+$ parcels;
- The drones require $\bar{t}_l$ and $\bar{t}_r$ time units to be launched and retrieved, respectively;
- A drone can be launched from customer nodes to perform at most one delivery and, later on, needs to return to the same truck;
- If a truck launches or retrieves more than one drone at a customer node, it is capable of performing the respective launching and retrieval setup operations in parallel;
- A set of $m \in \mathbb{Z}_0^+$ robot stations, each one housing a fleet of $|K| \in \mathbb{Z}_0^+$ homogeneous robots, is available and the maximum allowable number of active stations is given by $C \in \{0, ..., m\}$;
- Similarly to drones, robots can carry at most one parcel and need to return to the station to be resupplied and recharged for possible subsequent deliveries;
- Drones and robots have a limited endurance of ε_d and ε_r time units, respectively, and it is admitted that their batteries are recharged or swapped instantaneously;
- Trucks are restricted to the road network, whereas drones can move as the crow flies and robots may travel through roads or pedestrian areas;
- A station is activated for delivery once it is visited by a truck. Furthermore, the station should be activated by only one truck, i.e., split deliveries to the station are not allowed.

In order to mathematically formulate the VRPD-RS, it is necessary to establish a notation that describes the problem entities. The notation resembles the one proposed by Schermer et al. (2020b), except for the elements that describe the truck and drone fleets, as well as robot deliveries. The VRPD-RS admits that demand locations are known, as well as the location of the depot and potential robot station locations. These locations form a set of vertices V and a set of edges E, that connects every pair of vertices, included in a graph $\mathcal{G} = (V, E)$. The different nature of the elements in V leads to the definition of the following disjoint subsets:

- $\{0, n+1\} \subset V$ that denotes the depot at the start and end of a tour, respectively;
- $V_N = \{1, ..., n\} \subset V$ represents customer nodes. Additionally, $V_D \subseteq V_N$ includes customers that may be served by any vehicle in the system. Such subset is defined because some customers, belonging to $V_N \setminus V_D$, must be served by a truck, e.g., the parcel may exceed the maximum admissible payload or a signature may be required (Murray and Chu, 2015);
- $V_S = \{s_1, ..., s_m\} \subset V$ groups the potential robot station locations.

Note that $V = \{0\} \cup V_N \cup V_S \cup \{n+1\}$, where $0 \equiv n+1$. To simplify the expressions included in the formulation, subset $V_L = V \setminus \{n+1\}$, which marks the possible nodes from where a drone can be launched, and $V_R = V \setminus \{0\}$, specifying the locations from where the drone can be retrieved, are defined. A drone operation, or *sortie*, is characterized by a triple $(i, w, j) \in P$, where P denotes the set of all feasible operations. A feasible *sortie* is given by the following characteristics:

- Location i, which indicates from where the drone departs, belongs to V_L;
- Customer w, also referred to as the drone node, belongs to V_D and must differ from i;
- Location j, where the drone is retrieved, belongs to V_R and differs from w. If $i = j$, the operation is referred to as a cyclic operation or roundtrip. Conversely, if $i \neq j$, the operation is acyclic or multi-leg.

Finally, parameters d_{ij}, $\overline{d}_{ij}$ and $\widetilde{d}_{ij}$ represent the distance between vertices i and j for trucks, drones and robots, respectively. Likewise, parameters t_{ij}, $\overline{t}_{ij}$ and $\widetilde{t}_{ij}$ define the time needed to travel from i to j by each delivery mode, accordingly.

3.2 Mathematical Model

The VRPD-RS will be formulated as a mixed integer linear program according to two distinct minimization objectives: makespan and operational cost. The decision variables involved in both problems are the same and are defined in Table 3.1. Note that, besides the binary variables that specify routing elements, four groups of non-negative continuous variables are declared to ensure the temporal synchronization of the VRPD-RS.

Table 3.1: Decision variables of the VRPD-RS.

$\tau \in \mathbb{R}_0^+$	defines the makespan.	$\omega \in \mathbb{R}_0^+$	defines the operational cost.
$x_{ij}^v \in \{0,1\}$ $\forall i \in V_L, j \in V_R,$ $v \in F, i \neq j$	is equal to 1, iff truck v traverses arc (i, j).	$a_i^v \in \mathbb{R}_0^+$ $\forall i \in V$	represents the arrival time of truck v at i.
$y_{iw}^{vd} \in \{0,1\}$ $\forall i \in V_L, w \in V_D,$ $v \in F, d \in D, i \neq w$	is equal to 1, iff drone d from truck v travels from i to w in an acyclic operation.	$s_i^v \in \mathbb{R}_0^+$ $\forall i \in V$	represents the earliest departure time of truck v from i.
$y'^{vd}_{wj} \in \{0,1\}$ $\forall w \in V_D, j \in V_R,$ $v \in F, d \in D, w \neq j$	is equal to 1, iff drone d from truck v returns to j from w in an acyclic operation.	$\overline{r}_i^{vd} \in \mathbb{R}_0^+$ $\forall i \in V$	represents the earliest release time of drone d from truck v at i.
$\hat{y}_{iw}^{vd} \in \{0,1\}$ $\forall i \in V_L, w \in V_D,$ $v \in F, d \in D, i \neq w$	is equal to 1, iff drone d from truck v serves w in a roundtrip from i.	$\overline{s}_i^{vd} \in \mathbb{R}_0^+$ $\forall i \in V$	represents the earliest departure time of drone d from truck v from i.
$z_{sw}^{vk} \in \{0,1\}$ $\forall s \in V_S, w \in V_D,$ $v \in F, k \in K$	is equal to 1, iff robot k from station s, supplied by truck v, serves w.	$y_i^{vd} \in \{0,1\}$ $\forall i \in V$	is equal to 1, iff drone d from truck v is available to begin an operation at i.

3.2.1 Minimum Makespan VRPD-RS

The formulation of the minimum makepan VRPD-RS is given by the set of Expressions ranging from 3.1 to 3.31. The objective of the problem is given by Expression 3.1 and consists in the minimization of makespan. The makespan consists in the duration of the longest route performed by the vehicles. Therefore, Constraints 3.2 and 3.3 impose lower bounds for the makespan associated with the entities that can dictate its value: truck-drone tandems and robots, respectively. In fact, the makespan can be the latest arrival time of a truck at the depot or of a robot at its station.

$$min\ \tau \tag{3.1}$$

s.t.

$$s^v_{n+1} \leq \tau\ ,\ \forall v \in F \tag{3.2}$$

$$\sum_{v\in F} a^v_s + \sum_{v\in F}\sum_{w\in V_D} \left(\widetilde{t}_{sw} + \widetilde{t}_{ws}\right)\cdot z^{vk}_{sw} \leq \tau\ ,\ \forall s \in V_S,\ k \in K \tag{3.3}$$

Constraints 3.4 and 3.5 govern the flow of the trucks. Specifically, Constraints 3.4 explicit that each truck performs solely one route that begins and ends at the depot. Additionally, for each node reached by the truck, there should be conservation of flow, i.e., the truck should depart from it.

$$\sum_{j\in V_R} x^v_{0j} = \sum_{i\in V_L} x^v_{i,n+1} = 1\ ,\ \forall v \in F \tag{3.4}$$

$$\sum_{\substack{i\in V_L \\ i\neq h}} x^v_{ih} = \sum_{\substack{j\in V_R \\ h\neq j}} x^v_{hj}\ ,\ \forall h \in V_N \cup V_S,\ v \in F \tag{3.5}$$

Customers that can only be served by truck must have one truck assigned to them, which is imposed by Constraints 3.6. Constraints 3.7 state that the more flexible customers must be served exactly once, possibly either by a truck, a drone on a cyclic or an acyclic operation, or a robot.

$$\sum_{v\in F}\sum_{\substack{i\in V_L \\ i\neq w}} x^v_{iw} = 1\ ,\ \forall w \in V_N \setminus V_D \tag{3.6}$$

$$\sum_{v\in F}\sum_{\substack{i\in V_L \\ i\neq w}} x^v_{iw} + \sum_{v\in F}\sum_{d\in D}\sum_{\substack{i\in V_L \\ i\neq w}} \left(y^{vd}_{iw} + \hat{y}^{vd}_{iw}\right) + \sum_{v\in F}\sum_{s\in V_S}\sum_{k\in K} z^{vk}_{sw} = 1\ ,\ \forall w \in V_D \tag{3.7}$$

Constraints ranging from 3.8 to 3.9 characterize the synchronization between drones and their respective truck. More precisely, Constraints 3.8 indicate that a multi-leg flight cannot begin at i if the truck does not pass through that node. The same applies for roundtrips, as specified by Constraints 3.9. Moreover, as indicated by Constraints 3.10, a multi-leg flight cannot end at j if the truck does not reach that node to retrieve the drone. Constraints 3.11 establish flow conservation

for drone-serviced customers.

$$y_{iw}^{vd} \leq \sum_{\substack{j \in V_R \\ j \neq i}} x_{ij}^{v} \ , \ \forall i \in V_L,\ w \in V_D,\ v \in F,\ d \in D,\ i \neq w \tag{3.8}$$

$$\hat{y}_{iw}^{vd} \leq \sum_{\substack{j \in V_R \\ j \neq i}} x_{ij}^{v} \ , \ \forall i \in V_L,\ w \in V_D,\ v \in F,\ d \in D,\ i \neq w \tag{3.9}$$

$$y'^{vd}_{wj} \leq \sum_{\substack{i \in V_L \\ i \neq j}} x_{ij}^{v} \ , \ \forall j \in V_R,\ w \in V_D,\ v \in F,\ d \in D,\ w \neq j \tag{3.10}$$

$$\sum_{\substack{i \in V_L \\ i \neq w}} y_{iw}^{vd} = \sum_{\substack{j \in V_R \\ j \neq w}} y'^{vd}_{wj} \ , \ \forall w \in V_D,\ v \in F,\ d \in D \tag{3.11}$$

Constraints 3.12 impose that the earliest departure time of a truck occurs after its arrival time at a given node. Additionally, Constraints 3.13 indicate that, if a truck travels along arc (i, j), j will be reached in t_{ij} time units after having departed from i.

$$a_i^v \leq s_i^v \ , \ \forall i \in V,\ v \in F \tag{3.12}$$

$$M \cdot \left(x_{ij}^v - 1\right) + s_i^v + t_{ij} \leq a_j^v \ , \ \forall i \in V_L,\ j \in V_R,\ v \in F,\ i \neq j \tag{3.13}$$

Similarly to Constraints 3.13, the sets of Inequalities 3.14 and 3.15 dictate the relationship between drone time variables in the first and second legs of acyclic operations, respectively. Constraints 3.14 establish a lower bound on the earliest release time of drones at drone-serviced customer nodes and specify that a drone can only be released at w after having been prepared to launch and traveling across arc (i,w). Similarly, a lower bound on the earliest release time of drones at retrieval nodes is given by Constraints 3.15, which indicate that a drone can be released in a truck after having travelled along arc (w, j) and received the retrieval setup operations.

$$M \cdot \left(y_{iw}^{vd} - 1\right) + \bar{s}_i^{vd} + (\bar{t}_l + \bar{t}_{iw}) \leq \bar{r}_w^{vd} \ , \ \forall i \in V_L,\ w \in V_D,\ v \in F,\ d \in D,\ i \neq w \tag{3.14}$$

$$M \cdot \left(y'^{vd}_{wj} - 1\right) + \bar{s}_w^{vd} + (\bar{t}_{wj} + \bar{t}_r) \leq \bar{r}_j^{vd} \ , \ \forall w \in V_D,\ j \in V_R,\ v \in F,\ d \in D,\ w \neq j \tag{3.15}$$

Constraints 3.16 state that the earliest departure time of a drone at a given node occurs right after having been released there and all roundtrips starting from that node have been performed. The temporal synchronization between trucks and drones upon launches and retrievals is assured by Constraints 3.17 and 3.18, respectively. In what concerns launching operations, Constraints 3.17 establish a lower bound for the earliest departure time of a truck by defining that it can only restart its route after having prepared the drones to begin their multi-leg operations. Since the truck is capable of performing the setups for launch simultaneously, the earliest departure time of the truck will be given by the latest departure of a drone at the vertex. For retrieval operations, Constraints 3.18 dictate that any drone can be released at the truck t_r time units after the arrival of the truck at

the node. Again, retrieval operations can be performed simultaneously and, therefore, the earliest release time of each drone depends on the value of a_j^v.

$$\bar{r}_i^{vd} + \sum_{\substack{w \in V_D \\ i \neq w}} (\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wi} + \bar{t}_r) \cdot \hat{y}_{iw}^{vd} \leq \bar{s}_i^{vd} \ , \ \forall i \in V, \ v \in F, \ d \in D \tag{3.16}$$

$$\bar{s}_i^{vd} + \bar{t}_l \cdot \sum_{\substack{w \in V_D \\ i \neq w}} y_{iw}^{vd} \leq s_i^v \ , \ \forall i \in V, \ v \in F, \ d \in D \tag{3.17}$$

$$a_j^v + \bar{t}_r \cdot \sum_{\substack{w \in V_D \\ w \neq j}} y'^{vd}_{wj} \leq \bar{r}_j^{vd} \ , \ \forall j \in V, \ v \in F, \ d \in D \tag{3.18}$$

Constraints 3.19 and 3.20 establish that multi-leg and roundtrip operations, respectively, must not exceed the endurance of the drone. Note that, in multi-leg operations, drone waiting times can occur if the drone is expected to arrive at j sooner than $a_j^v + \bar{t}_r$. In such event, since it is desirable to have the sum of $\bar{t}_{iw} + \left(\bar{r}_j^{vd} - \bar{t}_r - \bar{s}_w^{vd}\right)$ below ε_d for a given operation (i, w, j), the solver will seek to minimize the difference between $\left(\bar{r}_j^{vd} - \bar{t}_r - \bar{s}_w^{vd}\right)$. Thus, Constraints 3.15 will be tight and, since Constraints 3.16 only impose a lower bound on $\bar{s}_w^{vd}$, this variable will exceed $\bar{r}_w^{vd}$ in the amount of potential waiting time units. Therefore, since Constraints 3.20 do not include this slack in the first side of the inequality, drone waiting times do not consume energy. Further considerations on this assumption will be discussed in Section 3.3.

$$(\bar{t}_{iw} + \bar{t}_{wi}) \cdot \hat{y}_{iw}^{vd} \leq \varepsilon_d \ , \ \forall i \in V_L, \ w \in V_D, \ v \in F, \ d \in D, \ i \neq w \tag{3.19}$$

$$\sum_{\substack{i \in V_L \\ i \neq w \\ i \neq j}} \bar{t}_{iw} \cdot y_{iw}^{vd} + \left(\bar{r}_j^{vd} - \bar{t}_r - \bar{s}_w^{vd}\right) \leq \varepsilon_d + M \cdot \left(1 - y'^{vd}_{wj}\right) \ , \ \forall w \in V_D, \ j \in V_R, \ v \in F, \ d \in D, \ w \neq j \tag{3.20}$$

Constraints 3.21 define an upper bound on the availability of a drone at node j, by keeping track of all operations related to a given arc (i, j) and the availability in the previous node i.

$$\left(1 - x_{ij}^v\right) + y_i^{vd} + \sum_{\substack{w \in V_D \\ i \neq w \\ w \neq j}} \left(-y_{iw}^{vd} + y'^{vd}_{wj}\right) \geq y_j^{vd} \ , \ \forall i \in V_L, \ j \in V_R, \ v \in F, \ d \in D, \ i \neq j \tag{3.21}$$

Constraints 3.22 through 3.24 define that, if a drone is available at a given vertex, it can begin or end a multi-leg operation, as well as perform at most n roundtrip operations. This last scenario would occur if the drone had sufficient endurance to address all customers from the depot.

$$\sum_{\substack{w \in V_D \\ i \neq w}} y_{iw}^{vd} \leq y_i^{vd} \ , \ \forall i \in V_L, \ v \in F, \ d \in D \tag{3.22}$$

$$\sum_{\substack{w \in V_D \\ w \neq j}} y'^{vd}_{wj} \leq y_j^{vd} \ , \ \forall j \in V_R, \ v \in F, \ d \in D \tag{3.23}$$

$$\sum_{\substack{w \in V_D \\ i \neq w}} \hat{y}_{iw}^{vd} \leq n \cdot y_i^{vd} \ , \ \forall i \in V_L, \ v \in F, \ d \in D \tag{3.24}$$

A station is considered to be activated if a truck reaches it. Therefore, the assumption that stations' demand cannot be split among trucks is expressed by Constraints 3.25. Additionally, Constraints 3.26 impose that the number of active stations should not surpass the maximum number allowed.

$$\sum_{v \in F} \sum_{\substack{i \in V_L \\ i \neq s}} x_{is}^v \leq 1 \ , \ \forall s \in V_S \tag{3.25}$$

$$\sum_{v \in F} \sum_{i \in V_L} \sum_{\substack{s \in V_S \\ i \neq s}} x_{is}^v \leq C \tag{3.26}$$

Constraints 3.27 indicate that at most n robot deliveries can be performed at a station if it is activated by a given truck v. The number of deliveries would be n if the truck went directly to the station from the depot and unloaded all n parcels to be delivered by its robots. Similarly to drones, Constraints 3.28 guarantee that operations performed by robots do not surpass their endurance.

$$\sum_{k \in K} \sum_{w \in V_D} z_{sw}^{vk} \leq n \cdot \sum_{\substack{i \in V_L \\ i \neq s}} x_{is}^v \ , \ \forall s \in V_S, \ v \in F \tag{3.27}$$

$$\left(\widetilde{t}_{sw} + \widetilde{t}_{ws}\right) \cdot \sum_{v \in F} z_{sw}^{vk} \leq \varepsilon_r \ , \ \forall k \in K, \ s \in V_S, \ w \in V_D \tag{3.28}$$

The VRPD-RS assumes that customers demand one parcel. Therefore, the load carried by a given truck is a discrete variable equal to the number of customers addressed by either the truck directly, associated drones or robots from stations it supplies. Constraints 3.29 ensures that the load carried by each truck does not exceed its capacity.

$$\sum_{i \in V_L} \sum_{\substack{j \in V_N \\ i \neq j}} x_{ij}^v + \sum_{d \in D} \sum_{i \in V_L} \sum_{\substack{w \in V_D \\ i \neq w}} \left(y_{iw}^{vd} + \hat{y}_{iw}^{vd}\right) + \sum_{s \in V_S} \sum_{k \in K} \sum_{w \in V_D} z_{sw}^{vk} \leq L \ , \ \forall v \in F \tag{3.29}$$

The expressions presented so far would be sufficient to characterize the VRPD-RS. However, in order to enhance solver performance, the approach of Schermer et al. (2020b) will be followed with the addition of valid inequalities. Valid Inequalities 3.30 indicate that the total time spent by the each truck travelling and waiting for the drones at each vertex is a lower bound on the makespan. Additionally, Valid Inequalities 3.31 provide lower bounds for the makespan given by the total time travelled by drones, including the setup operations performed on them.

$$\sum_{i \in V_L} \sum_{\substack{j \in V_R \\ i \neq j}} t_{ij} \cdot x_{ij}^v + \sum_{i \in V} (s_i^v - a_i^v) \leq \tau \ , \ \forall v \in F \tag{3.30}$$

$$\sum_{i \in V_L} \sum_{\substack{w \in V_D \\ i \neq w}} (\bar{t}_l + \bar{t}_{iw}) \cdot y_{iw}^{vd} + \sum_{w \in V_D} \sum_{\substack{j \in V_R \\ w \neq j}} (\bar{t}_{wj} + \bar{t}_r) \cdot y'^{vd}_{wj} + \\ + \sum_{i \in V_L} \sum_{\substack{w \in V_D \\ i \neq w}} (\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wi} + \bar{t}_r) \cdot \hat{y}_{iw}^{vd} \leq \tau \ , \ \forall v \in F, \ d \in D \tag{3.31}$$

3.2.2 Minimum Operational Cost VRPD-RS

The VRPD-RS may be concerned with achieving the minimal operational cost, which is a function of the travelled distance by each delivery mode. Considering c_t, c_d, $c_r \in \mathbb{R}^+$ to be the variable transportation cost per mile of trucks, drones and robots, respectively, the objective function in this context is given by Expression 3.32.

$$min \ \omega = c_t \cdot \sum_{v \in F} \sum_{i \in V_L} \sum_{\substack{j \in V_R \\ i \neq j}} d_{ij} \cdot x_{ij} + c_d \cdot \sum_{v \in F} \sum_{d \in D} \sum_{i \in V_L} \sum_{\substack{w \in V_D \\ i \neq w}} \left[(\bar{d}_{iw} + \bar{d}_{wi}) \cdot \hat{y}_{iw}^{vd} + \bar{d}_{iw} \cdot y_{iw}^{vd} \right] + \\ + \sum_{v \in F} \sum_{d \in D} \sum_{w \in V_D} \sum_{\substack{j \in V_R \\ w \neq j}} \bar{d}_{wj} \cdot y'^{vd}_{wj} + c_r \cdot \sum_{v \in F} \sum_{s \in V_S} \sum_{k \in K} \sum_{w \in V_D} \left(\widetilde{d}_{sw} + \widetilde{d}_{ws} \right) \cdot z_{sw}^{vk} \tag{3.32}$$

Similarly to the minimum makespan VRPD-RS, the minimum operational cost variant is subject to Constraints 3.2 through 3.31.

3.3 Changes relative to the TSP-D-RS

Since the VRPD-RS is an extension of the TSP-D-RS, the majority of the constraints presented in Section 3.2 are extended directly from the model proposed by Schermer et al. (2020b). Nevertheless, the operation of several trucks and drones requires making assumptions regarding the mode in which they operate, which affects how some constraints are defined. Additionally, the VRPD-RS introduces a subtle change with respect to endurance restrictions during multi-leg operations.

3.3.1 Extension to a Multi-Vehicle Setting

The VRPD-RS arises from a will to study the TSP-D-RS in a context where multiple trucks and drones are available. In order to do so, it is necessary to identify the composition of the fleet, i.e., the number of trucks and drones available. Additionally, the relationship between trucks and drones, i.e., establishing if a drone is assigned to a truck or if it can be shared, needs to be declared. Finally, considerations regarding launch and retrieval operations need to be done. On the one hand, drone setup operations could be performed in parallel, e.g., the trucks are equipped with automated platforms capable of switching or recharging the battery of drones. On the other hand, only one drone could be assisted or the drones would have to form a queue in a given location, e.g., the operations require the assistance of the truck driver (Otto et al., 2018).

Since the VRPD-RS is a novel problem related to drone routing literature that appeared with the introduction of the FSTSP by Murray and Chu (2015), it would be desirable to include the assumptions that the majority of past related works adopted. To better decide on the operational characteristics of the system, the studies included in Table 3.2 were analyzed.

Table 3.2: Review of assumptions related to the operation of several trucks and drones.

Publication	**Truck Fleet Size**	**Drone Fleet Size**	**Truck-Drone Relationship**	**Setup Operations**
Di Puglia Pugliese and Guerriero (2017)	k	$k \times 1$	assigned	negligible setup times
Kitjacharoenchai et al. (2019)	m	≥ 1	shared	at most 1 setup per node
Kitjacharoenchai et al. (2020)	K	D	assigned	at most 1 setup per node
Murray and Raj (2020)	1	$\|V\|$	assigned	formation of queue
Phan et al. (2018)	1	m	assigned	parallel setups
Poikonen et al. (2017)	m	$m \times k$	assigned	negligible setup times
Sacramento et al. (2019)	m	$m \times 1$	assigned	setup of a single drone
Schermer et al. (2018)	m	$m \times k$	assigned	negligible setup times
Schermer et al. (2019a)	$\|K\|$	$\|K\| \times \|D\|$	assigned	negligible setup times
Schermer et al. (2019b)	$\|K\|$	$\|K\| \times \|D\|$	assigned	parallel setups
Wang et al. (2017)	m	$m \times k$	assigned	negligible setup times
Wang and Sheu (2019)	K	D	shared	negligible setup times
This study	$\|F\|$	$\|F\| \times \|D\|$	assigned	parallel setups

Among the studies considered, there is a tendency to associate a fleet of drones to each truck, which lead to the establishment of a fleet of $|F|$ homogeneous trucks with each carrying $|D|$ homogeneous drones. Regarding setup operations, several studies assume the setup times to be negligible. However, since the TSP-D-RS considers the overhead times to launch and retrieve nodes, the VRPD-RS should adopt this characteristic. Among the most adopted approaches, Phan et al. (2018) and Schermer et al. (2019b) consider that these setup operations can be executed in parallel, whereas Kitjacharoenchai et al. (2019) and Kitjacharoenchai et al. (2020) suggest that only one launch or one retrieval occur in a given location. Since the studies where a single operation is allowed differ substantially from VRPD variants, the assumption of allowing parallel launches and retrievals was taken. Thus, the constraints that govern the synchronization of trucks and drones while separating or joining resemble the formulation presented by Schermer et al. (2019b).

Another modification caused by the inclusion of several trucks in the system relates to the characterization of robot operations. In the TSP-D-RS, a drone operation is characterized by a triple (k,s,w), where robot k from station s addresses customer w. However, while for the TSP-D-RS only one vehicle is capable of activating a given station, in the context of the VRPD-RS it is necessary to identify the truck assigned to activate a station. In order to avoid the definition of another set of binary decision variables to encode that assignment, the binary variables that characterize robot operations were attributed an additional index v. Therefore, in the VRPD-RS, a drone operation is given by a quadruple (k,s,w,v).

3.3.2 Modelling Drone Endurance Constraints

Throughout the conduction of the present study, a special focus was given to the temporal synchronization aspects of the problem. More specifically, the synchronization of multi-leg flights and the restriction on maximum flight times of such operations was carefully analyzed. As it was already mentioned, as part of the definition of the VRPD-RS as a MILP, it was considered that drone waiting times do not count as dissipating energy. In a real setting, this would mean that drones would be allowed to wait on the ground in case they arrived at the retrieval location earlier. This assumption follows the strategy adopted by Schermer et al. (2020b). However, Constraints 3.20 were modelled in a slightly different manner from how a direct extension from their work would resemble. Constraints 3.33 exemplify how such extension would be expressed.

$$\sum_{\substack{i \in V_L \\ i \neq w \\ i \neq j}} \bar{t}_{iw} \cdot y_{iw}^{vd} + \left(\bar{r}_j^{vd} - \bar{s}_w^{vd} \right) \leq \varepsilon_d + M \cdot \left(1 - y'^{vd}_{wj} \right) \ , \forall w \in V_D,\ j \in V_R,\ v \in F,\ d \in D,\ w \neq j \quad (3.33)$$

As it can be observed, the authors considered the duration of the second part of the multi-leg to be of $\bar{r}_j^{vd} - \bar{s}_w^{vd}$, which, by analysing constraints 3.15, is equivalent to $\bar{t}_{wj} + \bar{t}_r$. However, by looking back at how Constraints 3.19, relative to the maximum endurance in roundtrips, were modelled, one concludes that the retrieval time was not assumed to be dissipating energy. Therefore, for the sake of consistency, it was decided that these overhead times do not dissipate energy.

Another possibility to model the maximum flight time during multi-leg operations would be considering that drone waiting times consume energy, e.g., the drones hovers in the air while waiting (Murray and Raj, 2020). This option is worth mentioning because it is more realistic and it was an assumption considered while implementing the solution approach proposed in the present study. Ultimately, this assumption influenced the adopted solution representation structures and mechanisms to assess solution feasibility.

To model this assumption, one modification relative to the MILP formulation proposed in Section 3.2 would be to alter Constraints 3.20 to take into account the waiting times, denoted by the slack between the time variables at customer w. Thus, the endurance constraints for multi-leg operations would be given by Expression 3.34.

$$\sum_{\substack{i \in V_L \\ i \neq w \\ i \neq j}} \bar{t}_{iw} \cdot y_{iw}^{vd} + \left(\bar{r}_j^{vd} - \bar{t}_r - \bar{r}_w^{vd} \right) \leq \varepsilon_d + M \cdot \left(1 - y'^{vd}_{wj} \right) \ , \forall w \in V_D,\ j \in V_R,\ v \in F,\ d \in D,\ w \neq j \quad (3.34)$$

Constraints 3.34 use variable $\bar{r}_w^{vd}$ instead of $\bar{s}_w^{vd}$ to take into account the waiting times. However, given that the relationship between the time variables involved in multi-leg flights define solely lower bounds, the slack that previously occurred between the time variables at w in the original formulation could occur in a different set of constraints and might not be traced by Constraints 3.34. Figure 3.2 illustrates the interaction between the time variables.

In case the drone needs to wait for its truck, Constraints 3.18 will have no slack and $\bar{r}_j^{vd}$ will

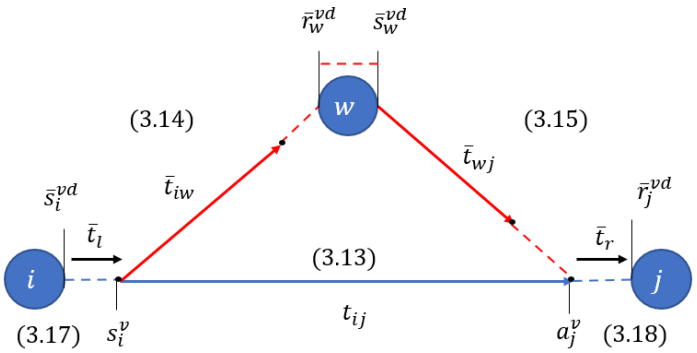

Figure 3.2: Illustration of the dependencies between time variables in multi-leg operations.

be equal to $a_j^v + \bar{t}_r$. Additionally, given the configuration of the modified maximum endurance constraints in multi-legs 3.34, the solver will seek to minimize the difference between $\bar{r}_j^{vd}$ and $\bar{r}_w^{vd}$. Therefore, on one hand, Constraints 3.15 will not present slack as well and $\bar{s}_w^{vd}$ will be strictly equal to $\bar{r}_j^{vd} - \bar{t}_r - \bar{t}_{wj}$. On the other hand, since Constraints 3.14 only impose a lower bound for $\bar{r}_w^{vd}$, this variable will shift further away in time to assume its upper bound, $\bar{s}_w^{vd}$. Thus, the waiting time will then shift to Constraints 3.14 as the slack in the inequality and would not be considered in the modified maximum endurance restriction. To overcome this situation, a new set of constraints capable of fixing $\bar{r}_w^{vd}$ to be strictly equal to $\bar{s}_i^{vd} + \bar{t}_l + \bar{t}_{iw}$ would need to be added. Such solution is represented by Constraints 3.35.

$$\bar{r}_w^{vd} \leq \bar{s}_i^{vd} + (\bar{t}_l + \bar{t}_{iw}) + M \cdot \left(1 - y_{iw}^{vd}\right) \ , \ \forall i \in V_L,\ w \in V_D,\ v \in F,\ d \in D,\ i \neq w \tag{3.35}$$

While Constraints 3.14 imposed a lower bound on $\bar{r}_w^{vd}$, Constraints 3.35 would establish the same value to be the upper bound of the variable. The option of adding this set of constraints instead of formulating the restriction on the maximum endurance during multi-legs for a given triple (i, w, j) was taken to avoid increasing the computational requirements of the MILP formulation.

Even though assuming that the drone hovers while waiting for a truck is more realistic and the algorithm was developed with such consideration, the MILP formulation presented and computational experiments were not performed under such case. The reason for this decision was based on the fact that the MILP solver used, Gurobi Optimizer 9.0.2 (Gurobi Optimization LLC, 2020), presents an integer feasibility tolerance which impedes the establishment of an equality through the use of Constraints 3.14 and 3.35. In fact, by default, integer and binary variables can assume a value differing from the nearest integer by a deviation of order 10^{-6}. Thus, by multiplying an arbitrarily large number M by $\left(1 - y_{iw}^{vd}\right)$ or $\left(y_{iw}^{vd} - 1\right)$, where y_{iw}^{vd} can assume a value slightly below or above 1, may lead to the creation of an interval where $\bar{r}_w^{vd}$ can vary, instead of the desired value. This problem could be addressed by finding an appropriate value for M. However, since the goal of this study is not to improve the exact formulation through the introduction of bound-tightening strategies, the initial formulation was selected to perform the experiments.

Chapter 4

Solution Approach

The VRPD-RS is reduced to the TSP, known to be *NP*-hard, if one considers the case where only one truck is available and the endurance of both drones and robots is null. Consequently, the VRPD-RS is *NP*-hard as well. As the results included in Chapter 5 will demonstrate, the VRPD-RS is already computationally demanding for small-sized instances. Therefore, to address large instances within reasonable computational times, it is necessary to develop efficient algorithms capable of finding near-optimal solutions.

The present chapter describes the GVNS algorithm developed to address the VRPD-RS. Initially, the structures used to encode VRPD-RS solutions will be presented, as well as the constructive heuristics. Afterwards, the local search operators developed and evaluation functions used to assess solutions will be presented. Finally, the configuration of the metaheuristic will be detailed.

4.1 Solution Representation

In order to accurately represent a feasible solution to the VRPD-RS, it is necessary to recognize the elements that compose it, as well as to understand the constraints of the problem. This last condition may lead to the creation of additional data structures to guarantee that solutions respect the imposed restrictions. Given these two perspectives, the representations used in the algorithmic implementation will be divided into a category concerned with routing elements and another representing auxiliary variables. In order to understand the role of each representation structure and how they are connected, an example of a solution for a problem with 10 customers and $(|F|,|D|,C,|K|) = (2,2,1,2)$ will be presented.

4.1.1 Representation of Routing Variables

A VRPD-RS solution is composed of three main routing decisions related to each transportation mode. Regarding trucks, the subsequence of nodes that each should visit must be detailed. In order to specify these elements, a matrix with two dimensions $R(v, pos)$ was created, where v identifies the vehicle and pos indicates a position in the truck's route. Figure 4.1 illustrates this representation for a case where truck 0 departs from the depot to visit customers 3 and 7, and then

returns back to the depot. Moreover, truck 1 only needs to supply station 12 in its route. Note that, at most, $L+2$ positions can store a node so that, beyond the maximum number L of customers that can be supplied by the truck, the initial and final depot nodes can be included. This decision was taken to ensure coherence with the structures that represent drone operations, since drone launches and retrievals may occur at nodes 0 and 11, respectively.

$R(v, pos)$

v	**pos. 0**	**pos. 1**	**pos. 2**	**...**	**pos. L**	**pos. L+1**
0	0	3	7	...	11	
1	0	12	11	...		

Figure 4.1: Representation of the trucks' routes.

Another set of decisions is related to scheduling drone operations. Since a drone operation is described by a triple (i, w, j), a structure capable of storing a list of triplets performed by each drone could be used. However, in order to align the representation with $R(v, pos)$ and facilitate the modeling of synchronization requirements with trucks, a different approach was followed.

For multi-leg operations, matrixes $D_D(v, d, pos)$ and $D_R(v, d, pos)$ store customer nodes served by drone and their corresponding retrieval nodes, respectively. In Figure 4.2, an example with four multi-leg operations is given. By observing matrix D_D, it is possible to see that customer 5 is served by drone 0, which was launched from the node in position 2 of truck 0's route, i.e., node 7. Through matrix D_R, it is possible that the drone will be retrieved in node 11. Hence, this structure defines multi-legs along the same column, where the position indicates where the launch occurred and elements i, w and j are specified in matrixes R, D_D and D_R, respectively.

$D_D(v, d, pos)$

(v, d)	**pos. 0**	**pos. 1**	**pos. 2**	**...**	**pos. L**	**pos. L+1**
(0,0)	4		5	...		
(0,1)		2		...		
(1,0)		1		...		
(1,1)				...		

$D_R(v, d, pos)$

(v, d)	**pos. 0**	**pos. 1**	**pos. 2**	**...**	**pos. L**	**pos. L+1**
(0,0)	7		11	...		
(0,1)		11		...		
(1,0)		11		...		
(1,1)				...		

Figure 4.2: Representation of customer and retrieval nodes in multi-leg operations.

The representation of roundtrips could be carried out by matrixes D_D and D_R as well by storing in D_R the truck node from where the operation begins. However, this would require an additional dimension to store the several operations that a drone may perform at a given position. In fact, at most n roundtrips can be performed from a given truck node and, after performing a given roundtrip, the drone can begin a multi-leg operation from the same node. Since both types of drone tasks possess distinctive operational characteristics and impact time variables differently, a separate representation for roundtrips was created through matrix $D_{round}(v, d, pos, operation)$. In the example of Figure 4.3, drone 0 serves node 6 in a roundtrip started in position 0 of truck 0's

route before initiating the multi-leg operation that serves customer 4. Moreover, drone 0 performs two successive roundtrips from the node in position 1 of truck 1's route. Even though D_D, D_R and D_{round} constitute sparse matrixes, auxiliary functions are used to avoid exploring blank spaces.

$D_{round}(v,d,pos,operation)$

(v, d)	pos. 0	pos. 1	pos. 2	...	pos. L	pos. L+1
(0,0)	[6]	[]	[]	...	[]	[]
(0,1)	[]	[]	[]	...	[]	[]
(1,0)	[]	[8,9]	[]	...	[]	[]
(1,1)	[]	[]	[]	...	[]	[]

Figure 4.3: Representation of customers addressed in roundtrip operations.

The third group of decisions is concerned with the activation of satellite stations and the assignment of robots to perform deliveries. To represent these elements, matrix $Z(s,k,customer)$ was created, whose elements represent a customer w addressed by robot k from station s. Despite being possible to identify which vehicle activates a station through R, an additional column was added to store this information and facilitate the implementation. Figure 4.4 demonstrates the representation of robot deliveries for the example analyzed so far. As it can be observed, truck 1 activates station 12 and delivers the parcel that robot 0 will deliver to customer 10.

$Z(s,k,operation)$

(s, k)	vehicle	1st customer	2nd customer	...	Lth customer
(12,0)	1	10		...	
(12,0)	1			...	
(13,0)				...	
(13,1)				...	

Figure 4.4: Representation of robot deliveries.

4.1.2 Representation of Auxiliary Variables

A VRPD-RS solution can be completely defined using the structures presented so far. Even though it would be possible to indirectly assess the time dependencies between vehicles through the routing representations, additional structures were created to facilitate the implementation. This option originates in the fact that every move performed upon a given solution has impact in subsequent or parallel operations due to the high degree of synchronization of the problem, e.g., inserting a drone operation could create truck waiting times at the retrieval node, leading to delays in subsequent nodes and possible increases in existing drone waiting times, which could lead to infeasible solutions in case it is assumed that drones hover in the air while waiting.

Similarly to variables a_i^v and s_i^v in the MILP formulation, matrixes $A(v,pos)$ and $S(v,pos)$ represent the arrival and earliest departure times at each position in a truck's route, respectively. These matrixes are structured similarly to R, but instead of storing visited nodes, express time instants. Additionally, if a given row in matrix R is not full, the time variables of the last node are repeated for the rest of the row in matrixes A and S. Hence, the makespan of truck-drone tandems is given by the maximum value among the elements in the last column of matrix S. Figure 4.5 illustrates A and S for the example under analysis.

$A(v, pos)$

v	pos. 0	pos. 1	pos. 2	...	pos. L	pos. L+1
0	0	18	26	...	37	37
1	0	20	48	...	48	48

$S(v, pos)$

v	pos. 0	pos. 1	pos. 2	...	pos. L	pos. L+1
0	13	19	28	...	41	41
1	0	38	49	...	49	49

Figure 4.5: Representation of truck time variables.

Matrixes $R_{bar}(v,d,pos)$ and $S_{bar}(v,d,pos)$ identify the earliest release and earliest departure times of drones in a given position of the truck's route, respectively. Such representations help identify the deviations between the time variables of trucks and drones caused by the occurrence of launches, roundtrips and retrievals. Figure 4.6 illustrates the drone time variables for the example.

$R_{bar}(v,d,pos)$

(v, d)	pos. 0	pos. 1	pos. 2	...	pos. L	pos. L+1
(0,0)	0	18	27	...	38	38
(0,1)	0	18	26	...	41	41
(1,1)	0	20	49	...	49	49
(1,1)	0	20	48	...	48	48

$S_{bar}(v,d,pos)$

(v, d)	pos. 0	pos. 1	pos. 2	...	pos. L	pos. L+1
(0,0)	12	18	27	...	38	38
(0,1)	0	18	26	...	41	41
(1,1)	0	37	49	...	49	49
(1,1)	0	37	48	...	48	48

Figure 4.6: Representation of drone time variables.

The next two elements presented handle the temporal constraints involved in multi-legs. Matrix $R_{D,bar}(v,d,pos)$ stores the earliest release times at drone-serviced customers. Structurally, this matrix resembles matrix D_D but replaces customer nodes by the corresponding times of arrival. Matrix $S_{D,bar}(v,d,pos)$ identifies the departure times that will allow the drones to be promptly retrieved by the truck. Therefore, if a multi-leg results in truck waiting times, the values of $R_{D,bar}$ and $S_{D,bar}$ will coincide. Conversely, if the drone waits, the difference between the corresponding elements in both matrixes provides the waiting time. As displayed in Figure 4.7, only the operation performed by drone 1 from truck 0 initiated in position 1 does not involve drone waiting times.

Finally, a direct adaptation of decision variables y_i^{vd} from the MILP formulation was made through matrix $Y_{availability}(v,d,pos)$. The information stored by this matrix is redundant but allows to readily assess if a given drone operation can be inserted between two nodes i and j without compromising existing operations already scheduled for the same drone. Furthermore, it may help determine if any drone launches or retrievals occur in a certain position. If a drone was unavailable for an operation in $pos-1$ and becomes available in pos, a retrieval occurs at pos. A similar perspective could be used to assess if drone launches occur at pos. However, this representation does not help identify if a launch or retrieval occurs for multi-leg operations where the truck does not visit any node between the launch and retrieval nodes. In such a case, $Y_{availability}$ would assume

$R_{D,bar}(v,d,pos)$

(v, d)	pos. 0	pos. 1	pos. 2	...	pos. L	pos. L+1
(0,0)	14		33	...		
(0,1)		36		...		
(1,0)		43		...		
(1,1)				...		

$S_{D,bar}(v,d,pos)$

(v, d)	pos. 0	pos. 1	pos. 2	...	pos. L	pos. L+1
(0,0)	15		34	...		
(0,1)		36		...		
(1,0)		45		...		
(1,1)				...		

Figure 4.7: Representation of drone time variables in multi-leg operations.

the value of 1 for both intervening truck nodes. Therefore, $D_R(v,d,pos-1)$, to assess if d was retrieved at pos, and $D_D(v,d,pos)$, to see if d was launched in pos, would need to be checked.

4.2 Constructive Heuristics

In this section, the procedures used to generate an initial VRPD-RS solution will be presented. The approach is based on three different stages: construction of a heuristic VRP solution, insertion of drone operations and insertion of robot operations. The order in which the stages are performed depends on the VRPD-RS variant considered. Should the minimal makespan be determined, after building the VRP routes, the algorithm inserts drone operations and then robot operations. Conversely, if the minimal cost is to be determined, the insertion of robot operations precedes drone insertions. This order inversion was adopted because robots constitute a delivery mode preferred in cost-minimization settings, since their variable transportation cost is lower than that of the drones.

4.2.1 Construction of a Heuristic VRP Solution

In order to produce a solution to the VRP, a cluster first, route second method is proposed, where the clustering step is performed using K-Means and the routing step is solved with a Cheapest Insertion heuristic. By building truck routes upon clustered customers, subsequent insertions of robot operations may be facilitated. Intuitively, satellite stations will be used to address customer agglomerates possibly within the range of robots. Therefore, those customers could be aggregated in a truck route and, if the assignment of customers to robots provides improvements in terms of makespan or cost, the truck could activate and deposit its load at the station.

The clustering algorithm K-means, proposed by MacQueen (1967), divides a given set of objects into k subsets. Each cluster is characterized by a centroid and objects are inserted in the cluster whose centroid is most similar to them. The algorithm starts with a random set of centroids and, iteratively, associates each object with the closest centroid and recalculates the centroids based on the assigned objects. In this context, procedure $KMeans()$ applies a version developed by Pedregosa et al. (2011), where k is equal to $|F|$ and the similarity measure used is the euclidean distance between nodes. Additionally, since K-means may render sets with a size

larger than truck capacity, a corrective procedure is applied to balance cluster size. As indicated in Algorithm 1, when selecting the objects to remain in the cluster, a dispatching rule p_w is used to privilege those closest to the centroid of the cluster under analysis, c_v, and farthest from the remaining centroids, $c_{v'}$ where $v' \in F \setminus \{v\}$. The Cheapest Insertion algorithm is a well-known TSP constructive heuristic that progressively introduces a new node k in the route between nodes i and j such that expression $c_{ik} + c_{kj} - c_{ij}$ is minimized, where c_{ij} designates the cost of travelling from i to j. In this case, this parameter indicates the truck travel time between any two nodes.

Algorithm 1: build_truck_routes()

Result: Heuristic VRP Solution
Initialization
For each v, set initial and final nodes to 0 and $n+1$, respectively;
Get X, the coordinates of each customer $w \in V_N$, and truck travel times t_{ij};
if $|F| > 1$ **then**
  $KMeans(|F|).fit(X)$;
  Create list C that containts clusters C_v in decreasing order of size; $N \leftarrow \{\}$;
  for $C_v \in C$ **do**
    for $w \in C_v \cup N$ **do** $p_w \leftarrow 2 \cdot t_{wc_v} / \sum_{v' \in F \setminus \{v\}} t_{wc_{v'}}$;
    Order nodes w in increasing order of p_w;
    Add, at most, the first L customers to C_v; Add remaining customers to N;
for $v \in F$ **do**
  $N \leftarrow C_v$; *Choose a random customer $i \in N$ to be inserted in the partial tour*;
  while $|N| \neq 0$ **do**
    Find customer $k \in N$ and edge (i, j) for which $c_{ik} + c_{kj} - c_{ij}$ is minimized;
    Insert k between i and j; $N \leftarrow N \setminus \{k\}$;

After applying *build_truck_routes*(), a VRPD-RS solution without drone and robot deliveries is obtained. Hence, at this point, the search-based metaheuristic could already be applied to explore the solution space. However, before initiating the GVNS, two additional procedures will be applied to enrich the solution with drone and robot operations. Even before performing drone and robot insertions, it could be argued that search operators could be used to refine the obtained VRP solution, since previous studies state that the VRPD can be solved more effectively starting from good VRP solutions (Schermer et al., 2018). Nevertheless, since it is desirable to maintain the organization of the routes as similar as possible to the initially computed clusters for subsequent robot insertions, it was decided not to improve the constructed truck routes.

4.2.2 Insertion of Drone Operations

The initial set of drone operations is generated through a greedy procedure that seeks to insert those operations that offer the most improvement. As described in Algorithm 2, the procedure begins by ordering drone eligible customers in decreasing distance to the centroid of their respective route. Here, the position of the centroid corresponds to the average of the coordinates of every node included in the route. This rule was inspired by the intuitive assumption that "drone delivery

would enable trucks to visit customers located centrally on the route and drones to visit farther-away customers" (Wang et al., 2017). Afterwards, for each customer, the best set of launch and retrieval nodes is determined and, if the operation is feasible, it is inserted. Furthermore, nodes that become launch or retrieval nodes cannot be tested for drone delivery to avoid destroying previously constructed drone operations.

Algorithm 2: build_drone_operations()

Result: VRPD Solution
Initialization
Get S, V_D, drone travel times $\bar{t}_{ij}$ and $\Omega \leftarrow \{\}$;
for $v \in F$ **do**
  Determine c_v; *Order list O of customers* $w \in R(v,:) \cap V_D$ *in decreasing distance to* c_v;
  for $w \in O$ **do**
    if $w \notin \Omega$ **then**
      $S_{best} \leftarrow S$; $f_{best} \leftarrow f(S)$;
      $S_{cand} \leftarrow remove\_truck\_node(v, R(v,:).index(w))$; $S \leftarrow S_{cand}$;
      for *every pair* $(i,j) \in R(v,:)$ **do**
        if $R(v,:).index(j) \geq R(v,:).index(i)$ ***and*** $\bar{t}_{iw} + \bar{t}_{wj} \leq \varepsilon_d$ **then**
          Find drone d available for an operation between i and j;
          if *drone was found* **then**
            $S_{cand} \leftarrow insert\_drone\_operation(v, d, i, w, j)$;
            if $f(S_{cand}) < f_{best}$ ***and*** $drone\_endurance\_feasibility(v)$ **then**
              $S_{best} \leftarrow S_{cand}$; $f_{best} \leftarrow f(S_{cand})$;
              Store best launch and retrieval nodes, i_{best} *and* j_{best};
      $S \leftarrow S_{best}$; $\Omega \leftarrow \Omega \cup \{i_{best}, j_{best}\}$;

Algorithm 2 uses two procedures, *remove_truck_node* and *insert_drone_operation*, which constitute steps used in some local search operators applied in the GVNS. Moreover, function *drone_endurance_feasibility* evaluates the drone operations associated with a given truck to determine if drone endurance constraints are respected. This function can be adapted to consider that waiting times dissipate energy by adding the differences between the elements in matrixes $S_{D,bar}$ and $R_{D,bar}$ to the flight time of drone operations.

4.2.3 Insertion of Robot Operations

Since robot insertions affect the activation of stations, beyond having to respect robot endurance constraints, special attention to the maximum allowed number of active stations should be given. Therefore, procedure *build_robot_operations* begins by selecting the potentially C best station locations, given the existing customers available to be addressed by robot. This is done by associating each eligible truck node with the closest station.

With the guarantee that the maximum number of active stations will not be surpassed, robot insertions can be tested for the C best stations in decreasing order of associated customers. For

each station, the eligible customers are ordered in terms of increasing distance to the station. If a station has not yet been activated for delivery, any truck node is eligible. In case the node acts as a launch or retrieval location, the station will assume its role and will perform the overhead operations. After adding the first customer to the station, the activating vehicle will be the truck that previously addressed that customer. After activation, just truck-only nodes are eligible for robot delivery to avoid affecting existing drone operations. Furthermore, adding robot deliveries to a station is allowed as long as the capacity of the activating vehicle is not surpassed. In Algorithm 3, the described procedure is presented.

Algorithm 3: build_robot_operations()

Result: VRPD-RS Solution with Robot Operations
Initialization
Get S, V_D, robot travel times $\widetilde{t}_{ij}$ and $\Omega \leftarrow \{\}$;
Select the C potentially best station locations as set V'_S;
for $s \in V'_S$ **do**
  Order eligible customers w in increasing distance to the station as list O;
  for $w \in O \setminus \Omega$ **do**
    $S_{best} \leftarrow S$; $f_{best} \leftarrow f(S)$;
    if *station has not been activated yet* **then**
      Swap w with s in the truck route; *Select the best robot to address w*;
    else
      if *w is a truck-only node* ***and*** *load of the truck that activates* $s \leq L$ **then**
        Select the best robot to address w;
    if $f(S_{cand}) < f_{best}$ ***and*** $drone\_endurance\_feasibility(v)$ ***and*** $\widetilde{t}_{sw} + \widetilde{t}_{ws} \leq \varepsilon_r$ **then**
      $S_{best} \leftarrow S_{cand}$; $f_{best} \leftarrow f(S_{cand})$; $\Omega \leftarrow \Omega \cup \{w\}$;

4.3 Solution Evaluation

In order to perform a local search for better VRPD-RS solutions, is is necessary to define the criteria that determines how well a solution addresses the problem. Since the VRPD-RS can assume two distinct objectives, the fitness functions used will have to be suited for each problem.

In terms of the minimization of the operational cost, the approach used to assess the quality of a solution is straightforward: a given solution S' is better than S if the operational cost of S' is lower than that of S, i.e., $f(S') < f(S)$, where $f(S) = \omega(S)$.

Regarding the minimization of makespan, the criteria used to compare two distinct VRPD-RS solutions involves three levels. Considering that τ represents the makespan and τ_{min} indicates the minimum makespan among all vehicles in the solution, a given solution S' is deemed better than S, if one of the following conditions applies: $\tau(S') < \tau(S)$; $\tau(S') = \tau(S)$ and $\tau_{min}(S') < \tau_{min}(S)$; $\tau(S') = \tau(S)$ and $\tau_{min}(S') = \tau_{min}(S)$ and $\omega(S') < \omega(S)$.

The evaluation of solutions in a minimal makespan scenario uses two levels to break ties when strict makespan improvements are not achieved. Since the makespan only regards improvements

in the longest route, improvements occurring in other parts of the solution may be overshadowed. Therefore, if a neighbor solution possesses the same makespan as the incumbent solution, the route with the shortest duration will be assessed to see if any improvement might have occurred, which is a similar approach as the one presented by Mbiadou Saleu et al. (2018) to address the PDSTSP. Furthermore, the cost of the two solutions is compared in case they have identical values of τ and τ_{min}. This condition was implemented because minimizing the makespan is aligned with cost reductions, since shorter delivery times typically involve reducing the distance covered by each vehicle. Furthermore, assessing cost variations allows to grasp improvements occurring in routes with a duration comprised between the longest and shortest routes. Throughout this chapter, fitness function f is considered to apply the appropriate evaluation criteria for each VRPD-RS variant.

4.4 Local Search Operators

The VRPD-RS models the cooperation of three transportation modes whose operations require a high-degree of synchronization. Consequently, while searching for neighbor solutions, several moves might be performed. In this section, the five different local search operators developed to aggregate these possible moves will be explained. Among them, three operators affect the number of tasks each delivery mode performs and are designated as *Relocate to Truck*, *Relocate to Drone* and *Relocate to Robot*. Additionally, operators *2-opt* and *Exchange* try to improve the solution while maintaining the activity level of each vehicle type.

4.4.1 2-opt

The 2-opt is a local search procedure for the TSP introduced by Croes (1958). This heuristic consists in selecting two non-consecutive nodes in the route and inverting the order in which the sequence in between them is visited, which may help untangle the route (Schermer et al., 2019a).

In the context of the VRPD-RS, *2-opt* constitutes a single-route improvement heuristic and is subject to adaptations to ensure that it produces feasible solutions. Since the 2-opt heuristic inverts the order of a subsequence of nodes, it impacts the relative positions of launch and retrieval nodes. Therefore, the *2-opt* operator can only be applied to nodes u and v where all nodes in the inverted subsequence must be either truck-only nodes or must aid drone operations fully contained within that sequence, i.e., do not involve outside launch or retrieval nodes. This restriction happens because, after applying this move, drone operations will need to swap launch and retrieval nodes to guarantee that they occur in the correct order. If a given drone performs an operation where a non-inverted truck node is involved, the resulting 2-opt neighbor solution might include overlapping operations, as represented in Figure 4.8 (c). In this situation, considering that both operations are performed by a single drone, the application of the 2-opt heuristic would lead to overlapping operations. Indeed, in v, a multi-leg operation would be solicited while the drone was performing an operation initiated at u. Differently, the situation depicted in Figure 4.8 (a) is feasible since all inverted nodes are truck-only nodes, as well as in Figure 4.8 (b), which includes a drone operation with launch and retrieval nodes belonging to the inverted sequence.

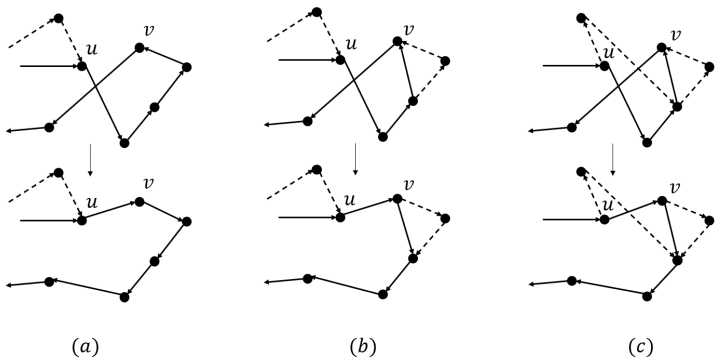

Figure 4.8: Example of solutions modified by the *2-opt* local search operator.

After applying *2-opt* and changing the routing data structures, procedure *update_times* is applied and drone endurance constraints are analysed. The *update_times* procedure is a crucial element of the implemented algorithm used after performing moves upon the incumbent solution. The rationale behind this procedure is, for a given vehicle, to start updating times from the earliest modified position until the end of the route. For the *2-opt* operator, the updates begin at node u.

4.4.2 Exchange

The *Exchange* operator performs permutations of any two nodes in the solution, regardless if they are truck, drone or robot nodes or associated with distinct trucks. Thus, a local search exploring neighborhoods generated by *Exchange* is both a single- and multi-route improvement heuristic.

This versatile operator leads to the creation of large-sized neighborhoods, but allows to consolidate several useful improvement mechanisms. Regarding truck routes, the exchange of two nodes belonging to the same truck constitutes a swap one for one. Moreover, if the nodes are related to different vehicles, a 'String Exchange' is performed, which might help correct errors made in the clustering stage, as depicted in Figure 4.9 (a). Additionally, the roles of each element in a triple (i,w,j) may be changed, e.g., the retrieval node may become the drone node and the triple would be given by (i,j,w), as depicted in Figure 4.9 (b). This particular move could be useful to address the bias of *build_drone_operations* that assumes that customers located on the outer zones of truck routes may be more promising for drone delivery. Another important role that *Exchange* plays is related to scheduling subproblems that exist at stations housing more than one robot and at nodes where roundtrips are completed by more than one drone. In this situations, an identical PMS problem exists and swapping jobs might help reducing the completion time of those tasks. Figure 4.9 (c) illustrates a case where the tasks are balanced between two robots, where previously one handled those with the longest completion time.

In terms of implementation, the move can be aborted in case a customer is exchanged to be assigned by an incompatible vehicle. Additionally, an exchange may not be allowed due to violations of endurance constraints. After performing the move, the *update_times* procedure is performed for the affected vehicles in their respective earliest positions of exchange.

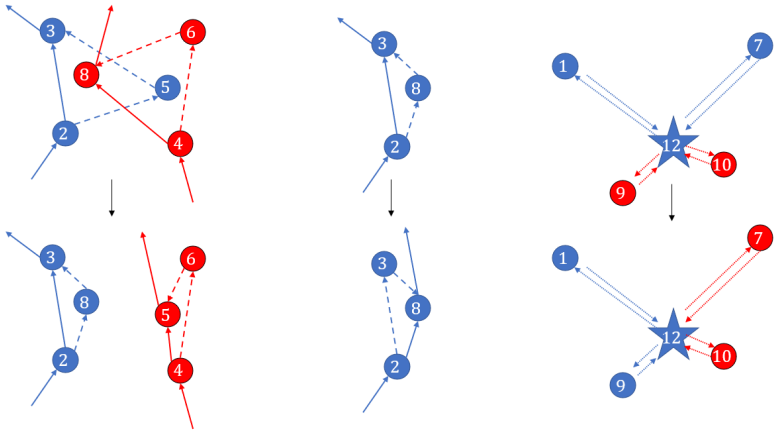

Figure 4.9: Example of solutions modified by the *Exchange* local search operator.

4.4.3 Relocate to Truck

Relocate to Truck transfers a customer to occupy a certain position in the route of a truck v. This relocation is performed using procedures *remove_generic_node* and *insert_truck_node*. By providing *remove_generic_node* with a customer and the truck that carries his parcel, it removes him from his current position regardless of the delivery mode that addresses him. If the customer is not a truck-only node, special corrective procedures are taken before removal, which include:

- Retrievals: if (i_r, w_r, j_r) designates an operation where j_r is to be removed, a new retrieval node is searched in subsequence $[i_r, j_r[$ of the route. In case all tested nodes lead to operations exceeding drone endurance, w_r is inserted as a truck node in a random position between i_r and j_r.
- Roundtrips: customers addressed in roundtrips are inserted at their cheapest insertion position in the truck's route.
- Launches: if (i_l, w_l, j_l) designates an operation where i_l is to be removed, a new launch node is searched in subsequence $]i_l, j_r]$ of the route. Similarly to retrieval operations, if drone endurance constraints are not respected for every candidate launch node, w_l is inserted in a random position between i_l and j_l.

Once a truck-only node is obtained, the removal can take place. Then, *insert_truck_node* inserts the node in the desired position of a given vehicle as a truck-only node. Similarly to *Exchange*, *Relocate to Truck* is both a single- and multi-route improvement operator. In fact, relocating a truck node in the same truck can be equivalent to a swap of two adjacent customers or a relocation of a customer to be before another. For a move between trucks, a 'String Relocation' is performed, which may balance the workload of the truck fleet. For this last situation, load capacity constraints need to be respected. For any relocation to truck, if drone waiting times consume energy, the feasibility of the solution regarding endurance constraints must be checked

since it may increase current waiting times, e.g., the node is relocated to a position in between the launch and retrieval nodes of an operation and delays the arrival of the truck at the retrieval point.

An important jump in the solution space allowed by *Relocate to Truck* is the deactivation of robot stations. This action occurs when the last customer supplied by a station is relocated to a truck route. When removing such a customer, *remove_generic_node* assesses the number of remaining deliveries made by the station and deactivates it if no deliveries take place.

4.4.4 Relocate to Drone

The search operator *Relocate to Drone* reassigns any customer node to be serviced by drone, even those already assigned to one. Similarly to *Relocate to Truck*, *remove_generic_node* is applied before calling procedure *insert_drone_operation*. Thus, the relocation of a truck node that aids drone operations will involve the complementary moves to reduce it to a truck-only node before removal as well. Moreover, this operator can also be responsible for the deactivation of stations.

By allowing relocations of customers currently addressed by drone, several improvement moves may be achieved. *Relocate to Drone* may help optimize the launch and retrieval nodes of a given operation, e.g., changing the retrieval node of drone operation $(2,8,3)$ depicted in Figure 4.10 (a) to be node 5 could possibly improve the synchronization of the two vehicles and reduce the total waiting time among them. Additionally, since *Relocate to Drone* can shift a drone operation performed by drone d to be performed by d' while maintaining the triple (i,w,j), it can contribute to more balanced roundtrip schedules, as illustrated in Figure 4.10 (b).

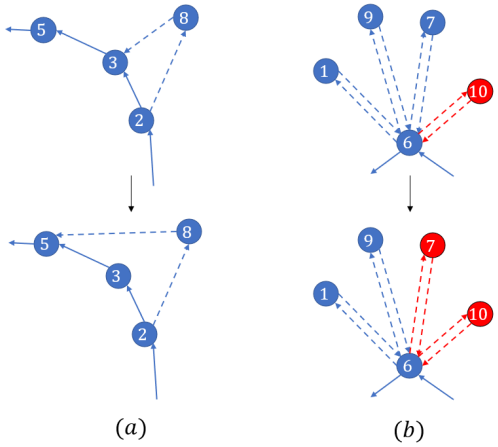

Figure 4.10: Example of solutions modified by the *Relocate to Drone* local search operator.

In terms of applicability, only customers $w \in V_D$ can be relocated. Furthermore, the selected drone must be available to perform the operation for the specified launch and retrieval nodes and its endurance limit should not be surpassed. In case the search operator performs a relocation between trucks, truck load constraints need to be respected.

4.4.5 Relocate to Robot

Relocate to Robot works similarly to the remaining relocation operators. Before performing the insertion of the customer as a robot node, *remove_generic_node* must be applied. However, *Relocate to Robot* can also be responsible for the activation of stations, whereas the remaining relocation operators could only perform a deactivation. Due to this feature, special measures need to be taken upon relocating a customer to be a robot node.

The first step of *Relocate to Robot* is assessing if the customer under relocation is already served by the station where we desire to place him. In such case, the deliveries at the station will possibly be balanced or the order of the customers addressed by a robot will be altered, which can easily be done without using procedure *remove_generic_node*. If the former condition is not verified, the search operator first checks if the station is activated. If not, the vehicle that addresses the relocated customer will activate the station. In case the customer was a truck node, the station will occupy its position in the truck route, whereas if the customer is served by drone, the station will be inserted in the cheapest insertion position of the route.

While performing a *Relocate to Robot* move, it is necessary to guarantee that the configuration of the resulting solution will include at most C activated stations. Additionally, load capacity constraints for the vehicle that supplies the station must be respected and the customer should be within the robot's range of operation. Since this operator may involve inserting in a route a station which may become a launch or retrieval node, drone endurance feasibility must be verified.

4.5 A General Variable Neighborhood Search Heuristic

The solution approach for the VRPD-RS proposed in the present study is based on the General Variable Neighborhood Search metaheuristic. The proposed GVNS applies five local search operators in an integrated manner to simultaneously address the routing, vehicle assignment and facility location decisions inherent to the VRPD-RS. The GVNS is based on the VNS metaheuristic introduced by Mladenović and Hansen (1997), that allows performing a local search upon varying neighborhood structures. The GVNS extends the VNS by considering that the local search can also be performed under different neighborhood structures.

Due to the possibility of combining several search operators and its simple implementation, the GVNS becomes an attractive framework to develop a VRPD-RS solution approach. Conceptually, the VRPD-RS is composed of an assignment subproblem, which dictates the activation of satellite stations as well, and a routing subproblem. Given the architecture of the GVNS, the shaking phase could be useful to predominantly change the assignments in the solution and, subsequently, the local search could improve routing for those assignments. Moreover, VNS is a framework applied in many state-of-the-art algorithms for vehicle routing problems (Salhi et al., 2014; Hemmelmayr et al., 2009). Specifically for routing problems with truck-drone tandems, competitive algorithms based on the VNS framework have already been proposed (de Freitas and Penna, 2018, 2020; Schermer et al., 2019a). Algorithm 4 describes the implemented GVNS.

Algorithm 4: GVNS()

Result: S
Initialization
$k_{max} = 4$; $l_{max} = 5$; Select the set of shaking procedures $\mathcal{N}_k$ with $k = 1, ..., k_{max}$
Select the set of neighborhood structures $\mathcal{N}_l$ used in the local search, with $l = 1, ..., l_{max}$
$stopping\_condition \leftarrow False$; $\Delta_t \leftarrow 0$; $t_i \leftarrow 0$; $t' \leftarrow 0$; $n_{max} \leftarrow \frac{n^2}{k_{max}}$; $cont \leftarrow 0$
$S \leftarrow build\_initial\_solution()$;
while ***not*** $stopping\_condition$ **do**
  $k \leftarrow 1$; $improvement \leftarrow False$;
  while $k \leq k_{max}$ **do**
    $S' \leftarrow \mathcal{N}_k(S)$;
    $S'' \leftarrow \text{RVND}(S')$;
    if $f(S'') < f(S)$ **then**
      $S \leftarrow S''$; $k \leftarrow 1$; $improvement \leftarrow True$;
    else
      $k \leftarrow k+1$;
  $t_f \leftarrow time()$;
  if *improvement=False* **then**
    $\Delta_t \leftarrow t_f - t'$; $cont \leftarrow cont + 1$;
  else
    $t' \leftarrow time()$; $cont \leftarrow 0$;
  if $(t_f - t_i) \geq \delta_T$ ***or*** $cont \geq n_{max}$ ***or*** $\Delta_t \geq \delta_P$ **then** $stopping\_condition \leftarrow True$;

As described in Algorithm 4, the heuristic begins by initializing the neighborhood structures used in both shaking and local search phases. The shaking procedures select a random solution from the neighborhoods generated by operators: (1) *Relocate to Robot*; (2) *Relocate to Truck*; (3) *Exchange*; and (4) *Relocate to Drone*. Since the adopted shaking procedures generate disjoint sets of neighbor solutions, the neighborhoods are not nested and cannot be ordered based on the hierarchical relations among them. Therefore, neighborhood structures $\mathcal{N}_k$ were ordered in terms of an increasing size of the neighborhood. The local search is performed by a RVND, which uses the same neighborhood structures as the shaking phase, as well as the *2-opt* operator. Due to the limited cases where the *2-opt* can be applied, generating random neighbors would require sampling several pairs of nodes until an allowed move could be performed. Thus, it was decided not to include it as a shaking procedure. Regarding the local search operators, because of the large neighborhood size of operators *Relocate to Truck*, *Exchange* and *Relocate to Drone*, a search strategy where the best solution from a sample of n^2 neighbors was adopted. For *2-opt* and *Relocate to Robot*, a best improvement strategy was followed.

The starting point of the GVNS heuristic is given by *build_initial_solution()*, which applies the heuristics described in section 4.2. Then, in the shaking phase, a random solution from the k^{th} neighborhood of S is generated. This newly found solution is improved through a local search performed by a RVND. In case the local optimum, S'', is better than the incumbent solution, S, the solution is updated and the search will progress from there for $k = 1$. Conversely, the search

returns to the incumbent solution and neighborhood $k = k + 1$ will be tested.

The local search employed in the present GVNS adopts the mechanism of the RVND proposed by de Freitas and Penna (2018, 2020). As opposed to the VND heuristic, the RVND randomizes the order of the neighborhood list at the start of the algorithm and every time an improvement is achieved (de Freitas and Penna, 2018). In a study performed on several variants of the heterogeneous fleet VRP, Penna et al. (2013) concluded that using a randomized version of the VND outperformed a version with a stable neighborhood order. Based on this fact and since the local search neighborhood structures do not share any relationship that could help establishing an order, this principle was adopted and it is described in Algorithm 5.

Algorithm 5: RVND(S')

Result: S"
Initialization
$l_{max} = 5$; Select the local search neighborhood structures $\mathscr{N}_l$ with $l = 1, ..., l_{max}$
Define the list of neighborhood structures $\mathscr{L}$, where $\mathscr{L} = [N_1, ..., N_{l_{max}}]$; *Shuffle* $\mathscr{L}$
$l \leftarrow 1$;
while $l \leq l_{max}$ **do**
  $S'' \leftarrow \mathscr{L}_l(S')$;
  if $f(S'') < f(S')$ **then**
    $S' \leftarrow S''$; $l \leftarrow 1$; *Shuffle* $\mathscr{L}$;
  else
    $l \leftarrow l + 1$;
$S'' \leftarrow S'$;

After all shaking procedures have been analyzed, i.e., $k = k_{max} + 1$, the stopping condition is assessed to determine if the algorithm should end or if a new exploration of the neighborhood structures should be performed. A termination might occur if the total running time of the algorithm exceeds δ_T time units. Moreover, if the algorithm is unable to improve the incumbent solution for n_{max} iterations, where $n_{max} \leftarrow n^2/k_{max}$, the execution ends. Alternatively, this condition is expressed as a time limit as well, where the algorithm terminates if it cannot improve the incumbent solution within a time frame of δ_P time units. This condition overlaps the n_{max} criteria but is added to ensure algorithm scalability. For larger instances, the exploration of the neighborhoods may be computationally expensive and using a number of iterations as the stopping condition might lead to significant time intervals where the incumbent solution is not altered.

Chapter 5

Results

The present chapter begins with a description of the instances used for testing. Then, the VRPD-RS will be analyzed to understand the impact of including drones and robot stations in a delivery system. Additionally, the performance of the proposed solution approach will be determined. Finally, the robustness of this delivery system when facing traffic congestion will be evaluated.

The MILP formulation presented in Chapter 3 was solved using the state-of-the-art solver Gurobi Optimizer 9.0.2 (Gurobi Optimization LLC, 2020) and the GVNS algorithm was implemented in Python. By default, the computational tests were performed on an Intel Xeon CPU E5-2650 v3 at 2.3 GHz with 40 CPUs executing three threads simultaneously.

5.1 Test Instances

The computational experiments were performed on the same parameter assumptions and instances as those used by Schermer et al. (2020b). Regarding the operational features of the vehicles, it is assumed that trucks move at 20 mph, a speed typical of urban settings, along the Manhattan (L_1) distance metric (Campbell et al., 2017). A combined truck-drone system benefits from larger relative drone speeds (Agatz et al., 2016). In this context, it is assumed that drone speed is equal to 25 mph and that the maximum flight time is of 20 min. Additionally, both the launch and retrieval times are set to be of 1 min (Murray and Chu, 2015). Schermer et al. (2020b) studied the TSP-D-RS with two different robot speeds: 5 mph for devices traveling at walking speed and 15 mph for faster robots. Their numerical results indicate that robot speeds of 5 mph have a reduced influence on makespan reductions, whereas speeds of 15 mph have a considerable impact. To have a more balanced utilization of the robots, a speed of 10 mph was adopted in this context, as well as an endurance of 40 min. Both autonomous vehicles are assumed to travel along the Euclidean (L_2) distance metric, since drones are not restricted to the road network and robots may access pedestrian areas otherwise unreachable by trucks (Schermer et al., 2020b). The cost coefficients c_t, c_d and c_r are expressed on relative terms and are given by 1, $\frac{1}{10}$ and $\frac{1}{20}$, respectively.

The instances used on the computational experiments are the ones proposed by Schermer et al. (2020a). These include 10 different graphs for each number of customers $n \in \{10, 15, 20\}$.

Each graph contains the locations of the depot and customer nodes, which were inserted randomly across a square region with 8x8 sqm. Additionally, two robot station locations are provided, where the first one assumes the coordinates given by the average of all customers' coordinates and the second one was inserted randomly. The first station could potentially be near customer clusters.

Regarding the configuration of the fleets of trucks and drones, the cases where $(|F|,|D|) = \{(1,1),\ (1,2),\ (2,1),(2,2)\}$ were studied. For each configuration, the maximum number of allowed stations and the number of robots could assume the values of $(C,|K|) = \{(0,0),\ (1,2),\ (2,1)\}$. By considering $(|F|,|D|) = (1,1)$, the impact of extending the TSP-D-RS to a multiple truck and drone context can be explored. Furthermore, by considering a situation where $C = 0$, the implications of including robot stations in a VRPD can be analyzed. Given all these scenarios and considering the two VRPD-RS objectives, a total of $2 \times 3 \times 10 \times 4 \times 3 = 720$ problem instances were solved. Finally, it was considered that all customers can be served by drones and robots, i.e., $V_N = V_D$. Such assumption is not realistic, but, from a computational perspective, corresponds to the most challenging case. Indeed, the assignment of either a truck, drone or robot delivery must be made for all customers, which corresponds to the scenario with the largest solution space (Schermer et al., 2020b). Additionally, since the delivered goods are small parcels and settings modelled by the VRPD-RS relate to same-day deliveries, truck capacity is not critical and was considered to be sufficient to serve all customers, i.e., $L = n$ (Ulmer and Thomas, 2018).

5.2 Main Experiments

In this section, the numerical results obtained by solving the MILP formulation with a time limit of 1 h will be exposed and an analysis of the features of the VRPD-RS will be performed. Additionally, the performance of the GVNS heuristic will be assessed, by comparing its solutions against those of the exact model. Based on the results achieved by the heuristic, an analysis to understand its strengths and limitations will be done.

5.2.1 Features of the VRPD-RS

5.2.1.1 Minimum Makespan VRPD-RS

The results included in Table 5.1 present the performance of the MILP solver in addressing minimum makespan VRPD-RS problems and the impact of fleet composition on the desired objective. Regarding performance assessment, the table indicates the average runtime in minutes, the number of times optimality was reached and the average MILP percentage gap. In terms of the impact of fleet composition, the average makespan among instances sharing the same parameters of $|F|$, $|D|$, C and $|K|$ is given, as well as the operational cost averaged across the combinations of $(C,|K|)$.

The results for the minimum makespan VRPD-RS emphasize its computational complexity. For small-sized 10 customer instances, the solver handles the problem effectively, as optimal solutions were not reached only for 7 cases out of 120 instances. However, as the number of customers increases, Gurobi fails to compute optimal solutions within the time limit. For 15 customers, the

Table 5.1: Results of the minimum makespan VRPD-RS using the MILP solver.

n	F,D	(C,|K|) (0,0)				(1,2)				(2,1)					
		τ^*	time	opt.	gap	τ^*	time	opt.	gap	τ^*	time	opt.	gap	$\overline{\tau}^*$	$\overline{\omega}$
10	**1,1**	75.4	1.0	10/10	0.0	69.0	1.5	10/10	0.0	70.9	3.7	10/10	0.0	71.8	23.6
	1,2	67.2	3.7	10/10	0.0	60.9	6.5	10/10	0.0	62.3	8.8	10/10	0.0	63.5	22.0
	2,1	54.3	3.8	10/10	0.0	52.2	12.3	10/10	0.0	52.2	23.4	9/10	1.1	52.9	32.3
	2,2	51.8	13.2	10/10	0.0	49.1	33.0	8/10	7.3	49.1	42.2	6/10	13.2	50.0	30.2
15	**1,1**	83.3	39.3	5/10	3.9	77.3	49.3	3/10	9.2	80.0	59.8	1/10	18.2	80.2	26.6
	1,2	73.5	60.0	0/10	26.3	68.8	60.0	0/10	37.6	69.4	60.0	0/10	36.2	70.6	24.4
	2,1	59.4	60.0	0/10	60.3	57.3	60.0	0/10	74.1	57.5	60.0	0/10	71.9	58.0	37.0
	2,2	55.0	60.1	0/10	118.2	54.5	60.1	0/10	170.4	54.9	60.1	0/10	157.6	54.8	34.9
20	**1,1**	97.5	60.0	0/10	19.8	86.9	60.0	0/10	25.2	91.1	60.0	0/10	29.0	91.8	30.3
	1,2	87.7	60.1	0/10	59.2	84.6	60.0	0/10	86.6	85.6	60.1	0/10	73.7	86.0	29.3
	2,1	67.7	60.0	0/10	82.1	64.3	60.0	0/10	103.4	66.4	60.0	0/10	101.1	66.1	42.4
	2,2	63.6	60.0	0/10	161.0	60.4	60.1	0/10	180.2	65.8	60.1	0/10	193.8	63.3	41.5

TSP-D-RS can be solved to optimality in less than half of cases, while the extensions to multiple trucks and drones already present considerable solver gaps. For 20 customers, the performance is further degraded as none of the instances were solved optimally. Indeed, for $(|F|,|D|) = (2,2)$, the gap was consistently larger than 100%, indicating that solutions with less than half of the makespan could potentially be obtained by Gurobi, if given unlimited time.

The addition of trucks and drones to a delivery system reduces the makespan, as indicated by the decreasing values of $\overline{\tau}^*$ for each instance size. Moreover, considering the TSP-D-RS to be the base scenario, adding one truck is more effective in reducing the makespan than including an extra drone. In fact, such strategy contributes to a higher degree of parallelization since the delivery area can be divided by two trucks and, compared to $(|F|,|D|) = (1,1)$, $(|F|,|D|) = (2,1)$ adds two resources as the new vehicle also carries a drone. However, this configuration is more expensive, since one vehicle with the highest operational cost is added. For $(|F|,|D|) = (1,2)$, both makespan and operational cost are reduced as opposed to the base scenario. This occurs because some of the workload of the truck can be transferred to the additional drone, which has a lower cost to operate. Table 5.2 presents the percentage of deliveries performed by trucks (T), drones in multi-legs (D_A) and roundtrips (D_C), and robots (R), as well as the percentage of times the station placed randomly (S_A) and the station whose coordinates are the average of all customer coordinates (S_W) are activated. The results confirm that when a drone is added, the sum of $T + D_A + D_C$ remains relatively stable and the utilization of both resources becomes more levelled.

It is also possible to observe that the inclusion of robot stations has positive outcomes. In fact, by allowing either a maximum of 1 or 2 stations to be activated, the makespan is reduced. Between cases where $(C,|K|)$ equals $(1,2)$ and $(2,1)$, there are no significant differences with respect to the optimal makespan, except for the fact that, when $C = 2$, the makespan is higher. This can be justified by the fact that both configurations contain the same number of robots, whose level of activity, R, is slightly decreased between the cases where C is equal to 2 instead of 1 (see Table 5.2). Thus, when $C = 2$, almost the same number of robot deliveries will be performed by robots placed at distinct locations, leading to a distortion in the truck's route to activate the second station.

Table 5.2: Percentage of deliveries performed by each mode and percentage of cases each station is selected for different minimum makespan VRPD-RS configurations.

F,D	(C,|K|)														
	(0,0)			(1,2)						(2,1)					
	Resources			Resources				Stations		Resources				Stations	
	T	D_A	D_C	T	D_A	D_C	R	S_w	S_A	T	D_A	D_C	R	S_w	S_A
1,1	72.2	27.8	0.0	54.5	20.0	0.0	25.5	53.3	46.7	56.9	22.6	0.0	20.5	90.0	60.0
1,2	53.1	46.6	0.3	42.1	37.3	0.0	20.6	53.3	43.3	41.2	40.9	0.0	17.9	90.0	46.7
2,1	73.2	25.4	1.4	62.1	22.1	1.0	14.8	63.3	33.3	59.8	24.9	1.0	14.3	90.0	60.0
2,2	56.7	39.4	3.9	46.9	33.7	4.9	14.6	86.7	13.3	48.6	38.8	2.8	9.8	83.3	43.3

A noteworthy feature of minimum makespan solutions is the predominance of multi-legs over roundtrips. In fact, the percentage of customers addressed in roundtrips is practically null for single-truck scenarios and reduced for two-truck scenarios. This feature was also verified by Schermer et al. (2020b) and its occurrence is intuitive as performing roundtrips leads to having the truck stopped at a location incurring on large waiting times. Contrarily, multi-legs admit a simultaneous execution of tasks, which helps reducing the makespan. Another observation related to robot stations is that, when $C = 1$, the sum of the utilization of the stations is equal to 100% or close to that value for the several fleet configurations. Thus, in almost every case, the stations are selected to reduce the makespan. Moreover, the station with the averaged coordinates, S_W, is preferred in the majority of cases when only one station can be selected and is almost always activated when $C = 2$.

5.2.1.2 Minimum Operational Cost VRPD-RS

As part of the analysis of the minimum cost VRPD-RS, for each solution, the corresponding makespan was obtained as well[1]. In Table 5.3, besides these two parameters, for each instance type, the average runtime in minutes and the percentage gap of the solver are included, as well as the number of instances where the optimal solution was reached.

By focusing on the performance of the solver, it is evident that the minimum operational cost VRPD-RS is less computationally demanding than the minimum makespan variant. In fact, Gurobi was capable of optimally solving the majority of the instances with 10 and 15 customers within relatively short runtimes. For 15 customers, only with $(|F|,|D|) = (2,2)$ did the execution become more complex, while for the remaining configurations, the solver consumed substantially less time. The only scenarios considered in this experiment that were also studied by Schermer et al. (2020b) were those where $|F| = 1$, $|D| = 1$ and $(C,|K|) = \{(0,0),(2,1)\}$. Contrarily to their experiments, where all instances were solved optimally, the solver failed to reach the optimal solution for four 20-customer instances. The reason for this might lie in the different resources used between both studies or the procedure in which the makespan was obtained. Moreover, with

[1]The objective function implemented in Gurobi encompassed not only expression 3.32, but also included a term with the makespan multiplied by a very small coefficient. Such approach is required because the cost objective does not trigger the minimization of time variables, which allows the makespan to be any non-negative real number.

Table 5.3: Results of the minimum operational cost VRPD-RS using the MILP solver.

n	F,D	(C,|K|) (0,0) ω^*	time	opt.	gap	(1,2) ω^*	time	opt.	gap	(2,1) ω^*	time	opt.	gap	$\overline{\omega}^*$	$\overline{\tau}$
10	**1,1**	18.7	0.0	10/10	0.0	16.7	0.1	10/10	0.0	16.6	0.1	10/10	0.0	17.3	130.8
	1,2	18.7	0.2	10/10	0.0	16.7	0.2	10/10	0.0	16.6	0.2	10/10	0.0	17.3	115.0
	2,1	18.7	0.1	10/10	0.0	16.7	0.3	10/10	0.0	16.6	0.6	10/10	0.0	17.3	119.6
	2,2	18.7	1.7	10/10	0.0	16.7	2.1	10/10	0.0	16.6	2.4	10/10	0.0	17.3	106.0
15	**1,1**	19.8	1.4	10/10	0.0	18.4	2.4	10/10	0.0	18.4	2.0	10/10	0.0	18.9	180.8
	1,2	19.8	6.1	10/10	0.0	18.4	7.8	10/10	0.0	18.4	12.7	10/10	0.0	18.9	148.9
	2,1	19.8	9.9	10/10	0.0	18.4	19.7	8/10	4.1	18.4	17.7	9/10	2.1	18.9	163.7
	2,2	19.8	49.1	4/10	15.9	18.4	43.5	5/10	20.8	18.5	49.9	4/10	20.5	18.9	138.8
20	**1,1**	20.9	23.6	8/10	2.8	19.4	24.3	8/10	4.7	19.2	21.5	8/10	7.2	19.8	219.2
	1,2	21.0	52.0	4/10	22.9	19.4	49.1	4/10	23.3	19.4	52.4	2/10	25.8	19.9	177.5
	2,1	21.0	53.4	3/10	20.5	19.4	51.7	2/10	27.1	19.3	51.7	3/10	29.2	19.9	209.2
	2,2	21.6	60.0	0/10	89.1	19.7	60.0	0/10	84.6	19.4	60.0	0/10	72.2	20.2	170.0

additional trucks and drones, the problem becomes progressively more complex, which resulted in failing to solve several 20-customer instances optimally.

One interesting result is that the optimal operational cost remains constant independently of the number of trucks and drones considered, for a given n and $(C,|K|)$. One reason that leads to this phenomenon is the fact that the operational cost objective is not concerned with reducing delivery times. Therefore, when the number of trucks and drones available increases, the solver will keep the optimal delivery routes achieved for $(|F|,|D|) = (1,1)$, if deploying an additional truck is not cost-effective. Nevertheless, increasing the size of the truck and drone fleets decreases the corresponding makespan of minimum cost solutions, as evidenced by the decreasing values of $\overline{\tau}$. Since the implementation seeks to minimize the makespan associated with a minimum cost VRPD-RS solution, the solver will seek to maintain the same vehicle operations but distribute the tasks among the available vehicles where possible in an attempt to reduce the delivery time. This phenomenon is illustrated in Figure 5.1, which compares scenario (a), where $(|F|,|D|) = (1,1)$, with scenario (b), where $(|F|,|D|) = (2,1)$. As it can be observed, the additional truck is only used for its drone that can perform the roundtrips initiated at the depot previously assumed by the original truck. Another reason for the observed behaviour may be the features of the instances solved, which might not include enough customer sparsity to justify deploying an additional truck.

Similarly to the minimum makespan VRPD-RS, the inclusion of robot stations has a positive effect on the objective function. Moreover, between scenarios where $C = 1$ and $C = 2$, there are no apparent differences. This can be explained by observing the utilization of the two robot stations for each case, shown in Table 5.4. Beyond indicating the utilization of each station, this table includes the percentage of deliveries performed by trucks (T), drones in multi-legs (D_A) and roundtrips (D_C), and robots (R), as well. By allowing one more station to be activated, the utilization of both stations is increased. However, the sum of the utilization of both stations when $C = 2$, $\overline{S_W} + \overline{S_A} = 83.3 + 16.7 = 100.0\%$, suggests that, in almost every instance, only 1 station is

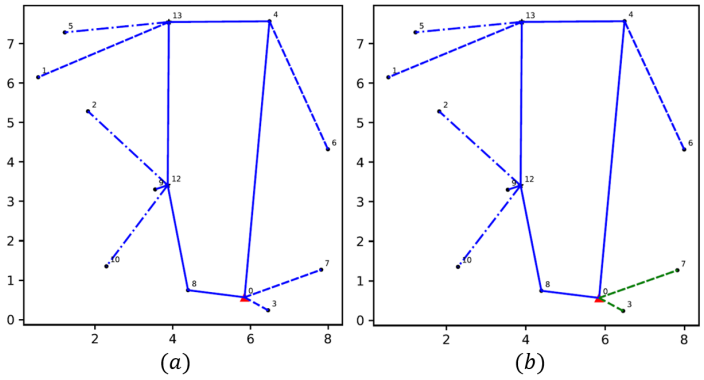

Figure 5.1: Illustration of the impact of additional trucks and drones on a minimum operational cost VRPD-RS solution.

activated. In fact, out of the 30 instances of this category, only 3 considered opening both stations. While this behaviour might not impact the operational cost, the makespan will be further increased in this scenario, since when $(C,|K|) = (2,1)$, only one robot will be performing the same tasks previously performed by two robots.

Table 5.4: Percentage of deliveries performed by each mode and percentage of cases each station is selected for the $(|F|,|D|) = (1,1)$ minimum operational cost VRPD-RS.

n	(C,|K|)															
	(0,0)			(1,2)						(2,1)						
	Resources			Resources				Stations		Resources				Stations		
	T	D_A	D_C	T	D_A	D_C	R	S_w	S_A	T	D_A	D_C	R	S_w	S_A
10	33.0	0.0	67.0	20.0	0.0	41.0	39.0	80.0	20.0	20.0	0.0	39.0	41.0	90.0	20.0
15	27.3	0.0	72.7	22.0	0.0	45.3	32.7	90.0	0.0	22.0	0.0	45.3	32.7	80.0	0.0
20	24.5	0.0	75.5	19.5	0.0	44.0	36.5	70.0	20.0	19.0	0.0	41.5	39.5	80.0	30.0
Avg.	28.3	0.0	71.7	20.5	0.0	43.4	36.1	80.0	13.3	20.3	0.0	41.9	37.7	83.3	16.7

Similarly to the results of Schermer et al. (2020b), in a cost minimization context, drones are used in roundtrips instead of multi-legs. Indeed, among all $(|F|,|D|) = (1,1)$ instances considered, none included a schedule with multi-legs. This phenomenon is intuitive since roundtrips are the drone operations that minimize the distance travelled by trucks, the most expensive resource.

To conclude the analysis of the VRPD-RS features, the two objectives will be compared. In Table 5.5, for each objective, the percentage increase in the secondary objective relative to its optimal value is indicated. This measure is obtained for the makespan and cost through Expressions 5.1 and 5.2, respectively.

$$r_\tau = \frac{\tau_{\omega^*} - \tau^*}{\tau^*} \cdot 100\% \tag{5.1}$$

$$r_\omega = \frac{\omega_{\tau^*} - \omega^*}{\omega^*} \cdot 100\% \tag{5.2}$$

where τ_{ω^*} and ω_{τ^*} designate the makespan associated with a minimum cost solution and the cost associated with a minimum makespan solution, respectively.

Table 5.5: Percentage increase in the alternative objective function caused by the minimization of the main objective.

n	τ^*	τ_{ω^*}	$r_\tau\,(\%)$	ω^*	ω_{τ^*}	$r_\omega\,(\%)$
10	59.5	117.9	100.5	17.3	27.0	55.7
15	65.9	158.0	142.9	18.9	30.7	62.6
20	76.8	194.0	157.5	20.0	35.9	79.7
Avg.	67.4	156.6	133.6	18.7	31.2	66.0

As the results evidence, there exists a trade-off between cost and delivery speed. When the minimal cost is preferred, makespan suffers an average increase of 133.6% relative to its optimal value. When minimizing makespan, the cost is also increased, since the utilization of trucks is favoured (Schermer et al., 2020b). Thus, addressing both objectives simultaneously might be more practically relevant and, given the average impact of each objective on the secondary one, it is expected that the corresponding optimal solutions will resemble more minimal makespan plans. To illustrate the differences between both objectives, the routing decisions for each objective on a particular instance are displayed in Figure 5.2, where $|F| = 2$, $|D| = 1$, $|K| = 1$ and $C = 2$. When makespan is optimized (Figure 5.2 (a)), the delivery area is partitioned in order to balance the workload among the two trucks, whereas for the minimum cost (Figure 5.2 (b)), only one truck is deployed. Furthermore, there is increased parallelization in Figure 5.2 (a) as drone-serviced customers are addressed in multi-legs, where first legs are represented using dashed lines and second legs using dotted lines. Conversely, in Figure 5.2 (b), the truck acts as a mobile station that parks in locations that offer the drones enough range to perform roundtrips.

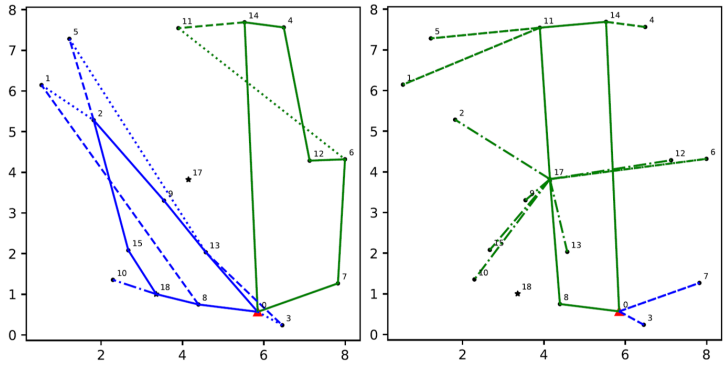

Figure 5.2: Illustration of the operational differences between minimal makespan and minimal cost VRPD-RS solutions.

5.2.2 Performance of the GVNS heuristic

5.2.2.1 Minimum Makespan VRPD-RS

In order to assess the performance of the proposed GVNS, computational experiments using the same problem instances as those used to evaluate the MILP solver were performed. The algorithm

described in Section 4.5 was tested with the following parameters: $\delta_T = 1$ h, $n_{max} = \frac{n^2}{k_{max}} = \frac{n^2}{4}$ and $\delta_P = 5$ min. The results are aggregated in Table 5.6, where columns 'gap' and 'gap*' designate the gap to all MILP solutions and gap to the provably optimal MILP solutions. Additionally, the average computational time, in seconds, is provided, as well as the average percentage improvement relative to the initial solution provided by the constructive heuristics. A measure of efficiency, ε_{GVNS}, which indicates the average amount of time, in seconds, needed to improve the incumbent solution by 1%, is shown. Finally, the number of instances where the GVNS algorithm was capable of matching or improving solutions relative to the solver is displayed.

Table 5.6: Aggregated performance of the GVNS heuristic for the minimum makespan VRPD-RS.

n	F,D	gap	gap*	time	improv.	$\varepsilon_{\text{GVNS}}$	opt.
10	(1,1)	0.4%	0.4%	30.1	12.9%	4.7	22/30
	(1,2)	2.9%	2.9%	76.1	15.1%	7.2	9/30
	(2,1)	1.3%	1.3%	57.6	13.9%	7.2	20/30
	(2,2)	1.6%	1.6%	71.0	14.7%	10.7	17/30
	Avg.	1.5%	1.6%	58.7	14.1%	7.5	68/120
15	(1,1)	0.0%	0.3%	179.3	16.9%	11.3	16/30
	(1,2)	0.8%	-	410.9	18.6%	32.9	19/30
	(2,1)	-2.0%	-	357.1	13.9%	36.2	25/30
	(2,2)	-5.3%	-	533.8	15.8%	40.3	29/30
	Avg.	-1.6%	0.3%	370.3	16.3%	30.2	89/120
20	(1,1)	-2.4%	-	822.3	21.8%	38.7	22/30
	(1,2)	-8.9%	-	1337.4	25.2%	54.0	27/30
	(2,1)	-7.8%	-	1111.7	15.5%	76.7	30/30
	(2,2)	-13.8%	-	1316.9	20.7%	71.8	30/30
	Avg.	-8.2%	-	1147.1	20.8%	60.3	109/120
Total Avg.		-2.8%	1.5%	525.3	17.1%	32.6	266/360

On average, the GVNS algorithm was capable of presenting solutions 2.8% lower than the ones provided by the solver within approximately 9 min of execution. The negative optimality gap is caused mainly due to the computational complexity of the problem. In fact, the gaps are negative already for 15- and 20-customer instances, which were the cases where the MILP solver started to be incapable of determining optimal solutions within the imposed time limit. Nevertheless, the gap provided by the proposed solution approach to optimal solver solutions of 1.5% is acceptable given that it consumes substantially less time. In fact, for $n = 15$, the average gap* was of 0.3% under 6 minutes of execution, whereas the solver required approximately 57 minutes, on average. Furthermore, the algorithm was capable of matching or improving the makespan computed by the solver for 266 cases out of 360 instances, i.e., approximately 74% of the tested instances.

By observing the gap among the several configurations of parameters $|F|$ and $|D|$, it is possible to see that, for settings with 1 truck, 2 drones and 10 or 15 customers, the performance of the heuristic is worse. For 20 customers, the gap is higher for the TSP-D-RS due to the fact that

the solver addresses this problem more effectively. Despite leading to large gaps, the percentage of improvement for 1 truck and 2 drones is constantly larger than for the remaining problems, independently of the instance size. Therefore, this could be connected with a lower performance of the constructive heuristics for this case, which leads to initial solutions with high makespans.

In terms of algorithm scalability, despite the approaches taken to contain the computational times for larger instances, it is possible to observe that these increase considerably. While 10-customer instances can be solved in under 1 min, 20-customer ones can be solved within approximately 19 min, on average. Even though the improvement achieved increases with the instance size, the algorithm becomes less efficient, since the number of seconds required to improve the solution by 1% increases. Nevertheless, when compared to the exact formulation, which requires the 60 min to run 20-customer instances, the algorithm is rather efficient and more effective.

As mentioned in Chapter 4, the defined local search operators are capable of activating and deactivating stations. More precisely, the 'Relocate to Robot' operator is capable of performing both tasks, while 'Relocate to Truck' and 'Relocate to Drone' only lead to a deactivation. However, since the defined algorithm relies on a local search to move across the solution space, in a setting where facility location aspects are stricter, i.e., $C < m$, the optimal selection of robot stations may be hindered and largely dependent on the initial selection performed by $build\_robot\_operations()$. Indeed, to transition between two solutions differing in terms of selected stations where C stations are active, a sequential removal of customers assigned to the station to be deactivated and at least one customer relocation to the station to be activated is required. However, the GVNS may not provide the necessary reach to perform such jump and the algorithm could be stuck in local optima. To assess if the solution approach is prone to such scenarios, consider Table 5.7, where the gap to MILP solutions of the GVNS heuristic is differentiated for the three $(C, |K|)$ settings.

Table 5.7: Performance of the GVNS heuristic in addressing facility location decisions.

F,D	**(C,|K|)**		
	(0,0)	**(1,2)**	**(2,2)**
1,1	-0.6%	0.1%	-1.5%
1,2	-1.7%	-1.3%	-2.2%
2,1	-2.4%	-3.4%	-2.8%
2,2	-5.5%	-4.7%	-7.3%
Avg.	-2.6%	-2.3%	-3.5%

Table 5.7 confirms that elements of facility location constitute a weakness of the solution approach as, on average, the gap to the solver is higher for scenarios where $C = 1$, i.e., where one station must be chosen among the two candidates. For $C = 0$, there are no elements of facility location problems. Additionally, when $C = 2$, every station can be activated independently of the state of the remaining ones, which does not require the previously mentioned sequence of moves to perform alternative station selections. Curiously, an exception occurred for $(|F|, |D|) = (2, 1)$, where the performance of the heuristic was actually better for $C = 1$, on average. Ultimately, even

though the performance is degraded for $C = 1$, the gap to Gurobi's solutions is very competitive, with the worst case occurring for $(|F|, |D|) = (1, 1)$ with a gap of 0.1%.

In an attempt to solve this problem, two algorithms were developed to incorporate a SA acceptance criterion in the GVNS, as opposed to a strict improvement criteria. Let $SA_{complex}$ denote a direct adaptation of the proposed GVNS to include a probabilistic acceptance criteria after performing the RVND. Its architecture resembles the GVNS heuristic and is carefully detailed in Appendix A. Since inserting the proposed algorithm in a simulated annealing framework may be computationally demanding, consider SA_{simple}, also detailed in Appendix A, which encompasses a GVNS with three shaking procedures, 'Relocate to Robot', 'Relocate to Truck' and 'Relocate to Drone', and a local search performed by a VND with the five search operators ordered in increasing neighborhood size. Table 5.8 presents the results of the experiments performed on $(|F|, |D|) = \{(1,1), (2,2)\}$ problem instances to assess the appropriateness of such modifications. These results were obtained using an Intel Core i7-8565U CPU at 1.80 GHz with 8 CPUs executing three threads simultaneously.

Table 5.8: Impact of adding a probabilistic acceptance criterion to the GVNS heuristic.

heuristic	**gap**	**gap***	**opt.**	**time**	**improv.**	ε	**gap$_{C=1}$**
GVNS	-1.9%	2.5%	107/180	564.1	17.7%	33.4	-0.6%
$SA_{complex}$	-2.9%	0.8%	125/180	1136.0	17.3%	79.7	-2.1%
SA_{simple}	-1.2%	2.1%	107/180	816.5	16.5%	66.5	-0.1%

Even though $SA_{complex}$ improved solution quality and solved $C = 1$ scenarios better than the GVNS heuristic, it is substantially less efficient as it more than doubles the computational times. Moreover, it is possible to conclude that the cost of simplifying the GVNS is not compensated by the increased diversification granted by the probabilistic acceptance criterion. Indeed, for SA_{simple}, both solution quality and efficiency decreased.

5.2.2.2 Minimum Operational Cost VRPD-RS

The same experimental conditions were arranged to study the performance of the GVNS algorithm under the minimization of the operational cost. The results obtained under this objective function are displayed in Table 5.9, where gap and gap* designate, respectively, the average gap to all solver solutions and the gap to the provably optimal solutions found by the solver. Additionally, the execution times, improvement relative to initially constructed solutions, efficiency and number of solution where the heuristic matched or improved the results of the solver are presented.

The proposed solution approach was capable of addressing minimum cost VRPD-RS problem instances with an average gap to the MILP solver of 2.5% within an average of 5 minutes. Curiously, the heuristic presented a gap to optimal solutions of 2.2%, which was lower than the global gap. This is due to the fact that Gurobi only failed to reach optimal solutions for a large share of 20-customer instances, where the global optimality gap was of 3.5%. Therefore, when computing the average gap*, the contribution of the gaps achieved for 20 customers will be reduced.

Table 5.9: Aggregated performance of the GVNS heuristic for the minimum operational cost VRPD-RS.

n	F,D	gap	gap*	time	improv.	$\varepsilon_{\text{GVNS}}$	opt.
10	(1,1)	0.6%	0.6%	22.8	25.6%	2.5	27/30
	(1,2)	0.4%	0.4%	30.1	25.4%	15.8	27/30
	(2,1)	0.5%	0.5%	26.9	41.0%	0.7	28/30
	(2,2)	0.6%	0.6%	23.6	39.5%	0.7	26/30
	Avg.	0.5%	0.5%	25.9	32.9%	4.9	108/120
15	(1,1)	2.1%	2.1%	175.2	29.1%	21.6	21/30
	(1,2)	2.6%	2.6%	226.4	28.0%	9.4	18/30
	(2,1)	4.4%	4.9%	194.2	39.3%	5.1	14/30
	(2,2)	4.6%	6.3%	176.8	38.7%	4.8	17/30
	Avg.	3.5%	4.0%	193.2	33.8%	10.2	70/120
20	(1,1)	3.0%	3.3%	587.3	35.2%	25.7	17/30
	(1,2)	3.5%	2.3%	743.5	35.4%	23.1	16/30
	(2,1)	4.3%	6.6%	706.0	44.3%	16.3	15/30
	(2,2)	3.3%	-	614.3	43.8%	14.4	17/30
	Avg.	3.5%	4.0%	662.8	39.7%	19.9	65/120
Total Avg.		2.5%	2.2%	293.9	35.4%	11.7	243/360

One aspect that contributes to higher optimality gaps for the cost objective is the good performance of the MILP solver in addressing the problem. In fact, for those 20-customer instances where Gurobi did not reach the optimal solution, by comparing the results obtained for the TSP-D-RS scenario, where 8 out of 10 instances were solved optimally, with the remaining $(|F|,|D|)$ scenarios, one can deduct that these were very close to optimality.

Another aspect that heavily influences the performance of algorithm for the minimum cost VRPD-RS is the implementation, which sought to be flexible enough to be applied for two distinct problem objectives. This can be demonstrated by observing the percentage of improvement achieved for problems where 2 trucks are available, which is considerable higher than for scenarios that employ a single truck. This is caused by the fact that procedure *build_truck_routes*() computes $|K|$ truck routes independently of the objective at stake, while the minimum cost VRPD-RS solutions considered involve solely one truck route. Moreover, the local search operators are not optimized to specifically address this problem. One evident example is the 'Relocate to Drone' operator which explores a neighborhood composed of both roundtrips and multi-legs, even though minimum cost solutions employ roundtrip operations.

In terms of computational times, the heuristic is efficient, even for 20-customer scenarios. Furthermore, judging by the impact of instance size on algorithm efficiency, the heuristic reveals to be capable of scaling up for bigger problems. Knowing that the minimum cost version of the GVNS algorithm presents reduced computational times and that it involves exploring neighborhoods composed of moves that contradict the rationale behind the computation of minimal cost solutions, the performance could have been better for this objective. Given that in a significant part

of the running time the heuristic may be exploring worse neighbors, e.g., relocating customers to the additional truck, the termination criteria could have been more relaxed. The results achieved in this test stress this remark. In fact, when the heuristic is applied in $|F| = 2$ scenarios, the gap is increased when compared to $|F| = 1$ settings, while the computational times remain stable.

Finally, it is important to mention that the features of the instances influenced algorithm performance. In this case, minimum cost VRPD-RS solutions require solely one truck route. However, in case the distribution of customers is more sparse, the use of a second truck may be viable. In those cases, relocating customers between trucks is not an ineffective local search move.

5.3 Robustness under Congested Scenarios

One clear advantage of performing deliveries using drones and robots relates to the fact that these do not depend on the road network, since drones are ably to fly in the 3D space and robots may utilize pedestrian areas. Consequently, such applications have the potential to not only reduce traffic but also to avoid congestion, which are phenomena typical of urban delivery settings (Agatz et al., 2016). Therefore, in the present section, an analysis to evaluate how drones and robots can help mitigate the effects of road congestion on delivery times will be performed.

To conduct this study, three 20-customer instances of Schermer et al. (2020a) were used. Besides the scenario where congestion is absent, two other scenarios with an increasing level of disturbance were created. The first step was to select which customers were affected by congestion. This was done by randomly placing the epicenter of disruption in the square region. Then, two additional instances were created where the 20% and 40%, respectively, closest customers to the epicenter were affected. For pairs of affected customers, the truck speed was reduced to half, while between affected customers and the remaining ones, truck speed was reduced by a quarter, i.e., not only would the truck have more difficulties moving inside the congested area, as well as when traveling to reach it. The velocity of drones and robots was assumed to not be affected.

The GVNS heuristic was applied using the same parameters as the ones used in Section 5.2, except for δ_T, which was reduced to 30 min. For these scenarios, truck-only systems where $|F| \in \{1, 2, 3\}$ were considered. In addition, tests were performed on the combinations of fleet sizes and maximum allowed stations used to analyze the VRPD-RS and the performance of the heuristic. In order to ensure a more realistic perspective, it was considered that drones hover in the air while waiting, i.e., drone waiting times dissipate energy. Similarly to the experiments of the SA heuristic variants, it was used an Intel Core i7-8565U CPU at 1.80GHz with 8 CPUs executing three threads simultaneously. The results are displayed in Table 5.10, where the minimal makespan and respective operational cost achieved by each configuration under a given congestion level are indicated, as well as the percentage deviation of each metric relative to the base scenario.

Upon settings where 20% of the customers are inserted in congested areas, truck-only scenarios are the ones most affected in terms of the delivery completion time, as these present the largest values of $\Delta_{\tau,20\%}$. Moreover, for these scenarios, the operational cost remains stable, which reveals

Table 5.10: Impact of a congested road network on the performance of several fleet compositions.

Fleet Composition				Congestion Level									
				0%		**20%**				**40%**			
|F|	|D|	C	|K|	$\tau_{0\%}$	$\omega_{0\%}$	$\tau_{20\%}$	$\omega_{20\%}$	$\Delta_{\tau,20\%}$	$\Delta_{\omega,20\%}$	$\tau_{40\%}$	$\omega_{40\%}$	$\Delta_{\tau,40\%}$	$\Delta_{\omega,40\%}$
1	0	0	0	124.5	41.5	143.1	41.5	15.0%	0.0%	158.3	41.9	27.2%	0.9%
	1	0	0	105.1	34.0	108.4	33.2	3.1%	-2.2%	122.2	32.2	16.2%	-5.4%
		1	2	98.4	33.4	100.8	34.2	2.5%	2.6%	107.5	29.6	9.2%	-11.2%
		2	1	96.4	32.9	101.3	32.6	5.1%	-1.0%	115.3	33.9	19.6%	3.1%
	2	0	0	91.8	31.0	95.1	32.3	3.6%	4.1%	109.1	32.1	18.8%	3.5%
		1	2	91.1	31.8	87.7	29.1	-3.7%	-8.5%	93.9	30.5	3.1%	-4.2%
		2	1	88.3	29.7	92.8	31.8	5.1%	7.0%	103.0	31.8	16.6%	7.0%
2	0	0	0	75.5	49.3	84.5	49.2	12.0%	-0.2%	96.1	50.6	27.4%	2.5%
	1	0	0	66.5	43.3	71.5	43.2	7.6%	-0.3%	83.6	45.2	25.8%	4.3%
		1	2	63.0	41.3	67.9	42.8	7.7%	3.5%	75.2	44.6	19.4%	8.0%
		2	1	65.4	41.6	71.0	41.6	8.7%	-0.1%	78.4	42.6	19.9%	2.2%
	2	0	0	62.3	37.7	65.6	40.6	5.4%	7.7%	68.9	38.8	10.7%	3.0%
		1	2	59.0	36.9	60.1	38.5	1.9%	4.3%	66.2	36.5	12.3%	-1.1%
		2	1	63.9	36.5	65.3	40.2	2.2%	10.2%	68.2	37.4	6.7%	2.4%
3	0	0	0	66.1	62.0	72.0	63.2	9.0%	1.9%	78.5	60.9	18.7%	-1.8%

that the truck routes used in a clear scenario are kept. Interestingly, in some scenarios that include drones and robots, the cost is reduced because more tasks are transferred to them.

Focusing on the delivery times with a congestion level of 20%, it is possible to see that including drones and robots reduces the makespan of the respective truck-only scenario as well as the impact of congestion. For $|F| = 1$, scenarios that allow $C = 2$ stations to be activated are more affected than when $C = 1$ or truck-drone tandem scenarios, because the resource responsible for the activation is slowed down. As a result, the instant of activation will be delayed and previously activated stations may not be worth being used. For $|F| = 2$, this situation is not so evident, which may be related to a lower utilization of robots, as evidenced by Table 5.2. Unexpectedly, for $(|F|,|D|,C,|K|) = (1,2,1,2)$, the makespan was reduced in the presence of congestion. However, since speed reductions only apply for trucks, the effect on makespan may be bidirectional and could modify the solution space to contemplate solutions with lower makespan and a higher utilization of drones and robots. In fact, the cost was reduced by 8.5% in this case, which evidences a higher preference for drones or robots.

For settings where 40% of the customers are affected, the impact on makespan is accentuated for truck-only scenarios, as well as situations where drones and robots can be deployed. However, focusing on a specific number of trucks, the addition of drones and robots makes the system more robust to congestion, as makespan increases are lower, and enhances the performance of the delivery system in terms of speed and cost-efficiency. One interesting example would be the difference in performance of the case where three trucks are employed and where two trucks, one drone and two stations are used. In this case, both system configurations allow to present roughly the same average makespan, but the second scenario is capable of doing so at a cost 30% lower.

Chapter 6

Conclusions and Future Work

The present study introduced the VRPD-RS, a VRP variant that models the cooperation of trucks, drones and satellite robot stations to perform deliveries. The problem was formally defined as a MILP, with both a makespan and operational cost objective, that extends the TSP-D-RS proposed by Schermer et al. (2020b) and a solution approach based on the GVNS metaheuristic framework was developed. This problem emerges as a consequence of an increasing interest on applications of autonomous delivery resources by both the academical community and companies.

6.1 Main Conclusions

The MILP formulation of the VRPD-RS was implemented and numerical studies were performed using a state-of-the-art solver. The obtained results for the makespan minimization variant evidenced that the addition of trucks and drones to a delivery system is capable of increasing delivery speed. While the makespan reduction granted by an additional truck may be more pronounced than for a drone, it comes at a price of an increased operational cost. In contrast, conceding one extra drone to trucks has a positive effect on both makespan and cost. Additionally, the usefulness of robot stations was proved by the makespan reductions achieved, as well as by the fact that when it was possible to utilize them, a station was activated in almost every case. Furthermore, the results demonstrate the importance of deciding on the location of stations because the station placed in the position given by the average of all customers' coordinates was almost always preferred.

The computational experiments performed for the minimum operational cost VRPD-RS proved that robot stations have a positive effect on the objective function. Curiously, for the tested instances, the addition of trucks and drones did not contribute to operational cost reductions relative to the respective TSP-D-RS scenario. In fact, the optimal TSP-D-RS routes remained stable regardless of the number of trucks and drones and only the corresponding makespan was affected. Furthermore, minimizing the operational cost neglects the performance in terms of delivery speed.

By assessing the performance of the solver, it was possible to verify the computational complexity of the VRPD-RS. In fact, already for 10 customer instances, Gurobi was incapable of reaching optimal makespan solutions. Even though the cost objective was less computationally

demanding, for some 15-customer instances, the solver failed to optimally solve the problem within the time limit. These results helped demonstrate the need for the development of solution approaches to present near-optimal VRPD-RS solutions in reasonable computational times.

The solution approach proposed in this study consists in a GVNS heuristic that combines five neighborhood structures exclusively developed to handle the features and synchronization requirements of the VRPD-RS. Additionally, this algorithm has the particularity that the local search is performed by a RVND. By solving the problem instances using this algorithm, a global optimality gap of -2.8% and a gap to optimal solutions found by the solver of 1.5% for minimum makespan VRPD-RS problems was achieved, within an average execution time below 9 minutes. For the minimum cost variant, a global optimality gap of 2.5% and a gap to provably optimal solutions computed by the MILP of 2.2% was obtained, for an average runtime below 5 minutes.

The results achieved by the algorithm were analyzed to unearth possible weaknesses of the approach. Given that the algorithm is based on local search procedures, handling elements of facility location of the problem may be hindered, since activating or deactivating a station requires sequential customer relocations. By comparing the performance of the algorithm between scenarios where a different number of stations could be activated, it was proven that, when it is necessary to choose one satellite station, the performance of the algorithm is affected. Even so, the optimality gaps for that scenario were very competitive and the worst-case performance was of 0.1% for a $(|F|,|D|) = (1,1)$ setting. Nevertheless, different heuristic configurations using an acceptance criterion from the SA framework were tested to relax the solution evaluation criteria and allow performing alternative station selections. Even though incorporating such a criterion in the GVNS improved solution quality, the resulting heuristic was less efficient. In fact, this algorithm required, on average, more than double of the computational time required by the normal GVNS.

With the development of an efficient algorithm capable of improving the solutions achieved by the solver, it was possible to use it to prove that delivery systems that include drones and robots are more robust in dealing with traffic congestion. In fact, such systems faced lower makespan increases than truck-only scenarios. Moreover, they were capable of improving the delivery performance in terms of speed and incurred operational cost relative to a traditional delivery concept.

6.2 Future Work

With the definition of the VRPD-RS and an efficient solution approach for it, several opportunities for future studies are created. A possibility would be to perform a sensitivity analysis on the relevant parameters of the problem, e.g., number of resources available for each vehicle type, endurance of the autonomous vehicles, vehicle speeds, capacity. Additionally, since the minimum cost VRPD-RS proved to be a more trivial problem that disregards delivery speed, it might be realistic to address settings where this objective should be optimized under a maximum makespan constraint, e.g., the makespan should not exceed the duration of a working shift. Alternatively, makespan and operational cost could be combined in a multi-objective problem.

In an attempt to further improve the performance of the solution approach, the GVNS version with a probabilistic acceptance criterion, i.e., $SA_{complex}$, could be refined to become more efficient. Alternatively, given the synchronization requirements of the VRPD-RS and the several subproblems it contains, the application of a metaheuristic framework such as Large Neighborhood Search or ALNS could be adopted to move across the solution space more freely. Such frameworks could be more effective in dealing with the satellite station selection.

In order to assess the performance of this delivery concept, more realistic scenarios could be studied. Since one might argue that implementing simultaneously drones and robots in a delivery fleet might be too far-fetched, robot operations could be treated as deliveries to parcel lockers where customers can go to retrieve their parcels. Additionally, since the algorithm uses data structures that store the time instants when customers are visited, the evaluation function could easily be adapted to study a problem with time-windows.

The VRPD-RS, like all models, presents assumptions that distance it from reality. Since drones are a rapidly-changing technology where designs are constantly under improvement, with a progressive adoption of these resources it will be expected that companies will possess heterogeneous resources (Murray and Raj, 2020). Therefore, assuming an heterogeneous fleet of drones, as well as of robots, would be reasonable. Furthermore, the assumption that several drones can be setup simultaneously on a truck may be unrealistic due to space and technology limitations. Therefore, imposing that at most one operation can take place in a given location or that drones form a queue to receive such operations is appropriate. Finally, since the application of autonomous resources appears in same-day delivery contexts, e.g., Amazon set out to perform delivers within 30 minutes of ordering, the VRPD-RS could be defined as a dynamic problem.

Bibliography

Agárdi, A., Kovács, L., and Bányai, T. (2020). Vehicle routing in drone-based package delivery services. In *Solutions for Sustainable Development - Proceedings of the 1st International Conference on Engineering Solutions for Sustainable Development, ICES2D 2019*, pages 151–159.

Agatz, N., Bouman, P., and Schmidt, M. (2016). Optimization Approaches for the Traveling Salesman Problem with Drone. *ERIM Report Series Reference No. ERS-2015-011-LIS*, pages 1–40.

Amorim, P., Dehoratius, N., Eng-Larsson, F., and Martins, S. (2020). Customer Preferences for Delivery Service Attributes in Attended Home Delivery attended home delivery. Chicago Booth Research Paper No. 20-07. Available at SSRN: `https://ssrn.com/abstract=3592597`.

Bouman, P., Agatz, N., and Schmidt, M. (2018). Dynamic programming approaches for the traveling salesman problem with drone. *Networks*, 72(4):528–542.

Boysen, N., Briskorn, D., Fedtke, S., and Schwerdfeger, S. (2018a). Drone delivery from trucks: Drone scheduling for given truck routes. *Networks*, 72(4):506–527.

Boysen, N., Schwerdfeger, S., and Weidinger, F. (2018b). Scheduling last-mile deliveries with truck-based autonomous robots. *European Journal of Operational Research*, 271(3):1085–1099.

Bryan, V. (2014). Drone delivery: DHL 'parcelcopter' flies to German isle. Reuters. `https://www.reuters.com/article/us-deutsche-post-drones/drone-delivery-dhl-parcelcopter-flies-to-german-isle-idUSKCN0HJ1ED20140924`. (accessed March 06, 2020).

Burns, S. (2017). Drone meets delivery truck. UPS. `https://www.ups.com/us/es/services/knowledge-center/article.page?kid=cd18bdc2`. (accessed March 06, 2020).

Campbell, J., Sweeney, D., and Zhang, J. (2017). Strategic Design for Delivery with Trucks and Drones. *Supply Chain Analytics Report SCMA-2017-0201*, pages 1–38.

Carlsson, J. G. and Song, S. (2018). Coordinated logistics with a truck and a drone. *Management Science*, 64(9):4052–4069.

Chang, Y. and Lee, H. (2018). Optimal delivery routing with wider drone-delivery areas along a shorter truck-route. *Expert Systems with Applications*, 104:307–317.

Croes, G. A. (1958). A Method for Solving Traveling-Salesman Problems. *Operations Research*, 6(6):791–812.

Cuda, R., Guastaroba, G., and Speranza, M. (2015). A survey on two-echelon routing problems. *Computers & Operations Research*, 55:185–199.

Daknama, R. and Kraus, E. (2017). Vehicle Routing with Drones. *ArXiv*, abs/1705.06431:1–24.

Dantzig, G. B. and Ramser, J. H. (1959). The Truck Dispatching Problem. *Management Science*, 6(1):80–91.

de Freitas, J. and Penna, P. (2018). A Randomized Variable Neighborhood Descent Heuristic to Solve the Flying Sidekick Traveling Salesman Problem. *Electronic Notes in Discrete Mathematics*, 66:95–102.

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

Di Puglia Pugliese, L. and Guerriero, F. (2017). Last-Mile Deliveries by Using Drones and Classical Vehicles. In *Springer Proceedings in Mathematics and Statistics*, volume 217, pages 557–565.

D'Onfro, J. (2019). Amazon's New Delivery Drone Will Start Shipping Packages 'In A Matter Of Months'. Forbes. `https://www.forbes.com/sites/jilliandonfro/2019/06/05/amazon-new-delivery-drone-remars-warehouse-robots-alexa-prediction/#3141e01b145f`. (accessed March 06, 2020).

Dorling, K., Heinrichs, J., Messier, G. G., and Magierowski, S. (2017). Vehicle Routing Problems for Drone Delivery. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(1):70–85.

Drexl, M. (2012). Synchronization in vehicle routing - A survey of VRPs with multiple synchronization constraints. *Transportation Science*, 46(3):297–316.

El-Adle, A. M., Ghoniem, A., and Haouari, M. (2019). Parcel delivery by vehicle and drone. *Journal of the Operational Research Society*, 0(0):1–19.

Es Yurek, E. and Ozmutlu, H. (2018). A decomposition-based iterative optimization algorithm for traveling salesman problem with drone. *Transportation Research Part C: Emerging Technologies*, 91:249–262.

Ferrandez, S. M., Harbison, T., Weber, T., Sturges, R., and Rich, R. (2016). Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm. *Journal of Industrial Engineering and Management*, 9(2):374–388.

Gonzalez-Feliu, J. (2011). Two-echelon transportation optimisation: Unifying concepts via a systematic review. *Working papers on operations management*, 2(1):18–30.

Gurobi Optimization LLC (2020). Gurobi Optimizer Reference Manual. Gurobi Optimization LLC. `http://www.gurobi.com`. (accessed February 28, 2020).

Ha, Q., Deville, Y., Pham, Q., and Hà, M. (2018). On the min-cost Traveling Salesman Problem with Drone. *Transportation Research Part C: Emerging Technologies*, 86:597–621.

Ha, Q. M., Deville, Y., Dung, P. Q., and Hà, M. H. (2015). Heuristic methods for the Traveling Salesman Problem with Drone. *ArXiv*, abs/1509.08764:1–13.

Ha, Q. M., Deville, Y., Pham, Q. D., and Hà, M. H. (2019). A hybrid genetic algorithm for the traveling salesman problem with drone. *Journal of Heuristics*, (26):219–247.

Hemmelmayr, V. C., Doerner, K. F., and Hartl, R. F. (2009). A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research*, 195(3):791–802.

JD.com (2016). JD.com's Drone Delivery Program Takes Flight in Rural China. JD.com. `https://corporate.jd.com/whatIsNewDetail?contentCode=6IhXLeeSAFLjLLlyuZatDA`. (accessed April 05, 2020).

Jeong, H., Song, B., and Lee, S. (2019). Truck-drone hybrid delivery routing: Payload-energy dependency and No-Fly zones. *International Journal of Production Economics*, 214:220–233.

Kim, S. and Moon, I. (2019). Traveling salesman problem with a drone station. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(1):42–52.

Kitjacharoenchai, P., Min, B.-C., and Lee, S. (2020). Two echelon vehicle routing problem with drones in last mile delivery. *International Journal of Production Economics*, 225(107598):1–14.

Kitjacharoenchai, P., Ventresca, M., Moshref-Javadi, M., Lee, S., Tanchoco, J., and Brunese, P. (2019). Multiple traveling salesman problem with drones: Mathematical model and heuristic approach. *Computers and Industrial Engineering*, 129:14–30.

Lim, S. F. W., Jin, X., and Srai, J. S. (2018). Consumer-driven e-commerce: A literature review, design framework, and research agenda on last-mile logistics models. *International Journal of Physical Distribution and Logistics Management*, 48(3):308–332.

Lipsman, A. (2019). Global Ecommerce 2019. eMarketer. `https://www.emarketer.com/content/global-ecommerce-2019`. (accessed March 02, 2020).

Liu, Y. (2019). An optimization-driven dynamic vehicle routing algorithm for on-demand meal delivery using drones. *Computers and Operations Research*, 111:1–20.

Macioszek, E. (2018). First and Last Mile Delivery – Problems and Issues. In Sierpiński, G., editor, *Advanced Solutions of Transport Systems for Growing Mobility*, pages 147–154, Cham. Springer International Publishing.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif. University of California Press.

Mangiaracina, R., Perego, A., Seghezzi, A., and Tumino, A. (2019). Innovative solutions to increase last-mile delivery efficiency in B2C e-commerce: a literature review. *International Journal of Physical Distribution and Logistics Management*, 49(9):901–920.

Marinelli, M., Caggiani, L., Ottomanelli, M., and Dell'Orco, M. (2018). En route truck-drone parcel delivery for optimal vehicle routing strategies. *IET Intelligent Transport Systems*, 12(4):253–261.

Mbiadou Saleu, R., Deroussi, L., Feillet, D., Grangeon, N., and Quilliot, A. (2018). An iterative two-step heuristic for the parallel drone scheduling traveling salesman problem. *Networks*, 72(4):459–474.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers and Operations Research*, 24(11):1097–1100.

Moeini, M. and Salewski, H. (2020). A Genetic Algorithm for Solving the Truck-Drone-ATV Routing Problem. In *Advances in Intelligent Systems and Computing*, volume 991, pages 1023–1032.

Mogg, T. (2016). Amazon wants to turn street lights and church steeples into drone docking stations. Digital Trends. `https://www.digitaltrends.com/cool-tech/amazon-prime-air-docking-stations/`. (accessed March 16, 2020).

Murray, C. and Chu, A. (2015). The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research, Part C: Emerging Technologies*, 54:86–109.

Murray, C. C. and Raj, R. (2020). The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones. *Transportation Research Part C: Emerging Technologies*, 110:368–398.

Otto, A., Agatz, N., Campbell, J., Golden, B., and Pesch, E. (2018). Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey. *Networks*, 72(4):411–458.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Penna, P., Subramanian, A., and Ochi, L. (2013). An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics*, 19(2):201–232.

Phan, A., Nguyen, T., and Pham, Q. (2018). Traveling salesman problem with multiple drones. In *Proceedings of the Ninth International Symposium on Information and Communication Technology*, SoICT 2018, pages 46–53. Association for Computing Machinery.

Poikonen, S. and Golden, B. (2020). Multi-visit drone routing problem. *Computers and Operations Research*, 113:104802.

Poikonen, S., Golden, B., and Wasil, E. A. (2019). A branch-and-bound approach to the traveling salesman problem with a drone. *INFORMS Journal on Computing*, 31(2):335–346.

Poikonen, S., Wang, X., and Golden, B. (2017). The vehicle routing problem with drones: Extended models and connections. *Networks*, 70(1):34–43.

Ponza, A. (2016). Optimization of Drone-assisted parcel delivery. Master's Thesis, Università Degli Studi Di Padova.

Robinette, T. (2014). HorseFly 'Octocopter' Primed to Fly the Future to Your Front Door. UC News. `https://www.uc.edu/news/articles/legacy/enews/2014/06/e19929.html`. (accessed March 15, 2020).

Sacramento, D., Pisinger, D., and Ropke, S. (2019). An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones. *Transportation Research Part C: Emerging Technologies*, 102:289–315.

Salhi, S., Imran, A., and Wassan, N. A. (2014). The multi-depot vehicle routing problem with heterogeneous vehicle fleet: Formulation and a variable neighborhood search implementation. *Computers & Operations Research*, 52:315–325.

Savuran, H. and Karakaya, M. (2016). Efficient route planning for an unmanned air vehicle deployed on a moving carrier. *Soft Comput. (Germany)*, 20(7):2905 – 20.

Schermer, D., Moeini, M., and Wendt, O. (2018). Algorithms for Solving the Vehicle Routing Problem with Drones. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10751 LNAI:352–361.

Schermer, D., Moeini, M., and Wendt, O. (2019a). A hybrid VNS/Tabu search algorithm for solving the vehicle routing problem with drones and en route operations. *Computers and Operations Research*, 109:134–158.

Schermer, D., Moeini, M., and Wendt, O. (2019b). A matheuristic for the vehicle routing problem with drones and its variants. *Transportation Research Part C: Emerging Technologies*, 106:166–204.

Schermer, D., Moeini, M., and Wendt, O. (2020a). Problem Instances used for the Drone-Assisted Traveling Salesman Problem with Robot Stations. Zenodo. `https://doi.org/10.5281/zenodo.3446016`. (accessed February 23, 2020).

Schermer, D., Moeini, M., and Wendt, O. (2020b). The Drone-Assisted Traveling Salesman Problem with Robot Stations. In *Proceedings of the 53rd Hawaii International Conference on System Sciences*, pages 1308–1317.

Schermer, D., Moeini, M., and Wendt, O. (2020c). The Traveling Salesman Drone Station Location Problem. *Advances in Intelligent Systems and Computing*, 991:1129–1138.

Sharma, S., Routroy, S., and Yadav, U. (2018). Vehicle routing problem: Recent literature review of its variants. *International Journal of Operational Research*, 33(1):1–31.

Sonneberg, M.-O., Leyerer, M., Kleinschmidt, A., Knigge, F., and Breitner, M. (2019). Autonomous Unmanned Ground Vehicles for Urban Logistics: Optimization of Last Mile Delivery Operations. In *Hawaii International Conference on System Sciences*, pages 1538–1547.

Tang, Z., Hoeve, W.-J., and Shaw, P. (2019). A Study on the Traveling Salesman Problem with a Drone. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11494 LNCS, pages 557–564.

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

UPS (2017). UPS Tests Residential Delivery Via Drone Launched From atop Package Car. UPS Pressroom. `https://www.pressroom.ups.com/pressroom/ContentDetailsViewer.page?ConceptType=PressReleases{&}id=1487687844847-162`. (accessed April 08, 2020).

Vanelslander, T., Deketele, L., and Hove, D. V. (2013). Commonly used e-commerce supply chains for fast moving consumer goods: comparison and suggestions for improvement. *International Journal of Logistics Research and Applications*, 16(3):243–256.

Vidal, T., Laporte, G., and Matl, P. (2019). A concise guide to existing and emerging vehicle routing problem variants. *European Journal of Operational Research*, 286:401–416.

Wang, K., Yuan, B., Zhao, M., and Lu, Y. (2019). Cooperative route planning for the drone and truck in delivery services: A bi-objective optimisation approach. *Journal of the Operational Research Society*, 0(0):1–18.

Wang, X., Poikonen, S., and Golden, B. (2017). The vehicle routing problem with drones: several worst-case results. *Optimization Letters*, 11(4):679–697.

Wang, Z. and Sheu, J.-B. (2019). Vehicle routing problem with drones. *Transportation Research Part B: Methodological*, 122:350–364.

Wing Medium (2019). Wing Unveils Plans for First-of-its-Kind Trial with FedEx and Walgreens. Medium. `https://medium.com/wing-aviation/wing-unveils-plans-for-first-of-its-kind-trial-with-fedex-and-walgreens-7f17350daa09`. (accessed March 06, 2020).

X Development LLC (2019). Delivery Drones - Wing. X Development LLC. `https://x.company/projects/wing/`. (accessed March 06, 2020).

Appendix A

Alternative Algorithms

A.1 $SA_{complex}$

The $SA_{complex}$ algorithm is a direct adaptation of the GVNS heuristic proposed in the present study to include the probabilistic acceptance criterion typical of Simulated Annealing-based algorithms. The overall architecture of this heuristic is given by Algorithm 6.

Algorithm 6: $SA_{complex}()$

Result: S
Initialization
$k_{max} = 4$; $l_{max} = 5$; Select the set of shaking procedures $\mathcal{N}_k$ with $k = 1, ..., k_{max}$
Select the set of neighborhood structures $\mathcal{N}_l$ used in the local search, with $l = 1, ..., l_{max}$
$halting\_criteria \leftarrow False$; $t_i \leftarrow 0$; $n_{max} \leftarrow \frac{n^2}{k_{max}}$; $cont \leftarrow 0$; $\alpha = 0.95$; $prob = 0.5$;
$S \leftarrow build\_initial\_solution()$; $T \leftarrow initial\_temperature\_complex(prob)$;
while ***not*** $halting\_criteria$ **do**
  $k \leftarrow 1$; $sol\_changed \leftarrow False$;
  while $k \leq k_{max}$ **do**
    $S' \leftarrow \mathcal{N}_k(S)$;
    $S'' \leftarrow \text{RVND}(S')$;
    if $f(S'') < f(S)$ **then**
      $S \leftarrow S''$; $k \leftarrow 1$; $sol\_changed \leftarrow True$;
    else
      if $random[0,1[< e^{-\frac{f(S'')-f(S)}{T}}$ **then**
        $S \leftarrow S''$; $k \leftarrow k+1$; $sol\_changed \leftarrow True$;
      else
        $k \leftarrow k+1$;
  $T \leftarrow \alpha \cdot T$; $t_f \leftarrow time()$;
  if *sol_changed=False* **then**
    $cont \leftarrow cont + 1$;
  else
    $cont \leftarrow 0$;
  if $(t_f - t_i) \geq \delta_T$ ***or*** $cont \geq n_{max}$ ***or*** $T \leq 0.05$ **then** $halting\_criteria \leftarrow True$;

Note that when the incumbent solution is updated to a worse solution obtained for shaking procedure $\mathcal{N}_k$, the algorithm advances to explore the $k+1^{\text{th}}$ neighborhood structure instead of returning to $k \leftarrow 1$. With such approach, diversification is promoted evenly for all shaking procedures instead of privileging the ones executed first. Moreover, the termination criteria for $\text{SA}_{\text{complex}}$ is changed relative to the GVNS heuristic. Here, the algorithm terminates in case one of the following conditions is verified: total execution time exceeds δ_T time units; the algorithm is incapable of updating the incumbent solution for n_{max} iterations; parameter T is less than or equal to 0.05.

In what concerns the functions called by the $\text{SA}_{\text{complex}}$, the RVND() is the same local search algorithm as the one presented in Algorithm 5. The function *initial_temperature_complex*(*prob*) computes the initial temperature T that would allow to accept the worst sampled case of $f(S'') - f(S)$ with a probability *prob*, which is considered to be 50% in this case. The function is further detailed in Algorithm 7. The GVNS used in this function to perform the sampling of solutions is more simple to achieve fast computations and considers the same four shaking procedures and a local search by VND using local search operators *2-opt* and *Exchange*.

Algorithm 7: initial_temperature_complex(prob)

Result: T
Initialization
$k_{max} = 4$; $l_{max} = 2$; Select the set of shaking procedures $\mathcal{N}_k$ with $k = 1, ..., k_{max}$
Select the set of neighborhood structures $\mathcal{N}_l$ used in the local search, with $l = 1, ..., l_{max}$
$worst\_case \leftarrow f(S)$; $n_{worst} \leftarrow 0$; $n_{samples} \leftarrow 0$; $sample\_size \leftarrow n$; $t_i \leftarrow 0$;
while $n_{samples} < sample\_size$ **do**
  $k \leftarrow 1$; $n_{samples} \leftarrow n_{samples} + 1$;
  while $k \leq k_{max}$ **do**
    $S' \leftarrow \mathcal{N}_k(S)$; $l \leftarrow 1$;
    while $l \leq l_{max}$ **do**
      $S'' \leftarrow \mathcal{N}_l(S')$;
      if $f(S'') < f(S')$ **then**
        $S' \leftarrow S''$; $l \leftarrow 1$;
      else
        $l \leftarrow l + 1$;
    $S'' \leftarrow S'$;
    if $f(S'') > worst\_fitness$ **then**
      $worst\_case \leftarrow f(S'')$; $n_{worst} \leftarrow n_{worst} + 1$;
    $k \leftarrow k + 1$; $t_f \leftarrow time()$;
    if $(t_f - t_i) \geq 2min$ **then** $n_{samples} \leftarrow sample\_size + 1$; $k \leftarrow k + 1$;
if $n_{worst} > 0$ **then**
  $T \leftarrow -\frac{worst\_case - f(S)}{ln(prob)}$;
else
  $T \leftarrow 10$;

A.2 SA$_{\text{simple}}$

The $\text{SA}_{\text{simple}}$ simplifies $\text{SA}_{\text{complex}}$ in terms of the GVNS applied for each temperature value. In the shaking phase, the neighborhood structures applied are those generated by operators: (1) *Relocate*

to Robot; (2) *Relocate to Truck*; and (3) *Relocate to Drone*. The local search is performed by a traditional VND that applies all local search operators in the following order: (1) *2-opt*; (2) *Relocate to Robot*; (3) *Relocate to Truck*; (4) *Exchange*; and (5) *Relocate to Drone*. Algorithm 8 illustrates the implementation of this heuristic.

Algorithm 8: $\text{SA}_{\text{simple}}()$

Result: S
Initialization
$k_{max} = 3$; $l_{max} = 5$; Select the set of shaking procedures $\mathcal{N}_k$ with $k = 1, ..., k_{max}$
Select the set of neighborhood structures $\mathcal{N}_l$ used in the local search, with $l = 1, ..., l_{max}$
$halting\_criteria \leftarrow False$; $t_i \leftarrow 0$; $n_{max} \leftarrow \frac{n^2}{k_{max}}$; $cont \leftarrow 0$; $\alpha = 0.95$; $prob = 0.5$;
$S \leftarrow build\_initial\_solution()$; $T \leftarrow initial\_temperature\_simple(prob)$;
while *not* $halting\_criteria$ **do**
  $k \leftarrow 1$; $sol\_changed \leftarrow False$;
  while $k \leq k_{max}$ **do**
    $S' \leftarrow \mathcal{N}_k(S)$; $l \leftarrow 1$;
    while $l \leq l_{max}$ **do**
      $S'' \leftarrow \mathcal{N}_l(S')$;
      if $f(S'') < f(S')$ **then**
        $S' \leftarrow S''$; $l \leftarrow 1$;
      else
        $l \leftarrow l + 1$;
    $S'' \leftarrow S'$;
    if $f(S'') < f(S)$ **then**
      $S \leftarrow S''$; $k \leftarrow 1$; $sol\_changed \leftarrow True$;
    else
      if $random[0,1[< e^{-\frac{f(S'')-f(S)}{T}}$ **then**
        $S \leftarrow S''$; $k \leftarrow k + 1$; $sol\_changed \leftarrow True$;
      else
        $k \leftarrow k + 1$;
  $T \leftarrow \alpha \cdot T$; $t_f \leftarrow time()$;
  if *sol_changed=False* **then**
    $cont \leftarrow cont + 1$;
  else
    $cont \leftarrow 0$;
  if $(t_f - t_i) \geq \delta_T$ ***or*** $cont \geq n_{max}$ ***or*** $T \leq 0.05$ **then** $halting\_criteria \leftarrow True$;

As can be observed, $\text{SA}_{\text{simple}}$ resembles the $\text{SA}_{\text{complex}}$ heuristic. Regarding the computation of the initial temperature, $initial\_temperature\_simple(prob)$ follows the same implementation as Algorithm 7, except for the fact that the *Exchange* operator is excluded from the shaking neighborhood structures.