Optimisation des livraisons urbaines avec drones et véhicules en parallèle

Gertrude Raïssa Mbiadou Saleu

Université Clermont Auvergne

École doctorale des
Sciences Pour l'Ingénieur de Clermont-Ferrand

T H È S E

présentée par

Gertrude Raïssa MBIADOU SALEU

pour obtenir le grade de

Docteur d'Université

Specialité : Informatique

Optimization of urban deliveries with drones and vehicles in parallel

---

Soutenue publiquement le 29 janvier 2021 devant le jury composé de :

Rapporteurs :		
Yasemin ARDA	-	Professeur des Universités, HEC Liège
Jorge MENDOZA GIMENEZ	-	Professeur des Universités, HEC Montréal
Examinateurs :		
Christelle GUÉRET	-	Professeur des Universités, Université d'Angers
Jean-Philippe GAYON	-	Professeur des Universités, Université Clermont Auvergne
Directeurs de thèse :		
Dominique FEILLET	-	Professeur des Universités, Mines Saint-Étienne
Alain QUILLIOT	-	Professeur des Universités, Université Clermont Auvergne
Co-encadrants :		
Laurent DEROUSSI	-	Maître de Conférences, Université Clermont Auvergne
Nathalie GRANGEON	-	Maître de Conférences, Université Clermont Auvergne

Acknowledgements

I would like to thank all the people who contributed in some way to the work described in this thesis.

First and foremost I want to thank my supervisors Alain Quilliot, Dominique Feillet, Laurent Deroussi and Nathalie Grangeon. I appreciate all their contributions of time and ideas to make my Ph.D. experience productive and stimulating. They were incredible and their guidance helped me in all the time of research and writing of this thesis. I could not have imagined having better supervisors for my Ph.D study.

I wish to thank the members of my dissertation committee: Yasemin Arda, Jorge Mendoza, Jean-Philippe Gayon and Christelle Gueret for generously offering their time, support and good will throughout the preparation and review of this document.

I also want to specially thank Helene Toussaint for helping me in using CPLEX and the laboratory servers for the various numerical tests. Thanks also to the LIMOS staff (Béatrice Bourdieu, Martine Caccioppoli, Severine Miginiac) for their professionalism and responsiveness.

I would like to thank all my PhD colleagues, with whom I have shared moments of deep anxiety but also of big excitement. Their presence was very important in a process that is often felt as tremendously solitaire.

I want to express my deep gratitude to my family: my dear husband Mr TOMEBA YOUMBI Patrick, my parents Mr SALEU René and Mrs SALEU Céline, for encouraging and supporting me throughout this experience.

The last word goes for Prince Hadriel, my baby boy, who was born in 2019, bringing so much light into my life and who has given me the extra strength and motivation to get things done.

This work was sponsored by a public grant overseen by the French National Research Agency as part of the “Investissements d’Avenir” through the IMobS3 Laboratory of Excellence (ANR-10-LABX-0016) and the IDEX-ISITE initiative CAP 20-25 (ANR-16-IDEX-0001). Financial support was also received from the European Union through the European Regional Development Fund program (ERDF – AURA region) and by the Auvergne-Rhône-Alpes region.

Résumé — La croissance de la livraison du dernier kilomètre et de la demande de service rapide poussent la logistique au-delà de la gestion traditionnelle des transports et de l'analyse de la chaîne d'approvisionnement. Une évolution récente de la logistique urbaine implique l'utilisation de véhicules aériens sans pilote à bord ou drones dans le processus de livraison. La livraison par drones offre de nouvelles possibilités mais induit également de nouveaux problèmes de routage complexes.

Les propositions de scénario pour la livraison par drone varient considérablement, les drones étant utilisés indépendamment ou en conjonction avec la livraison par camions. Dans cette thèse, nous abordons un scénario de livraison camion-drone dans lequel un ou plusieurs drones travaillent en collaboration avec un ou plusieurs camions de livraison traditionnels pour distribuer des colis aux clients depuis un dépôt. Le(s) camion(s) et les(s) drone(s) servent différents groupes de clients en parallèle. Un camion sert un sous-ensemble de clients avec un tour unique partant du dépôt et retournant au dépôt tandis qu'une livraison par drone implique un arrêt unique, le drone faisant des allers-retours entre le dépôt et les emplacements des clients. L'objectif est de minimiser l'heure à laquelle tous les camions et drones sont de retour au dépôt, tous les clients ayant été servis. Ce problème est appelé *Parallel Drone Scheduling Traveling Salesman Problem* (PDSTSP) lorsqu'un seul camion est disponible pour livrer. Quand il y a plusieurs camions, le problème est intitulé *Parallel Drone Scheduling Multiple Traveling Salesman Problem* (PDSMTSP).

Nous proposons une heuristique itérative en deux étapes pour le PDSTSP. Une étape de codage transforme une solution en une séquence de clients et une étape de décodage décompose la séquence de clients en un tour pour le véhicule et un sous-ensemble de clients affectés aux drones. Le décodage est exprimé comme un problème de plus court chemin bicritère et est réalisé par la programmation dynamique. Des expériences menées sur des instances de référence tirées de la littérature (de 10 et 20 clients) et de nouvelles instances plus grandes générées à partir de la TSPLIB[1] confirment l'efficacité de l'approche et les résultats permettent une nette amélioration par rapport à la littérature existante.

Pour résoudre le PDSMTSP qui étend le PDSTSP en considérant plusieurs véhicules, nous proposons une méta-heuristique hybride combinant recherche locale itérée et programmation dynamique. Cette heuristique est inspirée de l'heuristique itérative en deux étapes développée pour le problème restreint à un véhicule. 20 instances (de 50 à 199 clients) sont sélectionnées dans la CVRPLIB[2] pour des expériences. Les expérimentations comparant plusieurs variantes de la méta-heuristique hybride donnent un aperçu de ce système de livraison par drone.

Des formulations en Programme Linéaire en Nombres Entiers (PLNE) pour les deux problèmes sont également fournies et des algorithmes simples de type *Branch-and-Cut* sont développés.

Mots clés : livraison par drone, problème de tournées de véhicules, logistique urbaine, heuristiques, méta-heuristiques, MILP.

[1] TSPLIB est une bibliothèque d'exemples d'instances pour le TSP (et les problèmes associés)
[2] Capacited Vehicle Routing Problem LIBrary

Abstract — The growth of last-mile delivery and demand for next- and same-day service is pushing logistics beyond traditional transportation management and supply chain analytics. One recent evolution in urban logistics involves the usage of unmanned aerial vehicle (UAV) or drones in the delivery process. Delivery by drones offers new possibilities, but also induces new challenging routing problems.

Proposals for drone delivery vary widely, with drones being used independently or in conjunction with delivery by trucks. In this thesis we address a truck-drone delivery scenario in wich one or several drones work in collaboration with one or several traditional delivery trucks to distribute parcels to customers from a depot. Truck(s) and drone(s) serve different sets of customers in parallel. A truck serves customers with a single tour starting from the depot, visiting a subset of customers and returning back to the depot, while a drone delivery involves a single stop with the drone departing from and returning to the depot (a drone travels back and forth between the depot and the customer locations). The objective is to minimize the completion time i.e., the time at which all the trucks and drones are back to the depot, with the service of all customers carried out. This problem is called *Parallel Drone Scheduling Traveling Salesman Problem* (PDSTSP) when a single truck is available for delivery. When there are several trucks the problem is coined *Parallel Drone Scheduling Multiple Traveling Salesman Problem* (PDSMTSP).

We propose an iterative two-step heuristic for the PDSTSP. A coding step transforms a solution into a customer sequence, and a decoding step decomposes the customer sequence into a tour for the vehicle and a subset of customers assigned to drones. Decoding is expressed as a bicriteria shortest path problem and is carried out by dynamic programming. Experiments conducted on benchmark instances from the literature (which are instances of 10 and 20 customers) and new larger instances generated from the TSPLIB[3] confirm the efficiency of the approach and the results permit a clear improvement over the existing literature.

To solve the PDSMTSP which extends the PDSTSP by considering several vehicles, we propose a hybrid metaheuristic combining Iterated Local Search and Dynamic Programming. This heuristic is inspired from the iterative two-step heuristic developed for the same problem restricted to a single vehicle. 20 instances of sizes varying from 50 customers to 199 customers are selected from the CVRPLIB[4] for experiments. Computational experiments comparing several variants of the hybrid metaheuristic give some insights on this drone delivery system.

Mixed Integer Linear Programming (MILP) formulations for both problems are also provided and straightforward *Branch-and-Cut* algorithms are developped.

Keywords: drone delivery, vehicle routing problem, city logistics, heuristics, metaheuristics, MILP.

[3]TSPLIB is a library of sample instances for the TSP (and related problems)
[4]Capacited Vehicle Routing Problem LIBrary

Contents

List of Figures

List of Tables

Table of Acronyms

ALNS	*Adaptive Large Neighborhood Search*
CFRS	*Cluster First Route Second*
CVRP	*Capacited Vehicle Routing Problem*
DA	*Determinist Annealing*
DDP	*Drone Delivery Problem*
E-VRP	*Electric Vehicle Routing Problem*
FSTSP	*Flying Sidekick Traveling Salesman Problem*
GA	*Genetic Algoritm*
GRASP	*Greedy Randomized Adaptive Search Procedure*
GVNS	*General Variable Neighborhood Search*
HDP	*Heterogeneous Delivery Problem*
HFVRP	*Heterogeneous Fleet Vehicle Routing Problem*
HGVNS	*Hybrid General Variable Neighborhood Search*
ILP	*Integer Linear Program*
ILS	*Iterated Local Search*
IP	*Integer Programming*
LP	*Linear Program*
LPT	*Longest Processing Time*
LV-SAV	*Large Vehicle Small Autonomous Vehicle*
MC-DDP	*Minimum Cost Drone Delivery Problem*
mFSTSP	*Multiple Flying Sidekick Traveling Salesman Problem*
MILP	*Mixed Integer Linear Programming*
MT-DDP	*Minimum Time Drone Delivery Problem*
MTDRP	*Multi-Trip Drone Routing Problem*
MVDRP	*Multi-visit drone routing problem*

NW-TLSDP	*No-Wait Transit Last-Stretch Delivery Problem*
PDSTSP	*Parallel Drone Scheduling Traveling Salesman Problem*
PDSMTSP	*Parallel Drone Scheduling Multiple Traveling Salesman Problem*
PFA	*Parametric policy Function Approximation*
PMS	*Parallel Machine Scheduling*
RFCS	*Route First Cluster Second*
RTS	*Route Transform, Shortest path*
SA	*Simulated Annealing*
SDDPHF	*Same-Day Delivery Routing Problem with Heterogeneous Fleets*
SEC	*Subtour Elimination Constraint*
TLSDP	*Transit Last-Stretch Delivery Problem*
TSP	*Traveling Salesman Problem*
TSP-D	*Traveling Salesman Problem with Drone*
TSP-MD	*Traveling Salesman Problem with Multiple Drone*
UAV	*Unmanned Aerial Vehicle*
VRP	*Vehicle Routing Problem*
VRPD	*Vehicle Routing Problem with Drones*
VRPDR	*Vehicle Routing Problem with Drone Resupply*
VDRPTW	*Vehicle Drone Routing Problem with Time Windows*
2E-GU-RP	*two-echelon cooperated ground vehicle and its carried unmanned aerial vehicle routing problem*

Introduction

One of the most important revolutions in the world of aviation is the emergence of drones. These are unmanned aerial vehicles (UAVs), operated by a pilot who is on ground or autonomously through software. They have long been used mainly for military missions. Over time, technological advances have boosted the performance of these aircraft. Although the history of military drones dates back a hundred years, starting in the early 2000s, drones have also appeared in the civil domain. Since then, the development of civilian drones for professional and industrial use has experienced a boom that now seems to make them essential tools in many sectors. To date, there are several areas and applications in which drones excel. Examples include agriculture, aerial photography, surveillance of works of art or infrastructure, land use planning, cartography, fire safety, logistics and recreation.

Drone-based parcel delivery in urban areas is one of the most promising application of UAVs. As rapid urbanisation, evolving consumer demands and the emergence of new channels of distribution continue to change the face of the Transportation and Logistics Industries. The ease of the delivery and collection of goods in urban areas has a significant influence on the economic power, quality of life, accessibility and attractiveness of the city. For local governments, the cheap and easy movement of goods is then crucial for ensuring that their cities remain competitive, attractive and environmentally friendly. For logistics providers and transporters, last-mile logistics from the final delivery center to the customer's doorstep are becoming increasingly costly and complex to manage with urbanisation, ever more stringent consumer demands and the growth of new channels of distribution notably e-commerce. The E-commerce boom and new urban restrictions on truck traffic has led to innovative model of parcel distribution.

In December 2013, the chief executive officer of the largest online retailer Amazon Jeff Bezos shared with the world his vision of using flying robots to deliver products. «*I know this looks like science fiction. It's not*» he assured of the service called *Amazon Prime Air* [Ama]; a drone designed to deliver packages in just 30 minutes. Using unmanned aerial vehicles for door-to-door deliveries seemed like a laughable pipe dream when Amazon tossed out that fanciful idea. Even though some people didn't find the concept so far-fetched, many other compagnies and logistics providers like Google, DHL, UPS, Federal Express, Alibaba started getting interested in experimenting with drone delivery.

The transport of small parcels using autonomous drones has the potential to significantly accelerate delivery times, decrease delivery costs and reduce greenhouse gas emissions. Indeed, drones are lightweight, they consume less energy than standard vehicles and they are not subject to congestion problems since they do not follow the road network. On the other hand, drones are limited in terms of endurance, capacity and payload.

If drones are considered today as the future of home delivery, their large-scale use in urban areas is by no means obvious. Concerned about security issues, governments are drastically regulating the use of drones. The risks of collision (with buildings, trees, electric cables, etc.),

questions of liability in case of damage caused by a fall of the machine due to a malfunction, loss of the package to be delivered, noise annoyance caused by low-altitude flights and the saturation of urban airspace are so many factors that limit the popularization of drone delivery in urban areas.

Will we ever have flowers, medicines, or pizza delivered to our homes by drones? The futuristic idea isn't quite anymore. Commercial use of drones is now permitted in some countries that have regulations in place to limit the risks associated with their operation. Between 2019 and 2020, Amazon, UPS and Google have received authorization for commercial drone delivery from the Federal Aviation Administration (FAA) [5] and since 2016, the FAA had already granted temporary authorizations to several drone delivery companies for demonstration or short-distance delivery. European and Asian countries such as France, Germany, United Kingdom, Ukraine, China, Singapore are not left out. For example in France, the DGAC[6] has authorized DPDgroup, the international express subsidiary of Groupe La Poste, to deliver parcels by drone on a regular commercial line in the Var between Saint Maximin-la-Sainte-Baume et Pourrières in December 2016. Considering all of this, the specialists predict the presence of more than 800,000 devices in the sky, used for delivery in 2021.

The scientific community is getting more and more interested in investigating the design of drone delivery systems. Reaserchers adresse different aspects of drone for package delivery. They study environmental impact and energy implication [PKS18][Loh17][Gul17][GT18], delivery drone design and battery life [Bam15][PZC16], operationnal and security aspects (ability to detect or avoid obstacles, GPS navigation to the right address) [Sch16][GM16], routing optimisation (optimizing the delivery time or cost) [MC15][CAR18][Dor+17].

We are interested in the routing optimization aspect. There are several different scenarios for using drones for delivery, with drones being used alone or in combination with other means of delivery such as traditional trucks. They can be divided into two major groups. In the first group, a vehicle is considered to carry one or more drones. When the vehicle stops either at a customer location for delivery or along the road, a UAV can take off from the vehicle, carrying a package to serve a customer. After completing the task, the flying machine must return to the vehicle before it continues on its way. So, there is a kind of synchronization between the vehicle and the drone. In the second group, vehicles and drones make parallel deliveries. A vehicle departs from the depot, delivers a subset of customers and returns to the depot. A drone goes back and forth between the depot and customer locations. Vehicles and drones operate independently. The scenario with a drone being deployed from a truck has received the most attention in the literature because of the synchronization aspect which is a big challenge. Very little research has been done on the scenario where vehicles and UAVs work independently, yet this scenario seems to be also interesting. For this reason, we have chosen to explore this configuration. The fundamental research question we address is: how can hybrid vehicle-drone delivery system best be used to serve a region considering this scenario ?

[5] the U.S. government agency responsible for civil aviation regulations
[6] *Direction Générale de l'Aviation Civile*

Introduced in the literature for the first time by Murray and Chu [MC15], the problem of optimizing delivery times in a delivery system where a vehicle and one or several drones make parallel and independent deliveries to customers has been named Parallel Drone Scheduling Traveling Salesman Problem (PDSTSP). In this thesis we study two variations of the problem:

- the PDSTSP as defined by Murray and Chu, ie by considering that a single vehicle is involved in the delivery process;
- an extended version of the PDSTSP which considers the presence of several vehicles. This variant is named Parallel Drone Scheduling Multiple Traveling Salesman Problem (PDSMTSP).

This thesis is structured as follows:

Chapter 1 presents the context and motivation of this PhD study. The chapter starts by describing the drone's technology history and today's uses. Then it focuses on the use of drones for delivery by showing the motivation and challenges facing drone delivery as well as some current drone delivery running projects handled by actors like online retailers compagnies.

Chapter 2 provides the research background and a detailed literature review on drone delivery. Definitions, concepts and methodologies related to the problem at hand are introduced by exploring how and what trends are shaping transportation and logistics, how drones fit in future delivery scenarios and how the scientific community is approaching drone delivery problems.

Chapter 3 formally states the PDSTSP (where deliveries are split between a single vehicle and one or several drones) and the PDSMTSP (where several vehicles and a fleet of drones are considered). The chapter provides Mixed Integer Linear Programming (MILP) formulations for both problems and presents a simple branch-and-cut a procedure developed in an attempt of obtaining theoritical optimal solutions.

Chapter 4 presents an iterative two-step heuristic for the PDSTSP. The heuristic is composed of: a coding step that transforms a solution into a customer sequence, and a decoding step that decomposes the customer sequence into a tour for the vehicle and series of trips for the drone(s). Decoding is expressed as a bicriteria shortest path problem and is carried out by dynamic programming. To evaluate the proposed heuristic, experiments are carried out on benchmark instances from Murray and Chu's paper [MC15] which are instances with 10 and 20 customers. A new set of larger instances (containing 48 to 229 customers) generated from the TSPLIB is introduced. Sensitivity analyzes compare solutions obtained while changing some parameters of the problem such as the depot position, the percentage of drone-eligible customers, the drone speed factor and the number of drones. Results confirm the efficiency of the approach with a clear improvement over the existing litterature.

Chapter 5 provides a hybrid metaheuristic for the PDSMTSP adapted from the iterative two-step heuristic proposed for the PDSTSP in the previous chapter. The hybrid metaheuris-

tic starts by building a giant tour visiting all customers. A second step (decoding procedure) uses dynamic programming for efficiently partitioning the customers of the giant tour between the set of vehicles and the feet of drones with the restriction that each vehicle route follows the order deffined by the giant tour. Since introducing several vehicles has a huge computational impact on this step, several upper bounding and lower bounding techniques are introduced to cope with this difficulty. Finaly, an improvement step performs some local search moves to converge towards better solutions. Computational experiments are carried out on 20 instances of sizes varying between 50 and 199 customers taken from the CVRPLIB. Several variants of the hybrid metaheuristic are introduced and the results obtained with these variants are compared. The results demonstrate the importance of the decoding procedure and the relevance of the bounding techniques introduced in this procedure.

The conclusion shows the contributions of the thesis and identifies research opportunities.

List of publications during PhD thesis work

The research work carried out during this thesis gave rise to the publications presented below:

Published papers

R. G. Mbiadou Saleu, L. Deroussi, D. Feillet, N. Grangeon, A. Quilliot, An iterative two-step heuristic for the parallel drone scheduling traveling salesman problem, Networks 72 (4) (**2018**) 459-474 [MS+18].

Submitted papers

Raïssa G. Mbiadou Saleu, Laurent Deroussi, Dominique Feillet, Nathalie Grangeon, Alain Quilliot, The parallel drone scheduling problem with multiple drones and vehicles. Submitted to *EJOR (European Journal of Operational Research)*, **2020** [MS+20].

International Conferences

Raïssa G. Mbiadou Saleu, Laurent Deroussi, Dominique Feillet, Nathalie Grangeon, Alain Quilliot. Drone and truck deliveries: solving the parallel drone scheduling traveling salesman problem. *The seventh meeting of the EURO Working Group on Vehicle Routing and Logistics optimization* (VeRoLog), Seville, Spain, June 02-05, **2019** [MS+19].

Raïssa G. Mbiadou Saleu, Laurent Deroussi, Dominique Feillet, Nathalie Grangeon, Alain Quilliot. Optimization of urban delivery systems with drones. *The sixth meeting of the EURO Working Group on Vehicle Routing and Logistics optimization* (VeRoLog),

Amsterdam, Netherlands, July 10-12, **2017** [MS+17].

French Conferences

Raïssa G. Mbiadou Saleu, Laurent Deroussi, Dominique Feillet, Nathalie Grangeon, Alain Quilliot. Méthodes exacte et heuristique pour un système de livraison composé d'un véhicule et des drones travaillant sans synchronisation. *19ème congrès annuel de la Société française de Recherche Opérationnelle et d'Aide à la Décision* (ROADEF), Lorient, France, Février 21-23, **2018**.

Raïssa G. Mbiadou Saleu, Laurent Deroussi, Dominique Feillet, Nathalie Grangeon, Alain Quilliot. Optimisation de systèmes de livraison urbaine par drones. *18ème congrès annuel de la Société française de Recherche Opérationnelle et d'Aide à la Décision* (ROADEF), Metz, France, Février 22-24, **2017**.

Chapter 1

Thesis context

Contents

1.1 Introduction

To properly place our subject in its context, we must first study in more detail everything related to drones in general because this will be useful for the good understanding of our study.

This chapter provides an introduction to *Unmanned Aerial Vehicle* (UAV) talking about drone technology history and the current popular uses for drones. It gives details about drones for online delivery presenting the motivation to adopt drones for deliveries, the barriers to wider commercial adoption, the main actors (drone startups, logistics companies and retailers) actually involved in deliveries with drones and a list of current running drone delivery projets.

The chapter is organized as follows: Section 1.2.1 presents a short history on drone technology. This history was compiled from the following references [Con][Int]. Section 1.2.2 takes a look at the many uses and applications for drones (both for military and civilian purposes). Factors and motivation behind drone for delivery are discussed in Section 1.3.2. Section 1.3.3 presents technological challenges (payload, autonomy, flight range), security challenges (risk of collision, weather conditions) and legislative challenges (regulations in different countries) faced by drone delivery. Some current running projects dedicated to drone delivery are discussed in Section 1.3.4. The chapter ends up with a conclusion.

1.2 Drone Technology History and Today's Uses

Drones, also known as *Unmanned Aerial Vehicles* (UAV), are pilotless and non-crewed aircraft that are capable of flight either by remote control or through the use of on-board computers. UAV's can fly for long periods of time at a controlled level of speed and height and have a role in many aspects of aviation.

Drones have seen a rapid growth in consumer electronics with advances in technology. But, these unmanned aircrafts were originally built for military purposes. Especially, as weapons in the form of aerial missiles guided by remote controls through radio waves.

1.2.1 Brief History

Many cite that the origin of drones dates back to 1849, when Austria attacked Venice using unmanned balloons stuffed with explosives. However, these balloons do not meet the current definition of drones, which according to The Oxford English Dictionary is "a remote-less controlled piloted aircraft or missile". Going by the definition, the first pilotless aircrafts were built during the First World War in 1916. Shortly after, the U.S Army built the Kettering Bug. While continuing to develop UAV technology, in 1930 the U.S Navy began experimenting with radio-controlled aircraft resulting in the creation of the Curtiss N2C-2 drone in 1937. During WWII, Reginald Denny created the first remote-controlled aircraft called the Radioplane OQ-2. This was the first massed produced UAV in the U.S and was a breakthrough in manufacturing and supply drones for the military.

Drones were previously known to be an unreliable and an expensive toy, but in the 1980's this began to change. The Israeli Air Force's victory over the Syrian Air Force in 1982 contributed to this change. Israel used both UAVs and manned aircraft to destroy a dozen of Syrian aircraft with minimal losses. Further, in the 1980's, the U.S. created the Pioneer UAV Program to fulfil the need for inexpensive and unmanned aircraft for fleet operations. In 1986 a new drone was created from a joint project between the U.S. and Israel. The drone was known as RQ2 Pioneer, which was a medium sized reconnaissance aircraft.

Mini and micro versions of the UAVs were introduced in 1990, and the famous Predator was introduced in 2000, which was used in Afghanistan for the search of Osama Ben Laden. Then, in 2013, Amazon CEO Jeff Bezos announced that the company was considering using drones as a delivery method, igniting the public's interest in drone history.

Although many of the most notable drone flights have been for military purposes, technology is continuing to advance and receive more attention. The military has been a catalyst over the years for the development of UAV technology, which has allowed commercial drones to become cheaper, lighter, and more sophisticated. Today, drones aren't just for delivering military payloads in foreign lands. Individuals, commercial entities, and governments have come to realize that drones can have multiple uses (including civil applications).

Now that the technology's growing exponentially, it's hard to say what the future of our drone history will look like. However, for the military applications, drones are expected to become smaller and lighter with long battery, fuel or flight times. There will also be developments in improving the optics and other capabilities further. In the civilian market, there are developments in improving flight times to serve as delivery platforms, emergency service and also as a data collector for agriculture and forestry.

In the following section we are going to look at many of the ways in which drones are currently used.

1.2.2 Drone Uses

Drones most certainly are not the exclusive use of western governments. They have been used for a large variety of applications in recent years. We can categorize drone uses into two classes namely Military uses and Non-military uses.

1.2.2.1 Military uses

As mentionned before, drones were created mainly for military purposes. Countries at war have been using drones for a substantial period of time dating back as early as World War I. Drones made a lot of sense in wars given the fact that the life of pilots didn't have to be risked.

There are many uses for the drones in the military. In [UDO16], the authors present a review on UAV in military operations. Here is a summary of some military-based UAV missions:

1. **Intelligence, Surveillance and Reconnaissance** (ISR): ISR is the term commonly used to characterize operational missions that employ sensors rather than weapons. The main drone use overseas in war zones is reconnaissance of unknown areas/buildings, enemy tracking, and force protection (making sure our troops are safe and no one is approaching them). ISTAR which stands for Information, Surveillance, Target Acquisition, and Reconnaissance is a system using UAVs to gather enemy information, locate target and petrol hostile air space without risking lives of the operators. Gathering such information by reconnaissance UAVs is more effective and avoids putting soldier lives at risk.

2. **Electronic Attack** (EA): EA is a measure to reduce the effectiveness of radar systems, allowing aircraft to fly unharmed among radars and associated missiles. This is done by either distracting the radar with confusing or deceptive information, or by blinding the radar—making it unable to detect, track, engage, or destroy threats. Electronic Warfare (EW) is vital to all types of military operation. Today, small UAV can be used to knock out antiaircraft radar and surface-to-air missile guidance systems.

3. **Strike missions**: "strike" refers to operational missions that put weapons rather than sensors on target. Unmanned Combat Aerial Vehicles (UCAVs) are UAVs equipped for striking targets. They were developed in order to reduce the risk of the human pilots being behind enemy lines. The combat UAVs are capable of neutralizing targets deep in the battlefield with extreme precision and minimal collateral damage.

4. **Suppression and/or destruction of Enemy Air Defense** (SEAD): SEAD certainly qualifies as one of the most dangerous missions for modern air forces. SEAD operations, together with the related Destruction of Enemy Air Defences (DEAD), seek to disrupt, disable and/or destroy hostile, predominantly radar-based air-defence networks to the point were they are unable to respond effectively to the application of air power. UAVs are used to facilitate SEAD with their ability to detect, identify, locate, and track SEAD targets.

5. **Combat Search And Rescue** (CSAR): CSAR is an integral part of army aviation mission. UAV and UCAV platforms can support and even perform many of the key CSAR tasks on the modern battlefield to include finding, fixing, supporting and recovering isolated personnel. For instance, they could support CSAR aircraft involved in recovering isolated personnel by providing a variety of capabilities to include: SEAD, direct fires, surveillance, counter measures and communications relay in isolated areas.

6. **Communications Relay**: communications in military operations are crucial in completing desired missions under highly chaotic situations. Due to dynamic movements of troops, multiple units require wireless communications when they are separated from each other to coordinate the efforts for the mission. Relay stations are often used to enhance wireless signals. UAVs can be used as communication relay stations for supporting wireless communications in military operations.

1.2.2.2 Non-military uses

Drones have many uses outside the military. [Sha+19] present a survey of UAV civil applications and their challenges.

1. **Professional Photography and Videography**: aerial photography and video are certainly nothing new. Plenty of films, television shows, and commercials have used helicopters and planes. Drones have propelled the art of photography and videography to fantastic new heights. They have opened up a myriad of possibilities for photographers and videographers. Now protographers at ground level can remotely control mini electronic aircrafts containing built in or external cameras that can capture a range of shots from various heights, speeds and viewpoints [Che15].

2. **Express shipping and delivery**: Amazon made headlines when they first announced they were testing their Amazon Prime Air [Ama], a drone delivery service. Many assumed it was a joke or marketing ploy, but the concept proof to be viable as internet shopping continues its rapidly growing numbers. Drones can be used to deliver small

packages, pizzas, letters, medicines, etc. at short distances. Many other company's like UPS, DHL, Google, Alibaba, Dominos Pizza have started working on drone delivery projects. Of course, there is a lot of logistical issues that will have to be resolved before regular drone delivery becomes a reality, but it's an exciting idea nonetheless.

3. **Disaster management**: aerial views are critically helpful in large-scale disaster zones. After a natural or man made disaster, a drone provides a quick means to gather information [AF11]. Indeed, drones are designed to be agile, fast, robust and autonomously flown, thus they can access hard-to-reach areas and perform data-gathering tasks that are otherwise unsafe or impossible for humans. Equipped with high definition cameras and radars, drones can give rescuers access to a higher field of view without the need for wasting resources on manned helicopters. [Res+15] presents drone applications for supporting disaster management.

4. **Search and rescue operations**: usually, a rescue operation is a fight against time. The work needs to be done quickly and smoothly. When an incident threatens lives and livelihoods, emergency responders need information and real-time imagery in order to make better decisions and save time. UAVs can be used by emergency services such as police officers, firefighters or volunteer rescue teams. With the help of thermal sensors they can provide situational awareness over a large area quickly, reducing the time and the number of searchers required to locate and rescue an injured or lost person, greatly reducing the cost and risks of search and rescue missions [WT10].

5. **Geographic mapping**: UAV have had an enormous effect in the field of 3D geographic mapping. There are regions on the earth that are not easily accessible to humans. This might include some dangerous coastlines or unattainable mountain tops. But for the purpose of studying the terrain and preparing 3D maps, drones have been put to use. Thus, geologists now find it easier to collect data from these sites to pursue mapping processes [Tah+12].

6. **Building safety inspections**: some companies need to carry out regular inspections in order to ensure safety of their infrastructure. This includes surveying power lines, oil and gas pipelines, wind turbines, bridges and buildings under construction and the likes. Regular aerial monitoring can lead to significant improvements in constructing infrastructure leading to improved performances. Drones are being put to use for these purposes enabling the inspector or team to access the information from a safe position [Esc+12].

7. **Agriculture (Precision crop monitoring)**: low altitude aerial imagery paves the way for multiple agricultural applications. Calculation of input doses, monitoring of biomass production or tomorrow, the detection of leaf diseases and weeds, UAVs have become tools in the service of precision agriculture [ZK12]. Thanks to the images taken by their sensors, these flying technological jewels give agronomic indicators without having to take samples. While flying over a parcel, a drone records a multitude of geo-reference images with centimetric precision.

8. **Wildlife Monitoring**: the tracking, monitoring, and inspection of wildlife across a large area can present challenges relating to time and efficiency. The use of drones has become an important component of spotting, tracking and counting wildlife [Lin+15]; [San15]. Drones are able to provide GPS location and visual information to aid in wildlife inspection. They have a low noise footprint which does not scare the animals. Conservation drones can be equipped with different types of cameras and sensors, including thermal, providing complete and accurate information.

9. **Law enforcement and border control surveillance**: drones might come in handy in the fight against crime. Law enforcement agencies use drones to collect evidence and conduct surveillance. UAV's can help agencies gather essential information in dangerous situations while saving manpower and money [Cra17]; [FW12]. Drones are also increasingly being used as part of border control. For example, they are deployed at the Italy-France border to identify migrants crossing into France.

10. **Weather forecasting**: drones are used more and more to gauge weather, importantly, in real time and in places not monitored in the past. They are better able to predict storm forces than traditional methods, and that ability can be lifesaving when severe weather strikes [Bol03]. They could play an integral role in forecasting and relaying the most accurate information to scientists and the public. In fact, drones are set to be the (better) weather forecasters of the future.

11. **Recreational Purposes**: there are many small drones that are meant for recreational use and the number of hobby drone pilots is quite significant. In simple terms, recreational use refers to the operation of an unmanned aircraft based on personal interests and enjoyment. For instance, when you take aerial photos using a drone for your own personal use this would be considered as using it for recreational purposes. These drones are supposed to be flown by following some specific safety guidelines.

Nowadays, the use of drones takes extremely varied forms, and more and more numerous, both in the military and civil fields. Above we gave a non-exhaustive list of application areas that the use of a drone will upset. In this thesis, we are particularly interested in UAV's for Express shipping and delivery. In the next section, we detail the use of drones for online delivery and present some current running drone delivery projets around the world.

1.3 Drones for Online Delivery: motivation and current running projects

Logistics is essential to any business. Its purpose is to satisfy requests that concern the management of materials (transport, packaging, storage) and related information flows (traceability). For this, logistics is in charge of managing the means that achieve this goal (equipment, machinery) and mobilizes resources (human, material, financial) to achieve it. Goods transport is an important part of the logistics chain. For logistics providers and transporters,

Figure 1.1 Images illustration of some drone uses

last-mile logistics from the final delivery center to the customer's doorstep are becoming increasingly costly and complex to manage with urbanisation, ever more stringent consumer demands and the growth of new channels of distribution notably e-commerce. In urban areas, in the last mile logistics, companies face the problems of congestion and respect of the environment. As a result, city center supply systems must be restructured to fit eco-mobility or sustainable mobility for the design, implementation and management of modes of transport deemed less harmful to the environment, safe and economical, in particular with a lower contribution to greenhouse gas emissions.

Before presenting drones for online delivery, lets start by introducing the so-called *last mile delivery problem.*

1.3.1 Last mile delivery

Last mile logistics is the last link in the distribution chain, when the package moves from bulk goods transport to delivery to the end customer. The last mile is a crucial element in supply-chain. It corresponds to all the elements that must be taken into account in the context of the final delivery of the order to the customer. Last point of contact with the e-shopper in his buying process, it is often the main factor of satisfaction or disappointment of the customer. Indeed, for 62% of them, delivery is the most important criterion when buying online.

The last mile is, unsurprisingly, the most expensive part of the supply chain. As the product gets closer to its final destination, the unit cost of transportation increases and

therefore reaches its peak in the last mile. If the first miles are well controlled (tight flows between stocks and delivery centers), the unit costs of transport are often the highest when we get closer to the final customer. This is the "last mile challenge" faced by many logisticians and their e-merchant customers, among others. This last mile represents more than 20% of the overall cost of the delivery chain. On the environmental side, the cost is also very heavy. According to the city of Paris, 1 out of 5 vehicles circulates to deliver parcels: the transport of goods thus accounts for nearly 50% of diesel consumption and for more than 25% of the CO2 emitted.

Urban logistics therefore represent a complex challenge for logisticians and transporters who manage the flow of goods in the city. This challenge is even more ambitious to note that the e-commerce boom tends to impose delivery times as a strong argument in the purchase decision. The delivery in 24 or 48h becoming almost the new standard of the "Distance Selling", it is now necessary to deliver quickly, reducing the ecological footprint and costs.

Today several methods are used to reduce the impacts related to urban logistics of the last mile:

1. **The pooling of resources**: to share the costs and reduce the environmental impact, one can imagine the use of the same vehicle for different shippers. This would in particular increase the load factor of vehicles and thus reduce their number in circulation.

2. **New ways of collecting**: home delivery weighs heavily on the logistics cost of the last mile: if the addressee is absent (which happens in more than 20% of the cases according to the FEVAD [1]), the delivery person will have to make a new delivery attempt, which comes to increase the ecological and economic bill again. The solution seems to lie in the development of the points of contact. Some companies such as DHL, La Poste, Neopost or Inpost offer safe and accessible boxes 24h / 24h for the withdrawal of packages. These Click & Collect boxes are located in city centers, at railway stations (and even in Autolib stations or supermarket receptions) and complement the relay points which are more limited in time. Also, the relay points have individuals who provide storage areas. End customers travel to pick up their packages.

3. **Clean transport**: these include promoting the clean vehicle (electric, hybrid, using biofuels or soft transport) to ensure last mile logistics. Alternative solutions are already being tested: scooters, electric bicycles ...

4. **Delivery drones**: being studied in many countries, the delivery drone seems to offer great prospects for the development of last mile logistics. All players in logistics and e-commerce are looking into this solution: Amazon, Cdiscount, DHL, UPS, DPD, Geodis or even Google, put a lot of resources into research and development of solutions to quickly deliver customers e-commerce. In France as in many countries, the delivery poses many problems related to the safety of the airspace, but technically these delivery systems by drones already work. The delivery drone that does not have the capacity

[1] Fédération du E-commerce et de la Vente A Distance

today to benefit from regular commercial lines could ensure at least fast deliveries over short distances and in different or even complex geographies such as mountains, islands or isolated territories and realize operations in difficult weather conditions thanks to its low-altitude flight.

1.3.2 Motivation to adopt drones for delivery

Why so many expectations around deliveries by drones? Speed, accessibility, environmental impact ... the advantages of these flying machines are numerous.

The biggest advantage is obviously mobility. Places inaccessible by car or truck can be reached without problems by air. It is thus possible to make deliveries in hard to access areas like mountain and island for example or in rural area. Not having to use roads, drones can also reach their destination faster. This saves time, but also less pollution (greenhouse gas) since these machines are electric.

Delivery by drone will result in lower manual labor costs related to parcel delivery. Companies will benefit from a better return on investment. Drones could also reduce the impact on the environment compared to conventional delivery modes. Faster, cost-effective and environmentally friendly, they would provide a decisive competitive advantage for retailers.

However, everything is not rosy and the difficulties are many to overcome before seeing the sky filled with drones. The next section presents somes challenges faced by the drone delivery concept.

1.3.3 Challenges facing drone delivery

If a headline suggests that delivery by drone is becoming a reality, another suggests that it is not yet for tomorrow. What is it really? Is the technology mature? Will the necessary regulations be put in place? Do consumers have enough confidence?

- **Technological challenge**: autonomy is a first black point that limits the range of flying machines. The more we want to go far, the more the size of the batteries increases. And so the more the transportable load is reduced. Nevertheless, with electric multirotor drones, with a range of 30 to 40 minutes, today we can perform missions over a distance of about 20 km, to carry loads of about 2 kg. This can be enough for most individuals: 90% of the products delivered by Amazon weigh less than 2 kilos.
- **Security challenge**: delivery by drone raises the question of the safety of navigation in the airspace of big cities (risk of collision between drones, with electric wires, lampposts, birds ...). To fly a drone is not without risk, especially if it is automated. Multirotors should be designed to avoid obstacles, especially when flying over a city. Also the weather is an important element to take into account. Indeed, it is quite dangerous to fly a multirotor when the wind blows a little strong.

- **Legislative challenge**: drone delivery is subject to several regulatory issues. In the United States, the FAA (Federal Aviation Administration) states that UAVs must be controlled by a human operator and remain within his or her field of vision. Most countries do not allow out-of-sight flights. The potentially congested airspace still needs to develop an automated tracking management system. NASA and FAA are currently working together with major drone manufacturers to create a ground control system to manage drones that are not in their operator's field of vision, but this should not be completed until 2025. In June 2019, the European Union (EU) became the first region to publish a comprehensive set of rules for ensuring the safe, secure and sustainable use of drones. They cover both commercial and leisure use and aim to foster innovation and growth in the sector while ensuring safety. These are Regulation (EU) 2019/945 and Implementing Regulation (EU) 2019/947 [Eud].

  [Jon17] presents a report which summarizes the status of the international regulatory environment by then for drone delivery services. Table 1.1 gives an overview of which approach specific countries have taken to commercial drone legislation. Not all countries with drone legislation are represented in this table, either because there is not enough detail about legislation or legislation information is not accessible. A more recent report on drone regulations ordered by continents and by countries can be found in [Reg].

Table 1.1 Drone Legislation Approaches, by Country
(Source: Based on RAND research (as of 2017))

Approach	**Definition**	**Countries**	
Outright ban	Countries do not allow drones at all for commercial use.	• Argentina • Barbados • Cuba • India	• Morocco • Saudi Arabia • Slovenia • Uzbekistan
Effective ban	Countries have a formal process for commercial drone licensing, but requirements are either impossible to meet or licenses do not appear to have been approved.	• Algeria • Belarus • Chile • Colombia	• Egypt • Kenya • Nicaragua • Nigeria

Continued on next page

Table 1.1 – continued from previous page

Approach	Definition	Countries	
VLOS[2] required	Drones must be operated within VLOS of the pilot, thus limiting their potential range.	• Belgium • Bermuda • Bhutan • Botswana • Croatia • Ecuador • Jamaica • Latvia • Lithuania	• Luxembourg • Mexico • Nepal • Netherlands • Slovakia • South Africa • South Korea • Switzeland • Thailand
Experimental BVLOS[3]	Exceptions to the constant VLOS requirement are possible with certain restrictions and pilot ratings	• Australia • Austria • Brazil • Canada • China • Czech • Denmark • Finland • France • Germany • Greece • Guyana • Ireland	• Japan • New Zealand • Panama • Poland • Rwanda • Singapore • South Africa • Sri Lanka • Trinidad and Tobago • Uganda • United Kingdom • United States
Permissive	Countries have enacted relatively unrestricted legislation on commercial drone use. These countries have a body of regulation that may give operational guidelines or require licensing, registration, and insurance, but upon following proper procedures it is straightforward to operate a commercial delivery drone.	• Costa Rica • Iceland • Italy	• Norway • Sweden • United Arab Emirates

Whatever the technology or rules, consumer confidence will be decisive. Some studies

[2]Visual Line Of Sight
[3]Beyond Visual Line Of Sight

reveal that faced with the promise of almost immediate satisfaction, these will open their door to delivery by drone. Yet, according to other indicators, they will first want to be sure that drones will not harm their security and privacy.

Even if most consumers aren't sure they want drone delivery, private delivery companies and postal services are quickly moving forward with the notion. In the next section, we present some current running drone delivery projets.

1.3.4 Some drone delivery running projets

Whether in the United States, China, or Europe, several large companies are in the starting blocks and are preparing the moment when regulatory changes will make the delivery of parcels by drones possible. The multiple projects have different characteristics, whether in terms of target (delivery to private individuals, emergency deliveries, delivery in inaccessible areas ...), delivery modality (deposit of the package on the ground, collection point, parachute parcel ...), or technical characteristics and delivery capabilities (autonomy, speed, load supported). These projects are carried by different types of actors: retailers, logisticians, but also technological players.

Amazon brought the idea of drone delivery into the public's imagination in 2013 when it announced it was working on a drone system as part of its overall push to handle delivery in-house instead of relying on outside shippers such as UPS and FedEx. The concept of autonomous drone delivery was derided as pie-in-the-sky then, and some thought it was a joke. The project called ***Prime Air*** [Ama] is one of Amazon's most ambitious projects. It consists of delivering parcels by drone within 30 minutes maximum. The goal for the American giant is to design software that will integrate flying drones into the airspace beyond the field of vision in a secure manner. The drones will be able to fly at low altitude in complete safety thanks to this software, which must also integrate the management of the unforeseen and the bypass of the bad weather. Amazon hasn't provided an official launch date for the Amazon Prime Air drone delivery program. However, with Prime Air development centers in the U.S., France, Austria, United Kingdom, and Israel, the program is closer to launch than ever. The first test of Amazon Prime Air was conducted on the 7th December 2016 in Cambridge, England: a drone delivered an item to a customer in just 13 minutes after it was ordered. In June 2019 Amazon has unveiled the latest version of its Prime Air delivery drone, a hybrid aircraft that's capable of vertical takeoff and landing as well as sustained forward flight. Amazon envisions the drone delivery service being an exclusive benefit for Amazon Prime members, when it does launch. Amazon's UK trial, continued refining of drone design and technology, and Amazon's submission of airspace proposals for drone operation in the U.S. and other countries has largely silenced the skeptics. On August 31, 2020, Amazon received approval from the Federal Aviation Administration to operate its fleet of Prime Air delivery drones, bringing the company one step closer to its goal of 30-minute drone deliveries to Amazon Prime members in the US.

In August 2014, Google followed by revealing its drone delivery project called ***Wing***

[Win]. The project is handle by "X" (formerly Google X), the Alphabet company dedicated to experimental projects. Project Wing is an autonomous delivery drone service aiming to increase access to goods, reduce traffic congestion in cities, and help ease the CO2 emissions attributable to the transportation of goods. This project consists in designing a fleet of unmanned aircraft to collect packages from businesses and homes, carry them to a chosen location, and lower packages to the ground at the designated spot, such as near a doorstep or in a backyard. The automated aircrafts fly pre-planned routes and use sensors and software to avoid collisions with drones and other obstacles. The drone's route is mapped by project Wing's Unmanned Air System Traffic Management (UTM) platform, which allocates the drone's flight path and makes sure the aircrafts are able to follow routes that avoid each other, buildings, trees and other hazards. Wing has conducted tens of thousands of test flights both in the U.S. and in Australia over the past six years. The first public tests for project Wing were held in 2014 in rural Queensland, Australia where the Wing team successfully transported a first-aid kit, candy bars, dog treats, and water to farmers. Then in September 2016, the team delivered burritos to students at Virginia Tech in what was, at the time, the largest and longest drone delivery test on U.S. soil. Alphabet has continued to concentrate its drone testing on Australia, and in fall 2017 began making direct deliveries to homes in the rural Googong area outside of the city of Canberra, allowing a number of households in the region to order food and medicines using a smartphone app and have them delivered by drone. In early 2018, project Wing drones began deliveries in the neighboring region of Tuggeranong, which includes more built-up districts and homes with smaller backyards — in what appears to be a trial of how the system performs in more densely populated areas. In April 2019, the Australia aviation authority granted a regulatory approval to Google's Wing drones for public deliveries. The regulatory approval was given after an 18 month trial and 3000 deliveries. The service works by partnering with local businesses including coffee shops and pharmacies to deliver their products "in minutes." Wing's regulatory approval comes with restrictions. Drones will not be allowed to fly over main roads, they will only be allowed to fly between 7am and 8pm on Monday to Friday (or between 8am and 8pm on Sundays), and they will be restricted from flying too close to people. Customers in eligible homes will also be given a safety briefing about interacting with the drones. Also in April 2019, the Federal Aviation Administration (FAA) authorized Alphabet's Wing Aviation to start delivering goods via drones. In September 2019 Alphabet's Wing drone delivery service has finally taken off in the US. The drone company, owned by Google's parent launched a test program in Christiansburg, Virginia. For the test, Wing is partnering with FedEx and Walgreens for home deliveries.

In September 2014, DHL Deutsche Post, Germany's privatized postal system launched its ***parcelcopter***, a helicopter-style drone which could deliver medications and urgently needed goods to the remote North Sea island of Juist [Dhl]. In 2016, the third generation of the Parcelcopter was tested from January to March, flying packages from the German community of Reit im Winkl to a plateau located about 1.200 meters above sea level. DHL said the trip from base to mountain plateau took only eight minutes and was repeated for 130 deliveries during the testing period. A car making the same journey by road would take 30 minutes. The system was tested out using real members of the community who brought their packages to what DHL calls a "Packstation" or "Parcelcopter Skyport". The special facility is like a

small post office with a small helipad on top of it. When a package is inserted, a Parcelcopter swoops into action, grabbing hold of it. Then, the roof of the skyport opens and the drone zips off to deliver the goods to their destination. The entire system works automatically, without human. DHL's flyer has the ability to rise vertically like a helicopter and then convert to zip forward like an airplane. It can fly at 70 km/h (about 43 mph) carrying a parcel weighing up to 2.2 kg for 8.3 km (just over five miles). In 2018, together with the Deutsche Gesellschaft für Internationale Zusammenarbeit (GIZ) and drone manufacturer Wingcopter, DHL has successfully completed the pilot project in Tanzania. Over a period of six months, three experts tested the delivery of medicines using an autonomous drone. The DHL Parcelcopter 4.0 flew missions to an island in Lake Victoria. On average, the aircraft needed 40 minutes to complete the 60km flight from mainland to island. In total, the project included more than 180 take-offs and landings. The Parcelcopter 4.0 was in the air for about 2.000 flight minutes and flew over 2.200km. In May 2019, DHL launches its first regular fully-automated and intelligent urban drone delivery service. DHL Express and drone manufacturer EHang have entered into a strategic partnership to develop a fully automated drone delivery solution for China's metropolitan areas. Shipments are transported daily between the DHL Service Center in Liaobu, Dongguan, Guangdong Province and a customer's location eight kilometers away.

In February 2015, the giant Chinese online trading Alibaba conducted a first drone delivery experience via Taobao, its first online shopping arcade [Ali]. Alibaba launched its first deliveries by drones specifically to Beijing, Shanghai and Canton. The operation involved the delivery of 450 tea of the Taobao brand. The remote-controlled drones were not flown right to consumers' doors. They were landed outside residential buildings, where packages could be collected by human-type couriers for last-mile duty. In May 2018, the Chinese authorities gave their approval to Alibaba so that its online meal service "Ele.me" could use a fleet of drones to deliver users. In this way, flying systems will be able to move freely in the 58 square kilometers of the Jinshan Industrial Park in Shanghai.

In 2016 the American courier company UPS has launched tests on the use of drones to deliver parcels to remote or hard-to-reach areas, in collaboration with the manufacturer of drones CyPhy Works. The tests began with a simulation of urgent drug delivery from Beverly in Massachusetts to Children's Island at about five kilometers off the Atlantic coast. In February 2017, UPS successfully tested a new drone delivery method. Instead of having them go from a sorting center, UPS uses trucks as mobile launching bases [Ups]. The craft in question is an octocopter (***HorseFly***) capable of carrying a load of 4.5 kg and to fly independently for 30 minutes. It is installed on a platform located on the roof of the truck in which a hatch is arranged to let the transport basket attached under the device. The driver loads the package directly from the inside of the van and triggers the delivery via a touch screen installed on the dashboard. The drone will then follow the preprogrammed route associated with the package. Once the package is dropped, the octocopter takes off and rejoins the truck, which has meanwhile moved to another address. In March 2019, UPS and Matternet launched a new service using drones to transport blood and other medical samples between the various buildings at WakeMed Raleigh's medical campus in North Carolina. In October 2019, UPS has received a certification from the North American Federal Aviation

Administration (FAA), allowing to operate commercial delivery drones in the US, through a new subsidiary called UPS Flight Forward. This certification allows night flying, transport of goods weighing more than 25 kg and the possibility for drones to fly outside the line of sight of a pilot. The company will initially expand its drone delivery service further to support hospital campuses around the country, and to provide solutions for customers beyond those in the healthcare industry. UPS Flight Forward plans in the future to transport a variety of items for customers in many industries, and regularly fly drones beyond the operators' visual line of sight.

Since 2016, the Californian start-up Zipline has deployed a strong drone delivery business in Rwanda to facilitate the transport of blood to the most remote areas of the country to allow blood transfusion operations in rural clinics [Zip]. A total of 4000 flights were made and 7000 units of blood were transported. In 2018, the start-up announced their ambition to also break into the US market and take advantage of the upcoming announcements of the federal regulator authorizing experiments at broader scales. Thus, in April 2018, Zipline unveiled a new generation autonomous aircraft. It has been touted as the fastest delivery drone on the market (top speed: 60 mph, about 96 km/h). According to Zipline, the new drone is also capable of traveling 100 miles (about 160 km) in one go with a load of 3.8 pounds (about 1.72 kg).

7-Eleven, a chain of grocery stores with more than 56,000 points in 18 countries, obtained in July 2016 the authorization to test delivery by drones, which the brand realized immediately thanks to the support of Flirtey, a startup based in Reno, Nevada where this delivery service was set up [Lip]. It is then in Nevada that 7-Eleven has set up its drone delivery service for its customers in Reno. In just a few months, Flirtey's drones delivered 77 deliveries, mostly hot dishes, drinks or drugs. The order and the delivery follow-up were carried out via a mobile application that was distributed to a dozen customers at a 7-Eleven point of sale. They could order an item, then be notified when the order is prepared, when the drone takes flight and finally when it arrived at the GPS position of the customer. On average, deliveries were made in about ten minutes.

In August 2016, Domino's Pizza Enterprises Limited (Domino's) has partnered with New Zealand start-up Flirtey, which has made drone delivery its core business, to launch the world's first commercial pizza delivery service [Dom]. While the Australian branch of Domino's Pizza Enterprises has entered the top 20 Forbes innovative companies, Domino's Pizza Enterprises Limited and Flirtey have opened their partnership with a drone pizza delivery demonstration in Auckland, New Zealand. Tests were carried out within a radius of 1.5 km around equipped restaurants. The drones sent their pizzas at 30km/h and at an altitude of about 60 meters, before sending an alert to their customers once their order arrived. This successful demonstration took place under the aegis of the Civil Aviation Authority (CAA) and the New Zealand Minister of Transport. Domino's is studying the possibility to expand the delivery area to 10 km in diameter. Domino's also wants to extend the experiment in other countries, including Australia, Belgium, the Netherlands, Japan, Germany and even France. However few laws concerning drones still exist, and the regulations are very different in different countries.

In June 2014, DPDgroup [Dpd], the international parcel delivery network of GeoPost

International Express subsidiary of Groupe La Poste (France), in partnership with Atechsys, an SME in the Var department began testing to develop a drone. After numerous tests and more than 600 hours of flight, the DPDgroup drone demonstrated, in September 2015, its ability to fly in complete autonomy by transporting a 1.5kg package over a distance of 14 km. After two years of successful testing, DPDgroup has received authorization from the french General Direction of Civil Aviation (DGAC) to deliver packages by drone on a regular line of 15km. Once a week the DPDgroup drone connects Saint-Maximin-La-Sainte-Beaume to Pourrières, in the Var. This line allows to deliver an isolated business incubator that brings together a dozen of start-ups in the field of technology. Specifically, a drone (six electric rotors) DPDgroup flyes from a postal relay located in Saint-Maximin-La-Sainte-Baume to deliver a package (3 kg maximum) to a business incubator located about 15 kilometers, and this at the speed of 30 km/h. Note that all stages of delivery are automated: loading, takeoff, landing and storage. It is an operator approved by the DGAC who still pilot the drone remotely.

It is with the conviction that the future of parcel delivery is now played in the air that the e-retailer Cdiscount (Casino Group) has made delivery one of its major axes of innovation. In December 2016, this commitment was illustrated by a first test: Cdiscount thus delivered by drones toys to the children hospitalized with the Pellegrin hospital of Bordeaux. This unprecedented drone delivery operation in urban areas was a first in Europe. On the occasion of the international exhibition of aeronautics and space, which was held from 18 to 25 June 2017 at Le Bourget, the Bordeaux company Air Marine has formalized the kick-off of the ***Pelican*** project (for project study of delivery of air parcels in New Aquitaine) [Cdi]. The objective is to allow Cdiscount to develop drone delivery in the city center. In detail, the Pelican project aims to develop a drone to transport products up to 5 kg in urban areas. The drone will not be controlled manually but will move autonomously. On the other hand, a person will be in charge of monitoring the evolution of several aircraft at the same time and will be able to intervene if necessary. The idea is not to deliver parcels to customers homes, but to make the link between the warehouse and collection points in the city center, with the possibility for the customer to withdraw his parcel at this point relay or to be delivered on the last hundred meters by bicycle couriers. In Febuary 2018 The Region New Aquitaine voted a grant of 500.000€ to give a first impetus to the project, with a realization expected by 2022. The idea is to transport parcels up to 5 kg or 10 kg in urban areas. This collaborative project brings together leading multidisciplinary experts, working alongside Cdiscount, to eventually lead to an industrialization of this new mode of delivery: Air Marine, Thales, Ims, Robotics Industry, Onera and Serma Technologies.

In April 2016, Autralia post started testing the usage of remotely piloted drones internally for parcel delivery, with the support of the Civil Aviation Safety Authority [Aus]. The first tests began, with the drones of the company ARI Labs. Australia Post's drones are able to travel distances of about 15 km in 15 to 20 minutes before needing recharging, and their maximum payload is 1.5 kg. Australia Post sees drones as a way to serve its expansive rural zones. Their UAVs are not autonomous, unlike the Amazon Prime Air project, and need to be piloted by a drone pilot. In the Australia Post project, the pilot transports his drone into a car to approach the addressee and takes him off the road.

Switzerland's Swiss Post is testing drones for package delivery, especially for emergency supplies and medical deliveries [Swi]. They entered into partnership with a California start-up Matternet to develop an innovative means of transport by drone that allows two Swiss hospitals in Lugano to easily exchange laboratory samples. Since, the prototype has been tested on more than 70 flights between the two Lugano hospitals. The tests were carried out in cooperation with the Swiss Federal Office of Civil Aviation. In 2017, the Swiss Post has taken a new step by launching a test phase of the first commercial application of drones, in cooperation with the hospital network of Tessin. The drone of the Swiss Post is 80 cm in diameter, without its propellers. It weighs 9.5 kg, including the battery. It is designed to carry light goods weighing up to 2 kilos, at a cruising speed of 36 kilometers per hour. And that, at an altitude of 110 meters and at temperatures between -10 and 40 degrees Celsius. The autonomous drone is controlled via an application developed by Matternet for iPhone. Using a GPS, he locates the landing pad, which attracts it with an infrared signal.

In collaboration with the Infocomm Development Authority of Singapore (IDA), Singapore Post announced in October 2015 their successful trial of a drone-based point-to-point recipient-authenticated mail delivery system [Sin]. The test flight, which lasted five minutes and flew 2 kilometers (1.24 miles) between Lorong Halus and Pulau Ubin, carried a letter as well as a t-shirt over the Serangoon Harbour. What makes the SingPost delivery drone system unique is that it integrates a secure authentication system and an application for users to ensure that the package is delivered to the intended recipient. In addition, the app allows users to select their preferred delivery date and time.

The Ukrainian postal service, Ukrposhta, has been working with the Israeli company Flytrex Aviation to test the use of drones for parcel deliveries in the city of Bucha [Ukr]. This appears to be a move made to complement a recent bilateral agreement made with Kazakhstan to speed up e-commerce parcel delivery between the two countries. The pilot project started the 1st of June 2016 however the program is expected to be fully up and running by 2020.

Table 1.2 Drone delivery projects

Actor	**Origin**	**Project Name**	**Autonomous**	**Remotely-controlled**	**Max Load**	**Speed**	**Autonomy**
Amazon	US	PrimeAir	✓		2.5Kg	upto 96Km/h	
X (Alphabet)	US	Wing	✓		1.3Kg	upto 120Km/h	n.d.
UPS	US	HorsFly (WorkHorse)	✓		4.5Kg	upto 72Km/h	30 min
DHL	Germany	Parcelcopter	✓		2Kg	upto 70Km/h	8.3Km
7-Eleven	US	n.d.	✓		n.d.	n.d.	n.d.

Continued on next page

Table 1.2 – continued from previous page

Actor	Origin	Project Name	Autonomous	Remotely-controlled	Max Load	Speed	Autonomy
Alibaba	China	n.d.		✓	n.d.	n.d.	n.d.
Zipline	US	Zipline	✓		1.72Kg	upto 96Km/h	160Km
DPD Group	France	Atechsys	✓		3Kg	upto 30Km/h	20Km
Domino's Pizza	US	n.d.	✓		2Kg	upto 30Km/h	1.5Km
Cdiscount	France	Pelican	✓		5Kg	n.d.	n.d.
Australia Post	Australia	n.d.		✓	1.5Kg	n.d.	15Km
Swiss Post	Swiss	n.d.	✓		2Kg	36Km/h	10Km
SingPost	Singapore	n.d.	✓		0.5Kg	n.d.	2.3Km
Ukrposhta	Ukraine	n.d.			3Kg	n.d.	23Km

The list of drone delivery projects presented above is not an exhaustive list. Many other companies not mentioned here have launched similar projects. This shows that drone delivery is really an interesting and emerging field. More informations can be found in [Dro].

1.4 Conclusion

The utility of UAVs has been rapidly expanding. No more exclusively used in the military field, today, drones have several applications in the civilian field. Since sustainable last-mile delivery has become a topic of increased focus in logistics over the past few years, many companies have started looking towards drones as a cost-effective, environmentally friendly answer and they have started investigating the potentials of using drones in their delivery systems.

The current chapter's purpose was to provide a context to the problem under study, regarding the drone's technology history and characteristics, the motivations for introducing drones into delivery systems, the various challenges (technological, security, legislative) related to delivery of packages with drones and some drone delivery running projects.

Taking everything into consideration, it is apparent that introducing drone for package delivery requires a careful examination, given how the drones unique characteristics add new constraints that create different logistics challenges. The use of drones introduces new restrictions that make the logistics concepts and methodologies commonly applied for transportation in last-mile deliveries unprepared to handle drone deliveries properly.

The topic of last-mile drone-based delivery has gained a lot of attention among researchers from various fields. The scientific literature has started to study drone delivery with drones being used independently or in conjunction with other robots or means of transportation like trucks. Models and solution methods for combinatorial optimization problems such as Traveling Salesman Problem (TSP) and Vehicle Routing Problem (VRP) have been extended to deal with drone delivery.

In the next chapter, we first present a literature review on the Vehicle Routing Problem (VRP) as a preliminary foundation on which the truck-drone delivery problem builds upon. Then, we review the existing literature on the Vehicle Routing Problem with Drone (or Drone Routing Problem) where UAVs are integrated into the routing problem.

Chapter 2

Litterature Review

Contents

2.1 Introduction

The problem we are dealing with is related to the field of Routing Problems. The *Vehicle Routing Problem* (VRP) is one of the most well-studied problems in operations research. As a generalized case of the well known *Traveling Salesman Problem* (TSP), for a given fleet of vehicles and a set of customers, the VRP seeks for the optimal set of routes in order to deliver goods to the customers and satisfy their demand. The VRP plays a central role in the fields of transportation and logistics.

With the rise of the interest in integrating drones in delivery applications due to the acknowledgement of the potential advantages of employing drones in transportation, the scientific literature has started to study the last-mile delivery with drones, by extending traditional TSP and VRP models and solution methods. Several authors developed new models to address situations where drones are employed in transportation. Therefore, the following chapter will describe their contributions for delivery problems involving drones.

The chapter is organized as follows : first of all, we start by providing a literature review on the Vehicle Routing Problem (VRP) which serves as the logical starting point to establish a foundational understanding for the underlying methodology of drone delivery. VRP most relevant variants, exact and approximate existing solution methods are presented (see Section 2.2). Then, Section 2.3 focuses on papers that have integrated drones into a routing problem. We selected the most recent papers and provided an extensive classification and taxonomy of their problems and their formulation. Related problems are classified into several groups according to whether UAVs are used independently or combined with trucks and according to the roles assigned to drones in the combined operations. We also give a synthetic overview of the reviewed papers highlighting the problem name, the number of vehicles and drones considered, the drone capacity and the resolution techniques in Section 2.4.

2.2 Vehicle Routing Problems

The VRP was first introduced by Dantzig and Ramser [DR59] as a generalization of the older and fundamentally applied TSP. The TSP can be described as a problem in which a driver must visit a set of n cities exactly once and travel back to the origin point with the objective of minimizing either distance or time [Lin65]. While the TSP is defined as the optimization of a single tour with a single vehicle, the VRP builds upon this preliminary framework and incorporates additional complexities such as multiple vehicles and capacity constraints. There are different classes or variations of the VRP. In [KP12], Kumar et al. present a survey on the VRP and its variants. The following section presents some important problems derived from the VRP, focusing on those that show some connections with the problems studied in this thesis.

2.2.1 Most relevant variants of the VRP

In the literature, many different variants of the Vehicle Routing Problem were presented (Laporte [Lap92], Toth and Vigo [TV14]). In this section, we only present the most relevant variants that can be related to the issues encountered when drones are introduced into the routing problem.

Capacitated Vehicle Routing Problem (CVRP)

The Capacitated VRP (CVRP) is a VRP in which a fleet of identical vehicles of finite capacity, based at a single depot, must ensure delivery of items to several customers (or cities) each of which has requested a certain quantity of goods. The items have a quantity, such as weight or volume, and the vehicles have a maximum capacity that they can carry. The set of customers visited by a vehicle designates its tour. Each customer must be visited exactly once and each tour begins and ends at the depot.

The objective of the CVRP is to minimise the total cost, i.e., the sum of the distances or travel times of vehicle tours, while respecting the capacity constraint of the vehicles: the quantity of goods delivered on a tour must not exceed the capacity of the vehicle that insures it [Ral03].

Vehicle Routing Problem with Time Windows (VRPTW)

The VRPTW is an important variant of the VRP and a basic distribution management problem that can model many real-world problems. It consists in designing a minimum cost set of routes, starting and ending at a central depot, for a fleet of identical vehicles of finite capacity which services a set of customers with known demands. Each customer must be visited exactly once and has to be supplied within its time window (a time interval during which its service (e.g., loading or unloading of goods) must be completed). The total demand handled by any vehicle must not exceed its capacity.

A vehicle can arrive earlier, before the start of the time window, but it must wait until service is possible. In this case the total waiting time can be taken into account in the model and can be a goal to minimize [Del+07]. If it arrives later, the service can not be rendered and the corresponding client will never be satisfied. In this case, the problem is called "hard time window constraints". The number of vehicles can be fixed in advance, as in [LYY99]; [Zhu00] or be considered as one of the variables of the problem [Per+02]. The objective of the problem is then to minimize the number of vehicles and the total distance traveled to serve the customers without violating time window constraints. In so-called soft time window models, vehicles can serve the customers outside their time windows but at the cost of some penalty.

Vehicle Routing Problem with Pickup and Delivery (VRPPD)

In the VRP with Pickup and Delivery (VRPPD), a vehicle fleet must satisfy a set of transportation requests. Each request is defined by a pickup point, a corresponding delivery point, and a demand to be transported between these locations. The requested transport could involve goods or persons. The objective function generally minimizes system costs but can also imply quality of service. This model covers two categories of problems [PDH08]:

- In the first one (unpaired pickup and delivery points), customer requests are independent, which means that each unit picked up (at the depot or at a delivery center) can be used to deliver any receiving customer. Generally, this problem is referred to as Pickup and Delivery VRP (PDVRP) and is a single product problem, that is, all products transported are of the same type (see for example [CM99]).
- In the second category (paired pickup and delivery points), customer requests are dependent (linked): each transport must link a specific origin and destination. Two main variants of this model can be found: the Pickup and Delivery Problem (PDP) and the

Dial-A-Ride Problem (DARP). PDP is about freight transport [Amb+04], while DARP deals with passenger transport [AD03].

Multi-Depot Vehicle Routing Problem (MDVRP)

The Multi-Depot Vehicle Routing Problem (MDVRP) is a variant of the VRP in which more than one depot is considered [Min05]. Customers are served by several vehicles; each one is located in one of the several depots established in different places. In a MDVRP, the number and locations of the depots are predetermined. Each depot is large enough to store all the products ordered by the customers. Each vehicle starts and finishes at the same depot. The location and demand of each customer is also known in advance and each customer is visited by a vehicle exactly once. The objective of the problem is to find routes for vehicles to service all the customers at a minimal cost in terms of number of routes and total travel distance, without violating the capacity and travel time constraints of the vehicles.

The Heterogeneous Fleet Vehicle Routing Problem (HFVRP)

The HFVRP differs from the classical VRP in that it deals with a heterogeneous fleet of vehicles having various capacities, fixed costs and variable costs. Therefore, the HFVRP involves designing a set of vehicle routes, each starting and ending at the depot, for a heterogeneous fleet of vehicles which services a set of customers with known demands. Each customer is visited exactly once, and the total demand of a route does not exceed the capacity of the vehicle type assigned to it. The routing cost of a vehicle is the sum of its fixed cost and a variable cost incurred proportionally to the travel distance. The objective is to minimize the total of such routing costs.

There are a couple of HFVRP variants often found in the literature. They are basically related to the fleet limitation (limited or unlimited) and the costs considered (dependent and/or fixed). The HFVRP with unlimited fleet, also known as the Fleet Size and Mix (FSM), was proposed by Golden et al. [Gol+84] and it consists of determining the best fleet composition and its optimal routing scheme. Another HFVRP version, called Heterogeneous VRP (HVRP), was proposed by Taillard [Tai99] and it consists in optimizing the use of the available fixed fleet.

Two-Echelon Vehicle Routing Problem (2E-VRP)

The Two-Echelon Vehicle Routing Problem (2E-VRP) is an extension of the classical Capacitated VRP, where the delivery from a single depot to the customers is managed by routing and consolidating the freight through intermediate depots called satellites [Cra+10]. From a physical point of view, freight in 2E-VRP is delivered as follows:

- Freight arrives to the depot, where it is consolidated into 1st-level vehicles;

- Each 1st-level vehicle travels to a subset of satellites, and then returns to the depot;
- At each satellite, freight is transferred from 1st-level vehicles to 2nd-level vehicles;
- Each 2nd-level vehicle starts from a satellite, performs a route to serve the designated customers, and then returns to the same satellite for its next cycle.

The goal is to serve customers by minimizing the total transportation cost, and satisfying the capacity constraints of the vehicles and satellites.

Vehicle Routing Problem with Profits (VRPP)

The VRPP simultaneously choose a subset of customers from potential customers and design the routes to serve these customers. In these problems, visiting each customer has a profit that represents its relative importance compared to the other customers. Hence, there are two conflicting objectives: travel-cost minimization and profit maximization. Depending on how these objectives are compromised, three classes of VRPP have been studied in the literature:

- Profitable Tour Problems (PTP): in the PTP, both objectives are combined in a single objective, which is to maximize the difference between the total collected profit and the total travel cost;
- Team Orienteering Problems (TOP): in the TOP, the objective is to maximize the total collected profit where the total travel cost must not exceed a given threshold;
- Prize Collecting Routing Problems (PCRP): in the PCRP, the objective is to minimize the total travel cost such that the total collected profit is greater than a predetermined amount.

Feillet et al. [FDG05] surveyed TSP with profits, a subclass of VRPP. [VSVO11] and [GLV16] reviewed papers on TOP. For a general survey of VRPP, one can refer to [ASV14].

Electric Vehicle Routing Problem (E-VRP)

The use of Electric Vehicles (EVs) in freight and personal transportation is starting to get momentum. Because of the economic benefits and the environmental regulations, several companies in different sectors have started to use EVs in their operations. The E-VRP is an extension of the VRP for goods delivery but with the incorporation of EVs [Zha+18][Mon16][EC19].

Thus the problem consists of designing routes to serve a set of customers using a fleet of EVs. Due to their relatively short driving range, EVs may need to detour to charging stations (CSs) to replenish their battery, especially in the context of mid-haul or long-haul routing [SSW18][Vil+18]. Therefore, key decisions in E-VRPs concern not only the sequence

in which the customers are served, but also where and by how much to charge the vehicles. So the objective of the E-VRP is to obtain a set of routes with the minimum operational cost that serve customers requests while satisfying the driving range limitations and charging requirements of electric vehicles.

The problems we are dealing with in this thesis enrich the VRP by adding a particular type of vehicle: drones. They can be assimilated to a heterogeneous fleet vehicle routing problem in which drones are electric vehicles having a limited driving range, requiring battery recharge and having a capacity of 1. In a delivery system where vehicles and drones are working independently, once the customers are partitioned between vehicles and drones, the routing of the vehicles part corresponds to the resolution of a VRP problem. This VRP can take several forms taking into account capacity constraints, time window constraints, multiple depot, etc.

2.2.2 Exact Methods

Many researchers over the years have proposed a wide range of exact and approximate approaches to find solutions for VRP (with its variants) and TSP. Surveys on solution methods for the VRP and the CVRP have been published by Toth and Vigo [TV02][TV14], Baldacci et al. [BMR12], Laporte [Lap09], Golden et al. [GRW08] and Cordeau et al. [Cor+07]. Therefore, the academic environment for these problems has consequently developed into a robust foundation on which we are able to build upon to establish a preliminary understanding of vehicle routing with drones.

Exact methods rely on the use of algorithms that lead safe to the optimal solution. Several exact methods have been developed. The book edited by Toth and Vigo [TV14] suggests that exact methods for the VRP can be classified into three main categories: Branch-and-Bound algorithms, Branch-and-Cut algorithms and Branch-and-Price algorithms. Below, we present these three types of methods.

1. **Branch-and-Bound** :

   This method belongs to the class of tree search methods. It has been proposed for the first time by Land and Doig [LD60]. Primarily, it consists of building a search tree representing the solution space and pruning branches of this tree that do not contain any interesting or feasible solution. The sub-problems that form the tree are called nodes. There are three types of nodes in the Branch-and-Bound tree. The current node that is being processed, the active nodes that are in the problem queue, and the inactive nodes that were pruned during the running of the algorithm. A linear programming solver is generally used to try to find an entire optimal solution. If this fails, a phase of decomposition (Branch) of the problem into sub-problems is necessary. Applied to the VRP, another possible implementation consists in gradually building routes arcs by arcs. When a node is processed, a branch is then created for each arc extension possibility. Combinatorial bounds are then used to prune branches. In general, solution

times are very long and this method only allows solving small size problems.

2. **Branch-and-Cut** :

   Branch-and-Cut is a generalization of Branch-and-Bound. The term "Branch-and-Cut" has been introduced by Padberg and Rinaldi [PR87] for the solution of the TSP. It is used when integer programming formulations can be reinforced with large-size sets of valid inequalities or when the number of constraints is naturally very large. When the linear programming bound is computed at a node of the tree, separation algorithms are called to identify violated inequalities and add them to the formulation. This way, the formulation is progressively reinforced, which permits to improve the bound while keeping a limited number of constraints [TV02].

3. **Branch-and-Price** :

   Another type of exact method for the VRP is based on set partitioning formulations. In these formulations, a binary variable is introduced for every feasible vehicle route and indicates if the route is selected or not in the solution. A feasible VRP solution is then a selection of vehicle routes that satisfy all customer requests. Set partitioning formulations are obtained from a problem reformulation called Dantzig-Wolfe decomposition [DW60]. The advantage of this reformulation is that it provides a tighter linear programming relaxation, the inconvenient being that it generates a model with a very large number of variables. “Very large” here means that it is practically impossible to even enumerate all the variables. Branch-and-Price is a combinatorial optimization method for solving these programs. It combines the Branch-and-Bound algorithm with column generation. Column generation is a linear programming technique aiming at solving large-scale linear programs and is applied at each node of the search tree. The approach is based on the observation that most variables will be non-basic, i.e.,will be equal to zero in an optimal solution. Thus, the vast majority of columns (variables) are irrelevant and can be excluded from the LP formulation. Resolution is initiated with a very limited number of columns and a pricing problem is iteratively executed to find columns that should be present to guarantee optimality. This pricing problem identifies variables with a negative reduced cost. Column generation permits to reduce computational and memory requirements. A tutorial on column generation and Branch-and-Price for vehicle routing problems can be found in [Fei10]. If cutting planes are used to tighten LP relaxations within a Branch-and-Price algorithm, the method is known as *Branch-Price-and-Cut* or *Branch-and-Cut-and-Price* [Sav97][Bar+98][Fuk+06][BCM08].

2.2.3 Approximate Methods

Exact methods usually have problems with real life applications, and may require significant computation time even on small-size instances. It is for this reason that the use of the approximate methods has proven to be very useful. These methods make it possible to obtain good quality solutions in reasonable amount of time for larger-size problems, but without any guarantee of optimality. They can be subdivided into two classes: heuristics and metaheuristics.

2.2.3.1 Heuristics

The principle of a heuristic method is to find, in a reasonable amount of time a solution of hopefully good quality. Classical VRP heuristics can be further classified into three broad categories: construction heuristics, two-phase heuristics and improvement heuristics.

- **Construction heuristics** : construction heuristics build a single solution gradually. Like greedy methods, they operate a series of partial and definitive choices (such as the insertion of a client or the junction of two routes) which cannot be revoked subsequently. Constructive heuristics are usually employed to provide a starting solution to an improvement heuristic. Several such heuristics have been proposed over the years and are fully described in the first edition of Toth and Vigo book [TV02]). This kind of heuristics includes:
  - *Savings* algorithm in two versions parallel and sequential proposed by Clarke and Wright [CW64]. The *savings* algorithm initially constructs back and forth routes $(0, i, 0)$ for each customet i from depot 0 and gradually merges them by applying a saving criterion.
  - *Nearest Neighbour* (NN) algorithm which consists of constructing a route from a randomly chosen point and, extending the route interactively by inserting a node that has not been visited, which is nearest to the current node.
  - *Insertion heuristics* proposed by Mole and Jameson [MJ76]; and Laporte et al. [Lap+00] where the solution is created by inserting customers who have not been assigned to a route.
- **Two-phase heuristics** : two-phase heuristics divide the problem into two stages: customer allocation to route (clustering of customers into feasible routes) and determination of the order of visit (actual route construction). Inside this category Fisher and Jaikumar [FJ81] proposed the strategy Cluster-First Route-Second (CFRS) while Beasley proposed the strategy Route-First Cluster-Second (RFCS) [Bea83]. In the first case, vertices are first organized into feasible clusters, and a vehicle route is constructed for each of them. In the second case, a tour is first built on all vertices and it is subsequently segmented into feasible vehicle routes. An essential building block of Route-First Cluster-Second heuristics is the *Split* Algorithm.

  Algorithms using the CFRS method include:
  - *Fisher and Jaikumar algorithm* [FJ81] that solves a Generalized Assignment Problem to form the clusters;
  - *Sweep algorithm* [GM74] where feasible clusters are initially formed by rotating a ray centered at the depot and a vehicle route is then obtained for each cluster by solving a traveling salesman problem;
  - *Petal Algorithm* [RBL96] which contains two main phases: a *route generation* phase that generates a bundle of routes (one route is called a petal), and a *route selection* phase to select the best combination of routes by solving a set partitioning problem.

The *Split* procedure was proposed by Beasley [Bea83] as a second phase of a RFCS approach for the Capacited VRP. The first phase generates a giant tour visiting all customers (solving a TSP) by relaxing both vehicle capacity and maximum tour length. Then a decomposition of this sequence using a shortest path algorithm leads to a solution of the initial routing problem. This technique has led to successful metaheuristics for various Vehicle Routing Problems in the last decade. In 2001 the *Split* was used in a memetic algorithm for solving the Capacitated Arc Routing Problem [LPRC01]. In 2004 appeared one of the best method for the VRP, also using a *Split* procedure [Pri04]. This approach has since been adopted for several related issues, in particular problems with: time windows, heterogeneous fleet, dial-a-ride, location routing, or truck tours with trailer.

Two-phase heuristics can be classified under constructive heuristics.

- **Improvement heuristics** : these methods are also called local search methods and they are based on the concept of a neighborhood. A neighborhood of a solution p is a set of solutions that are in some sense close to p, for example because they can be easily computed from p or because they share a significant amount of structure with p. In improvement heuristics, in each iteration, improvements are applied to an initial solution until a local optimum is met. The initial solution can be generated randomly or even through a construction heuristic. Iterative improvement searches in the neighborhood of the initial solution a lower cost solution. If such a solution is found, it replaces the current solution and the search continues. Otherwise, the algorithm returns a locally optimal solution. There are several variations of this basic algorithm. *First improvement* generates the neighborhood incrementally and selects the first solution of better cost than the current one. *Best improvement* generates the complete neighborhood and selects the best solution within this neighborhood. Classical improvement heuristics perform intra-route and inter-route moves.

  In the first case, one can apply neighborhood structures designed for the TSP, such as λ-OPT exchanges (Lin [Lin65]), in which λ edges are removed and replaced by λ other edges. A generalization of this simple principle (where λ is modified dynamically throughout the search) forms the basis for one the most effective approximate algorithms for solving the symmetric TSP, the Lin-Kernighan algorithm [LK73]. We can also mention the Or-OPT operator [Or77] which consists in moving strings of 3, 2, and 1 consecutive vertices to another location in the tour.

  Inter-route improvement moves are carried out by operating on several routes simultaneously. Van Breedam[VB94][VB96] classifies the improvement operations as:

  - *String Cross* : two strings of vertices are exchanged by crossing two edges in two different routes. (2-opt*: two edges from different routes are replaced by two new edges)
  - *String Exchange* (swap): two strings of at most k vertices are exchanged between two routes.
  - *String Relocation*: a string of at most k vertices is moved from one route to another (usually $k = 1$ or 2)

– *String Mix*: *String Exchange* and *String Relocation* operators are combined and the operation that improves the most the current solution is selected.

Inter-route improvement moves also include *cyclic transfers* [TP93] where b routes are considered and k customers from each route are shifted to the next route of the cyclic permutation.

2.2.3.2 Metaheuristics

Metaheuristics are methods designed to escape local minima. The term "metaheuristic" was mentioned for the first time in [Glo86]. The goal of metaheuristics is similar to that of heuristics: to obtain good quality solutions in a reasonable time. However, unlike a heuristic, the general scheme of metaheuristics is totally independent of the problem to be treated. In addition, the metaheuristics contain exploitation mechanisms (local search, crossover, ...) and exploration mechanisms (perturbation, mutation, ...) which make it possible to avoid the blocking in a local minimum and without going in cycles. Thus, they make it possible to explore the search space efficiently in order to give very good solutions but also without guarantee of optimality. Metaheuristics can be subdivided into three classes: metaheuristics based on local search, population-based metaheuristics, and hybrid metaheuristics.

- **local search based metaheuristics** :

  Local search methods explore the solution space by moving at each iteration from a solution to another solution in its neighborhood. Metaheuristics based on this approach include:

  – *Simulated Annealing* (SA) [RR94][Tan+01]: based on the Monte Carlo algorithm, SA was conceived in the 80s and has since then seen widespread use to find approximate solutions to the TSP and many other discrete optimization problems. SA always accepts a better or equal cost solution as a new current solution, and accepts a worse solution with a certain probability, which allows the search to escape from local minima. A well-known application of SA to the VRP is that of Osman [Osm93].

  – *Determinist Annealing* (DA) [DH93][DS90]: is a deterministic variant of *Simulated Annealing*. The key concept of deterministic annealing algorithm is to provide a coefficient to guide the walking direction of incumbent solution instead of random walk. DA for the VRP is discussed in [KXX03].

  – *Tabu Search* (TS) [Glo89][Glo90]: TS uses a neighborhood search procedure to iteratively move from one potential solution x to an improved solution x' in the neighborhood of x, until some stopping criterion has been satisfied. The basic principle of TS is to continue the search for solutions even when a local optimum is met by allowing moves that do not improve the solution and using an *adaptative memory* that forbids or penalizes certain moves that would return to a recently visited solution (cyclical movements). In [ZK10] a TS providing good results for the VRP is provided.

- *Iterated Local Search* (ILS) [LMS19][Thi09]: the key idea of ILS is to iteratively build a chain of solutions returned by some underlying algorithm, typically a local search heuristic until a stop criterion is verified (e.g., a time limit, a number of outer iterations or a number of consecutive iterations without solution improvement). More precisely, a search by descent is applied to an initial solution to generate a better solution. Then a new search by descent is applied on this new solution after having "pertubed" it. The solution obtained is compared with the initial solution to know if it replaces it or not. All of this represents an iteration of ILS. [CZH18] is a paper on solving a last mile delivery problem using an ILS approach.
- *Greedy Randomized Adaptive Search Procedure* (GRASP) [FR89]: GRASP is an iterative metaheuristic, in which each iteration consists of two phases. A *construction* phase that builds a feasible solution and a *local search* phase that investigate the neighborhood of the previous obtained solution until a local minimum is found. The best overall solution is kept as the result.
- *Variable Neighborhood Search* (VNS) [MH97]: VNS consists of systematically changing from one neighborhood to another every time a local optimum is found (the goal is to identify better local optima). The algorithm starts with an initial solution and iteratively applies several neighborhoods (for example 2-OPT, 3-OPT, etc.) in a descent way until no further improvement is possible. A new cycle can be restarted after the last neighborhood has been applied. This is repeated until a given termination condition is met (a preset number of cycles or when no further improvement is possible). VNS applied to the VRP is discussed in [Kyt+07].

- **population-based metaheuristics** : unlike previous methods that were trying to improve a single "individual solution", population-based metaheuristics work explicitly with a population of solutions. The basic principle is that these methods treat a population of solutions globally. At each iteration, they build a new population based on the previous one. In other words, they evolve from a population of solutions, in order to obtain a set of the most adapted solutions. This evolution is based on transformations and cooperation between individuals, which individually represent a solution. Among these methods, we can mention:

  - *Genetic Algorithm* (GA) [GH88]: GAs have emerged from the work of Holland [Hol+92]. They make use of techniques inspired from evolutionary biology such as selection, mutation, inheritance and recombination to solve a problem. Starting from some initial population, the search mechanism of a simple GA is divided into four phases: (1) evaluation of each solution in the population, (2) selection of parent solutions, (3) application of crossover and mutation operators to parent solutions to generate offspring solutions and (4) replacement of the old population by the new population of offspring solutions. This process is repeated for a number of iterations or until the system does not improve anymore [Pot09]. A successfull application of GA to the VRP is the work of Prins [Pri04].
  - *Ant Colony System* (ACS)[Dor92][DMC96]: ACS is based on the actual behavior of ants when they are looking for food. The core of ant's behaviour is the communication between the ants by means of chemical pheromone trails, which enables

them to find shortest paths between their nest and food sources. The basic idea of ant colony algorithms is to work on a population of solutions. Each ant moves through the research space. A common data structure, shared by all the ants, contains information about the pheromone accumulated in this space. The ants mark the best solutions, and take into account the previous markings to optimize their research. An ant colony algorithm for the CVRP is given in [ML04].

- *Scatter Search* (SS) [Glo77]: the principle of the SS method is to generate diversified solutions which are then improved. From these improved solutions, we extract a set of reference solutions that will serve as the basis for the generation of new solutions resulting from the combination of solutions from the reference set. Examples of the application of this approach to the VRP are available in [SW06][TZP10].

- **hybrid metaheuristics** : the trend is currently to use hybrid methods. There are several possible types of hybridizations. The hybridization mode that seems most fertile is the combination of metaheuristic methods based on the local search and metaheuristic methods based on population. The essential idea of this hybridization is to fully exploit the power of both types of methods by applying local search to solutions in a population before combining them. For example, an evolutionary method can detect good regions in the search space, while a local search effectively explores promising regions by limiting the risk of missing out on an optimal solution without seeing it. Hybridization can also refer to the combination of metaheuristic methods with exact methods [Pri04][EM09][ZZ09] (ofen also called *matheuristics*). For example, a metaheuristic can provide bounds to a Branch and Bound method. The combination increases the power of these methods. Unfortunately, the necessary computing times can become prohibitive because of the number of individuals handled in the population. To solve this problem, we can apply the parallelization of these algorithms on parallel machines [VA95]; [THG97].

Figure 2.1 gives a summary of VRP solution techniques.

2.3 Integration of Drones in Last-Mile Delivery

The problem of parcel delivery with drone has received increasing attention these last years. Proposals for drone delivery vary widely, with drones being used independently or in conjunction with other robots or means of transportation like trucks. Delivery drones may be launched and recovered at fixed facilities (e.g., Delivery Centers or retail store locations), from relocatable facilities (e.g., platforms that can be moved to different locations), or from trucks themselves.

In the combined operations of aerial drones and vehicles, synchronization may or may not be required. No synchronization is required if UAVs and vehicles perform independent tasks, such as independent deliveries from the central warehouse.

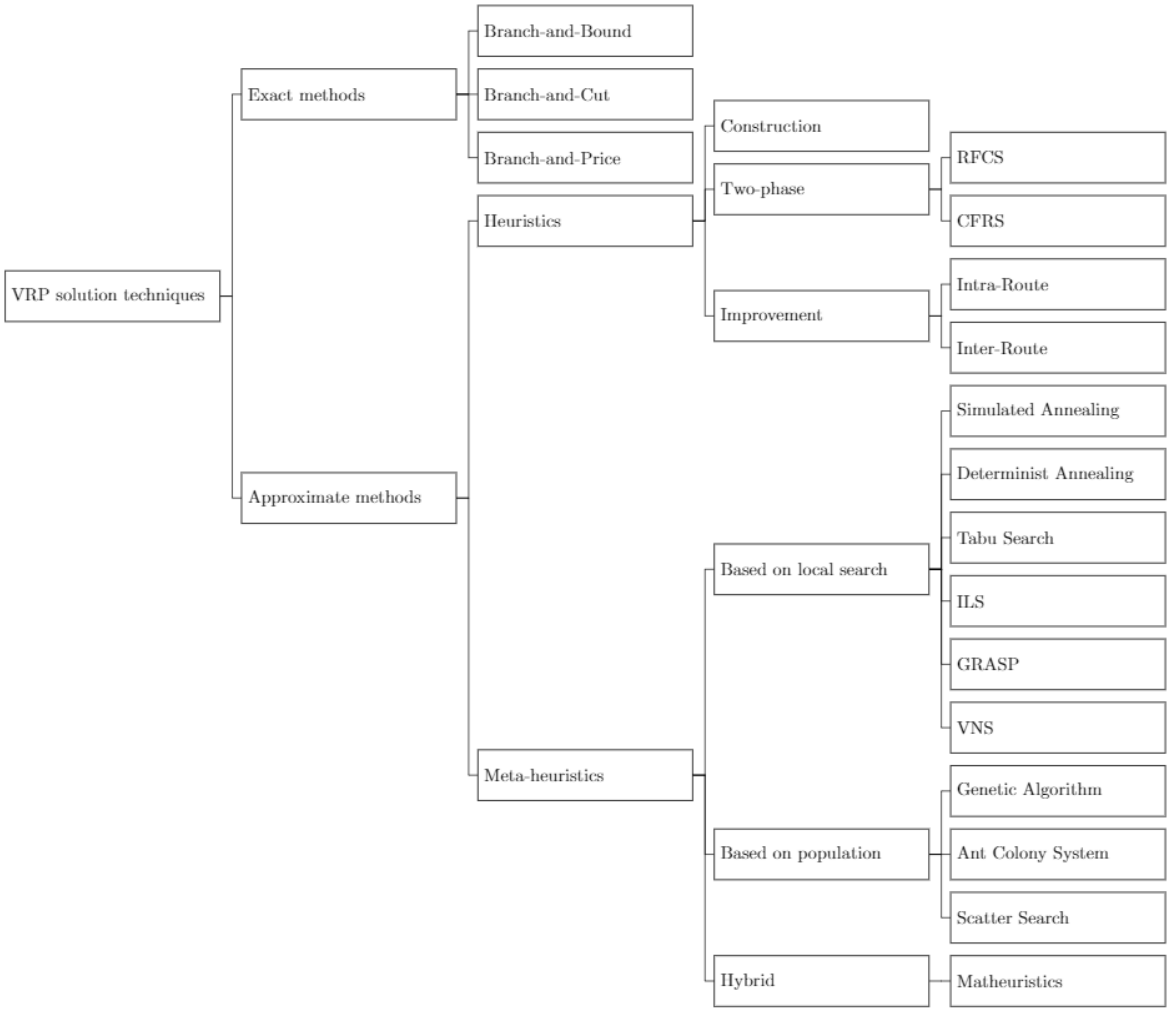


Figure 2.1 VRP solution techniques

The problem of combining a drone with a traditional delivery truck for parcel delivery was first formally defined by Murray and Chu [MC15]. They introduced two different problems : the *Flying Sidekick Traveling Salesman Problem* (FSTSP), where deliveries are performed by a single vehicle and a single drone working in tandem (the drone travels on the truck, and the drone can make deliveries on its own) and the *Parallel Drone Scheduling Traveling Salesman Problem* (PDSTSP), where multiple drones are launched from the depot to serve nearby customers, independent of the truck delivery. Since, many studies have explored different variants of truck-drone combination for parcel delivery.

In [Ott+18], the authors proposed a literature survey on optimization approaches to civil applications of drones. They classify articles according to the roles assigned to drones and vehicles in combined operations.

- **Vehicles supporting operations of drones**: drones may serve as the main performance driver in combined operations. All the deliveries are assigned to drones and vehicles just carry UAVs to their deployment location and/or serve as mobile refueling or battery recharging stations.
- **Drones supporting operations of vehicles**: in other applications, vehicles may serve as the main performance driver in combined operations and UAV operations play only a subordinate role. This variant is usualy related to applications like telecommunication or environmental protection and disaster management. Drones may provide communication to ground vehicles. For example, as communication relays, they may connect rescue vehicles to the ground control station and with each other.
- **Drones and vehicles performing independent tasks**: as another option, both drones and vehicles may work in parallel and perform independent tasks. The vehicle and drone serve different sets of customers concurrently, with the drone traveling back and forth between the depot and customers while a vehicle visits other customers.
- **Drones and vehicles as synchronized working units**: in some delivery system, vehicle and UAV operations are synchronized. One or more UAVs travel on a vehicle, which serves as a mobile depot. In other words, a UAV picks up a package from the vehicle (which continues on its route), and after delivering the package, the UAV returns to the vehicle to pick up the next package. Deliveries can be made by both UAVs and the vehicle. Thus, the simultaneous movements of the vehicle and UAVs need to be synchronized to allow for the UAVs to return to the truck at discrete locations within their allowable flight time. The vehicle, while serving delivery requests itself, effectively extends the service range of the UAVs beyond their original travel range.

Khoufi et al. [KLA19] provided a survey of the most recent papers that proposed extended variants of the Traveling Salesman Problem (TSP) and Vehicle Routing Problem (VRP) for UAVs. They provided an extensive classification through a taxonomy based on: type and number of vehicles, type (and subclass) of routing problem (TSP or VRP), application (transportation and delivery, communication, surveillance, monitoring, tracking, logistic processes, disaster management) and resolution techniques.

In this survey, we review the most recent studies on drone delivery focussing specifically on routing optimization problems where vehicles are combined with UAVs for deliveries. However we also give some references on drone-only delivery systems. Most of these articles introduced extended models of TSP or VRP and validate their designed models by considering exact algorithms and heuristics using instances from the well known TSPLIB (i.e., Traveling Salesman Problem Library) while others proposed their own test instances.

The reviewed papers will be classified in four groups and discussed in the following sections. Figure 2.2 shows an illustration of the four groups. The first group (Figure 2.2(a)) contains the coordinated use of trucks and UAVs. That is when together trucks and drones perform delivery services in a coordinated way. The second group reviews the works where drones are assisted by other vehicles to perform their services (Figure 2.2(b)). In this case, drones are the only vehicles that bring packages to final deliveries, and the other vehicles are used as a launching platform, or to help the drones reach the desired targets. The third group (Figure 2.2(c)) considers UAVs and vehicles performing delivery services independently that means no synchronization is required between them. The fourth group (Figure 2.2(d)) focuses on the works where drones work alone or with the only support of fixed infrastructure, like distribution centers or depots.

For each group, we detail the different variants of the problem and give a description of resolution methods that have been applied by each article to solve the problem.

We end up by presenting a synthetic summary of the reviewed papers in Section 2.4.

2.3.1 UAVs and vehicles as synchronized working units

In this class of problems, we consider a set of customers, each of whom must be served exactly once by either a truck or a drone. Not all customers are eligible for drone delivery, because of practical constraints, such as the limited payload or the flying range of drones. Customers that are eligible for drone delivery are consistently called *drone-eligible*. Both vehicle and drone must start from and return to the depot exactly once. A vehicle, during its trip, can launch a drone when serving a customer, the drone performs a delivery within it's flight endurance limit and returns to the truck, possibly at a different customer location (to pick up more items for delivery). The rendezvous between the truck and the drone requires coordination (synchronization) mechanisms in both time and location. The objective is to minimize the delivery completion time, that is, the time at which the vehicle and all drones are back to the depot, with the service of all customers carried out.

Figure 2.3 illustrates a truck-only tour versus truck-drone working in tandem.

The main difficulty of this problem is attributed to the synchronization between vehicle and drone. Three important decisions are made in this problem:

- Which customers will be served by a truck and which ones will be served by a drone?

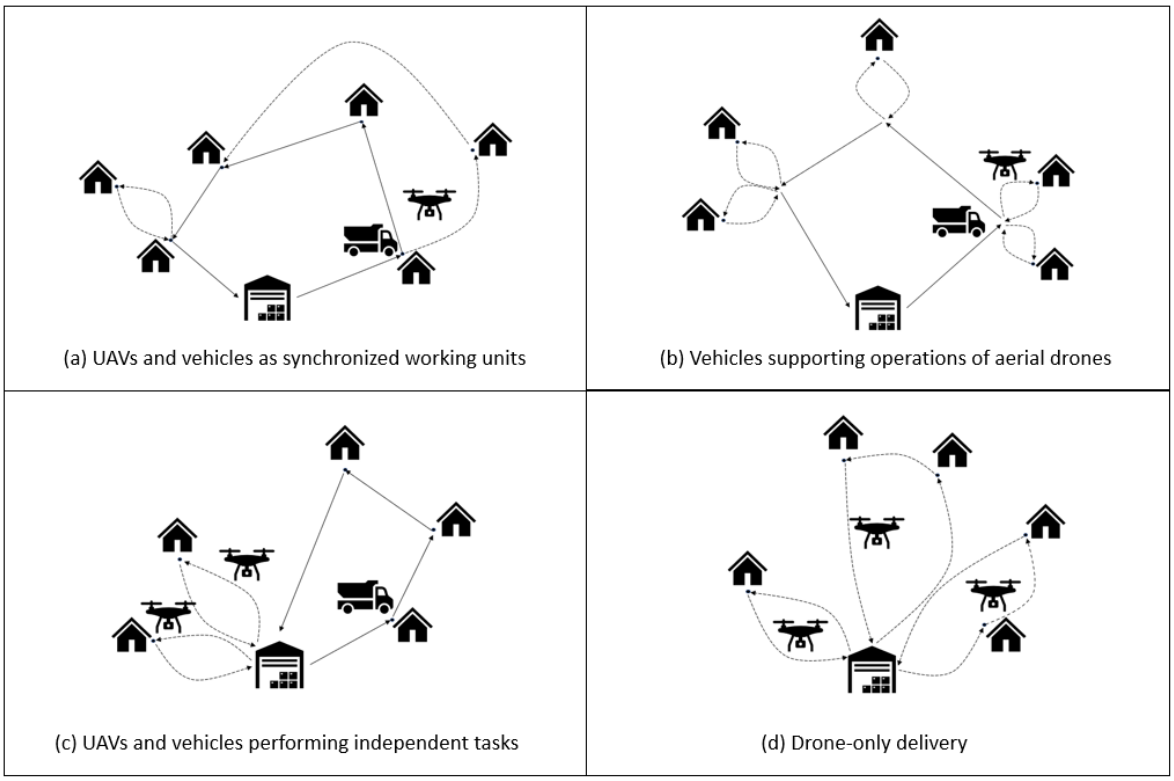


Figure 2.2 Illustrative examples of different types of drone delivery systems.
Straight arrows depict trajectories of the vehicle, bending arrows depict trajectories of the UAVs, small homes correspond to customer locations, larger facilities correspond to depots

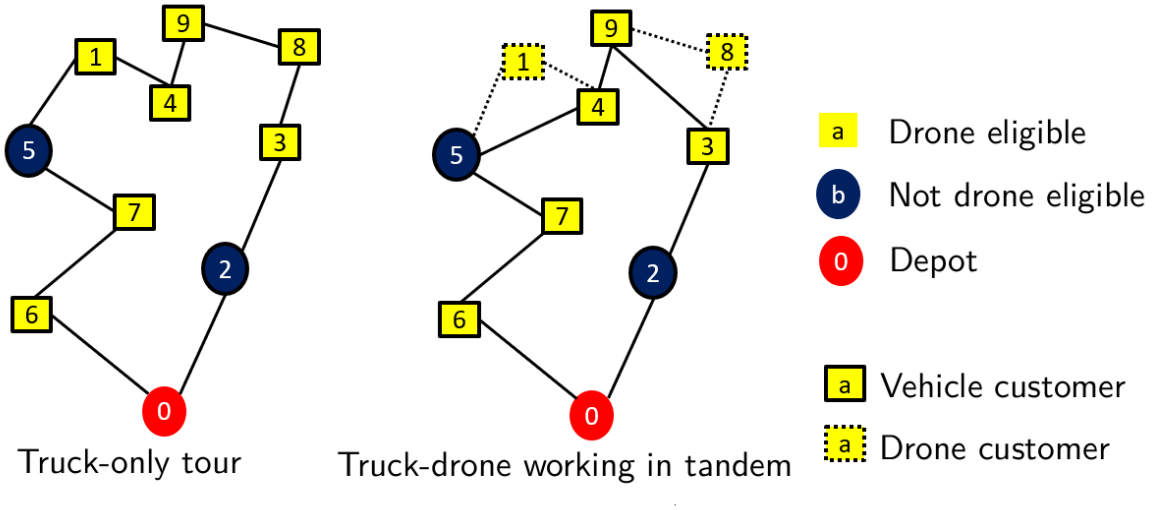


Figure 2.3 Illustration of the synchonization mechanism

- Where are the launch and the pickup locations (that means where do the drone launches from the vehicle and where does it return to the vehicle)?
- In which order will the vehicle visit the customers that are assigned to him?

2.3.1.1 Delivery with 1-truck 1-drone

The single-truck single-drone variant is adressed in several papers.

Murray and Chu [MC15] introduced the *Flying Sidekick Traveling Salesman Problem* (FSTSP) which is a generalization of the (TSP) and the vehicle routing problem (VRP). In the FSTSP, a single truck and a single drone are coupled and synchronized to perform deliveries. The drone and truck start and end at a depot, the drone travels on the truck, and the drone can make deliveries on its own. Each drone delivery is for a single stop, departing from and returning to the truck at sequential truck delivery stops. Thus, the drone cannot launch and recover at the same customer site (truck stop). The drone can only visit eligible customers, i.e., customers that the parcel does not exceed the vehicle payload capacity and respects the endurance of the vehicle to complete the trip. The objective is to minimize the total time for both truck and drone to return to the depot, which includes travel time, launch and recover time for the drone, and waiting time for the drone or truck at the stop where recovery occurs. The authors first proposed a mixed integer linear programming (MILP). However, due to the NP-Hard nature of classic TSP problems, FSTSP inherently faces the same challenge in terms of scaling to larger practical customer sets using a purely mathematical MILP approach. Thus, they formulate a route and re-assign heuristic to generate efficient results, which serves to capture decision trade-offs between drone usage and truck usage to serve each customer in order to yield overall time savings. They do this by defining a list of drone-eligible customers, generating a solution for the truck-only TSP, and then finally reassigning the route by inserting

drones and assessing time savings. This last step is iterated several times until no more savings or improvements can be achieved.

Several studies have since explored variations of the single-truck single-drone problem. For example, the FSTSP is also adressed in [Pon16]; [CS18]; [FP20]. Ponza [Pon16] proposes a simulated annealing heuristic. Julia et al. [FP20] approach is a heuristic framework where the initial solution is created from the optimal TSP solution reached by a Mixed-Integer Programming solver. Next, an implementation of the General Variable Neighborhood Search (GVNS) metaheuristic is used to obtain the delivery routes of the truck and the drone. The approach is named Hybrid General Variable Neighborhood Search (HGVNS).

In [KM17], Kundu et al. study the FSTSP by considering the effect of wind and UAV battery-power consumption. They address this issue of routing the truck and UAV in an environment with a static velocity of wind and optimized airspeed, according to different objectives, using a variant of a route and reassign heuristic (RFCS) that is common in vehicle routing problems. Experimental scenarios using different wind-velocities and number of delivery points have been simulated and the tour completion time metric is compared.

Marinelli et al. [Mar+17] extend the FSTSP by allowing the launch and rendezvous operations to be performed not only at a node but also along a route arc. In this way, the operations of a drone are not strictly related to the customers position, but it can serve a wider area along the route. The authors proposed a Greedy Randomized Adaptative Search Procedure to solve the problem. They tested the metaheuristic on benchmark instances and analysed the benefits introduced with the "en-route" approach. Dell'Amico et al. [DMN19] consider two versions of the FSTSP: one in which the drone is allowed to wait at the customers, as in the literature, and another where waiting is allowed only in flying mode. They propose three-indexed and two-indexed formulations and a set of inequalities that can be implemented in a branch-and-cut fashion which are adapted to both versions. A comparison between the two versions is provided.

Agatz et al. [ABS18] study a slightly different problem called the "Traveling Salesman Problem with Drone" (TSP-D) in which the drone may be launched and return to the same location, while this is forbidden in the FSTSP. The authors proposed an Integer Programming (IP) formulation and a route first - cluster second (RFCS) heuristic based on local search and dynamic programming. They prove worst-case approximation ratios for the heuristics and test their performance by comparing the solutions to the optimal solutions for small instances. The (TSP-D) was also studied in [Ha+15][BAS18][YO18][MF+16].

Bouman et al. [BAS18] present exact solution approaches for the TSP-D based on dynamic programming and provide an experimental comparison of these approaches. They claimed their numerical experiments show that their approach can solve larger problems than the mathematical programming approaches that have been presented in the literature thus far. In [PGW19] Poikonen et al. also proposed an exact solution approach which uses branch-and-bound, whereby each node of the branch-and-bound tree corresponds with a potential order to deliver a subset of packages. An approximate lower bound at each node is given by solving a dynamic program. Exact methods for the TSP-D are also explored in [RR19].

They propose a compact mixed-integer linear program (MILP) for several TSP-D variants that is based on timely synchronizing truck and drone flows. Furthermore, they introduce dynamic programming recursions to model several TSP-D variants and show how these dynamic programming recursions can be exploited in an exact branch-and-price approach based on a set partitioning formulation and using *ng*-route relaxation (which is one of the most efficient route relaxations in the literature proposed in [BMR11]), dual variable stabilization, and a three-level hierarchical branching.

Yurek and Ozmutlu [YO18] provide an iterative heuristic approach for solving the TSP-D that is based on a decomposition of the problem into two stages:

- (i) Determine the truck route (determination of the truck route also provides the customer assignments because the customer nodes that are not located along the truck route have to be served by the drone.)
- (ii) Determine the drone route considering the fixed truck route and drone nodes (to determine the drone route, the launch and the pickup nodes have to be determined for each drone node that is obtained in the first stage).

First, they generate all possible solutions to the Traveling Salesman Problem (TSP). They keep track of the global upper bound in their solution set and propose an iterative way of solving the drone assignment problem through a mathematical model that minimizes the waiting time of the truck at potential retrieval vertices. They compare their results to (Murray and Chu [MC15], Agatz et al. [ABS18]) and report an improvement, by being able to solve instances with 10 to 12 customer locations in shorter computation time. However, the authors believe that, due to the concept of the global upper bound, their approach is not well-suited to clustered problem instances.

Ha et al. [Ha+15] adopted the name TSP-D for the FSTSP. They proposed a cluster first - route second (CFRS) and a route first - cluster second (RFCS) heuristics. In [Ha+18], the same authors investigated a different objective function: the minimization of the total transportation cost for both the vehicle and the drone. They coined this problem as min-cost TSP-D. They proposed a MILP model and two heuristics: one based on a Greedy Randomized Adaptive Search Procedure (GRASP) and another one called TSP-LS (Traveling Salesman Problem - Local Search) adapted from the approach proposed by Murray and Chu [MC15] for FSTSP in which an optimal TSP solution is converted to a feasible TSP-D solution by local searches.

Liu et al. [Liu+20] study the two-echelon routing problem for truck and drone (2E-RP-T&D) where a truck and a drone are used to cooperatively complete the deliveries of all parcels. The truck starts from the depot, taking all the parcels and the drone. The truck can only travel on the road network and deliver the parcel to its customer. When the truck delivers the parcel, the drone can also carry some small parcels and take off from the truck to complete some deliver tasks simultaneously (the drone can carry several parcels). The drone can only take off / land on the truck when it stops at customer nodes (or depot),

where the drone can be charged or change the battery. After completing the delivery of all parcels, the truck must return to the depot with the drone. The objective is to minimize the overall traveling cost of the truck and drone. A two-stage route-based modelling approach is proposed to optimize both the truck's main route and the drone's adjoint flying routes. A hybrid heuristic integrating nearest neighbor and cost saving strategies is developed to quickly construct a feasible solution. A simulated annealing algorithm is applied to improve the quality of the solution, where a Tabu list is employed to improve the search efficiency.

2.3.1.2 Delivery with 1-truck m-drone

In the case of single-truck multi-UAV problems, we can mention the work of Ferrandez et al. [MF+16]. The authors investigate the notion of the reduced overall delivery time and energy for a truck-drone network by comparing the in-tandem system with a stand-alone system (truck-only delivery). They assume that the truck follows the route generated by solving a TSP and that from each truck stop one or several drones can be launched. Each truck stop is a hub for drone deliveries and drone flight range constraints are not considered here. They use K-means clustering algorithm to find truck stops (drones launch locations) and a genetic algorithm to determine the truck TSP tour.

In [CL18] Chang et al. study a delivery problem in which a truck carrying delivery items departs from a depot and travels the shifted centers of clusters bundled by several delivery locations, and finally returns to the depot. In each cluster, several drones leave the truck at the same time and return to the truck after visiting delivery locations. In these cases, the total delivery time is determined by the longest flight time among drones in each cluster. The proposed model aims at minimizing the total delivery time by having a truck moving among the centers of clusters as drones operate in each cluster. They proposed a research model with three steps :

- (i) Clustering delivery locations (locations–nearby delivery locations within the drone's service range are clustered using the K-means clustering technique);
- (ii) Routing the centers of clusters (truck's delivery route among the centers of clusters is set up to minimize its traveling time using TSP);
- (iii) Finding shift-weights that move the centers of clusters to make for wider drone-delivery areas along shorter truck-route (a nonlinear programming model is proposed).

Yoon [Yoo18] considers a single truck that may launch multiple UAVs, with the UAVs returning to the truck at a different location conversely to problems studied in [MF+16] and [CL18] where the drones have to return to the truck before the truck departs for its next destination. A MILP formulation is provided, which is tested on instances with up to 10 customers.

Tu et al. [TDD18] investigate an extension of the TSP-D problem in which a truck travels with m ($m > 1$) drones (called TSP-mD) instead of one drone in TSP-D. They adapt the

GRASP proposed by Ha et al. [Ha+18] and propose an Adaptive Large Neighborhood Search (ALNS) heuristic to the resolution of this problem.

Murray et al. [MR20] extend the original FSTSP which they called mFSTSP. The mFSTSP considers an arbitrary number of heterogeneous UAVs that may be deployed from the depot or from the delivery truck. These UAVs may have different travel speeds, payload capacities, service times, and flight endurance limitations. The new problem also features a more realistic treatment of the operating conditions, to better reflect the realities associated with the complex nature of this coordinated vehicle routing problem. In particular, the mFSTSP considers a thorough treatment of UAV flight endurance, a constraint that is of significant importance in drone-based applications. Rather than simply assuming that each UAV has an endurance specified in total flight time, they acknowledge that endurance is a function of payload weight, travel distance while carrying a parcel, and travel distance without the burden of the parcel. They assume that because the delivery truck is typically too small to safely accommodate multiple drones landing or launching simultaneously, the mFSTSP queues the aircraft in both the launch and retrieval phases. To solve this problem the authors proposed a MILP and a three-phased heuristic:

- In Phase I, customers are partitioned into two sets : those that will be served via truck and those served via UAVs;
- In Phase II, sorties for the UAV-assigned customers (as determined in Phase I) are generated. These sorties define the launch and recovery locations associated with each UAV customer, as well as the UAV assigned to each sortie;
- In Phase III, an MILP is solved to determine the exact timing of the launch, recovery, and service activities for the truck and the UAVs. Phase III also determines the queueing sequences for the UAVs.

2.3.1.3 Delivery with k-truck m-drone

The use of multiple trucks and multiple UAVs is considered by Wang et al. [WPG17]. They study an extension of the FSTSP called the Vehicle Routing Problem with Drones (VRPD) in which they consider a homogeneous fleet of trucks, each one of them carrying several identical drones. A drone launched from a truck must be picked up by the same truck at the same or at a different location. The objective is to minimize the completion time. They provide a worst-case analyse and obtain theoretical bounds on the benefits achieved using drones. This problem is also adressed in [CSZ17] where the authors use continuous approximation modeling techniques to derive general insights. Zheng et al. [WS19] addressed one distinctive feature of the VRPD which is that a drone may travel with a truck, take off from its stop to serve customers, and land at a service hub to travel with another truck as long as the flying range and loading capacity limitations are satisfied. They proposed a mixed integer programming model, and developed a branch-and-price algorithm. Daknama and Kraus [DK17], and Schermer et al. [SMW18] explored neighborhood search based heuristics to solve the VRPD.

Pugliese and Guerriero [PG17] extend the VRPD by considering time window constraints. The problem is modeled as a variant of the Vehicle Routing Problem with Time Windows (VRPTW) where each vehicle is equipped with drones (VDRPTW). A Mixed-integer linear programming formulation for the VDRPTW is proposed. The numerical results, collected on instances generated to be very close to reality made them conclude that the use of drones is not economically convenient in the classical terms. However, when considering negative externalities related to the use of classical vehicles and quality of service requirements, the benefit of using drones becomes relevant.

Dayarian et al. [DSC20] consider a home delivery system in which a fleet of drones and a fleet of vehicles collaboratively perform home deliveries of online orders from a fulfillment center. Here they are considering a different use of drones. A drone can carry several items. The delivery vehicles are regularly resupplied by drones. Resupply can take place whenever a delivery vehicle is stationary and a drone can land on the vehicle's roof. The introduced problem is named Vehicle Routing Problem with Drone Resupply (VRPDR).They present several policy function approximations to analyze different routing and assignment strategies.

Using trucks as mobile landing and take-off platform for UAVs is also considered in [Boy+18]. Given a fixed sequence of stops constituting a truck route and a set of customers to be supplied, they aim at a drone schedule (i.e., a set of trips each defining a drone's take-off and landing stop and the customer serviced), such that all customers are supplied and the total duration of the delivery tour is minimized. They differentiate whether multiple drones or just a single one are placed on a truck and whether or not take-off and landing stops have to be identical. They provide an analysis of computational complexity for each resulting subproblem and introduce efficient mixed-integer programs.

2.3.2 Vehicles supporting operations of aerial drones

In this class of truck/UAV delivery problems, drones are the single vehicles reaching the recipients, and other vehicles, such as trucks, are used to bring the drones closer to the recipients, or as recharge stations. An illustration of this class of problems is shown in figure 2.4.

In [PG20] Poikonen and Golden introduced the Multi-Visit Drone Routing Problem (MVDRP) which considers a tandem between a truck and drone. The drone can launch from the truck with one or more packages to deliver to customers. The drone may return to the truck to swap/recharge batteries, pick up a new set of packages, and launch again to customer locations. A part from the possibility for a drone to carry multiple heterogeneous packages, the model also allows the specification of a drone energy drain function that takes into account each package weight. The goal of MVDRP is to minimize completion time. To solve the problem, the author proposes a flexible solution heuristic and the solution methodology is then extended to a truck and multi-drone model, titled k-MVDRP. The solution is titled *Route, Transform, Shortest Path* (RTS).

Mathew et al. [MSW15] studied a similar problem called the Heterogeneous Delivery

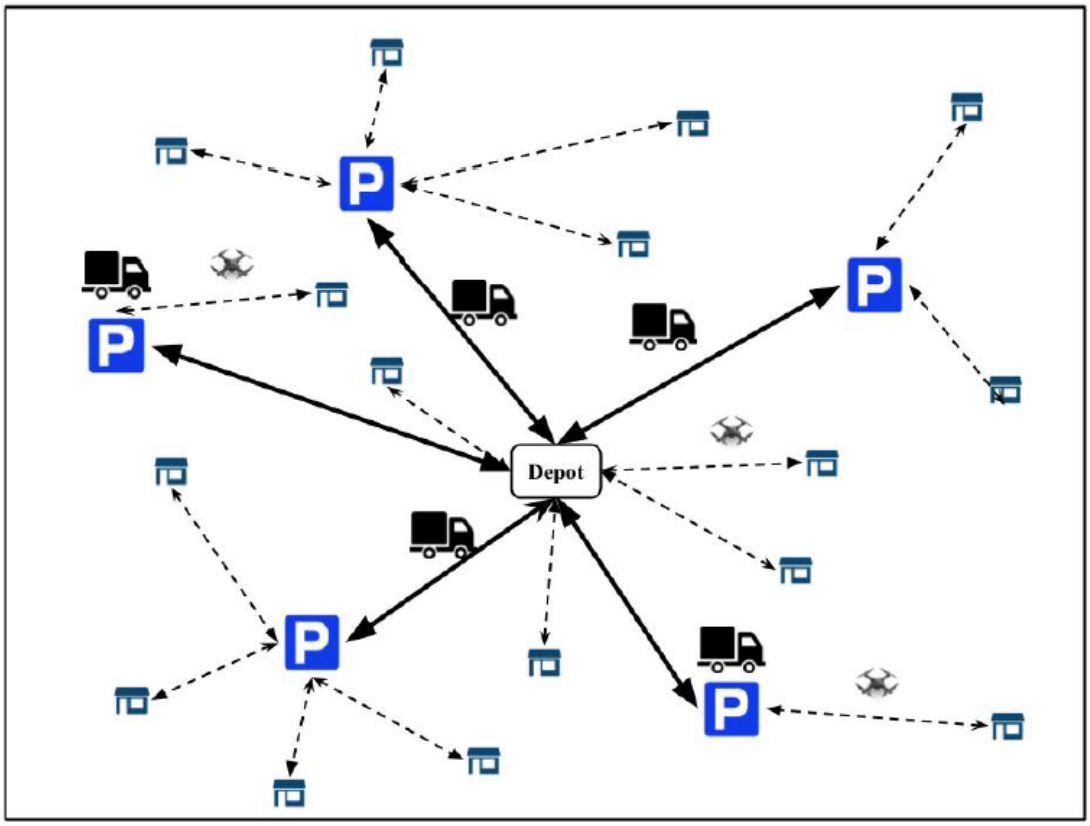


Figure 2.4 Illustration of truck-UAV combined but only UAVs deliveries (Source [DYKB19])

Problem (HDP). This problem considers a team of one truck and one drone in which the truck travels along a street network and the drone can be deployed from the truck to perform deliveries. All the deliveries are assigned to the drone, and the truck is only used to carry the drone. The truck can wait at a street vertex for the drone to come back or go to the next vertex to rendezvous with the drone. The problem is formulated as an optimal path planning problem on a graph and the goal is to find the shortest cooperative route enabling the drone to deliver items at all requested locations. They prove the HDP is NP-hard and propose an efficient reduction to the Generalized Traveling Salesman Problem.

Othman et al. [Oth+17] studied a scenario in which a delivery truck is dispatched carrying a shipment of parcels to be delivered to customers, and it is required to end its route at a predetermined location, which is not necessarily the same as the starting location. A drone is charged with making the last-stretch delivery of a parcel from the truck to a customer's doorstep. The objective is to deliver all parcels to end customers in the minimum amount of time. They introduce two problems: the No-Wait Transit Last-Stretch Delivery Problem (NW-TLSDP), and the Transit Last-Stretch Delivery Problem (TLSDP). In the first setting, while the drone is making a delivery of a parcel, the truck is not allowed to wait for the drone to return at the last rendezvous point. In particular, the truck can only intercept the drone at any rendezvous point at most once. In the second setting, the truck is allowed to wait for

the drone to return at the last rendezvous point, or re-visit some rendezvous point multiple times. They modeled this problem as a problem of finding a special type of a path in a graph of a special structure, and proposed a polynomial-time approximation algorithm for each of the problem settings.

Dukkanci [DYKB19] introduced the Drone Delivery Problem (DDP) in which drones are used to make deliveries to a number of customers and the drones themselves are transported by traditional vehicles that act as launch points. The DDP consists of choosing launch points to use from a candidate set of sites, assigning customers to launch points, and determining the speed at which drones travel between customers and launch points, subject to all deliveries made within a given amount of time and the range of a drone. The problem minimizes the total variable cost that comprises the operational cost of trucks and the cost of energy consumption arising from the use of drones. A nonlinear model for the DDP is provided. The model is reformulated using second order cone programming and solved using an off-the-shelf optimization software (IBM ILOG CPLEX Optimization Studio version 12.6.1.0).

In [LLS17], Luo et al. addressed the same kind of problem they called two-echelon ground vehicle and UAV cooperated routing problem (2E-GU-RP) which considers a set of targets, each of whom must be visited (or served) exactly once by the UAV. Deliveries are only made by the UAV which is able to pick up several paquets and stop at several targets before going back to the ground vehicle. They provide an integer programming model and two heuristics. Vosooghi et al. [Vos+19] consider a two-echelon urban delivery problem using small autonomous vehicles (SAV)s for 2nd-level route delivery. A Large vehicle (LV) is used to transport the SAVs on the 1st-level route and drops off or pick them up in the rendezvous nodes, while the SAV handles customer service on the 2nd-level route. LVs are used only for carrying SAVs, and so no direct shipping from LVs to customers is allowed. An SAV is allowed to visit multiple customers during a dispatch rather than only visit one customer. Each pick-up or drop-off (rendezvous) node can only be visited at most once by the same vehicle. Each customer node must be visited by just one SAV exactly once. In addition, customer nodes and depot have their time windows based on real demand in city logistics. The objective is to minimize the number of LVs and the total transportation cost of 1st-level and 2nd-level routes. This problem applies for delivering parcels or other small commodities to pedestrianized areas such as campuses or residential cluster (where LVs are often banned). They indroduced a MILP for the problem and proposed construction heuristics and a hybrid metaheuristic approach to solve larger instances.

2.3.3 UAVs and vehicles performing independent tasks

In this class of problems, we consider a set of customers, each of whom must be served exactly once by either a truck or a drone. A truck delivers customers with a single tour starting from the depot, visiting a subset of customers and returning back to the depot. Drones operate back and forth trips between the depot and the customers, delivering a single customer in each trip. It is assumed that the depot is located in close proximity to customers. Not all customers are eligible for drone delivery, because of practical constraints such as the limited

payload or the flying range of drones. The objective is to minimize the delivery completion time, i.e., the time at which all vehicles and all drones are back to the depot, with the service of all customers carried out. Figure 2.5 illustrates truck-drone performing independent tasks.

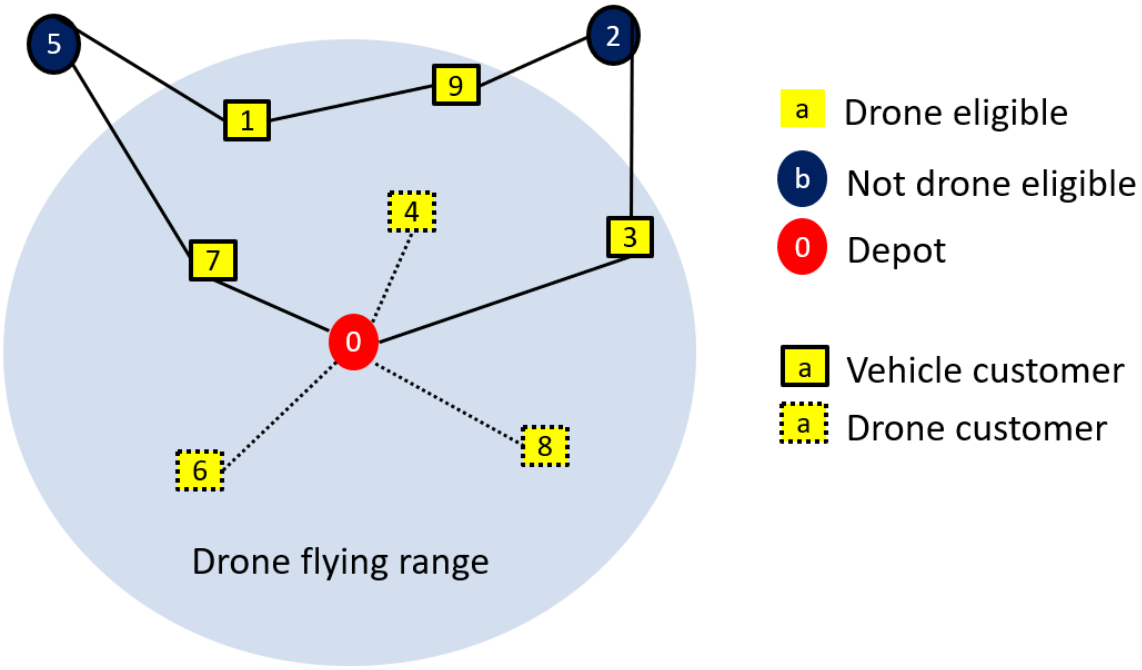


Figure 2.5 Truck-drone performing independent tasks (no synchronization)

One example is a study by Murray and Chu [MC15], who set up a delivery problem with customers located in close proximity to a warehouse. The problem is called the Parallel Drone Scheduling TSP (PDSTSP). In PDSTSP, a single depot exists, from which a single delivery truck and a fleet of one or more identical UAVs must depart and return. Packages may be delivered either by drone, which transports a maximum of one package per sortie, or by truck, which can transport several packages but moves at a slower speed. Drones cannot land on the truck and must return to the warehouse to pick up another package. Unlike the FSTSP, there is no synchronization between a UAV and a truck in the PDSTSP. The objective of PDSTSP is to minimize the latest time that a vehicle returns to the depot, such that each customer is served exactly once. They propose the first MILP formulation for the problem and a simple greedy heuristic. The principle of the heuristic is to partition the set of customers into two subsets (a *vehicle set* and a *drone set*). Then, a TSP tour is computed for the customers assigned to the vehicle and a Parallel Machine Scheduling (PMS) problem is solved to assign customer requests (jobs) to drones (machines). At the end, an improvement step is performed to reassign customers either to the drone set or to the vehicle set in order to better balance vehicle and drones activities. Several methods are used to solve the TSP. Among them a MILP solved to optimality and the savings heuristic. To solve the PMS problem they use a MILP solved to optimality and the Longuest Processing Time heuristic (LPT). Dell'Amico at al. [DMN20] propose a MILP and some matheuristic approaches for the same problem,

all relying on the MILP model with the aim to cover a spectrum of methods able to provide different trade-offs between the quality of the final solution and the computation time required to produce it.

A problem where multiple drones, multiple trucks and multiple depots are considered is presented in [Ham18]. The author studies an extended version of the PDSTSP by considering two different types of drone tasks: drop-off and pickup. After a drone finishes the delivery task, it can either fly back to depot to deliver the next parcels or fly directly to another customer to pick up a returned parcel. Customer can order multiple products with different shipping priorities (or time-window). Recharging time of the drone can be negated, e.g. by swapping batteries. A constraint programming approach is proposed and tested with problem instances of m-truck, m-drone, m-depot, and hundred-customer distributed across an 8-mile square region.

Ulmer and Thomas [UT18] study a dynamic variant of the PDSTSP. They called their problem Same-Day Delivery Routing Problem with Heterogeneous Fleets (SDDPHF). The SDDPHF considers two fleets of vehicles and drones that deliver goods from the depot to customers who dynamically request services. When a new customer request arrives, a dispatcher has to decide whether or not to accept the customer for the same-day delivery and determine the according assignment and routing decisions. The SDDPHF takes into account a time window for each request, the loading time for both vehicles and drones, the time to drop off a package from a vehicle or a drone as well as the recharging or battery swap time for drones. The objective is to maximize the expected number of customers served the same day. To solve the problem, the authors proposed an adaptive dynamic programming approach called parametric Policy Function Approximation (PFA).

2.3.4 Drone-only delivery

In this group of problems, all the deliveries are performed by UAVs rather than cooperative delivery by drones and trucks. A set of drones can serve in parallel the customers from a depot but there is no truck. Drones can serve multiple customers, have a capacity and a maximum operation time, making the treated problem a generalization of the vehicle routing problem. Planning the routes for drones is challenging due to multiple operational characteristics, including multi-trip operations, recharge planning, as well as calculating energy consumption. For this group of problems which may seem simpler, the characteristics of drones are taken into account in more depth.

2.3.4.1 Delivery with 1-UAV

Sundar and Rathinam [SR14] study a single UAV routing problem where multiple depots are available for refueling the drone. The objective is to find a path for the UAV such that each target is visited at least once, the fuel constraint is never violated along the path for the UAV, and the total fuel required by the UAV is a minimum. They developped an approximation

algorithm for the problem, and propose fast construction and improvement heuristics.

The Travelling Salesman Problem (TSP) for UAV path planning is addressed in [CYL17]. In this problem, path planning algorithms based on TSP are embedded in UAVs. The authors establish a multi-objective multi-constrained combinatorial optimization model-TSP. In the model, the UAV is considered as the traveling salesman, and the mission target is regarded as the traveling city. To solve the model, they propose two parallel optimization algorithms. One is the Improved Genetic Algorithm (IGA), and the other is the particle-swarm-optimization-based ant colony optimization algorithm (PSO-ACO).

2.3.4.2 Delivery with m-UAV

Choi and Schonfeld [CS17] study an automated drone delivery system with multiple UAVs available. The study assumes that a drone can lift multiple packages within its maximum payload and serve customers in a service area of given radius. They use the relationship among battery capacity, payload, and flight range to optimize the drone fleet size for a service area, by minimizing the total costs of the delivery system. To study the sensitivity of the system outputs they explored four parameters: working period, drone operating speed, demand density of service area and battery capacity.

In [Tro+18], Troudi et al. also discuss the issue of dimensioning the size of the drone fleet, the stock of batteries to dispose of and the strategy of battery charging in a drone delivery system deployed in an urban perimeter. They provide an analytical model expressing the proposed problem of the optimal drones delivery mission taking into account the issues of autonomy and energy consumption related to the drone's technical specification. The objective consists of minimizing simultaneously the total distance resulting from served parcels, the total used drones and the total batteries used during these missions.

Dorling et al. [Dor+17] introduced a problem called Drone Delivery Problem (DDP). They considered two variants of the DDP: the Minimum Cost Drone Delivery Problem (MC-DDP) that aims at minimizing the total delivery cost and the Minimum Time Drone Delivery Problem (MT-DDP) that aims at minimizing the total delivery time. In these problems, deliveries are only performed by drones, and the energy consumption is explicitly considered for the drones, taking into account their payload. The authors established the relationship with the Multi-Trip Vehicle Routing Problem (MTVRP). They proposed a MILP formulation and a Simulated Annealing (SA) metaheuristic.

Cheng et al [CAR18] study the multi-trip drone routing problem (MTDRP) where a fleet of homogeneous drones based at a depot is available to deliver parcels to customers. The problem consists in designing a set of drone routes such that each route starts and ends at the depot, every customer is visited exactly once, drone weight capacity constraint, battery energy constraint and customers time windows must be respected. They proposed two modeling schemes and developped exact algorithms based on Branch-and-cut for their formulations.

In [SLC16] San et al. describe the implementation steps used to assign a swarm of unmanned aerial vehicles tasked to effectively deliver items to target locations. They consider several constraints, including drone's payload, flight range, and the volume of each item. A Genetic algorithm is used.

Two related problems, where a set of drones can serve in parallel the customers from a depot are also proposed in [Gha+18] and [TLK18].

2.4 Synthesis

Tables 2.1 to 2.4 report a summary of the main papers dealing with UAVs involved in the delivery process. The empty lines in columns labeled "*Problem*" correspond to papers for which the authors did not give a particular name to the problem.

Table 2.1 UAVs and vehicles as synchronized working units

Reference	**Problem**	**Vehicles**	**Drones**	**Drone capacity**	**Solution method**
[MC15]	FSTSP	1	1	1	MILP,Heuristics
[Pon16]	FSTSP	1	1	1	Simulated annealing
[CS18]	FSTSP	1	1	1	Mathematical analysis
[FP20]	FSTSP	1	1	1	HGVNS
[Mar+17]	FSTSP en-route	1	1	1	GRASP
[DMN19]	FSTSP	1	1	1	MILP, Branch-and-cut
[Ha+15]	TSP-D	1	1	1	CFRS, RFCS
[ABS18]	TSP-D	1	1	1	IP, RFCS
[BAS18]	TSP-D	1	1	1	Dynamic programming
[YO18]	TSP-D	1	1	1	Iterative algorithm combining TSP and MILP
[Ha+18]	min-cost TSP-D	1	1	1	MILP, GRASP, TSP-LS
[PGW19]	TSP-D	1	1	1	branch-and-bound
[RR19]	TSP-D	1	1	1	MILP, Dynamic Programming Recursions
[Liu+20]	2E-RP-T&D	1	1	*multiple*	Two-stage route-based modelling (Nearest neighbor, SA)
[MF+16]	TSP-D	1	m	1	K-means and GA

Continued on next page

Table 2.1 – continued from previous page

Reference	Problem	Vehicles	Drones	Drone capacity	Solution method
[CL18]		1	m		K-means clustering and non-linear programming model
[Yoo18]	TSP-MD	1	m	1	K-means clustering and non-linear programming model
[TDD18]	TSP-MD	1	m	1	GRASP,ALNS
[MR20]	mFSTSP	1	m	1	Worst-case analysis
[WPG17]	VRPD	k	m	1	Worst-case analysis
[DSC20]	VRPDR	k	m	multiple	Route generation, order release heuristic
[CSZ17]		k	m	1	Continuous approximation models
[PG17]	VDRPTW	k	m	1	MILP, GRASP
[WS19]	VRPD	k	m	1	MILP, branch-and-price

Table 2.2 Vehicles supporting operations of aerial drones

Reference	Problem	Vehicles	Drones	Drone capacity	Solution method
[MSW15]	HDP	1	1	1	Reduce to TSP then use TSP solver
[Oth+17]	NW-TLSDP	1	1	1	Aproximation algorithm
	TLSDP	1	1	1	
[LLS17]	2E-GU-RP	1	1	multiple	TSP route and split, route and re-assign
[Vos+19]	LV-SAV	k	m	multiple	MILP, hybrid metaheuristic
[DYKB19]	DDP	k	m	multiple	Second order cone programming
[PG20]	MVDRP	1	1	multiple	RTS Heuristic
	k-MVDRP	1	m	multiple	

Table 2.3 UAVs and vehicles performing independent tasks

Reference	Problem	Vehicles	Drones	Drone capacity	Solution method
[MC15]	PDSTSP	1	m	1	MILP, Heuristics
[UT18]	SDDPHF	k	m	1	Adaptive dynamic programming (PFA)
[Ham18]	PDSTSP drop&pickup	k	m	multiple	Constraint programming
[DMN20]	PDSTSP	1	m	1	MILP, matheuristic
This work [MS+18]	**PDSTSP**	**1**	$\boldsymbol{m}$	**1**	**MILP, Iterative two-step heuristic**
This work	**PDSMTSP**	$\boldsymbol{k}$	$\boldsymbol{m}$	**1**	**MILP, Hybrid metaheuristic (ILS + dynamic programming)**

Table 2.4 Drone-only delivery

Reference	Problem	Vehicles	Drones	Drone capacity	Solution method
[SR14]		0	1	multiple	Route construction and improvement heuristics
[SLC16]		0	m	1	GA
[Dor+17]	DDP	0	m	multiple	MILP, SA
[CS17]	DDP	0	m	multiple	Mathematical analysis
[CAR18]	MTDRP	0	1	multiple	MILP, Exact algorithms

2.5 Conclusion

Research and practice state of the art on drone delivery have recently made huge advances. Researchers have extended the TSP and VRP problems to the UAV context in order to meet the requirements of new emerging UAV's applications.

Despite the fact that a significant amount of literature exists for the TSP and VRP, the introduction of drones into these problems have inspired the development of a dedicated branch of literature. Although the approach is fundamentally close to classic VRP, drone-based routing problems must address additional complexities since drones have a limited flight endurance and carrying capacity. Features like multiple trips to the depot, and battery weight and payload weight effect on energy consumption may need to be considered. In this chapter

we identified the preexisting survey literature on related topics and then focused on a recent selection of articles on drone-based routing problems.

The strongest asset of UAVs is their very competitive delivery time in a point-to-point service, especially when the ground deliveries are affected by congestion. However, only in very special situations will drones be able to replace traditional urban delivery methods, but they emerge as another option to complement exisiting delivery networks.

The most studied problem in research is the coordinated use of drones and trucks, where drones are mainly used to provide a higher delivery capacity. On the other hand, the use of drones to accelerate the supply chain from the warehouses to recipient has not been deeply discussed in the research context. Thus, this scenario needed to be explored.

The following chapters of this thesis focus on drone-base delivery systems where vehicle(s) and UAVs are used in parallel to deliver parcels to customers. Vehicle(s) and drones serve different sets of customers concurrently, with a drone traveling back and forth between the depot and customers while a vehicle visits other customers.

Chapter 3

Exact solution of parallel drone scheduling problems

Contents

3.1 Introduction

In this chaper, we focus on parallel drone scheduling problems, introduced by Murray and Chu [MC15], which model a drone delivery system with no synchronization between vehicles and drones. In these problems, deliveries are split between one or several vehicles and one or several drones. Vehicles perform a classical delivery tour from the depot, while the drones are constrained to perform back and forth trips. The objective is to minimize the overall completion time. We chose to focus on this delivery system because we found it more relevant on a short term than sysnchronized systems in view of actual delivery practices, but also because it has drawn less attention in the literature. We study two problems. In the basic version, called PDSTSP by Murray and Chu [MC15], deliveries are split between a single vehicle and a fleet of identical drones. In the extended version, that we rename into Parallel

Drone Scheduling Multiple Traveling Salesman Problem (PDSMTSP), deliveries are split between a fleet of vehicles and a fleet of drones. In this chapter we propose MILP formulations for both problems and develop straightforward *Branch-and-Cut* algorithms.

This chapter is a preliminary work for the following chapters (Chapter 4 and Chapter 5) which propose heuristic solution approaches for the same problems. The objective is to try to find optimal solutions or feasible solutions to be able to compare them with the results of the heuristics. The chapter is organized as follows : in Section 3.2, the components of a generic Branch-and-Cut algorithm are described and illustrated on the traveling salesman problem (TSP). In Section 3.3, we provide problem statement and a MILP formulation for both the PDSTSP and the PDSMTSP. Section 3.4 presents a Branch-and-Cut procedure for both problems. Experiments carried out on benchmark instances from the literature and new instances generated from the TSPLIB and CVRPLIB libraries are presented and results are discussed in Section 3.5. We end the chapter with a conclusion.

3.2 Branch-and-Cut method: application to the asymmetric TSP

Branch-and-Cut is a combinatorial optimization method for solving integer linear optimization problems. A Branch-and-Cut algorithm implements a combination of a cutting plane method with the widely known Branch-and-Bound algorithm to solve any combinatorial optimization problem. This procedure works by solving a sequence of linear programming relaxations of the integer programming problem. Cutting plane methods improve the relaxation of the problem to more closely approximate the integer programming problem, and Branch-and-Bound algorithms proceed by a sophisticated divide and conquer approach to solve problems. The basic idea is based on the identification of the violated valid inequalities of the integer linear program. Thus, at each step of the algorithm, violated inequalities, which are identified by solving the separation problems, are added to the formulation and the corresponding linear program is resolved.

For a better understanding of how the procedure works, we take the asymmetric traveling salesman problem as an example.

3.2.1 The Traveling Salesman Problem

3.2.1.1 Problem description

The Traveling Salesman Problem (TSP) is the problem of finding a least-cost sequence in which to visit a set of cities, starting and ending at the same city, and in such a way that each city is visited exactly once.

Let $G = (V, E)$ be a directed or undirected graph with set of vertices V and set of edges

$E = \{(i,j) \mid i,j \in V\}$. Vertices correspond to cities and edges correspond to paths between cities. Let $d_{ij}(i \neq j\ ,\ (i,j) \in E)$ be the distance covered when traveling from city i to city j. Let $\mathbb{H}$ be the set of all Hamiltonian cycles (a cycle that visits each vertex exactly once) in G. The traveling salesman problem is to find the tour $h \in \mathbb{H}$ such that the sum of the distances $d_{ij}(i \neq j; (i,j) \in h)$ in the tour is minimized. Minimizing the sum of the distances for Hamiltonian cycle is equivalent to identifying the shortest path in which each city is visited only once. The TSP can be divided into two classes depending on the nature of the distance matrix:

- *Symmetric traveling salesman problem (sTSP)*: G is undirected and the distance between cities is the same in both directions $(\forall\ (i,j) \in E : d_{ij} = d_{ji})$

- *Asymmetric traveling salesman problem (aTSP)*: G is directed and distances are not always the same in both directions $(\exists\ (i,j) \in E : d_{ij} \neq d_{ji})$

This problem is without doubt one of the oldest combinatorial optimization problems and certainly one of the most studied due in part to its wide applicability in practice. A key issue has been the question of whether there exists a polynomial-time algorithm for solving the problem. Researchers have for years been attempting without success to find polynomial-time algorithms for this problem. The TSP is classified as a NP-hard problem [Kar72][PC77].

3.2.1.2 Mathematical model

The TSP can be formulated as an integer programming problem.

We introduce decision variables x_{ij} for each $(i,j) \in E$ such that

$$x_{ij} = \begin{cases} 1 & if\ city\ j\ is\ visited\ immediatly\ after\ city\ i \\ 0 & otherwise. \end{cases}$$

The integer linear programming formulation for an Asymmetric TSP is given by:

$$\text{minimize} \sum_{(i,j) \in E} d_{ij} x_{ij} \tag{3.1}$$

subject to

$$\sum_j x_{ij} = 1 \qquad i \in V \tag{3.2}$$

$$\sum_i x_{ij} = 1 \qquad j \in V \tag{3.3}$$

$$\sum_i \sum_j x_{ij} \leq |S| - 1 \qquad S \subset V, 2 \leq |S| \leq |V| - 1 \tag{3.4}$$

$$x_{ij} \in \{0, 1\} \qquad (i, j) \in E \tag{3.5}$$

Objective (3.1) is to minimize the total travelled distance. To ensure that the result is a valid tour, several contraints must be satisfied. Contraints (3.2) are called *go-to contraints*. They state that after visiting a city i, the salesman must visit exactly one city next. Contraints (3.3) are *come-from constraints*. They express that when visiting a city, the salesman must have come from exactly one city. Subtour elimination constraints (SECs) are provided by (3.4). These constraints ensure that a tour is fully connected, i.e. without subtours. They require that for each proper (nonempty and with more than 1 city) subset S of V, the number of edges between nodes in S must be at most $|S| - 1$.

If the set of cities V is of size n, the number of binary variables becomes $n(n-1)$, and the problem has an exponential number of subtour elimination constraints (SECs): $O(2^n)$.

3.2.2 Branch-and-Cut for the TSP

As mentionned before, the Branch-and-Cut algorithm is an improved version of the Branch-and-Bound algorithm. It is an hybrid algorithm which combines the Branch-and-Bound decision tree with cutting planes.

The cutting plane method was introduced by Dantzig et al. (1954). The method solves the linear programming relaxation of a problem iteratively, and adds cuts after each iteration. In the relaxed version of the TSP, the integrality constraints (that means not allowing variables to be fractional) and the exponentially many subtour elimination constraints are removed. With each cut, the feasible region of the linear programming relaxation is shrunk without deleting possible feasible tours. An inequality that cuts the solution space of the linear programming relaxation, but does not cut any feasible solutions to the original integer programming problem, is called a *valid inequality*, or *facet-defining*. Identifying these constraints is called the *separation* problem. The procedure of solving the linear programming relaxation and searching for valid cuts is repeated until a feasible solution for the original problem is obtained.

In the Branch-and-Cut algorithm, the first step is to initialize a linear programming relaxation of the original problem. Initially, a cutting plane procedure is used until no more valid inequalities can be found. The best solution for the original problem and the relaxed problem is stored. Then, the first branching step is taken on a fractional variable, which

means this fractional variable is restricted to be either 0 or 1. This yields two new nodes in the so-called Branch-and-Cut tree. In every node, the new linear programming relaxation of the problem is solved. If the solution of the relaxed problem is higher than the best solution found for the original problem, this means there is no room for improvement in this branch of the tree and the node is pruned, i.e. cut off. Otherwise, the procedure of solving, looking for valid inequalities and branching is repeated. Whenever a better incumbent solution is found, bounds throughout the tree are updated.

The Branch-and-Cut procedure, thus, consists of performing branching and applying cuts at the nodes of the tree:

- A *branch* is the creation of two new nodes from a parent node, as in the Branch-and-Bound. After branching constraints have been added to the model, the two resulting nodes represent distinct parts of the solution space.

- A *cut* is a constraint added to the model. The purpose of adding cuts is to limit the size of the solution domain for the continuous LP problems represented at the nodes, while not eliminating integer solutions. The addition of these cuts can yield significant performance improvements, because it reduces the number of branches required to solve the MILP problem.

The overall Branch-and-Cut procedure is summurized in Figure 3.1.

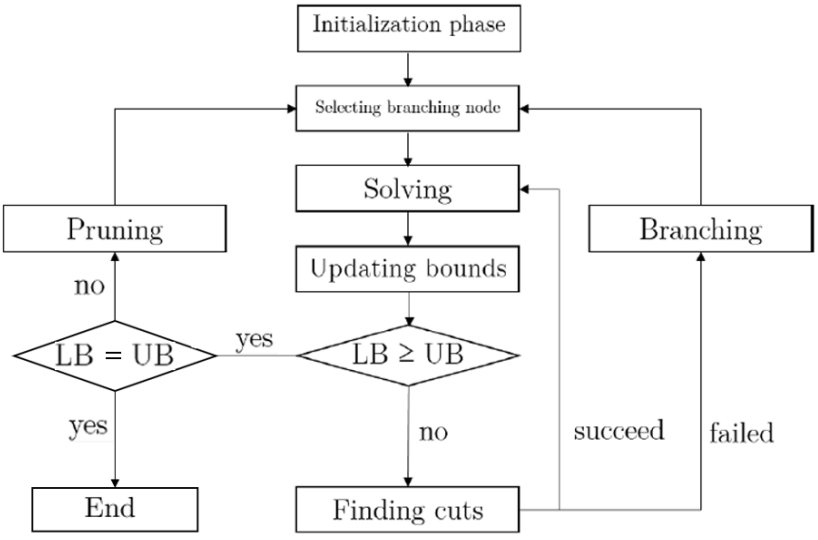


Figure 3.1 Outline of the Branch-and-Cut Algorithm

3.3 Problem statements and MILP formulations

This section provides formal definitions for the PDSTSP and the PDSMTSP, as well as a MILP formulation for each problem. A summary of the notation is given in Table 3.1.

Table 3.1 Notation.

K	Number of vehicles
M	Number of UAVs (drones)
N	Set of customers $N = \{1, \dots, n\}$
N_d	Subset of customers that may be served by UAVs (*drone-eligible*) $N_d \subset N$
$\{0\}$	Depot
A	Arc set; $A = \{(i, j) : \ i, j \in N \cup \{0\}, i \neq j\}$
t_{ij}	Travel time incurred when a vehicle goes from node i to node j; $i, j \in N \cup \{0\}$
$\widehat{t}_i$	Travel time incurred when a drone serves customer i ($\widehat{t}_i$ include back and forth trip times); $i \in N_d$

3.3.1 Parallel Drone Scheduling Traveling Salesman Problem

3.3.1.1 Problem Statement

The Parallel Drone Scheduling Traveling Salesman Problem (PDSTSP) deals with the optimal assignment and service sequence of a set of customers to a single vehicle and a fleet of drones performing deliveries independently, with drones traveling back and forth between the depot and customer locations while the vehicle visits another set of customers. A formal definition of the PDSTSP is given hereafter.

We consider a complete directed graph $G = (N \cup \{0\}, A)$ where $N = \{1, \dots, n\}$ is a customer set and 0 is the depot. A single vehicle ($K = 1$) and a fleet of M drones are available to deliver parcels to customers. The vehicle delivers customers with a single tour starting from the depot, visiting a subset of customers and returning back to the depot. Drones operate back and forth trips between the depot and the customers, delivering a single customer in each trip. Not all customers are eligible for drone delivery, because of practical constraints such as the limited payload or the flying range of drones. The subset of customers that can be served by a drone is denoted N_d. These customers are consistently called *drone-eligible* in the rest of the document. Figure 3.2 depicts a solution of the PDSTSP on an instance with 10 customers, among which 5 are drone-eligible.

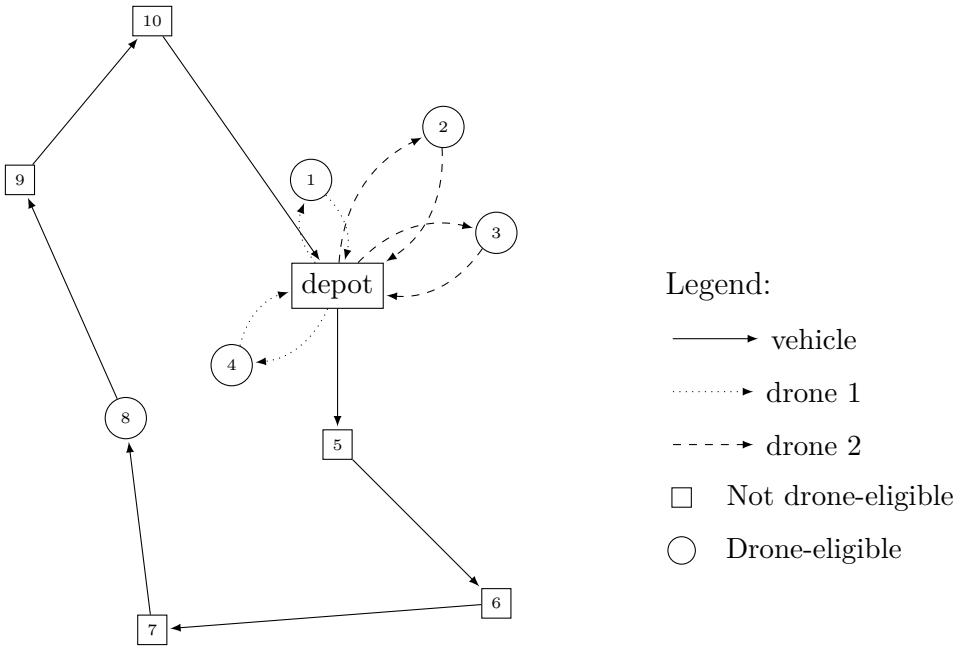


Figure 3.2 A set of 10 customers served by 1 vehicle and 2 drones.

A travel time t_{ij} is incurred when the vehicle goes through an arc $(i, j) \in A$. A travel time $\widehat{t}_i$ is incurred when a drone serves a customer $i \in N_d$. Assuming that both the vehicle and the drones can start from the depot at time 0, the objective of the PDSTSP is to minimize the delivery completion time, *i.e.*, the time at which the vehicle and all drones are back to the depot, with the service of all customers carried out.

3.3.1.2 MILP Formulation

In this section, we express the PDSTSP with a MILP formulation. We introduce the following binary decision variables :

$$z_i = \begin{cases} 1 & if\ customer\ i\ is\ visited\ by\ the\ vehicle\ (i \in N) \\ 0 & if\ customer\ i\ is\ visited\ by\ a\ drone. \end{cases}$$

$$x_{ij} = \begin{cases} 1 & if\ arc\ (i,j)\ belongs\ to\ the\ vehicle\ tour\ ((i,j) \in A) \\ 0 & otherwise. \end{cases}$$

$$y_{im} = \begin{cases} 1 & if\ customer\ i\ is\ assigned\ to\ drone\ m \\ 0 & otherwise\ (i \in N_d, 1 \leq m \leq M). \end{cases}$$

In addition, we introduce a nonnegative variable T that represents the completion time.

The model is:

$$\text{minimize } T \tag{3.6}$$

subject to

$$T \geq \sum_{(i,j)\in A} t_{ij}x_{ij} \tag{3.7}$$

$$T \geq \sum_{i\in N_d} \widehat{t}_i y_{im} \qquad (1 \leq m \leq M) \tag{3.8}$$

$$z_i = 1 \qquad (i \in N \setminus N_d) \tag{3.9}$$

$$\sum_{(i,j)\in A} x_{ij} = z_i \qquad (i \in N) \tag{3.10}$$

$$\sum_{1\leq m\leq M} y_{im} = 1 - z_i \qquad (i \in N_d) \tag{3.11}$$

$$\sum_{(0,j)\in A} x_{0j} \leq 1 \tag{3.12}$$

$$\sum_{(i,j)\in A} x_{ij} = \sum_{(k,i)\in A} x_{ki} \qquad (i \in N \cup \{0\}) \tag{3.13}$$

$$\sum_{j\in S}\sum_{k\notin S} x_{jk} \geq z_i \ (\forall\ i \in S,\ S \subseteq N, S \neq \emptyset) \tag{3.14}$$

$$z_i \in \{0,1\} \qquad (i \in N) \tag{3.15}$$

$$x_{ij} \in \{0,1\} \qquad ((i,j) \in A) \tag{3.16}$$

$$y_{im} \in \{0,1\} \qquad (i \in N_d, 1 \leq m \leq M) \tag{3.17}$$

$$T \geq 0 \tag{3.18}$$

The objective (3.6) is to minimize the completion time T. Constraints (3.7) and (3.8) give lower bounds on T expressed using vehicle assignment and drone assignment variables. Constraints (3.9) ensure that all the customers that are not drone-eligible are served by the vehicle. Constraints (3.10) and (3.11) guarantee that every customer is served either by the vehicle or a drone, exactly once. Constraint (3.12) stipulates that the vehicle leaves the depot at most once. Constraints (3.13) ensure flow conservation for the vehicle tour. Subtour elimination constraints (SECs) are provided by (3.14). These constraints ensure that given a non empty subset of customers $S \subseteq N$, if all the customers in S are visited by the vehicle, there is at least one outgoing arc from S. Finally, constraints (3.15) to (3.18) define the decision variables.

3.3.2 Parallel Drone Scheduling Multiple Traveling Salesman Problem

In this section, we introduce the Parallel Drone Scheduling Multiple Traveling Salesman Problem (PDSMTSP) which is an extended version of the PDSTSP (presented in Section 3.3.1) by considering K vehicles ($K > 1$).

3.3.2.1 Problem Statement

The PDSMTSP aims at finding the optimal set of routes to be performed by a fleet of vehicles and a fleet of drones working in parallel to serve a given set of customers. Each vehicle delivers customers with a single tour starting from the depot, visiting a subset of customers and returning back to the depot, while UAVs perform back and forth trips between the depot and the customer locations. Therefore, the PDSMTSP extends the PDSTSP formulation by including multiple vehicle routes and uses the same notation.

A solution of the PDSMTSP on an instance with 10 customers, among which 5 are drone-eligible is illustrated in Figure 3.3.

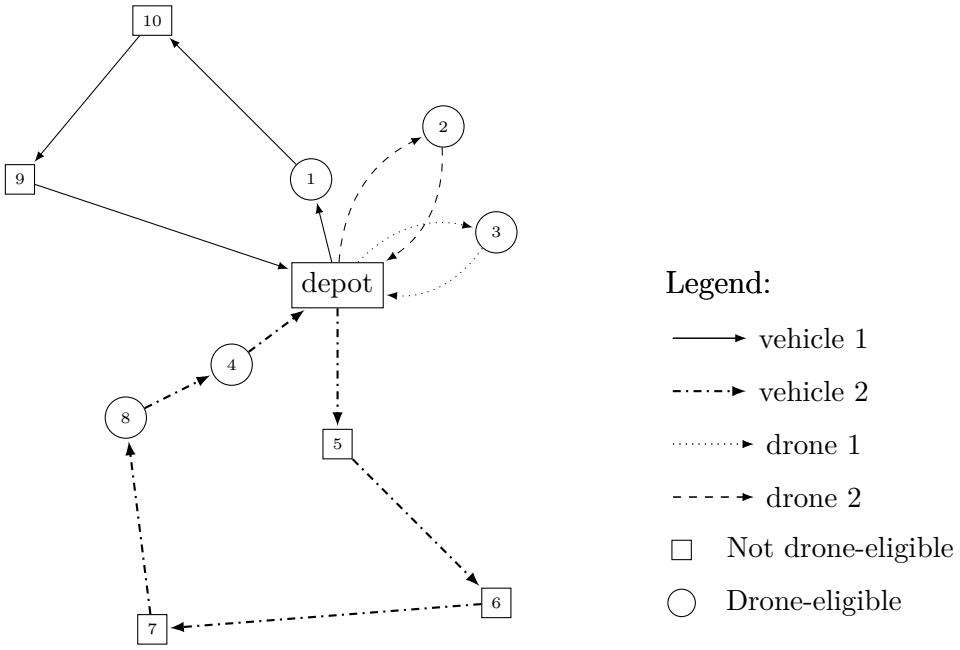


Figure 3.3 A set of 10 customers served by 2 vehicles and 2 drones.

3.3.2.2 MILP Formulation

To express the PDSMTSP with a MILP formulation, we introduce the following decision variables:

$$z_i = \begin{cases} 1 & if\ customer\ i\ is\ visited\ by\ a\ vehicle\ (i \in N) \\ 0 & if\ customer\ i\ is\ visited\ by\ a\ drone. \end{cases}$$

$$x_{ijk} = \begin{cases} 1 & if\ arc\ (i,j)\ belongs\ to\ vehicle\ tour\ k\ ((i,j) \in A) \\ 0 & otherwise. \end{cases}$$

$$w_{ij} = \begin{cases} 1 & if\ arc\ (i,j)\ belongs\ to\ a\ vehicle\ tour\ ((i,j) \in A) \\ 0 & otherwise. \end{cases}$$

$$y_{im} = \begin{cases} 1 & if\ customer\ i\ is\ assigned\ to\ drone\ m \\ 0 & otherwise\ (i \in N_d, 1 \leq m \leq M). \end{cases}$$

$T \geq 0$ *indicating completion time*

The model is:

$$\text{minimize } T \tag{3.19}$$

subject to

$$T \geq \sum_{(i,j)\in A} t_{ij} x_{ijk} \qquad (1 \leq k \leq K) \tag{3.20}$$

$$T \geq \sum_{i\in N_d} \widehat{t}_i y_{im} \qquad (1 \leq m \leq M) \tag{3.21}$$

$$z_i = 1 \qquad (i \in N \setminus N_d) \tag{3.22}$$

$$\sum_{1\leq m\leq M} y_{im} = 1 - z_i \qquad (i \in N_d) \tag{3.23}$$

$$w_{ij} = \sum_{1\leq k\leq K} x_{ijk} \qquad ((i,j) \in A) \tag{3.24}$$

$$\sum_{(i,j)\in A} w_{ij} = z_i \qquad (i \in N) \tag{3.25}$$

$$\sum_{(0,j)\in A} x_{0jk} \leq 1 \qquad (1 \leq k \leq K) \tag{3.26}$$

$$\sum_{(i,j)\in A} x_{ijk} = \sum_{(j,i)\in A} x_{jik} \qquad (i \in N \cup \{0\},\ 1 \leq k \leq K) \tag{3.27}$$

$$\sum_{j\in S} \sum_{l\notin S} w_{jl} \geq z_i \qquad (\forall i \in S, S \subseteq N, S \neq \emptyset) \tag{3.28}$$

$$z_i \in \{0,1\} \qquad (i \in N) \tag{3.29}$$

$$x_{ijk} \in \{0,1\} \qquad ((i,j) \in A,\ 1 \leq k \leq K) \tag{3.30}$$

$$w_{ij} \in \{0,1\} \qquad ((i,j) \in A) \tag{3.31}$$

$$y_{im} \in \{0,1\} \qquad (i \in N_d, 1 \leq m \leq M) \tag{3.32}$$

$$T \geq 0 \tag{3.33}$$

The objective (3.19) is to minimize completion time T. Constraints (3.20) and (3.21) give lower bounds on T expressed by completion times of each vehicle and drone. Constraints (3.22) ensure that customers that are not drone-eligible are served by a vehicle. Constraints (3.23) guarantee that every drone-eligible customer is also served, either by a vehicle or a drone. Constraints (3.24) and (3.25) express that if a customer is served by a vehicle, it is

assigned to a vehicle tour. Constraints (3.26) stipulate that each vehicle leaves the depot at most once. Constraints (3.27) ensure flow conservation for vehicle tours. Subtour elimination constraints are provided by constraints (3.28). These constraints ensure that given a non empty subset of customers $S \subseteq N$, if at least one customer in S is visited by a vehicle, there exists at least one outgoing arc from S. Finally, constraints (3.29) to (3.33) define decision variables.

Previous models could be used to solve exactly the problems with a MILP solver, but, because of the weak LP relaxation quality and the exponential size of subtour elimination constraints (3.13), it may only allow solving very small-size instances. In order to address slightly larger instances, we implemented a simple Branch-and-Cut algorithm in which subtour elimination constraints are dynamically added to the formulation.

3.4 Branch-and-Cut Algorithm

In both MILP formulations described in section 3.3.1.2 and section 3.3.2.2, subtour elimination constraints (equations 3.14 and 3.28) are indeed in exponential number. Thus these formulations cannot be direct handled by a MILP solver: entering all these constraints at once is usually not practical, except for very small instances.

To solve these linear programs, we start by solving (with a Branch-and-Bound method) a relaxation version of the MILP without subtour elimination constraints and without integrality constraints. Then we use a *separation* algorithm (cutting plane method) to dynamically add the subtour elimination constraints which ensure that a tour is fully connected i.e. without subtours. In the following we describe the general principle of the method together with the separation algorithms which detect violated subtour elimination constraints.

3.4.1 General principle of the method

We solve the MILP proposed for the PDSTSP and the PDSMTSP by using a Branch-and-Cut method which is a combination of Branch-and-Bound method with the cutting plane algorithm. In the Branch-and-Cut approach, the LP relaxation of problem (LP without SECs and without integrality constraints) is solved at the root node of a search tree and if the solution is integral the algorithm terminates with the optimal solution to the original MILP. If the solution is fractional then cuts (violated SECs) are identified by solving a *separation* problem. These cuts are added to the constraint region and the LP relaxation is re-solved. When the effect of the additional cuts becomes marginal the search tree is extended by branching

Designing a Branch-and-Cut algorithm from scratch stretches far out of the scope of this work. We will be content with using CPLEX solver for making choices regarding preprocessing, choosing branching variables and whether to branch or cut at a current subproblem. Nevertheless, there are options in the CPLEX solver allowing us to add subtour inequalities

ourselves. Subtour inequalities can be added using both *Lazy Constraint Callbacks* and *User Cut Callbacks*. Whereas the *Lazy Constraint Callbacks* cut the integer solution space and are only called when the algorithm has found an integer solution, the *User Cut Callbacks* only cut off fractional solutions and are called at any point in the algorithm. Cuts generated in callbacks are returned to the MILP solver engine which adds these cuts to the Cut Pool. These cuts are merged with the cuts generated with the solver builtin cut generators and a subset of these cuts is included to the LP relaxation model. Note that in the Branch-and-Cut algorithm context cuts are optional components and only those that are classified as *good* cuts by the solver engine will be accepted, i.e., cuts that are too dense and/or have a small violation could be discarded, since the cost of solving a much larger linear program may not be worth the resulting bound improvement.

As the subtour inequalities are actually a part of PDSTSP and PDSMTSP formulations and we would like the algorithm to find violated subtour inequalities in every node, we have chosen to implement the callback for these cuts as both *Lazy Constraint Callbacks* and *User Cut Callbacks*. In the next section, we detail the description of the proposed *separation* procedures.

3.4.2 Separation of subtour elimination constraints

In this section, the separation of SECs is explained.

Let's consider the PDSTSP. The subtour elimination constraints are expressed by equation (3.14). These constraints ensure that given a non empty subset of customers $S \subseteq N$, if all the customers in S are visited by the vehicle, there is at least one outgoing arc from S. For the separation of SECs, we implemented two algorithms: One for the separation of SECs when the solution is integer and another one for the separation of SECs when the solution is fractional.

The separation of subtour elimination constraints for fractional solutions amounts to the solution of a max-flow or min-cut problem. Meanwhile identifying subtour elimination constraints violated by an integer solution is much easier (by using a simple scan).

3.4.2.1 Separation of SECs for integer solution

Algorithm 1 describes the procedure for finding subtours when the solution is integer (that means when values of vectors x, y and z obtained after solving a relaxation of the MILP model by ignoring integrality constraints and SECs are integer). Given a solution of the MILP relaxation, let $G_1 = (V_1, A_1)$ be the graph such that :

- $V_1 = \{0\} \cup \{i \in N \text{ such that } z_i = 1\}$
- $A_1 = \{(i,j) \text{ such that } i,j \in V_1 \text{ and } x_{ij} = 1\}$

V_1 is a set containing the customers assigned to the vehicle as well as the depot and A_1 is the set of travelled arcs between nodes in V_1

The algorithm takes graph $G_1 = (V_1, A_1)$ as input and returns as output the list of subtours found in G_1. We consider that a subtour doesn't contain the depot ($S \subseteq N$). Lines 3 to 14 describe how to find a subtour S starting from a vertex $i \in V_1,\ i \neq 0$ not yet visited. S is considered as a subtour if $0 \notin S$ and $|S| \geq 2$. Subtours are circular paths, that have no connections to the depot. A circle containing a depot is a valid tour. Once all the subtours S are found, contraints 3.14 are added to the model.

Algorithm 1 SeparateLazyConstraints

Input : $G_1 = (V_1, A_1)$
Output : *list of subtours*

1: $subToursList \leftarrow \emptyset$
2: $Visited[i] \leftarrow false\ \forall\ i \in V_1,\ i \neq 0$
3: **while** there exists $i \in V_1 \backslash \{0\}$ with $Visited[i] = false$ **do**
4: $start \leftarrow i\ ,\ S \leftarrow \{i\}$
5: $Visited[i] \leftarrow true$
6: $containsDepot \leftarrow false$
7: **while** the successor j of i ($x_{ij} = 1$) is not equal to $start$ **do**
8: $i \leftarrow j$
9: $Visited[i] \leftarrow true$
10: $S \leftarrow S \cup \{i\}$
11: **if** $i = 0$ **then**
12: $containsDepot \leftarrow true$
13: **end if**
14: **end while**
15: **if** $containsDepot = false$ **then**
16: $subToursList \leftarrow subToursList \cup \{S\}$
17: **end if**
18: **end while**
19: **return** $subToursList$

Figure 3.4 illustrates an integer vehicle tour obtained after solving the LP relaxation. Applied to this graph, Algorithm 1 returns a subtour $S = \{3, 4, 5\}$ that violates contraints 3.14 ($\sum_{j \in S} \sum_{k \notin S} x_{jk} = 0 < z_3 = z_4 = z_5 = 1$).

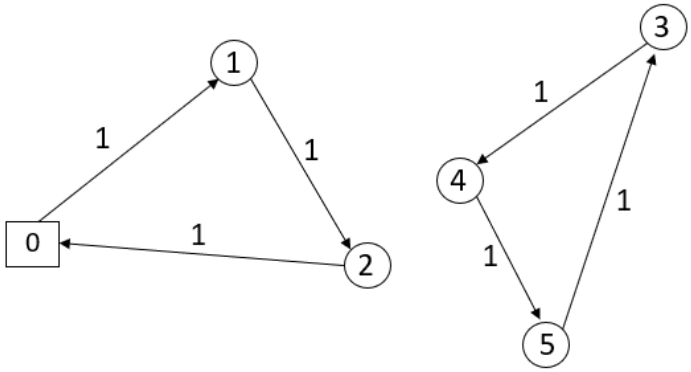


Figure 3.4 Variables x_{ij} equal to 1 after solving the LP relaxation

3.4.2.2 Separation of SECs for fractional solution

When the values of vectors x and z (obtained after the resolution of the linear model with integrality constraints and SECs relaxed) are fractional, the separation algorithm to find a subset $S \subseteq N$ that violates constraints (3.14) becomes more complex.

As when separating SECs in a classic TSP when the solution is fractional, we use *min cut/max flow* algorithm [PR91][App+98][FJF15].

In the context of the TSP, the separation of these SECs is made by a maximum flow calculation in the subgraph made up of arcs with positive flow. A subtour is found be trying to send 1 unit of flow from a source node (starting city) to each other node of the subgraph.

In our context, the *max flow* algorithm is applied on the graph $G_2 = (V_2, A_2)$ such that:

- $V_2 = \{0\} \cup \{i \in N \text{ such that } z_i \neq 0\}$
- $A_2 = \{(i,j) \text{ such that } i,j \in V_2 \text{ and } x_{ij} \neq 0\}$
- Every arc $a = (i,j) \in A_2$ has a capacity $c(a) = x_{ij}$

For each client node $i \in V_2$, we try to send a flow F of z_i units from $\{0\}$ to i such that for each crossed arc (i,j), $F(i,j) \leq x_{ij}$ (the flow passed over the arc (i,j) must not exceed its capacity which is x_{ij}). If we do not succeed in sending this flow of z_i units from $\{0\}$ to i, then, all of the node clients not reached during the search for an augmenting chain from $\{0\}$ to i will form the set S which violates the constraint (3.14).

Algorithm 2 describes the search procedure for a subset S which violates constraints (3.14). In Line 1, all client nodes are defined as not processed ($Visited[i] \leftarrow false \; \forall \; i \in V \backslash \{0\}$). The flow on every arc $a \in A_2$ is initialized to 0 and a not yet processed client node i is chosen (Lines 4 and 5). We try to send a flow of z_i units from 0 to i by searching each time for

an augmenting chain and by updating the residual capacities of the arcs of the chain found as follows: let δ be the residual capacity of the augmenting chain μ found (Line 12). If we are on a *forward arc* $a \in \mu^+$, the flow on the arc is incremented by δ (Line 15 : a flow of δ unit is passed through the arc). If we are on a *backward arc* $a \in \mu^-$, the flow on the arc is decremented by δ (Line 17 : a flow of δ unit which had already been validated on the arc is removed). If we do not succeed in sending a flow of z_i unit from 0 to i, we deduce the subset S as being the set of all clients not reached during the search for an augmenting chain linking 0 to i which failed (Line 25). Finally, if there is no such subset S which violates the constraint (3.14), the algorithm returns $Fail$.

Algorithm 2 SeparateUserCuts

Input : $G_2 = (V_2, A_2)$
Output : Subset S that violates contraints 3.14 or $Fail$

1: $Visited[i] \leftarrow false \ \forall \ i \in V_2 \backslash \{0\}$
2: $Fail \leftarrow false$
3: **while** $Fail = false$ and there exists $i \in V_2 \backslash \{0\}$ with $Visited[i] = false$ **do**
4: choose i such that $Visited[i] = false$
5: $f(a) \leftarrow 0, \ \forall \ a \in A_2$
6: $TargetFlow \leftarrow z_i, \ stop \leftarrow false$
7: **while** $TargetFlow \neq 0$ and $stop = false$ **do**
8: $\mu \leftarrow findAugmentingChain(G, i)$
9: **if** $\mu = \emptyset$ **then**
10: $stop \leftarrow true$
11: **else**
12: $\delta \leftarrow min(min_{a \in \mu^+}\{c(a) - f(a)\}; min_{a \in \mu^-}\{f(a)\})$
13: **for** $a \in \mu$ **do**
14: **if** $a \in \mu^+$ **then**
15: $f(a) \leftarrow f(a) + \delta$
16: **else**
17: $f(a) \leftarrow f(a) - \delta$
18: **end if**
19: **end for**
20: $TargetFlow \leftarrow TargetFlow - \delta$
21: **end if**
22: **end while**
23: **if** $stop$ **then**
24: $Fail \leftarrow true$
25: $S \leftarrow i \in V_2$ not reached from 0 when searching for μ
26: **return** S
27: **else**
28: $Visited[i] \leftarrow true$
29: **end if**
30: **end while**
31: **return** $Fail$

The procedure for finding an augmenting chain is described in Algorithm 3. The algorithm

takes as input the graph $G_2 = (V_2, A_2)$ described above as well as a node called "*target*" ($target \in V_2 \backslash \{0\}$) not yet processed ($Visited[target] = false$). Its objective is to find an augmenting chain connecting 0 to *target*.

The procedure starts at the depot 0 as the current node. At each current node, its eligible neighbors are investigated. The eligible neighbors of a node u are :

- the successors v of u such that the arc (u, v) is not saturated;
- the predecessors v' of u such that the arc (v', u) has a positive flow.

So, while there are still eligible neighbors not yet processed and the augmenting chain is not yet found (that means *target* is not reached), we choose a neighbor to continue building the augmenting chain. If the choosen neighbor is *target* then the constructed path is the augmenting chain μ and we validate the flow. At the end of the procedure, the algorithm returns the chain μ ($\mu = \emptyset$ if no augmenting chain is found).

Algorithm 3 findAugmentingChain(G_2, $target$)

Input : $G_2 = (V_2, A_2)$, $target$ ($target \in V_2 \backslash \{0\}$)
Output : An augmenting chain μ from 0 to $target$

$\mu \leftarrow \emptyset$, $Visited[0] \leftarrow true$, $Visited[i] \leftarrow false \ \forall \ i \in V_2 \backslash \{0\}$
$cur\_node \leftarrow 0$
while $Visited[target] = false$ and there is an eligible neighbor j of cur_node
    not yet visited **do**
    if $(a = (cur\_node, j) \in A_2$ and $f(a) < c(a))$ or $(a = (j, cur\_node) \in A_2$ and $f(a) > 0)$ **then**
        $Visited[j] \leftarrow true$
        $\mu \leftarrow \mu \cup a$
    end if
    $curr\_node \leftarrow j$
end while
if $Visited[target] = false$ **then**
    $\mu \leftarrow \emptyset$
end if
return μ

Illustration

To illustrate how this algorithm works, we consider the fractional solution shown in Figure 3.5. We assume that we have 1 vehicle and 1 drone. For each node i of the graph, the values of the variables z_i and y_{i1} are represented in the table. On each arc (i, j) of the graph is mentioned the value of the variable x_{ij} which corresponds to the arc capacity.

We want to send a flow of $z_3 = 0.9$ units from the 0 to node 3. Figure 3.6 shows the iterations of the Algorithm. In this Figure:

- on each arc, the value pair (f, c_f) is mentioned where f is the current flow on the arc and c_f is the residual capacity of the arc;
- arcs in bold are saturated arcs ($c_f = 0$);
- every augmenting chain connecting node 0 to node 3 is shown in dashed lines;
- any update of the value pair (f, c_f) is materialized in bold and italic;
- dashed and dotted arcs (iteration 4) are those crossed during the search for an augmenting chain connecting node 0 to node 3 which failed.

	i=1	i=2	i=3	i=4	i=5	i=6	i=7
z_i	1	0.9	0.9	0.9	1	1	1
y_{i1}	0	0.1	0.1	0.1	0	0	0

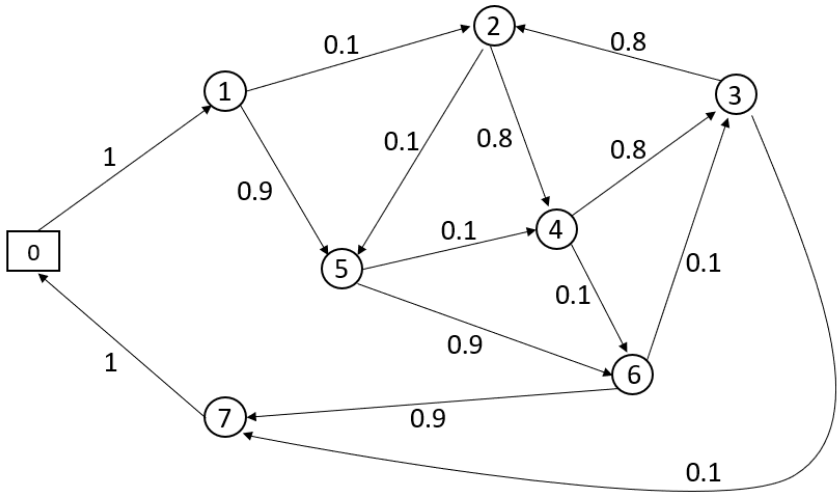


Figure 3.5 A fractional solution of the LP relaxation

The iterations shown in this figure (Figure 3.6) are described as follows :

- at iteration 0, the flow is initialized to 0 on every arc $a = (i, j)$ of the graph and the residual capacity of each arc corresponds to its initial capacity ($c_f(a) = c(a) = x_{ij}$);
- at iteration 1, a flow of 0.1 unit is sent from node 0 to node 3 through the augmenting chain $0 - 1 - 2 - 4 - 3$. Now the arc $(1, 2)$ is saturated.
- at iteration 2, an additional flow of 0.1 unit is sent from node 0 to node 3 through the augmenting chain $0 - 1 - 5 - 4 - 3$. After this flow passage, the arc $(5, 4)$ is saturated.

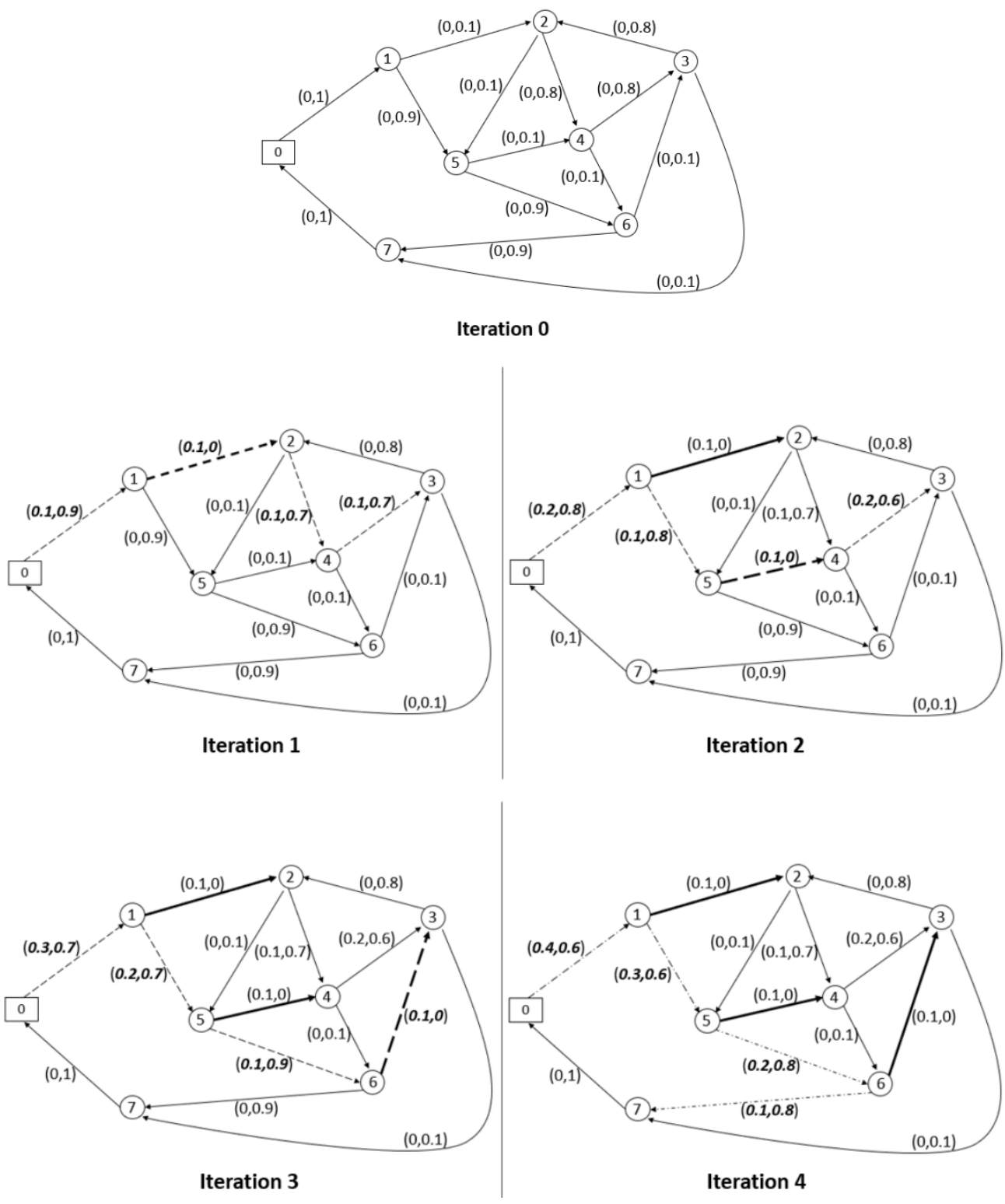


Figure 3.6 Iteration of Algorithm 2 when trying to send a flow of $z_3 = 0.9$ units from node 0 to node 3

- at iteration 3, a new increasing chain allowing to pass a flow of 0.1 units from node 0 to node 3 is found: $0-1-5-6-3$. The arc $(6,3)$ becomes saturated.
- at iteration 4, we try again to send the maximum possible flow starting from node 0 and trying to reach node 3. However, we do not succeed. Finally we succeed in sending only a flow of 0.3 unit from node 0 to node 3. The search for an augmenting chain having failed, we deduce the subset $S = \{2,3,4\}$ corresponding to the nodes which could not be reached when searching for the augmenting chain. S does not satisfy constraints 3.14 ($\sum_{j\in S}\sum_{k\notin S} x_{jk} = 0.3$ and $z_2 = z_3 = z_4 = 0.9$).

The separation algorithms described above are the same used for the PDSMTSP. The only change is to rename x_{ij} into w_{ij} in the definition of the graphs G_1 and G_2.

3.5 Experiments and results

In this section we test the effectiveness of the Banch-and-Cut algorithms for the PDSTSP and the PDSMTSP. The environment used for the computational work is Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60 GHz with 2x8 cores 25M cache and 62.5GB of RAM. C++ language is used for the implementation part and the MILPs of Sections 3.3.1.2 and 3.3.2.2 are solved by using IBM Concert Technology and CPLEX 12.6.

3.5.1 PDSTSP: 1 vehicle available

For the PDSTSP, we conducted experiments on two sets of instances. First, we evaluate the efficiency of our MILP on Murray and Chu‘s [MC15] benchmark instances. Unfortunately, they are of small size (10-20 customers) and it is difficult to draw definitive conclusions with them. For this reason, we introduced instances of larger size from the TSPLIB [1] on which we conducted a second set of experiments. The following subsections describe the instances used and the results obtained.

3.5.1.1 Experiments on Murray and Chu's instances

Murray and Chu's [MC15] instances were generated with either 10 or 20 customers. The vehicle and drone speeds were fixed at the same speed of 25 miles/h. However, distances were computed with a Manhattan distance for the vehicle and with a Euclidean distance for the drones. The depot location was selected as being either near the center of all customers, near the edge of the customer region, or at the origin. Customer locations were generated such that either 20%, 40%, 60%, or 80% of all customers were located within the drone's range from the depot, with the drone having a flight endurance of 30 minutes. Finally, 10%-20% of

[1] TSPLIB is a library of sample instances for the TSP (and related problems)

customers were arbitrarily set as not drone-eligible because of excessive parcel weights. Each of the above parameter settings was repeated 10 times, resulting in 120 instances with 10 customers and 120 instances with 20 customers. All these instances were solved with a single truck and either 1, 2, or 3 available UAVs, resulting in 720 test instances.

In their paper Murray and Chu [MC15] provided a MILP formulation for the PDSTSP which they solved via Gurobi, with a 3-min time limit per problem. Computational work was conducted on an HP 8100 Elite desktop PC with a quad-core Intel i7–860 processor and 4 GB RAM running Ubuntu Linux 14.04 in 64-bit mode. Gurobi was able to find the optimal solution for all of the 10-customer problems. However, Gurobi was terminated at the time limit on 87 of the 360 20-customer instances.

Table 3.2 presents a comparison of the results obtained by the linear model of Murray and Chu with those obtained by our linear model solved by Branch-and-Cut. The comparison focuses on the execution time in seconds and the number of optimal solutions found. Our linear model manages to find the optimal solutions for all these instances with a relatively low execution time.

Table 3.2 Comparison Murray and Chu MILP versus our MILP

MILP	10 Customers			20 Customers		
	CPU (s)			CPU (s)		
	Avg	Max	#Sol	Avg	Max	#Sol
Murray et al. IP	0.3194	2.02	360	77.775	180.00	273
Our IP	0.0692	0.23	360	0.563	13.57	360

3.5.1.2 Experiments on new instances generated from TSPLIB

To evaluate the MILP solver on larger instances, we generated another set of instances by using TSPLIB files *att48.tsp, berlin52.tsp, eil101.tsp, gr120.tsp, pr152.tsp and gr229.tsp*. The number in the file name corresponds to the number of customers which are represented by their coordinates x and y. To stay consistent with Murray and Chu's instances, we used the Manhattan distance for the vehicle trip and the Euclidean distance for the drone trip. From each file, we generated what we call a reference instance and several instances that can be derived from this reference instance by changing some parameters. These instances are characterized by:

- **The position of the depot**: A fictitious point was added to represent the depot. The depot location was selected as being either near the center of all customers ($x_0 = \frac{\max_{i \in N}(x_i) - \min_{i \in N}(x_i)}{2}$, $y_0 = \frac{\max_{i \in N}(y_i) - \min_{i \in N}(y_i)}{2}$) or at the left-bottom corner of the region ($x_0 = \min_{i \in N}(x_i)$, $y_0 = \min_{i \in N}(y_i)$).

- **The percentage of drone-eligible customers**: Instances were generated with either 0%, 20%, 40%, 60%, 80% or 100% of drone-eligible customers. We proceeded as follows. Let P be the target percentage of drone eligible customers. We introduced L ($L = 30\%$) the percentage of non-drone-eligible customers that are not drone-eligible because of parcel weights. We selected these customers according to their index n: customers not drone-eligible because of their parcel weight are customers such that $n \pmod{\lfloor \frac{1}{(1-P)\times L} \rfloor} = 0$. Remaining customers were then set as drone-eligible in the increasing order of their distance to the depot, until reaching $P\%$ of drone-eligible customers.

- **The drone speed**: The drone can go at the same speed as the vehicle or 2, 3, 4 or 5 times faster. In this case, vector $\widehat{d}$ is simply divided by the speed factor. The speed of the vehicle is assumed to be one unit of native distance per unit of time.

- **The number of drones**: The drone fleet can be composed of 1, 2, 3, 4 or 5 drones.

In reference instances, the depot is located at the center, there are $P = 80\%$ of drone eligible customers, the speed factor is 2 and a single drone is used.

Note that the method for generating drone-eligible customers might appear a little bit twisted, but it has the advantage of being fully deterministic and easily reproducible.

The first goal with these new instances is to analyse more in details the branch-and-Cut performances when solving the MILP of Section 3.3.1.2. Besides that we also conduct some sensitivity analyzes. We compare solutions obtained while changing the depot position, the percentage of drone-eligible customers, the drone speed factor and the number of drones, respectively. For that purpose, we consider the six reference instances and construct new instances by changing a single parameter at a time.

Results are presented in Tables 3.3–3.6. For each instance, the following information is reported:

Gap1 the percentage gap between the solution cost for the relaxed linear program $\mathtt{z}_{lr}$ at the root (the integrity constraints are relaxed but the SECs are applied) and the cost of the best integer solution $\mathtt{z}_{ub}$ (Gap1 $= 100 \times \frac{\mathtt{z}_{ub}-\mathtt{z}_{lr}}{\mathtt{z}_{ub}}$);

Gap2 the percentage gap between the best lower bound $\mathtt{z}_{lb}$ and the best integer solution cost $\mathtt{z}_{ub}$ (Gap2 $= 100 \times \frac{\mathtt{z}_{ub}-\mathtt{z}_{lb}}{\mathtt{z}_{ub}}$);

#Nodes the number of nodes explored in the Branch-and-Bound tree (when the number is zero that means the process ended at the root node);

#LC the number of SECs added with LAZYCONSTRAINTCALLBACK (used to search for constraints violated by an integer solution)

#UC the number of SECs added with USERCUTCALLBACK (used to search for constraints violated by a fractional solution)

CPU the Branch-and-Cut running time in seconds;

#Dro. the number of drones used in the best integer solution;

D.C. the number of customers assigned to the drones in the best integer solution;

C.T. the completion time (ie. the Branch-and-Cut best integer solution cost $\mathbf{z}_{ub}$)

We can tell from the results that the Branch-and-Cut succeeds to find the optimal solution (Gap2 = 0.00) for all instances of size $n < 100$ (*att48*, *berlin52*). For these instances, it was possible to enumerate fewer than 160 nodes within the 10800 seconds time limit, and in some cases only the solution of the root-node relaxation was completed. An optimal solution could be found for most tests made on eil101 instances (except test on *eil101* with 3 drones for which we get an integrality gap of 2.21%). Also some few tests on eil101 couldn't be completed (no integer solution found) within the time limit.

The Branch-and-Cut was unable to find any integer solution for *pr152* and *gr229* instances within the time limit (except test on *pr152* with 0% of drone eligible customers). The Branch-and-Cut also fails to find an integer solution for some tests made with *gr120* instance (there are some missing lines in Tables 3.4 to 3.6). However for tests that could be completed on *gr120* instance, we can see that the root gap (**Gap1**) and the integrality gap (**Gap2**) are close to 0. The largest gap between the upper bound and lower bound is seen to be 0.78%. Thus the upper bound found by the Branch-and-Cut is close to the optimal value.

We can also observe that there are more user cuts (**UC**) than lazy constraints (**LC**) added to the model. One can also underline the increase of the total number of cuts (**LC+UC**) when the size of the instance increases.

Regarding the sensitive analyses comparing solutions obtained while changing the depot position, the percentage of drone-eligible customers, the drone speed factor and the number of drones, the results show that in general the completion time is better when the depot is located at the center of all the customers (see Table 3.3). Furthermore less customers are assigned to the drone when the depot is located at the corner. These observations seem to be in line with the expected results. In Table 3.6 we can notice that faster the drone is, better the completion time is also, and the drone is able to visit more customers. Also when the number of drones increases, the completion time is improved. Finally, table 3.4 shows that the more we have drone-eligible customers the more we have chance to find a better completion time value.

Table 3.3 Impact of depot position (80% of drone-eligible customers, 1 drone, speed factor 2)

Instance	Position	CPLEX Branch-and-Cut details						Solution details		
		Gap1 (%)	Gap2 (%)	#Nodes	#LC	#UC	CPU (s)	#Dro.	#D.C.	C.T.
att48	center	0.02	0.00	4	37	2206	12.61	1	17	29954.00
	corner	0.49	0.00	158	36	2904	14.29	1	9	33798.00
berlin52	center	0.43	0.00	37	35	4696	122.04	1	18	6386.48
	corner	0.76	0.00	63	38	3862	12.70	1	8	7830.00
eil101	center	0.37	0.00	2557	67	36004	4978.93	1	28	564.00
	corner	0.54	0.00	941	51	19968	1952.34	1	18	648.98
gr120	center	0.89	0.78	697	54	36678	10097.20	1	31	1417.65
	corner	0.92	0.39	4065	90	34002	10521.00	1	18	1730.00

Table 3.4 Impact of the percentage of drone-eligible customers (depot at the center, 1 drone, speed factor 2)

Instance	DE(%)	CPLEX Branch-and-Cut details						Solution details		
		Gap1 (%)	Gap2 (%)	#Nodes	#LC	#UC	CPU (s)	#Dro.	#D.C.	C.T.
att48	0%	0.00	0.00	0	47	275	0.63	0	0	42136.00
	20%	0.00	0.00	0	40	1282	2.00	1	8	38662.00
	40%	0.00	0.00	0	32	2218	8.54	1	17	31592.00
	60%	0.10	0.00	7	29	1624	5.96	1	16	30788.80
	80%	0.02	0.00	4	37	2206	12.61	1	17	29954.00
	100%	3.74	0.00	34	51	3409	19.64	1	16	27784.00
berlin52	0%	0.21	0.00	7	49	180	0.92	0	0	9675.00
	20%	0.27	0.00	17	37	1536	4.53	1	7	9350.00
	40%	0.00	0.00	0	31	1983	7.08	1	17	8300.00
	60%	1.11	0.00	80	35	5156	105.98	1	26	7410.00
	80%	0.43	0.00	37	35	4696	122.04	1	18	6386.48
	100%	3.55	0.00	129	47	5187	94.05	1	14	6192.00
eil101	0%	0.24	0.00	130	107	1837	104.07	0	0	819.00
	20%	0.25	0.00	10	82	14198	1229.59	1	20	736.00
	60%	0.32	0.00	217	63	14371	868.34	1	27	578.00

Continued on next page

Table 3.4 – continued from previous page

Instance	DE(%)	CPLEX Branch-and-Cut details						Solution details		
		Gap1 (%)	Gap2 (%)	#Nodes	#LC	#UC	CPU (s)	#Dro.	#D.C.	C.T.
	80%	0.37	0.00	2557	67	36004	4978.93	1	28	564.00
	100%	0.46	0.00	3035	62	28972	7122.93	1	26	560.00
gr120	0%	0.36	0.00	1042	110	4068	1448.52	0	0	2006.00
	60%	0.61	0.00	2131	83	24613	4982.51	1	30	1494.00
	80%	0.89	0.78	697	54	36678	10097.20	1	31	1417.65
pr152	0%	0.68	0.00	522	223	9603	3234.24	0	0	86596.00

Table 3.5 Impact of the number of drones (depot at the center, 80% of drone-eligible customers, speed factor 2)

Instance	Drone(s)	CPLEX Branch-and-Cut details						Solution details		
		Gap1 (%)	Gap2 (%)	#Nodes	#LC	#UC	CPU (s)	#Dro.	#D.C.	C.T.
att48	1	0.02	0.00	4	37	2206	12.61	1	17	29954.00
	2	0.02	0.00	17	40	3308	57.61	2	24	28686.00
	3	12.36	0.00	7	41	4650	60.92	3	31	28610.00
	4	0.00	0.00	0	39	5566	297.21	4	31	28610.00
	5	0.00	0.00	0	44	8292	667.01	5	34	28610.00
berlin52	1	0.43	0.00	37	35	4696	122.04	1	18	6386.48
	2	0.31	0.00	76	37	4510	117.6	2	31	5290.91
	3	0.00	0.00	0	34	2617	53.82	3	36	5190.00
	4	0.00	0.00	0	26	3280	125.13	4	38	5190.00
	5	0.00	0.00	0	17	6288	438.23	3	36	5190.00
eil101	1	0.37	0.00	2557	67	36004	4978.93	1	28	564.00
	2	0.46	0.00	2102	50	41839	9193.01	2	40	456.00
	3	2.33	2.21	24	41	19350	10120.50	3	51	400.09
	4	0.27	0.00	1083	46	30294	7739.19	4	59	346.00
gr120	1	0.89	0.78	697	54	36678	10097.2	1	31	1417.65

Table 3.6 Impact of the drone speed (depot at the center, 80% of drone-eligible customers, 1 drone)

Instance	Speed	CPLEX Branch-and-Cut details						Solution details		
		Gap1 (%)	Gap2 (%)	#Nodes	#LC	#UC	CPU (s)	#Dro.	#D.C.	C.T.
att48	1	1.03	0.00	90	30	2160	14.38	1	10	33234.00
	2	0.02	0.00	4	37	2206	12.61	1	17	29954.00
	3	0.28	0.00	16	42	2475	21.49	1	21	29142.00
	4	0.02	0.00	5	37	4053	112.35	1	24	28686.00
	5	0.00	0.00	0	45	4163	140.63	1	28	28610.00
berlin52	1	0.97	0.00	102	42	5287	49.40	1	13	7450.00
	2	0.43	0.00	37	35	4696	122.04	1	18	6386.48
	3	0.30	0.00	79	37	5416	175.34	1	23	5656.56
	4	0.30	0.00	52	26	4523	156.45	1	31	5290.65
	5	0.00	0.00	0	36	4071	142.50	1	37	5190.00
eil101	1	0.67	0.00	3833	72	21863	3386.33	1	18	650.00
	2	0.37	0.00	2557	67	36004	4978.93	1	28	564.00
	3	0.47	0.00	1617	43	22112	5308.76	1	34	503.19
	4	0.46	0.00	840	60	23984	3532.11	1	40	456.00
	5	0.36	0.00	723	51	25846	5101.09	1	47	420.83
gr120	1	2.07	1.72	3207	80	34790	0419.90	1	19	1610.00
	2	0.89	0.78	697	54	36678	10097.20	1	31	1417.65

3.5.2 PDSMTSP: K vehicles available

In this section, the effectiveness of the solution method proposed for the PDSMTSP is examined.

3.5.2.1 Problem instances

Not being aware of instances in the literature for the PDSMTSP, we selected 20 instances from CVRPLIB[2] (a repository of instances for the Capacited Vehicle Routing Problem). The selected instances where choosen with the depot location being near the center of all customers. The instances size varies from 50 customers up to 199 customers. A detailed

[2]Capacited Vehicle Routing Problem LIBrary

description of the selected instances is given hereafter :

- 5 instances from *Christofides, Mingozzi and Toth* (1979) [CMT] benchmark :
  - **CMT1** : 50 customers, minimum number of vehicles K = 5;
  - **CMT2** : 75 customers, minimum number of vehicles K = 10;
  - **CMT3** : 100 customers, minimum number of vehicles K = 8;
  - **CMT4** : 150 customers, minimum number of vehicles K = 12;
  - **CMT5** : 199 customers, minimum number of vehicles K = 17.
- 3 instances from Set E (*Christofides and Eilon*,1969) benchmark :
  - **E-n51-k5** : 50 customers, minimum number of vehicles K = 5;
  - **E-n76-k8** : 75 customers, minimum number of vehicles K = 8;
  - **E-n101-k8** : 100 customers, minimum number of vehicles K = 8.
- 2 intances from Set M (*Christofides, Mingozzi and Toth*,1979) benchmark
  - **M-n151-k12** : 150 customers, minimum number of vehicles K = 12;
  - **M-n200-k16** : 199 customers, minimum number of vehicles K = 16;
- 7 instances from Set P (*Augerat,1995*) benchmark
  - **P-n51-k10** : 50 customers, minimum number of vehicles K = 10;
  - **P-n55-k7** : 54 customers, minimum number of vehicles K = 7;
  - **P-n60-k10** : 59 customers, minimum number of vehicles K = 10;
  - **P-n65-k10** : 64 customers, minimum number of vehicles K = 10;
  - **P-n70-k10** : 69 customers, minimum number of vehicles K = 10;
  - **P-n76-k5** : 75 customers, minimum number of vehicles K = 5;
  - **P-n101-k4** : 100 customers, minimum number of vehicles K = 4;
- 3 instances from *Uchoa et al.* (2014) benchmark
  - **X-n110-k13** : 109 customers, minimum number of vehicles K = 13;
  - **X-n115-k10** : 114 customers, minimum number of vehicles K = 10;
  - **X-n139-k10** : 138 customers, minimum number of vehicles K = 10;

In each instance file, customers are represented by their coordinates x and y. For the instances to fit our problem (involving vehicles and drones), we divided the minimum number of vehicles K by 2. Then we considered that the ceiling of the obtained value corresponds to the number of vehicles and the floor corresponds to the number of UAVs. For all instances, we assume that all the customers are drone eligible. We could make this assumption because we are interested in an academic study of the problem. In the experiments, we use the Manhattan distance for the vehicle trip and the Euclidean distance for the drone trip.

3.5.2.2 Results

Solving the MILP directly in CPLEX requires a very long computing time. The calculations were interrupted after a pre-defined time limit of 3 hours (10800 CPU seconds). The detailed results are listed for each instance in Table 3.7. For each instance we mentioned:

Gap1 the percentage gap between the solution cost for the relaxed linear program $\mathbf{z}_{lr}$ at the root (the integrity constraints are relaxed but the SECs are applied) and the cost of the best integer solution $\mathbf{z}_{ub}$ (Gap1 = $100 \times \frac{\mathbf{z}_{ub}-\mathbf{z}_{lr}}{\mathbf{z}_{ub}}$);

Gap2 the percentage gap between the best lower bound $\mathbf{z}_{lb}$ and the best integer solution cost $\mathbf{z}_{ub}$ (Gap2 = $100 \times \frac{\mathbf{z}_{ub}-\mathbf{z}_{lb}}{\mathbf{z}_{ub}}$);

#Nodes the number of nodes explored in the Branch-and-Bound tree (when the number is zero that means the process ended at the root node);

#LC the number of SECs added with LAZYCONSTRAINTCALLBACK (used to search for constraints violated by an integer solution)

#UC the number of SECs added with USERCUTCALLBACK (used to search for constraints violated by a fractional solution)

CPU the Branch-and-Cut running time in seconds;

#Veh. the number of vehicles used in the best integer solution;

#Dro. the number of drones used in the best integer solution;

D.C. the number of customers assigned to the drones in the best integer solution;

C.T. the completion time (ie. the Branch-and-Cut best integer solution cost $\mathbf{z}_{ub}$)

We can tell from the results that for our addressed problem, it is not a good choice to use the mixed integer programming to solve it. Indeed the numerical experiments demonstrate that the Branch-and-Cut fails to find an integer solution for all instances of size $n > 100$ within the time limit. Besides that we can also observe that for none of the test instances (solved within the time limit) was an optimal solution verified. For the 13 instances that could be solved, the root gap (**Gap1**) and the integrality gap (**Gap2**) after the Branch-and-Cut running for 3 hours is huge. This show that we are still far from the optimal solution. It may be very difficult on a regular computer to solve a problem with more than 100 customers in a acceptable running time. As a result, we should ask for help in heuristics.

Table 3.7 Result after solving the MILP with 3 hours limit time

Instance	CPLEX Branch-and-Cut details						Solution details			
	Gap1 (%)	Gap2 (%)	#Nodes	#LC	#UC	CPU (s)	#Veh.	#Dro.	#D.C.	C.T.
CMT1 (50,3,2)	22.79	22.41	20100	51	23726	10668.50	3	2	11	188.00
CMT2 (75,5,5)	97.20	97.20	957	49	20101	10653.80	0	1	75	3630.86
CMT3 (100,4,4)	96.45	96.45	732	44	14515	10650.80	1	4	94	4537.11
E-n51-k5 (50,3,2)	22.79	22.41	20210	50	23302	10668.80	3	2	11	188.00
E-n76-k8 (75,4,4)	95.73	95.73	2954	38	13712	10606.00	0	4	75	2975.51
E-n101-k8 (100,4,4)	96.45	96.45	732	44	14515	10650.40	1	4	94	4537.11
P-n51-k10 (50,5,5)	64.74	64.63	2896	47	26372	10616.40	4	5	13	230.00
P-n55-k7 (54,4,3)	67.24	67.05	5834	94	17239	10651.90	3	3	11	308.00
P-n60-k10 (59,5,5)	65.87	65.73	5842	48	23596	10560.40	5	5	12	246.00
P-n65-k10 (64,5,5)	83.69	83.68	2782	76	19446	10615.60	5	5	14	580.00
P-n70-k10 (69,5,5)	96.86	96.86	2570	44	18956	10644.10	1	5	68	3166.25
P-n76-k5 (75,3,2)	35.26	35.23	2776	70	15555	10620.20	3	2	11	280.00
P-n101-k4 (100,2,2)	93.20	93.19	991	39	14748	10581.50	1	2	99	4725.47

3.6 Conclusion

In this Chapter, we have provided Branch-and-Cut algorithms to solve the MILP formulations proposed for the PDSTSP and the PDSMTSP. We have implemented two subtour elimination constraints separation algorithms to identify vilotated SECs in integer and fractional solution respectively. The MILPs where solved with the well known and most widely used large-scale optimization solver IBM ILOG CPLEX. To evaluate the proposed exact approach, we conducted experiments on different sets of instances. For the PDSTSP, we used Murray and Chu‘s [MC15] benchmark instances which are instances of 10 and 20 customers. To evaluate the MILP solver on larger instances, we generated another set of instances by using TSPLIB sample instances for the TSP. Regarding the PDSMTSP, experiments where carried on 20 instances selected from CVRPLIB.

To summarize the results, we have to register that the exact solution method Branch-and-Cut, which we applied, is not suitable for solving large-sized instances and cannot even provide solutions of low quality in acceptable time for these instances (especially for solving the PDSMTSP). However, CPLEX gave very good performances regarding tests made for the PDSTSP. In the other hand, we could observe very high CPLEX optimality gaps when

solving the PDSMTSP. Nevertheless, this behaviour of CPLEX was expected due to the complexity (huge number of variables an constraints in the MILP) of the problem considering several vehicles.

To tackle large-sized instances and get good solutions in a raisonnable amount of time, heuristics might be a better choice. Thus, in the next Chapters, we propose heuristic solution for both PDSTSP and PDSMTSP and we analyse their performances.

Chapter 4

An iterative two-step heuristic for the parallel drone scheduling traveling salesman problem

Contents

4.1 Introduction

In this chapter we present a heuristic solution approach for the PDSTSP in which a single vehicle is used in parallel with one or several UAVs for package deliveries. We propose an iterative two-step heuristic. The procedure starts by building a TSP tour τ (also called giant tour) visiting all the customers. This tour is then decomposed into a subsequence $\tau_{vehicle}$ and a complementary subset τ_{drones}, which respectively give a tour for the vehicle and a set of customers assigned to the drones. This decomposition is performed by dynamic programming, with a labeling mechanism. Vehicle tour $\tau_{vehicle}$ is then reoptimized with the Lin-Kernighan heuristic [Hel00]. The assignment of customers from τ_drones to individual drones is solved as a Parallel Machine Scheduling (PMS) problem with a greedy heuristic. A new giant tour that will be used for the subsequent iteration is reconstructed by inserting in the vehicle tour all the customers served by drones, with a randomized best insertion strategy.

Experiments are conducted on the same instances described in Section 3.5.1.1 and Section 3.5.1.2 of the previous chapter. Results obtained for Murray and Chu's benchmarck instances

are compaired with those obtained with solutions methods provided in their paper [MC15] for the same problem. Moreover, we also compaire the results of the iterative two-step heuristic with the results obtained by the Branch-and-Cut procedure presented in the previous chapter. In addition, with the instances generated from the TSPLIB, we investigate in more details the behavior of our heuristic and analyze the benefits of using drones.

The PDSTSP has been described in section 3.3.1.1(in Chapter 3). Here we just recall the main notation used in the problem statement.

- $G = (N \cup \{0\}, A)$ is a complete directed graph where $N = \{1, \ldots, n\}$ is a customer set and 0 is the depot,
- We dispose of 1 vehicle and M drones,
- $N_d \subset N$ is the set of *drone-eligible* customers;
- t_{ij} is the travel time incurred when the vehicle goes through an arc $(i, j) \in A$
- $\widehat{t}_i$ is the travel time incurred when a drone serves a customer $i \in N_d$ ($\widehat{t}_i$ include back and forth trip times).

The remainder of the chapter is organized as follows: Section 4.2.1 presents the general scheme of the iterative two-step heuristic. The decoding procedure when a single drone is used ($M = 1$) is provided in Section 4.2.2. Section 4.2.3 describes the adaptation of the decomposition scheme to handle several drones ($M > 1$). Experiments conducted on Murray and Chu's benchmark instances and the results obtained are presented and discussed in Section 4.3.1. Section 4.3.2 presents experiments and results related to the instances generated from the TSPLIB. A conclusion for the chapter is given in Section 4.4.

4.2 Iterative two-step heuristic

In the proposed heuristic we consider the PDSTSP as a bilevel problem in which:

- at the first level, customers are *partitioned* between the vehicle and the fleet of drones;
- the second level manages the *routing optimization*, which consists in solving two NP-Hard problems, namely: a TSP for the vehicle and a Parallel Machine Scheduling (PMS) problem for drones.

Our iterative two-step heuristic alternates between these two levels. Given a solution of the PDSTSP, a coding step transforms this solution into a customer sequence. Then, a decoding step decomposes this sequence into a tour for the vehicle and series of trips for the drones. These tours / trips are optimized and the process is repeated. The general solution framework does not depend on the number of drones, but the decoding procedure does. When

$M = 1$, *i.e.*, a single drone is available, the decoding scheme is exact. When $M > 1$, it is heuristically extended.

The following is organised like this: in Section 4.2.1, we describe the general scheme of the heuristic. The decoding procedure is detailed in Section 4.2.2 when $M = 1$. We explain how it is adapted to $M > 1$ in Section 4.2.3. In these sections, we adopt the following notation. A solution S is represented as a vector of $M+1$ customer sequences $(\tau_{vehicle}, \tau_1, \ldots, \tau_M)$, where $\tau_{vehicle}$ indicates the visit order of the customers for the vehicle and τ_1 to τ_M this order for the M drones. We denote $c^1(S)$ the completion time for the vehicle in solution S and $c^2(S)$ the completion time for the fleet of drones. Finally, $c(S) = \max(c^1(S), c^2(S))$ is the solution cost.

4.2.1 General scheme of the iterative two-step heuristic

The general scheme of our heuristic is summarized in Algorithm 4 and is explained hereafter.

Algorithm 4 General scheme of the iterative two-step heuristic

1: $\tau \leftarrow solveTSP()$
2: $bestSol \leftarrow (\tau, \emptyset, \ldots, \emptyset)$
3: **while** $bestSol$ is improved **do**
4: $\quad(\tau_{vehicle}, \tau_{drones}) \leftarrow split(\tau)$
5: $\quad\tau^{opt}_{vehicle} \leftarrow reoptimizeTSP(\tau_{vehicle})$
6: $\quad(\tau^{opt}_1, \ldots, \tau^{opt}_M) \leftarrow optimizePMS(\tau_{drones})$
7: $\quad$**if** solution $(\tau^{opt}_{vehicle}, \tau^{opt}_1, \ldots, \tau^{opt}_M)$ is better than $bestSol$ **then**
8: $\quad\quad bestSol \leftarrow (\tau^{opt}_{vehicle}, \tau^{opt}_1, \ldots, \tau^{opt}_M)$
9: $\quad\quad\tau \leftarrow bestInsertion(\tau^{opt}_{vehicle}, \tau^{opt}_1, \ldots, \tau^{opt}_M)$
10: $\quad$**end if**
11: **end while**

In lines 1 and 2, we build a giant TSP tour τ visiting the depot and all the customers. For that matter, we apply a nearest-neighbor construction procedure. The TSP tour provides a starting solution $(\tau, \emptyset, \ldots, \emptyset)$ for our algorithm, where all the customers are visited by the vehicle in the order of sequence τ and no customer is assigned to the drones.

The main part of the algorithm is given by lines 4 to 9 that are repeated as long as they enable finding better solutions. A solution S is considered to be better than another solution S' if one of the two following conditions hold:

- $c(S) < c(S')$
- or $c(S) = c(S')$ and $\min(c^1(S), c^2(S)) < \min(c^1(S'), c^2(S'))$

Decoding procedure *split* on Line 4 decomposes τ in two complementary subsequences: one assigned to the vehicle ($\tau_{vehicle}$) and one assigned to the fleet of drones (τ_{drones}). This pro-

cedure, which is central in the algorithm, is detailed below. Vehicle route $\tau_{vehicle}$ is then reoptimized, with Helsgaun's implementation of Lin-Kernighan heuristic [Hel00] (Line 5) and a PMS solution is obtained from τ_{drones} (see Section 4.2.3). The new solution $(\tau^{opt}_{vehicle}, \tau^{opt}_1, \ldots, \tau^{opt}_M)$ is compared with *bestSol* and the latter is updated if needed. Finally, in this case, all the customers from $\tau^{opt}_1, \ldots, \tau^{opt}_M$ are reinserted in $\tau^{opt}_{vehicle}$, in this order and with a best insertion policy (Line 9). Otherwise the algorithm stops.

Note that when $M = 1$, the PMS problem is trivial, hence one can replace Line 6 with $\tau^{opt}_1 \leftarrow \tau_{drones}$.

4.2.2 Decoding procedure when $M = 1$

We first consider the case $M = 1$. Procedure $split(\tau)$ computes the best partition of the customer set between the vehicle and the unique drone with the constraint that the customers in the vehicle tour follows the same order as in sequence τ. This problem is modeled as a bicriteria shortest path problem [CM82] in an acyclic directed graph and is solved by dynamic programming.

Definition of the acyclic graph

We introduce $G^\tau = (V^\tau, A^\tau)$ an acyclic graph defined as follows. $V^\tau = \{0, 1, 2, \ldots, n, n+1\}$ represents the set of customers completed by two copies of the depot: the original depot 0 and its copy $n + 1$. In the bicreteria shortest path problem, 0 will be the origin and $n + 1$ the destination. Given i and j in V^τ, with $i \neq j$, arc (i, j) exists in A^τ if and only if the two following conditions are both satisfied:

- $i = 0$, or $j = n + 1$, or i is before j in τ (in the three cases we note $i <_\tau j$)
- all the customers between i and j in τ are drone-eligible: $i <_\tau k <_\tau j \Rightarrow k \in N_d$

With every arc $(i, j) \in A^\tau$, we associate a cost vector (c^1_{ij}, c^2_{ij}). The first cost component c^1_{ij} represents the cost incurred for the vehicle if it travels directly from i to j: $c^1_{ij} = d_{ij}$. The second cost component c^2_{ij} represents the corresponding cost induced for the drone. If the vehicle travels directly from i to j, all customers k in-between (which by definition of A^τ are all drone-eligible) are assigned to the drone: $c^2_{ij} = \sum_{\{k \in V^\tau:\ i <_\tau k <_\tau j\}} \widehat{d}_k$.

Figure 4.1 shows an illustrative example for an instance with 5 customers. Customers 2 and 4 are not drone-eligible. The vehicle travel cost matrix is reported in Table 4.1(a) in which we add the copy of the depot (node 6). Table 4.1(b) presents the drone-trip costs for drone-eligible customers. We assume $\tau = (1, 2, 3, 4, 5)$.

The set of paths from 0 to $n + 1$ in G^τ exactly matches the set of acceptable routes for the vehicle that can be extracted from sequence τ. Furthermore, given a path P, the cost

	0	**1**	**2**	**3**	**4**	**5**	**6**
0	0	8	8	11	6	8	0
1		0	10	7	10	12	8
2			0	13	8	6	8
3				0	11	7	11
4					0	5	6
5						0	8
6							0

(a) Vehicle cost matrix d_{ij}

Customer	1	3	5
Drone cost	16	12	20

(b) Drone cost vector $\widehat{d}_i$

Table 4.1 Illustrative instance

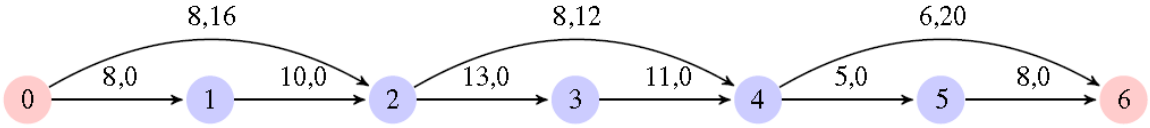


Figure 4.1 Graph G^τ for the instance of tables 4.1(a) and 4.1(b), with $\tau = (1, 2, 3, 4, 5)$.

vector $(c^1(P), c^2(P)) = (\sum_{(i,j)\in P} c^1_{ij}, \sum_{(i,j)\in P} c^2_{ij})$ corresponds to the costs of the vehicle and the drone, if the route of the vehicle follows P and the remaining customers are assigned to the drone. Taking again the example depicted with Figure 4.1, path $P = (0, 2, 4, 5, 6)$ should be interpreted as a solution S with a vehicle tour $(0, 2, 4, 5, 6)$ and customers 1 and 3 assigned to the drone; $c^1(P) = 29$, $c^2(P) = 28$, $c(S) = \max(29, 28) = 29$.

For that reason, the aim of the *split* procedure is to find the path P from 0 to $n+1$ in G^τ that minimizes $\max(c^1(P), c^2(P))$. We now describe the dynamic programming procedure developed to compute this path.

Dynamic programming scheme.

To understand the procedure applied to find that shortest path, let us first consider the following definitions:

- **partial path**: a path going from 0 (origin depot) to any node i of graph G^τ;
- **label**: the cost vector (c^1, c^2) of a partial path calculated by adding the costs of the crossed arcs.

The shortest path is found by applying the procedure described in Algorithm 5. The principle is to progressively associate a list of labels $\mathcal{L}(i)$ with each node i of G^τ. At the

initialization, $\mathcal{L}(i)$ is set to $\emptyset$ for every node i (Line 1). The procedure then starts by assigning label $(0, 0)$ to $\mathcal{L}(0)$ (Line 2).

The procedure goes through the graph from the origin depot node 0 to the destination depot node $n + 1$ by following the order defined in sequence τ. Function *increment* on Line 5 is introduced for this purpose. For a given vertex i, list $\mathcal{L}(i)$ is constructed by adding a new label for every label in the list of labels of a predecessor node (Lines 6-13). The new label is simply defined by adding the arc cost (Line 8). In order to limit the number of labels generated in the lists as the procedure advance in the graph, some bounding mechanisms (Line 9) and a dominance rule (Line 10) are introduced. These two components are detailed below.

At the end, the shortest path is retrieved through a backtrack mechanism considering the best label found in $\mathcal{L}(n + 1)$. This shortest path constitutes the vehicle tour $\tau_{vehicle}$ and the nodes out of the path are assigned to the drone.

Algorithm 5 Procedure $split(\tau)$

1: $\mathcal{L}(i) \leftarrow \emptyset$ for $0 \leq i \leq n + 1$
2: $\mathcal{L}(0) \leftarrow \{(0, 0)\}$
3: $i \leftarrow 0$
4: **while** $i <_\tau n + 1$ **do**
5: increment i
6: **for all** j such that arc $(j, i) \in A^\tau$ **do**
7: **for all** label $L \in \mathcal{L}(j)$ **do**
8: $L' \leftarrow (c^1(L) + c^1_{ji}, c^2(L) + c^2_{ji})$
9: **if** L' is not pruned with the bounding mechanisms **then**
10: $\mathcal{L}(i) \leftarrow addWithDominance(\mathcal{L}(i), L')$
11: **end if**
12: **end for**
13: **end for**
14: **end while**
15: $bestLabel \leftarrow$ label L in $\mathcal{L}(n + 1)$ with minimum completion time $\max(c^1(L), c^2(L))$
16: $\tau_{vehicle} \leftarrow$ path that led to label $bestLabel$
17: $\tau_{drones} \leftarrow$ nodes that are not in the path

Dominance rule.

The dominance rule is very simple. A label $L_a = (c^1_a, c^2_a)$ dominates a label $L_b = (c^1_b, c^2_b)$ if $(c^1_a < c^1_b$ and $c^2_a \leq c^2_b)$ or $(c^1_a \leq c^1_b$ and $c^2_a < c^2_b)$. Before adding a new label to a list of label, dominance tests are tried with all the labels in the list, both to check whether the new label should be discarded or if an existing label should be removed.

The dominance rule guarantees that the number of labels attached to a node is limited by the number of values that can take label components c^1 or c^2. As this number can still be very large, bounding mechanisms are helpful to limit the combinatorial explosion.

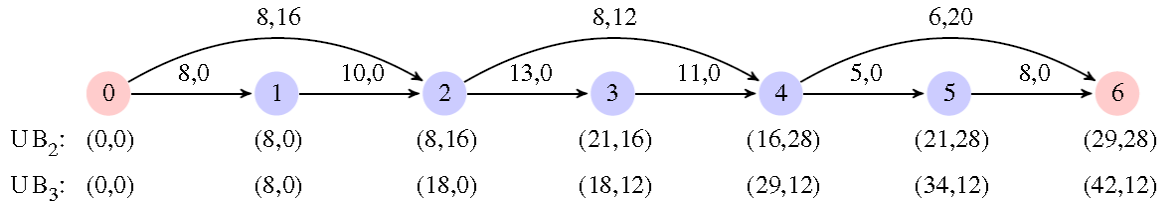


Figure 4.2 Upper bounds for the graph of Figure 4.1.

Upper bound generation.

The bounding mechanisms rely on the computation of several upper bounds. A first upper bound UB_1 is given by the value of the best solution found so far. A second upper bound UB_2 is obtained by applying Algorithm 5 with the following modifications:

- the bounding mechanisms of Line 9 are not considered;
- the dominance rule is changed to: L_a dominates L_b if $c(L_a) \leq c(L_b)$; with this new rule, a single label is kept at each node of the graph.

The third upper bound UB_3 results from the following greedy heuristic. The nodes are considered in the order defined by $<_\tau$. Nodes are added to the vehicle tour $\tau_{vehicle}$ except when the two following conditions hold: node i is drone-eligible and adding i to the drone does not increase the completion time of the current partial solution. In this case, node i is assigned to the drone. UB is set as the best (minimal) upper bound among these three bounds: $UB = \min(UB_1, UB_2, UB_3)$.

Figure 4.2 represents the application of these algorithms to compute UB_2 and UB_3 on the example introduced with Figure 4.1. Assuming that no previous solution is known ($UB_1 = +\infty$), we obtain $UB_2 = 29$, $UB_3 = 42$ and $UB = 29$.

Lower bound generation.

We introduce two lower bounds, $LB^{tot}(i)$ and $LB^{veh}(i)$, at each customer node of the graph. Lower bound $LB^{tot}(i)$ indicates the minimal total cost $c^1(P)+c^2(P)$ of any path P between i and $n+1$ in G^τ. Lower bound $LB^{veh}(i)$ indicates the minimal vehicle cost $c^1(P)$ of any path P between i and $n+1$ in G^τ. Both bounds are computed simultaneously with a backward exploration of the acyclic (topologically-ordered) graph G^τ.

Figure 4.3 details these lower bounds on the example introduced with Figure 4.1.

Bound mechanisms.

We define the three following bounding rules:

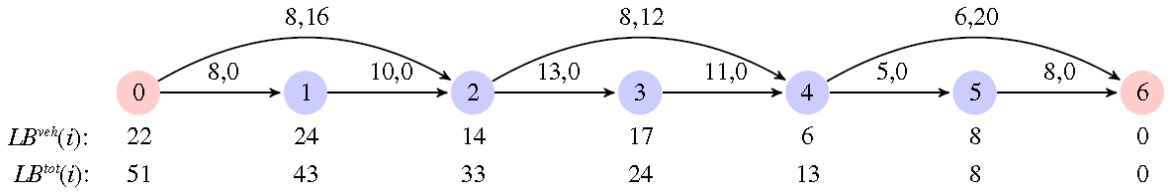


Figure 4.3 Lower bounds for the graph of Figure 4.1.

- **R1**: prune label L if $c^2(L) \geq UB$
- **R2**: prune label L if $c^1(L) + LB^{veh}(i) \geq UB$
- **R3**: prune label L if $c^1(L) + c^2(L) + LB^{tot}(i) \geq 2 \times UB$

Rule R1 identifies labels whose completion time is already at least UB, because of the customers already assigned to the drone. Rule R2 identifies labels whose vehicle tour cannot finish before UB, because of the subtour already assigned to the vehicle and the minimal remaining path to $n+1$. Rule R3 considers both the vehicle and drone completion times. It is based on the fact that whatever the path P extending the subpath associated with L to $n+1$, the completion time of the resulting solution is $\max(c^1(L) + c^1(P), c^2(L) + c^2(P)) \geq \frac{c^1(L)+c^1(P)+c^2(L)+c^2(P)}{2} \geq \frac{c^1(L)+c^2(L)+LB^{tot}(i)}{2}$.

Hence, none of the labels for which R1, R2 or R3 holds, can result in a solution better than UB. They can be pruned.

Figure 4.4 illustrates the *split* subroutine on the graph presented in Figure 4.1. List $\mathcal{L}(i)$ is reported under every node i. In Figure 4.4(a), the bounding mechanisms are not applied. Labels can only be removed by dominance. Dominated labels are crossed out. In Figure 4.4(b), the bounding mechanisms are reinserted. The bounding rule(s) enabling to delete a label is reported on the right of the strikethrough label.

In both cases, the optimal path and the associated labels are represented in red. The optimal decomposition is to assign customer sequence $(2, 4, 5)$ to the vehicle and customers $(1, 3)$ to the drone. The cost of this solution is 29.

4.2.3 Decoding procedure when $M > 1$

When $M > 1$, the preceding decomposition scheme could naturally be adapted by defining labels of size $M + 1$, where the first field represents the vehicle cost and the following M fields represent the cost for each drone. This would however induce a more complex labeling algorithm and, above all, many more labels, with probably intractable computing times except for fairly small instances. We preferred to handle this situation with a different approach.

We reproduce exactly Algorithm 5 with the sole modification that arc costs c^2_{ij} are divided by M. This change simulates the fact that the total drone delivery time can be shared equally

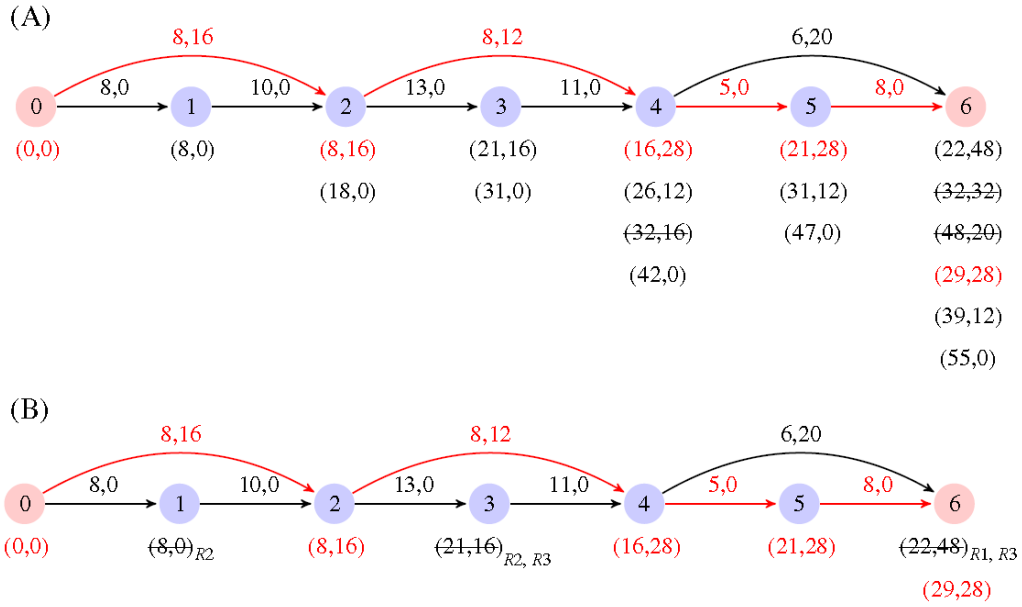


Figure 4.4 Illustration of the *split* procedure on the graph of Figure 4.1, without or with the bounding mechanisms.

between the M drones. In other words, it also means that the PMS problem is relaxed, allowing both preemption of tasks and parallel execution of these tasks.

Sequence τ_{drones} returned by procedure *split* is then transformed into a feasible PMS solution $(\tau_1^{opt}, \ldots, \tau_M^{opt})$ with a greedy heuristic. We propose to apply the well known Longest Processing Time (LPT) heuristic [Wei95]. In LPT, the tasks (drone deliveries) are first sorted in the decreasing order of their processing time $(\widehat{d}_i)$. Then, they are successively assigned to the drone with the minimal current completion time.

4.3 Experiments and results

We conducted two sets of experiments. In the first set, we evaluate the efficiency of our heuristic on Murray and Chu's [MC15] benchmark instances and present some further analyzes. As far as we know, these instances are the only ones existing for this problem. Unfortunately, they are of small size (10 to 20 customers) and it is difficult to draw definitive conclusions with them. For this reason, we introduced instances of larger size on which we conducted a second set of experiments. In this second set, we investigate in more details the behavior of our heuristic and analyze the benefits of using drones.

The environment used for the computational work is Intel core(TM) i5-6200U CPU @ 2.30Ghz 2.40Ghz; 8GB RAM; Windows 10; 64 bits. C++ language is used for the implementation part.

For the experiments, we considered two different ways of applying our two-step heuristic:

1. **Single-start two-stepH**: the algorithm is executed exactly as it is described in Section 4.2.

2. **Multi-start two-stepH**: the algorithm is repeated several times, with a randomized initialization and until a time limit is reached. Randomization is introduced in the nearest neighbor heuristic, by randomly selecting one of the K closest neighbors instead of the closest. The solution returned is the best solution among all the solutions found. In the experiments, we arbitrarily set $K = 3$. The computing time limit depends on the experiments and is indicated when needed. The algorithm is not stopped until the split procedure is completed, even if the limit time is exceeded. Thus, this can lead to computation times exceeding the time limit in some cases.

4.3.1 First set of experiments: Murray and Chu's instances

Murray and Chu [MC15]'s instances are described in section 3.5.1.1 (in Chapter 3).

Before analyzing the performance of our heuristic, let us recall the principle of Murray and Chu [MC15]'s PDSTSP heuristic. Their idea is to first assume that the drones will serve all the drone-eligible customers and that all remaining customers will be delivered by the vehicle. Based on this partition, a TSP and a PMS are solved for the vehicle and the drones, respectively. An improvement step reassigns individual customers to either a drone or the vehicle if it allows decreasing the overall completion time. This improvement step is implemented as follows. If the completion time for the drones exceeds the duration of the vehicle tour, a customer is moved from a drone to the vehicle. The move affording the greatest net savings is chosen. If no move yields a savings in the overall completion time a swap is investigated. The swap operator explores all pairwise exchanges of customers between the drones and the vehicle. If the completion time for the vehicle determines the overall completion time, the swap operator is again employed. The process of reallocating customers between the drones and the vehicle is repeated as long as improvements are carried out. Three TSP solution methods were evaluated for computing the vehicle tour, namely: integer programming (with the guarantee that the optimal TSP solution is obtained), the savings heuristic [Lys97], and the nearest neighbor heuristic. Similarly, two PMS solution methods were evaluated for the drones: integer programming (exact solution) and the longest processing time (LPT) heuristic [Wei95].

Table 4.2 presents a summary on the performance of our heuristics compared to Murray and Chu's. The first four lines correspond to Murray and Chu's methods. Note that we ignored the less efficient among their methods in this table. The fifth line provides information on the computation of exact solutions by Murray and Chu with their IP formulation. The results obtained with our two-step heuristic are presented for both single-start two-stepH and multi-start two-stepH, with a time limit of 3 seconds for the latter. Columns #Sol give the number of times each method found the best solution among all methods. Columns Gap give

the average and maximal gaps to these best solutions, for each method. Columns Runtime report computing times, in seconds.

Table 4.2 Comparison with Murray and Chu results

Solution approach	10 Customers					20 Customers				
	Gap (%)		Runtime (s)			Gap (%)		Runtime (s)		
	Avg	Max	Avg	Max	#Sol	Avg	Max	Avg	Max	#Sol
IP/IP	0.12	10.13	2.4856	29.97	299	0.31	23.47	495	21510	302
IP/LPT	0.12	10.13	2.3093	28.85	300	0.41	23.47	498	21521	292
Savings/IP	1.57	20.68	0.2373	8.26	209	3.57	18.83	3.721	80.68	87
Savings/LPT	1.58	20.68	0.0003	0.01	209	3.98	18.83	0.008	0.07	80
Exact (IP)			0.3194	2.02	360			77.775	180.00	352
Single-start two-stepH	0.12	8.51	0.1660	0.32	278	0.51	23.47	0.210	0.56	225
Multi-start two-stepH	0.02	4.45	3.2060	3.26	313	0.15	23.47	3.072	3.32	337

Regarding best solutions, Murray and Chu [MC15] indicate that their IP formulation was able to find the optimal solution for all of the 10-customer instances but was not able to solve in the imparted time 87 of the 360 20-customer instances. Column #Sol however shows that for these 87 instances, IP still found the best solution among all tried methods in all cases except 8.

We observe that the single-start version of the two-step heuristic and the Savings/LPT are both extremely fast. In comparable amounts of time, single-start two-stepH is able to achieve far better optimality gaps than Savings/LPT. Regarding the multi-start version, we can notice that with a time limit of only 3 seconds, the two-step heuristic yields very small optimality gaps. The IP/IP and IP/LPT methods are also effective but very slow. In general, this comparison shows that our two-step descent algorithm is able to converge very quickly towards very good solutions, at least for small instances with 10 or 20 customers. Clearly, the multi-start version of the algorithm should be preferred to the single-start version, as it permits to reduce optimality gaps, with a controlled additional amount of time and with a very limited additional effort in the implementation.

Table 4.3 and Table 4.4 show in more details the behavior of Multi-start two-stepH with 1, 2 or 3 drones for instances with 10 and 20 customers, respectively.

We notice that the average completion time value decreases when the number of drones increases. This is a logical expected result but it greatly depends on the number of drone-eligible customers and it is not always very significant. At some points, indeed, all the drone-eligible customers might be assigned to drones; then, if the vehicle tours is longer than

Table 4.3 Multi-start two-stepH (time limit: 3 sec) performance on 10 customer instances

	1 drone	2 drones	3 drones
AVG Gap (%)	0.00	0.05	0.00
MAX Gap (%)	0.00	4.45	0.19
AVG Obj.	114.07	105.09	91.90
#SplitCall	2	2	2
#Start (3 sec)	17	19	8

Table 4.4 Multi-start two-stepH (time limit: 3 sec) performance on 20 customer instances

	1 drone	2 drones	3 drones
AVG Gap (%)	0.20	0.06	0.18
MAX Gap (%)	23.47	2.41	6.33
AVG Obj.	141.90	139.93	139.82
#SplitCall	2	2	2
#Start (3 sec)	15	16	15

drone tours, adding new drones does not help. Results also show that the split procedure is called 2 times on average.

Seeing that our decoding scheme is optimal when $M = 1$ (a single drone) while it is not when $M > 1$, we could have also expected larger gaps for instances with several drones. This is however not clearly observed with these results, probably showing the quality of the approximation introduced in the multiple-drones case.

4.3.2 Second set of experiments: New instances generated from TSPLIB

Similarly to what we did in Section 3.5.1.2 of the previous chapter, here we conduct the same sensitivity analyzes to see the performances of the proposed heuristic compaired to that of CPLEX. We compare solutions obtained while changing the depot position, the percentage of drone-eligible customers, the drone speed factor and the number of drones, respectively. For that purpose, we consider the 6 reference instances and construct new instances by changing a single parameter at a time. The method used for these experiments is Multi-start two-stepH with a time limit set to 5 minutes (a large amount of time is given so as to obtain optimality gaps as small as possible and make the results more reliable).

Results are presented in tables 4.5 to 4.8. In these tables, columns labeled with C.T. indicate the value of the completion time and columns labeled with #D.C. indicate the number

of customers assigned to the drones. Lines labeled "H" and "IP" correspond respectively to the results obtained with the two-step heuristic and the MILP of Section 3.3.1.2 solved with CPLEX.

The first observation to be made is that the completion time found with the two-step heuristic is the same to that of CPLEX for almost all instances. There is even a case where the two-stepH yields a better completion time than CPLEX result (test on instance *gr120* with depot at the center, 80% of drone eligible customers, speed=2, using 1 drone).

Table 4.5 investigates the impact of the depot location. We observe that completion times are higher when the depot is located at the left-bottom corner. Furthermore less customers are assigned to the drone. These observations are consistent with expectations.

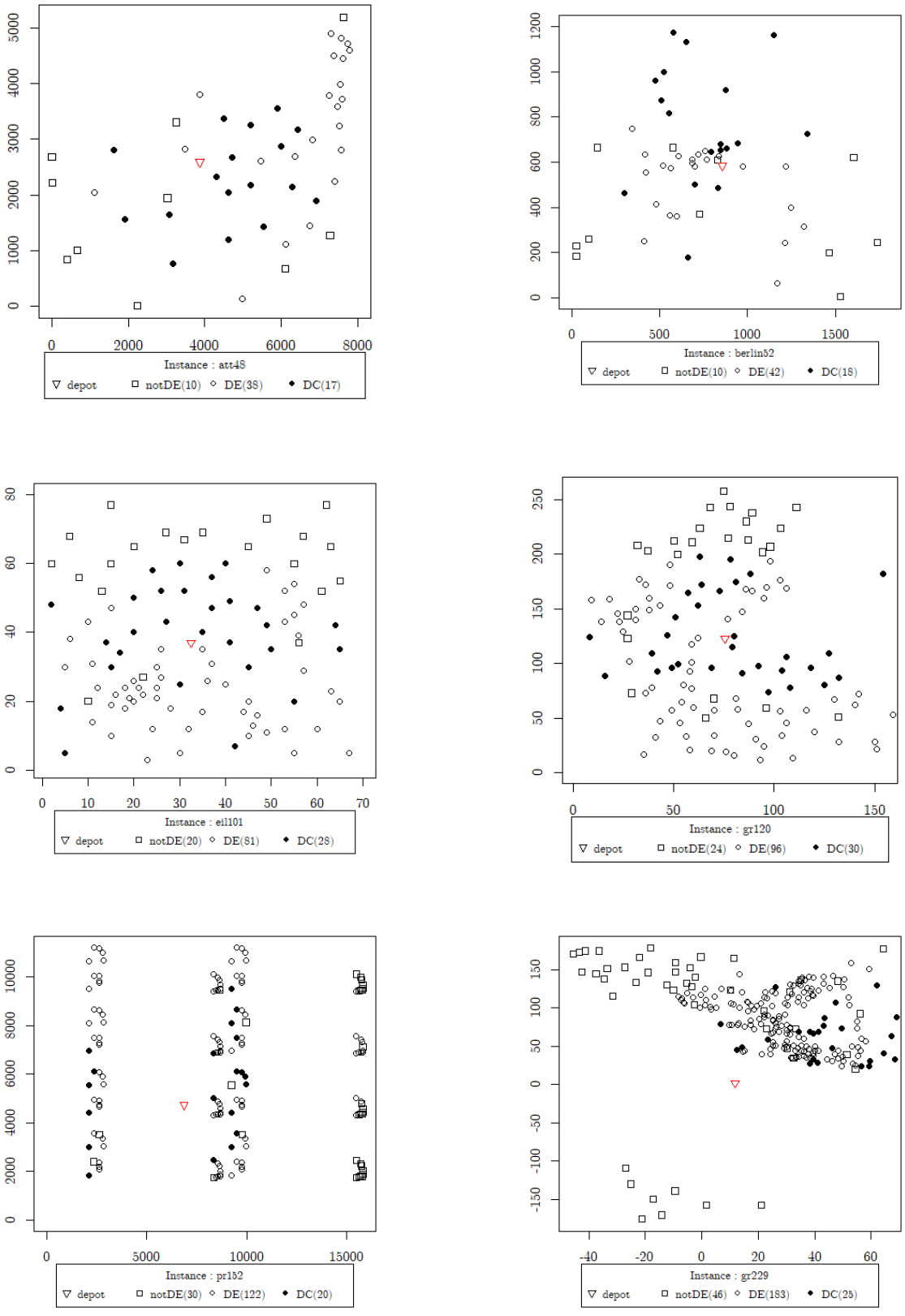
Table 4.5 Impact of depot position (80% of drone-eligible customers, 1 drone, speed factor 2)

Instance		Center		Corner	
		C.T.	#D.C.	C.T.	#D.C.
att48	H	29954	17	33798	9
	IP	29954	17	33798	9
berlin52	H	6386.48	18	7830	8
	IP	6386.48	18	7830	8
eil101	H	564	28	650	19
	IP	564	28	648.98	18
gr120	H	1414	30	1730	18
	IP	1424	31	1730	18
pr152	H	76008	20	76556	19
gr229	H	1794.84	25	1913.74	9

In complement to this table, we also investigate if the drones tend to serve customers near the depot, in other words if there are any “rules of thumb” that dictate which customers should be served by the drones. Figure 4.5 shows the solutions obtained for our 6 reference instances (depot position: center, 80% of drone-eligible customers, drone speed: 2, one drone). The depot is represented by a triangle. Customers that are not drone-eligible are represented by squares and drone-eligible customers are represented by circles. The circle is in plain when the given drone-eligible customer is served by the drone. Images show no clear correlation between the position of a drone-eligible customer and its selection for a delivery by a drone.

Table 4.6 investigates the impact of the percentage of drone-eligible customers. The table shows that, in general, completion times are improved when the percentage of drone-eligible

Figure 4.5 Results of the two-step heuristic for the 6 reference instances



customers increases. However, it does not necessarily imply that the number of customers assigned to the drone increases (except of course for the case with 0 drone-eligible customers). Actually, at first, when a few customers are drone-eligible, they tend to be assigned to the drone to relieves the vehicle. Then, when a good balance in vehicle and drone completion times is reached, two conflicting phenomena are met: adding drone-eligible customers allows a better partition of the customers between the vehicle and the drone, and thus should result in an higher number of drones visited per unit of time; but, it also allows decreasing completion times, and thus should result in a smaller total number of customers assigned to the drone.

Table 4.6 Impact of the percentage of drone-eligible customers (depot at the center, 1 drone, speed factor 2)

Instance		0%		20%		40%		60%		80%		100%	
		C.T.	#D.C.	C.T.	#D.C.	C.T.	#D.C.	C.T.	#D.C.	C.T.	#D.C.	C.T.	#D.C.
att48	H	42136	0	38662	7	31592	15	30788.80	16	29954	17	27784	16
	IP	42136	0	38662	8	31592	17	30788.80	16	29954	17	27784	16
berlin52	H	9675	0	9350	4	8300	15	7410	25	6386.48	18	6192	14
	IP	9675	0	9350	7	8300	17	7410	26	6386.48	18	6192	14
eil101	H	819	0	738	18	646	31	578	27	564	28	561.41	26
	IP	819	0	736	20	-	-	578	27	564	28	560	26
gr120	H	2006	0	1736	18	1624	25	1494	29	1414	30	1414.80	23
	IP	2006	0	-	-	-	-	1494	30	1417.65	31	-	-
pr152	H	86596	0	82504	22	77372	23	76786	20	76008	20	74468	23
	IP	86596	0	-	-	-	-	-	-	-	-	-	-
gr229	H	2020.16	0	1862.76	19	1828.02	22	1807.50	23	1794.84	25	1498.05	11

In Table 4.7 and Table 4.8, we vary the speed and number of drones, respectively. We can notice that when the speed of the drone increases, the completion time is improved and the drone is able to visit more customers. Similar results are obtained when increasing the number of drones. Actually, this is not a surprise seeing that in the algorithm, changing the speed or the number of drones almost has the same effect. For example, the only difference between solving an instance with 2 drones having a speed ratio 2 and an instance with 1 drone of speed ratio 4, concerns the solution of the PMS. However, one should mention that when the number of drones increases the average number of customers assigned to each drone decreases. Finally, in some cases, adding drones does not allow improving the solution, for the reasons already developed in Section 4.3.1 (see att48 and berlin52).

With further experiments, we analyze the behavior of our two-step heuristic general scheme. For these experiments, we only consider the reference instance obtained from *gr229.tsp* and execute Single-start two-stepH. Results are presented in Tables 4.9 and 4.10.

Table 4.9 presents the results obtained while varying the percentage of drone-eligible customers. It provides the value of the completion time, the average number of calls to

Table 4.7 Impact of the drone speed (depot at the center, 80% of drone-eligible customers, 1 drone)

Instance		1		2		3		4		5	
		C.T.	#D.C.	C.T.	#D.C.	C.T.	#D.C.	C.T.	#D.C.	C.T.	#D.C.
att48	H	33234	10	29954	17	29142	21	28686	24	28610	26
	IP	33234	10	29954	17	29142	21	28686	24	28610	28
berlin52	H	7450	13	6386.48	18	5656.56	23	5290.65	31	5190	35
	IP	7450	13	6386.48	18	5656.56	23	5290.65	31	5190	37
eil101	H	650	17	564	28	504	36	456	39	420.83	47
	IP	650	18	564	28	503.19	34	456	40	420.83	47
gr120	H	1592	18	1414	30	1289.27	37	1189.71	43	1112	50
	IP	1610	19	1417.65	31	-	-	-	-	-	-
pr152	H	80164	12	76008	20	72936	31	70412	41	67798	31
gr229	H	1865	13	1794.84	25	1735.16	36	1679.33	48	1642.04	58

Table 4.8 Impact of the number of drones (depot at the center, 80% of drone-eligible customers, speed factor 2)

Instance		1		2		3		4		5	
		C.T.	#D.C.	C.T.	#D.C.	C.T.	#D.C.	C.T.	#D.C.	C.T.	#D.C.
att48	H	29954	17	28686	24	28610	26	28610	26	28610	27
	IP	29954	17	28686	24	28610	31	28610	31	28610	34
berlin52	H	6386.48	18	5299.81	31	5190	35	5190	35	5190	35
	IP	6386.48	18	5290.91	31	5190	36	5190	38	5190	36
eil101	H	564	28	456	40	395	52	346.68	59	319.74	69
	IP	564	28	456	40	400.09	51	346	59	-	-
gr120	H	1414	30	1188.51	43	1044.65	54	946.04	65	880	73
	IP	1417.65	31	-	-	-	-	-	-	-	-
pr152	H	76008	20	70244	42	65062.10	41	60027.40	56	56336.10	60
gr229	H	1794.84	25	1686.75	50	1603.90	66	1518.62	84	1483.68	98

the split procedure, the average number of labels generated in the split procedure (summed over all nodes and including labels eventually deleted by dominance or with the bounding mechanisms) and the percentage of labels deleted with the bounding mechanisms.

Table 4.9 Impact of the percentage of drone-eligible customers on the behavior of Single-start two-sepH (depot at the center, 1 drone, speed factor 2)

Drone-eligible	C.T.	#SplitCall	#LabelsGenerated.	LabelsDeleted (%)
0%	2020.16	0	0	0
20%	1867.60	3	9450	28.26
40%	1826.22	6	32969	53.10
60%	1812.84	6	139849	76.85
80%	1809.84	4	385309	88.43
100%	1500.45	4	3662748	99.05

Several interesting conclusions can be drawn from this table. First, one can observe an increase in the number of calls to the split procedure following that of the number of drone-eligible customers. It indicates that, thanks to the flexibility provided by the new drone-eligible customers, the method gets less easily trapped into a local optimum. However, the method still converges quickly (a few iterations) and towards good-quality solutions: the gaps with the solutions found in 5 minutes with Multi-start two-stepH (last line of Table 4.6) remain limited. Second, one can see a quick increase in the number of generated labels. It can clearly be explained by the shape of graph G^{τ}. Having more drone-eligible customers favors longer arcs and result in more arcs in the graph, and thus more possibilities for extending labels. Fortunately, bounding mechanisms are capable of getting rid of most of these labels: more than 99% when all the customers are drone-eligible.

Table 4.10 investigates the quality of the approximation made in the split procedure when $M > 1$. It shows the average and maximum gaps between the drone completion time found by the split procedure (assuming a single drone with a speed multiplied by M) and the completion time obtained after having applied the LPT heuristic. We observe that the gaps grow when the number of drones increases, but remain very limited.

Table 4.10 Average gap between the drone completion time found by the split procedure and the completion time found with the LPT heuristic

Number of drones	1	2	3	4	5
Avg Gap (%)	0	0.01	0.73	0.85	0.99
Max Gap (%)	0	0.01	1.41	1.35	1.86

Finally, with Table 4.11, we provide a benchmark set of best known solutions on our new instances, for future researches. In this table, we consider our 6 references instances with a

drone fleet size going from 1 to 5 drones. It gives rise to a new instance set of 30 instances. The results provided in the table were obtained with a single execution of Multi-start two-stepH with a time limit of 5 minutes.

Table 4.11 Benchmark based on reference instances with 1 to 5 drones applying Multi-start two-setpH with a time limit of 5 mins

	1	2	3	4	5
att48	29954	28686	28610	28610	28610
berlin52	6386.48	5299.81	5190	5190	5190
eil101	564	456	395	346.68	319.74
gr120	1414	1188.51	1044.65	946.04	880
pr152	76008	70244	65062.10	60027.40	56336.10
gr229	1794.84	1686.75	1603.90	1518.62	1483.68

4.4 Conclusion

In this chapter, we studied heuristic approach to solve the PDSTSP which is the combination of a single vehicle and drones for parcel delivery in an approach without synchronization between the vehicle and drones. We propose an iterative two-step heuristic, composed of: a coding step that transforms a solution into a customer sequence, and a decoding step (split procedure with labeling mechanism) that decomposes the customer sequence into a tour for the vehicle and trips for the drones. Decoding is expressed as a bicriteria shortest path problem and is carried out by dynamic programming. To limit the number of labels generated in the procedure, some bounding mechanisms and a dominance rule were introduced. Experiments were carried by considering two different ways of applying our two-step heuristic. A single-start version and a multi-start version where the algorithm is repeated several times, with a randomized initialization and until a time limit is reached.

Experimental results show that the proposed two-step heuristic is close to CPLEX in terms of average gap (for most instances, the heuristic succeeds to find the optimal solution), while being much faster. Regarding experiments carried out on Murray and Chu's benchmark instances, the two-step heuristic was able to achieve far better optimality gaps than the solution method proposed in Murray and Chu's paper [MC15] to solve the PDSTSP. The sensitivity analysis performed on larger instances generated from the TSPLIB allowed us to investigate the impact of several parameters of the problem including the position of the depot (center or corner), the percentage of drone eligible customers, the number of drones and the speed factor of drones.

Nevertheless, there are still a lot of possible improvements. First, we mainly focused in this study on the split procedure. Clearly the quality of the results could be slightly

improved by solving better the PMS that are repeatedly solved in the solution scheme. In the current implementation, the algorithms used for solving these problems are very simple. They have the advantage to be very quick, but it is certainly possible to gain in effectiveness with a limited increase in computing times. Most importantly, the iterative algorithm could certainly beneficially be inserted within a meta-heuristic scheme. The multi-start two-step heuristic can already be interpreted as a GRASP, but more promising meta-heuristic schemes exist. Iterated Local search, or the hybridization with evolutionary algorithms, would for example be natural candidates.

The content of this chapter is the subject of a paper published in a special issue of the international journal Networks [MS+18].

In the next chapter, we focus on designing a heuristic solution for the PDSMTSP where multiple vehicles and multiples drones are used for parcel delivery still in an approach without synchronization between the vehicles and drones.

Chapter 5

A hybrid metaheuristic for the parallel drone scheduling traveling salesman problem with multiple drones and vehicles

Contents

5.1 Introduction

In Chapter 4, we have proposed an original iterative two-step heuristic for the Parallel Drone Scheduling Travelling Salesman Problem (where a single vehicle is used in parallel with one or several UAVs for package deliveries). In this chapter, we study heuristic method for solving the Parallel Drone Scheduling Multiple Traveling Salesman Problem (PDSMTSP) which extends the PDSTSP by considering several vehicles instead of a single vehicle.

The PDSMTSP has been described in section 3.3.2.1 (in Chapter 3). Here is a brief reminder of the main notation used in the problem statement.

- $G = (N \cup \{0\}, A)$ is a complete directed graph where $N = \{1, \ldots, n\}$ is a customer set and 0 is the depot,

- We dispose of K vehicle and M drones,
- $N_d \subset N$ is the set of *drone-eligible* customers;
- t_{ij} is the travel time incurred when the vehicle goes through an arc $(i, j) \in A$
- $\widehat{t}_i$ is the travel time incurred when a drone serves a customer $i \in N_d$ ($\widehat{t}_i$ include back and forth trip times).

We propose a hybrid metaheuristic combining Iterated Local Search and Dynamic Programming to solve the PDSMTSP. This heuristic is inspired from the heuristic developed in the previous chapter for the same problem restricted to a single vehicle. However, considering several vehicles adds more complexity especially in the decomposition step. Decomposing the giant tour τ becomes much harder because a tour is expected for each one of the K vehicles. Several bounding mechanisms are introduced in the decomposition to preserve acceptable computing times to the detriment of the quality of solution obtained. To improve the solution obtained after the decomposition step, we introduce some quick local search operators moving customers between vehicles or between vehicles and drones. Experiments are conducted on 20 instances selected from the CVRPLIB (the instances are decribed in Section 3.5.2.1 of Chapter 3). We investigate how results are impacted by the different components in the hybrid metaheuristic by introducing several variants and compairing them.

The chapter is organized as follows: in Section 5.2.1, the general scheme of the hybrid metaheuristic is provided. Local search moves that will be implemented to improve the solution after the decomposition of the giant tour are described in Section 5.2.2. Section 5.2.3 details the decoding procedure which decomposes the giant tour into a set of vehicles tours (each vehicle tour following the order defined by the giant tour) and a set of customers assigned to drones. Experiments and the results obtained are presented and discussed in Section 5.3. We close the chapter with a conclusion in Section 5.4.

5.2 Hybrid metaheuristic

In this section we describe the solution approach that we proposed. As already explained, it extends a procedure developed in Chapter 4 for the case with a single vehicle (PDSTSP). We quickly recall this procedure below.

The procedure starts by building a TSP tour τ (also called giant tour) visiting all the customers. This tour is then decomposed into a subsequence $\tau_{vehicle}$ and a complementary subset π_{drones}, which respectively give a tour for the vehicle and a set of customers assigned to the drones. This decomposition is performed by dynamic programming, with a labeling mechanism. Vehicle tour $\tau_{vehicle}$ is then reoptimized with the Lin-Kernighan heuristic [Hel00]. The assignment of customers from π_{drones} to individual drones is solved as a Parallel Machine Scheduling (PMS) problem with a greedy heuristic. The next step is to reconstruct a new giant tour that will be used for the subsequent iteration. This is carried out by inserting in the vehicle tour all the customers served by drones, with a randomized best insertion strategy.

The solution method developed for the PDSMTSP basically follows the same idea, but with some important changes. Decomposing the giant tour τ becomes much harder because a tour is expected for each one of the K vehicles. To preserve acceptable computing times, we heavily restrict the possibilities in the decomposition. Roughly speaking, vehicle tours will correspond to successive subsequence of τ: having two subsequences that overlap is not allowed. For that reason, the quality of the decomposition cannot be entirely ensured and we introduce some quick local search operators to intensify the search. The heuristic is coined hybrid metaheuristic.

In Section 5.2.1, we describe in more details the general scheme of the heuristic. The decoding procedure, decomposing the giant tour, is presented in Section 5.2.3.

5.2.1 General scheme of the hybrid metaheuristic

We adopt the following notation. A solution S is represented as a vector of K customer sequences and M sets $(\tau_1, \tau_2, \ldots, \tau_K, \pi_1, \ldots, \pi_M)$, where $\tau_1, \tau_2, \ldots, \tau_K$ indicates the visit order of the customers for the K vehicles and π_1 to π_M the sets of assigned customers for the M drones. We denote $c_k^1(S)$ the completion time for vehicle k in solution S and $c_m^2(S)$ the completion time for drone m. Finally, $c(S) = \max(c_1^1(S), \ldots, c_K^1(S), c_1^2(S), \ldots, c_M^2(S))$ is the solution cost. To illustrate the notation, let us consider the solution depicted in Figure 3.3: the solution vector is $((1, 10, 9), (5, 6, 7, 8, 4), (3), (2))$.

The general scheme of the heuristic is summarized in Algorithm 6 and is explained hereafter. Note that it can be interpreted as an Iterated Local Search [LMS03], except that the metaheuristic explores the space of permutations, and that the local search exploits the space of solutions. The link between these two spaces is managed by the decoding procedure. Seeing the complexity of each iteration of the metaheuristic, we accept new solutions according to a better-walk mechanism, that is, only improving solutions are accepted.

Algorithm 6 General scheme of the hybrid metaheuristic

1: $\tau \leftarrow solveTSP()$
2: $bestSol \leftarrow (\tau_1^{rec} = \tau, \tau_2^{rec} = \emptyset, \ldots, \tau_K^{rec} = \emptyset, \pi_1^{rec} = \emptyset, \ldots, \pi_M^{rec} = \emptyset)$
3: **while** the computing time limit is not reached **do**
4: $\quad (\tau_1, \tau_2, \ldots, \tau_K, \pi_{drones}) \leftarrow split(\tau)$
5: $\quad (\tau_1^{opt}, \tau_2^{opt}, \ldots, \tau_K^{opt}) \leftarrow reoptimizeTSP(\tau_1, \tau_2, \ldots, \tau_K)$
6: $\quad (\pi_1^{opt}, \ldots, \pi_M^{opt}) \leftarrow optimizePMS(\pi_{drones})$
7: $\quad (\tau_1^{imp}, \tau_2^{imp}, \ldots, \tau_K^{imp}, \pi_1^{imp}, \ldots, \pi_M^{imp}) \leftarrow improveSol(\tau_1^{opt}, \tau_2^{opt}, \ldots, \tau_K^{opt}, \pi_1^{opt}, \ldots, \pi_M^{opt})$
8: $\quad$ **if** solution $(\tau_1^{imp}, \tau_2^{imp}, \ldots, \tau_K^{imp}, \pi_1^{imp}, \ldots, \pi_M^{imp})$ is better than $bestSol$ **then**
9: $\quad\quad bestSol \leftarrow (\tau_1^{imp}, \tau_2^{imp}, \ldots, \tau_K^{imp}, \pi_1^{imp}, \ldots, \pi_M^{imp})$
10: $\quad$ **end if**
11: $\quad \tau \leftarrow buildGiantTour(bestSol)$
12: **end while**

Initialization (Lines 1 and 2). We initialize the algorithm with a giant TSP tour τ where

a vehicle is visiting the depot and all the customers. This tour is obtained with a nearest-neighbor construction procedure. It provides a starting solution ($\tau_1 = \tau, \tau_2 = \emptyset, \ldots, \tau_K = \emptyset, \pi_1 = \emptyset, \ldots, \pi_M = \emptyset$), where all the customers are visited by the first vehicle in the order of sequence τ. No customers are assigned to other vehicles nor to the drones.

Decoding (Line 4). Procedure *split* decomposes sequence τ in K complementary subsequences: $\tau_1, \tau_2, \ldots, \tau_K$ for customers assigned to the K vehicles and a set π_{drones} for customers assigned to the fleet of drones. This complex procedure is detailed in Subsection 5.2.3.

Route reoptimization (Line 5). Every vehicle route τ_1 to τ_K is reoptimized with the TSP Lin-Kernighan heuristic, using Helsgaun's implementation [Hel00].

Drone assignement (Line 6). The output of the decoding procedure does not provide a detailed planning for the drones; it only indicates which customers will be served by drones. The assignment of customers to individual drones is a PMS problem and is solved with the well-known longest processing time heuristic [Wei95].

Local search (Line 7). Procedure *improveSol* performs local search moves to improve the current solution. These moves and the local search strategy are described in Subsection 5.2.2.

Update of the current solution (Lines 8 and 10). The new solution $(\tau_1^{imp}, \tau_2^{imp}, \ldots, \tau_K^{imp}, \pi_1^{imp}, \ldots, \pi_M^{imp})$ is compared with *bestSol* and the latter is updated if needed. A solution S is considered to be better than another solution S' if one of the two following conditions hold:

- $c(S) < c(S')$
- $c(S) = c(S')$ and $\sum_{k=1}^{K} c_k^1(S) + \sum_{m=1}^{M} c_m^2(S) < \sum_{k=1}^{K} c_k^1(S') + \sum_{m=1}^{M} c_m^2(S')$

This way, when two solutions have the same objective value, the one that minimizes completion times in general is preferred.

Construction of the new giant tour (Line 11). The improved vehicle tours are first concatenated in a random order. then, customers assigned to drones are randomly inserted. The resulting giant tour is optimized with 2-opt.

5.2.2 Local search moves

Figures 5.1 and 5.2 describe the different types of local search moves that we implemented, respectively between a vehicle and a drone, or between two vehicles:

- *Tranfer move* (figure 5.1(a), figure 5.1(b)): a drone-eligible customer is transferred from a vehicle to a drone or from a drone to a vehicle.

- *Exchange move drone-veh* (figure 5.1(c)): a drone-eligible customer in a vehicle tour and a drone customer are exchanged.

- *Relocate move* (figure 5.2(a)): a customer located at one route is moved to another route.

- *Exchange move veh-veh* (figure 5.2(b)): two customers of two different routes are interchanged between the two routes.

- *Cross move* (figure 5.2(c)): two sequences of customers are exchanged by crossing two edges in two different routes.

The moves are applied until a local optimum is obtained. If the maximal completion time is due to a vehicle, the above order is followed when applying moves. If it is by a drone, the search is limited to the transfer and exchange moves, in this order. A first improvement strategy is employed: as soon as an improving move is found, the solution is updated and the neighborhoods are explored again from the beginning.

In all cases, the search is restricted to neighbor solutions that involve the vehicle or drone with the maximal completion time. Indeed, no improvement is possible otherwise. When a customer is moved from this vehicle/drone, it is first tentatively moved to the vehicles/drones with minimum completion time.

The transfer neighborhood has a size $O(n \times M)$ when the transfer is from a vehicle to a drone, $O(n \times |N_d|)$ otherwise. The size for the exchange move between a drone and a vehicle is also $O(n \times |N_d|)$. Other neighborhoods have a size $O(n^2)$. Every solution in every neighborhood can be evaluated in constant time $O(1)$.

5.2.3 Decoding procedure $split(\tau)$

In this section, we explain how K vehicle tours and a set of customers assigned to drones are extracted from a sequence τ with the *split* procedure. We introduce the following notation. $i <_\tau j$ indicates that i precedes j in τ; $j = pred_\tau(i)$ means that j is the direct predecessor of i in τ; equivalently, $j = succ_\tau(i)$ means that j is the direct successor of i. Given i and j with $i <_\tau j$, $time_\tau(i, j)$ is the path length (in terms of the sum of travel times) from i to j by following arcs in τ; to simplify further notation, $time_\tau(i, i)$ is also introduced and set to 0.

Procedure $split(\tau)$ decomposes sequence τ by assigning each element to one of the vehicles or to the set of drone customers with the following constraints: if $i <_\tau j$ and $i, j \in \tau_k$ then $i <_{\tau_k} j$; if $i <_\tau j$, $i \in \tau_k$ and $j \in \tau_l$ with $k \neq l$, then $k < l$. We perform this decomposition by introducing an acyclic directed graph and solving a multi-criteria shortest path problem [CM82] by dynamic programming.

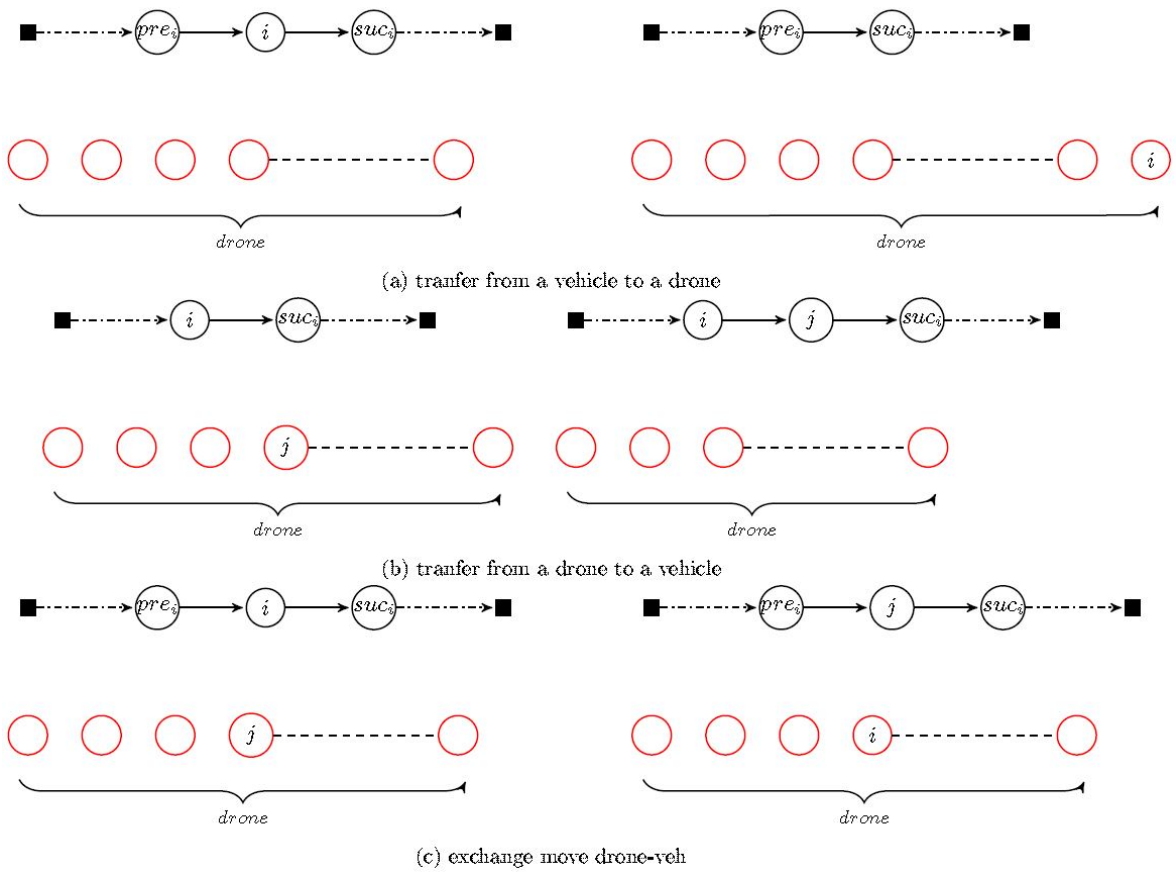


Figure 5.1 Local search operators between a vehicle and a drone

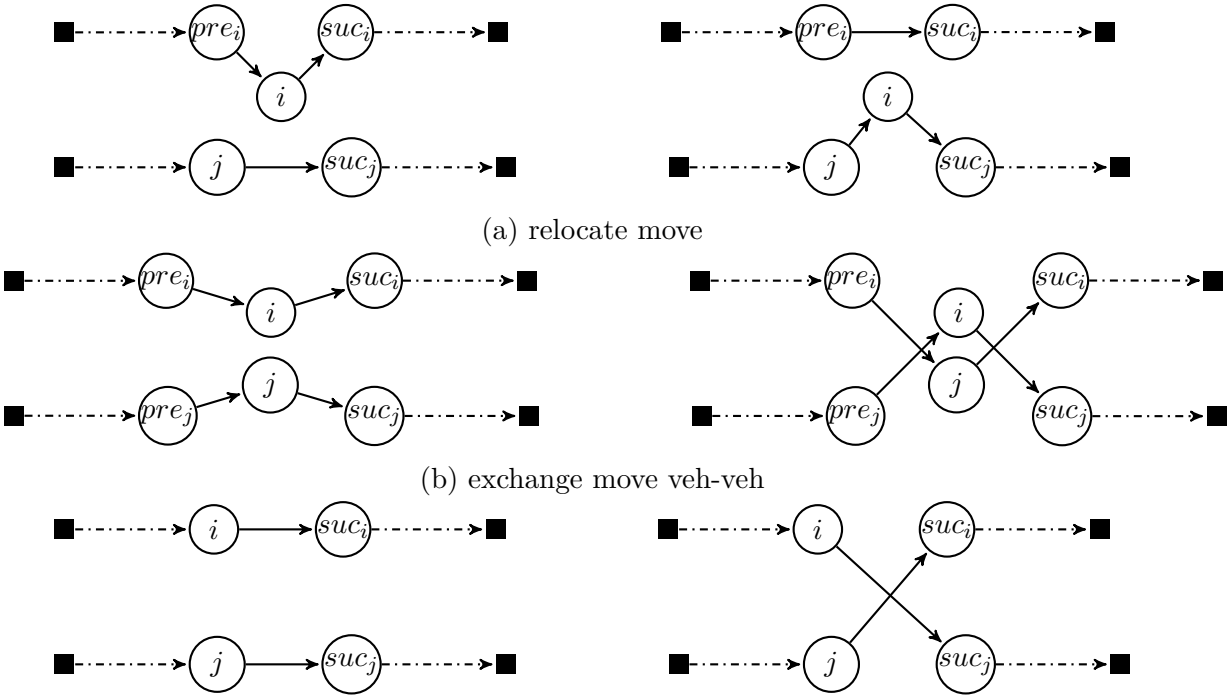


Figure 5.2 Local search operators between two vehicles

Definition of the acyclic graph

We introduce $G^\tau = (V^\tau, A^\tau)$ an acyclic graph defined as follows. $V^\tau = \{0, 1, 2, \ldots, n, n+1\}$ represents the set of customers completed by two copies of the depot: the original depot 0 and its copy $n+1$. In the multi-creteria shortest path problem, 0 will be the origin and $n+1$ the destination. Given i and j in V^τ, with $i \neq j$, arc (i, j) exists in A^τ if and only if the two following conditions are both satisfied:

- $i <_\tau j$
- all the customers between i and j in τ are drone-eligible: $i <_\tau \mu <_\tau j \Rightarrow \mu \in N_d$

With every arc $(i, j) \in A^\tau$, we associate a cost vector (c^1_{ij}, c^2_{ij}). The first cost component c^1_{ij} represents the cost incurred if a vehicle travels directly from i to j: $c^1_{ij} = t_{ij}$. The second cost component c^2_{ij} represents the corresponding cost induced for the drones. If the vehicle travels directly from i to j, all customers μ in-between (which by definition of A^τ are all drone-eligible) are assigned to the drones: $c^2_{ij} = \dfrac{1}{M} \times \displaystyle\sum_{\{\mu \in V^\tau:\ i <_\tau \mu <_\tau j\}} \widehat{t}_\mu$. This value does not exactly gives the contribution of these customers to the completion time of the drones, which could only be obtained by solving a PMS problem, but provides an optimistic (lower-bound) value and a good approximation.

Figure 5.3 shows an illustrative example for an instance with 5 customers. Customers 2 and 4 are not drone-eligible. The vehicle travel cost matrix is reported in Table 5.1(a) in which we add the copy of the depot (node 6). Table 5.1(b) presents the drone-trip costs for drone-eligible customers. We assume $\tau = (1, 2, 3, 4, 5)$ and a single drone.

	0	**1**	**2**	**3**	**4**	**5**	**6**
0	0	8	9	11	6	8	0
1		0	10	7	10	12	8
2			0	13	8	6	8
3				0	11	7	11
4					0	5	6
5						0	8
6							0

(a) Vehicle cost matrix t_{ij}

Customer	1	3	5
Drone cost	16	12	20

(b) Drone cost vector $\widehat{t}_i$

Table 5.1 Illustrative instance

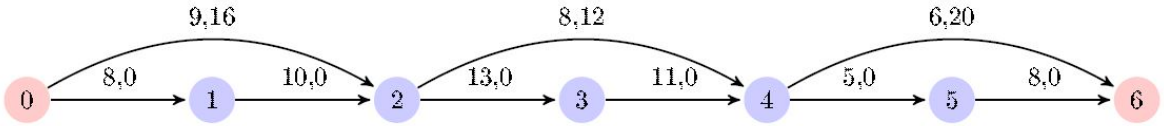


Figure 5.3 Graph G^τ for the instance of tables 4.1(a) and 4.1(b), with $\tau = (1, 2, 3, 4, 5)$.

Dynamic programming scheme

Paths starting from node 0 in graph G^τ can be matched with partial decompositions of sequence τ. The information about this decomposition is captured in a label associated with the path. A label is defined with the following attributes:

- n^{veh} is the index of the current vehicle; the tours (sequences τ_k) of vehicles with index $k < n^{veh}$ are known; the tours of vehicles with index $k > n^{veh}$ are still empty;
- c^{preVeh} is the cost of the longest of the tours that are already known ;
- c^{curVeh} is the cost of the current vehicle tour;
- c^{drone} is the approximated drone cost;
- *pred* is the label from which this label was obtained.

We note $L = (n^{veh}, c^{preVeh}, c^{curVeh}, c^{drone}, pred)$. Equivalently, the atributes of a label L are denoted $n^{veh}(L)$, $c^{preVeh}(L)$ and so on.

The dynamic programming procedure is described in Algorithm 7. The principle is to progressively associate a list of labels $\mathcal{L}(i)$ with each node i of G^τ. At the initialization, $\mathcal{L}(i)$ is set to $\emptyset$ for every node i (Line 1). The procedure then starts by assigning label $(1, 0, 0, 0, \emptyset)$ to $\mathcal{L}(0)$ (Line 2).

The procedure goes through the graph from the origin depot node 0 to the destination depot node $n+1$ by following the order defined in sequence τ. Function *increment* on Line 5 is introduced for this purpose. The list of labels $\mathcal{L}(i)$ of a given node i is obtained by extending labels in the list of labels of its predecessor nodes (Lines 6-19). For any predecessor node j and any label L of $\mathcal{L}(j)$, two possible extensions may occur:

1. A first extension is to add arc (j, i) to the current vehicle tour. The new generated label L' is defined by (Line 8):

$$\begin{cases} n^{veh}(L') = n^{veh}(L) \\ c^{preVeh}(L') = c^{preVeh}(L) \\ c^{curVeh}(L') = c^{curVeh}(L) + c^1_{ji} \\ c^{drone}(L') = c^{drone}(L) + c^2_{ji} \\ pred(L') = L \end{cases}$$

2. A second extension consists in closing the tour of the current vehicle and starting a new vehicle tour with an available vehicle. This extension is not allowed when: $j = 0$, $i = n + 1$ or $n^{veh}(L) = K$; furthermore, the extension is not considered when pursuing with the current vehicle does not increase the current global cost: indeed, it would then be suboptimal to start the tour of a new vehicle (Line 12). When the extension is carried out, the new label L' is given by (Line 13):

$$\begin{cases} n^{veh}(L') = n^{veh}(L) + 1 \\ c^{preVeh}(L') = \max(c^{preVeh}(L), c^{curVeh}(L) + t_{j,0}) \\ c^{curVeh}(L') = t_{0,i} \\ c^{drone}(L') = c^{drone}(L) + c^2_{ji} \\ pred(L') = L \end{cases}$$

In order to limit the number of labels generated in the lists as the procedure advances in the graph, some bounding mechanisms (Lines 9 and 14) and a dominance rule (Lines 10 and 15) are introduced. These two components are detailed below.

At the end, the vehicle tours are retrieved through a backtracking mechanism based on attribute *pred* and starting with the best label found in $\mathcal{L}(n+1)$, *i.e.*, the label that minimizes $\max(c^{preVeh}(L), c^{curVeh}(L), c^{drone}(L))$. Nodes that are not in the vehicle tours are assigned to the drone.

Algorithm 7 Procedure *split*(τ)

$\mathcal{L}(i) \leftarrow \emptyset$ for $0 \leq i \leq n+1$
$\mathcal{L}(0) \leftarrow \{(1, 0, 0, 0, \emptyset)\}$
$i \leftarrow 0$
while $i <_\tau n+1$ **do**
    increment i
    for all j subject to $(j, i) \in A^\tau$ **do**
        for all $L \in \mathcal{L}(j)$ **do**
            $L' \leftarrow$ extend L by adding arc (j, i) to the current tour
            if L' is not pruned by bounding mechanisms **then**
                insert with dominance L' in $\mathcal{L}(i)$
            end if
            if extending L by starting a new vehicle tour should be considered **then**
                $L' \leftarrow$ extend L by adding arc $(j, 0)$ to the current tour and starting a new tour with arc $(0, i)$
                if L' not pruned by bounding mechanisms **then**
                    insert with dominance L' in $\mathcal{L}(i)$
                end if
            end if
        end for
    end for
end while
$L^* \leftarrow$ best label in $\mathcal{L}(n+1)$
derive $\tau_1, \tau_2, \ldots, \tau_K$ by backtracking from L^*
$\pi_{drones} \leftarrow$ nodes not in $\tau_1 \cup \tau_2 \cup \ldots \cup \tau_K$

Insertion with dominance

When a label has to be added to a list of label, dominance tests are tried. These tests are done with all the labels in the list, both to check whether the new label should be discarded or if an existing label should be removed. The dominance rule is very simple. Given a node $i \in \tau$ and two labels L_a and L_b in $\mathcal{L}(i)$, L_a dominates L_b if the following inequalities are all satisfied:

$$\begin{cases} n^{veh}(L_a) \leq n^{veh}(L_b) \\ \max\left(c^{preVeh}(L_a), c^{curVeh}(L_a) + t_{i,0}, c^{drone}(L_a)\right) \leq \max\left(c^{preVeh}(L_b), c^{curVeh}(L_b) + t_{i,0}, c^{drone}(L_b)\right) \\ c^{curVeh}(L_a) \leq c^{curVeh}(L_b) \\ c^{drone}(L_a) \leq c^{drone}(L_b) \end{cases}$$

Upper bound generation

The bounding mechanisms rely on three upper bounds that are computed before starting the dynamic programming algorithm, for a given sequence τ. A first upper bound UB_1 is given by the value of the best solution found so far by the iterative algorithm. The other bounds rely on Algorithm 8. This algorithm computes the minimal cost $\chi(i, k)$ that can be

achieved to serve all customers up to i in the order of sequence τ using k vehicles ($i \in \tau$, $1 \leq k \leq K$). Compared to Algorithm 7 the possibility to serve customers with drones is ignored. Algorithm 8 applies the following recursion:

$$\begin{cases} \chi(i,1) = time_\tau(0,i) + t_{i,0} \quad (i \in \tau), \\ \chi(0,k) = 0 \quad (2 \leq k \leq K), \\ \chi(i,k) = \min\limits_{j <_\tau i} \Big(\max\Big(\chi(j,k-1), t_{0,succ_\tau(j)} + time_\tau(succ_\tau(j),i) + t_{i,0}\Big)\Big) \quad (i \in \tau : i \neq 0, 2 \leq k \leq K). \end{cases} \tag{5.1}$$

Algorithm 8 Procedure *decompose*(τ)

$\chi(i,1) \leftarrow time_\tau(0,i) + t_{i,0}, \ \forall i \in \tau$
$\chi(0,k) \leftarrow 0, \ \forall k = 2..K$
for $k = 2..K$ **do**
  $i \leftarrow 0$
  while $i \neq n+1$ **do**
    $i \leftarrow succ_\tau(i)$
    $\chi(i,k) \leftarrow \min\limits_{j <_\tau i} \Big(\max\Big(\chi(j,k-1), t_{0,succ_\tau(j)} + time_\tau(succ_\tau(j),i) + t_{i,0}\Big)\Big)$
  end while
end for
return χ

The second upper bound UB_2 is then obtained as follows :

1. Determine the *acceleration coefficient* $\alpha = \frac{\chi(n+1,1)}{\chi(n+1,K)}$; this coefficient approximates the gains obtained when K vehicles are used instead of 1;
2. Apply procedure *split*(τ) assuming a single vehicle, changing costs c^1_{ij} to $\frac{t_{ij}}{\alpha}$, desactivating the bounding mechanisms and keeping only the best label on each node;
3. Consider the sequence τ' obtained with this procedure and re-apply Algorithm 8 to decompose it into K vehicles tours $\tau_1, \tau_2, ..., \tau_K$ (these tours can be obtained backwardly from $\chi(n+1,K)$);
4. Solve the PMS problem with the longest processing time heuristic for the remaining customers;
5. Compute the completion time and set UB_2 to this value.

The third upper bound UB_3 is obtained as UB_2 but changing the acceleration coefficient α to K. UB is set as the best (minimal) upper bound among these three bounds: $UB = \min(UB_1, UB_2, UB_3)$.

Lower bound generation

We introduce a lower bound $LB^{tot}(i, L)$ which is defined at each customer node i of the graph. $LB^{tot}(i, L)$ requires to precompute value $SP(i)$ which represents the cost of the shortest path in G^{τ} between i and $n+1$, with modified arc costs set to $c^1_{ij} + M \times c^2_{ij}$. Values $SP(i)$ are computed for all nodes i with a single backward exploration of the acyclic (topologically-ordered) graph G^{τ}. They indicate the minimal total distance that will be traveled by the vehicles and drones between i and $n+1$ in the solution space defined by the $split(\tau)$ procedure. Figure 5.4 reports $SP(i)$ for the example introduced with Figure 5.3.

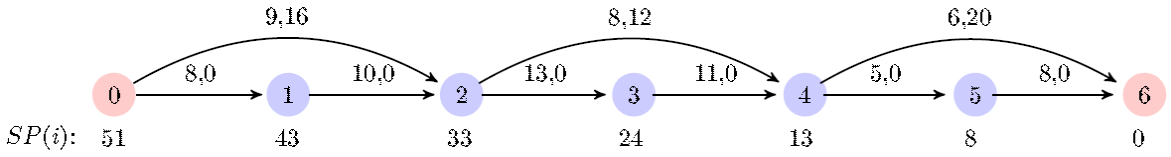


Figure 5.4 Values $SP(i)$ for the graph of Figure 5.3 ($M = 1$).

Then, given $i \in \tau$ and $L \in \mathcal{L}(i)$, the total distance traveled by vehicles $n^{veh}(L)$ to K and by the M drones is at least $c^{curVeh}(L) + M \times c^{drone}(L) + SP(i)$. Hence, at least one of those vehicles or drones cannot complete its duty before time $LB^{tot}(i, L)$ with:

$$LB^{tot}(i, L) = \frac{c^{curVeh}(L) + M \times c^{drone}(L) + SP(i)}{K - n^{veh}(L) + 1 + M}$$

Bounding mechanism

A label $L \in \mathcal{L}(i)$ is pruned if one the following rules applies:

- R1: $\max\left(c^{preVeh}(L), c^{curVeh}(L) + t_{i,0}, c^{drone}(L)\right) \geq UB$;
- R2: $LB^{tot}(i, L) \geq UB$.

Rule R1 identifies labels whose completion time is already at least UB. Rule R2 apply lower bound $LB^{tot}(i, L)$.

Figure 5.5 illustrates the $split(\tau)$ subroutine on the graph presented in Figure 4.1, with $K = 2$ and $M = 1$. List $\mathcal{L}(i)$ is reported under every node i. In Figure 5.5(a), the bounding mechanisms are not applied. Labels can only be removed by dominance. Dominated labels are crossed out. The best label is indicated in red with a '*' symbol beside in list $\mathcal{L}(6)$, as well as labels from which it inherits (and that allow to reconstruct the different vehicle tours). In Figure 5.5(b), the bounding mechanisms are reinserted. The bounding rule(s) enabling to delete a label is also reported at the right of the strikethrough label. After applying the procedures described above for the upper bound computation the value of the upper bound is

$UB = \min(UB1, UB2, UB3) = 27$. In this figure, $\mathcal{L}(6) = \emptyset$ because the upper bound cannot be improved. In both cases, the vehicle tours obtained with the procedure are: $\tau_1 = (0, 1, 2, 0)$ and $\tau_2 = (0, 4, 5, 0)$. Customer 3 is assigned to the drone. The cost of this solution is 27.

5.3 Experiments and results

In this section, the effectiveness of the proposed solution method is examined. The environment used for the computational work is Intel core(TM) i5-6200U CPU @ 2.30Ghz 2.40Ghz; 8GB RAM; Windows 10; 64 bits. Solution methods are implemented in C++ language.

For these experiments, we used instances from CVRPLIB described in Section 3.5.2.1.

5.3.1 Limitation of the number of labels

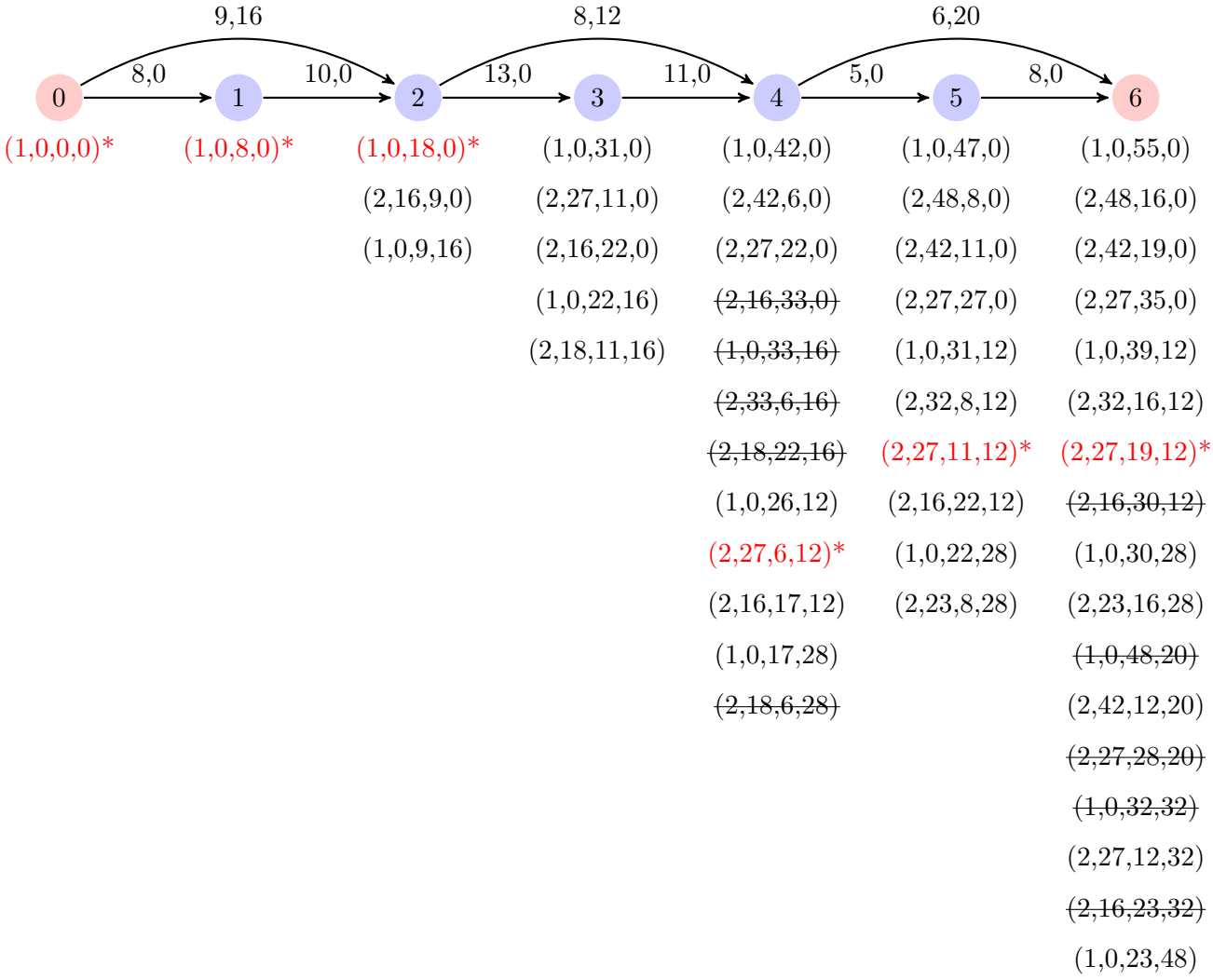
Seeing that the number of generated labels in procedure $split(\tau)$ can grow exponentially according to instance sizes, we proposed a method to limit this number. This method complements bounding mechanisms, previously presented, that might not be enough. The method is based on a threshold $\mathcal{T}$ that will limit the number of labels at a node $x \in \tau$ to at most $K \times \mathcal{T}$. To do so, we first group labels according to their value $n^{veh}(L)$. Then, in each group, we sort labels in the increasing order of value $c^{preVeh}(L) + c^{curVeh}(L) + c^{drone}(L)$ and only keep the $\mathcal{T}$ first labels.

To analyze the efficiency of this method, we conducted a set of experiments by varying parameter $\mathcal{T}$. We only considered the five CMT instances, which represents a sample of instances with different sizes. For each instance, we generated 25 sequence τ with a randomized version of the nearest neighbor heuristic. Then, for each sequence, we applied procedure $split(\tau)$ with different values of $\mathcal{T}$. Table 5.2 presents aggregated results. In the table, we report the average (Avg), maximum (Max) and standard deviation (SD) of the number of labels generated in the procedure, the relative error related to the difference of makespan value without and with limitation of labels and the computing time.

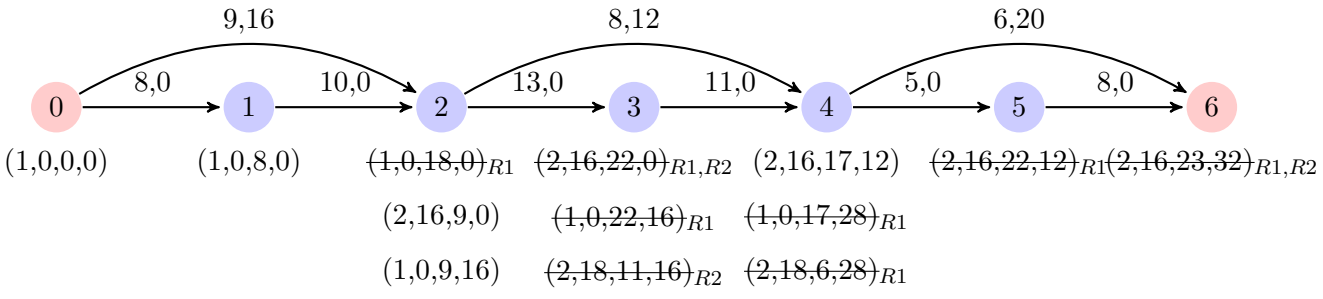
As expected, results show that when $\mathcal{T}$ increases, the number of labels generated in procedure $split(\tau)$ and the computing time increase while the relative error decreases. In view of these results, we considered that the best trade-off between solution quality (low error rate) and execution time was obtained with $\mathcal{T} = 60$. For the remainder of the experiments, when it is indicated that labels are limited, we keep this value.

5.3.2 Results

In this section, we finally evaluate the hybrid metaheuristic, called **HM** hereafter. In order to investigate how results are impacted by the different components in **HM**, we introduce several variants and compare our results to those obtained with these variants:



(a)



(b)

Figure 5.5 Illustration of the *split* procedure on the graph of Figure 5.3, without or with the bounding mechanisms ($K = 2$, $M = 1$).

Instance	$\mathcal{T}$	#LabGen			Error (%)			CPU (s)		
		Avg	Max	SD	Avg	Max	SD	Avg	Max	SD
CMT1 (50,3,2)	10	32592.76	41065	3221.32	11.96	22.61	6.32	0.05	0.06	0.004
	20	58203.16	71635	5515.62	10.44	27.54	6.79	0.06	0.07	0.01
	30	77896.36	96438	7897.76	9.22	27.54	6.32	0.09	0.12	0.02
	40	93623.92	116711	10518.46	7.86	19.40	6.06	0.08	0.10	0.013
	50	105935.48	134491	12916.17	6.99	19.40	5.98	0.09	0.13	0.019
	60	116271.88	148682	14477.09	5.72	19.40	5.62	0.13	0.20	0.03
	70	125979.28	161281	15664.12	4.88	19.40	5.37	0.12	0.16	0.02
	80	135242.32	171434	16660.01	4.71	16.69	5.11	0.13	0.20	0.03
CMT2 (75,5,5)	10	119888.48	145406	12242.51	18.66	29.76	5.06	0.12	0.16	0.01
	20	224724.88	259763	22649.44	13.41	25.28	4.93	0.17	0.22	0.02
	30	317329.20	360712	29324.43	11.03	24.42	4.90	0.28	0.38	0.05
	40	398036.44	442525	34354.79	9.48	20.93	4.50	0.32	0.46	0.06
	50	467528.88	525287	40268.87	8.84	21.40	5.02	0.41	0.59	0.09
	60	529162.64	605701	46719.84	6.96	19.75	4.04	0.66	0.99	0.15
	70	586593.72	676084	51625.91	6.17	17.44	3.70	0.58	0.90	0.13
	80	639260.80	733429	55147.65	5.67	17.44	3.71	0.68	1.15	0.16
CMT3 (100,4,4)	10	170724.80	217266	16096.19	16.43	25.62	4.83	0.19	0.25	0.02
	20	314293	393205	32492.02	14.18	21.49	4.37	0.24	0.32	0.03
	30	448786.28	549272	44498.56	12.80	22.03	4.82	0.42	0.71	0.09
	40	567207.48	690704	58322.19	11.98	19.17	4.43	0.49	0.71	0.10
	50	677780.20	824038	70712.04	11.45	17.95	4.55	0.65	0.96	0.13
	60	779789.40	943959	81315.07	10.21	17.95	4.52	1.02	1.54	0.26
	70	867494.60	1047865	91286.82	9.77	16.95	4.10	0.98	1.52	0.24
	80	947767.12	1142010	102206.65	9.17	16.95	3.98	1.21	2.08	0.31
CMT4 (150,6,6)	10	547245.72	609538	39888.51	20.98	29.52	4.36	0.55	0.67	0.04
	20	1045624.48	1172400	74623.51	19.64	27.11	4.42	0.77	0.88	0.06
	30	1514593.56	1737610	115363.92	18.33	25.71	4.10	1.28	1.60	0.17
	40	1947959.24	2246024	149516.15	17.14	26.04	4.84	1.57	1.96	0.18
	50	2359557.20	2750357	177303.68	16.40	23.81	4.45	2.10	2.67	0.25
	60	2730814.68	3206762	209068.25	15.36	23.81	4.41	2.63	3.30	0.32
	70	3079424.64	3607688	233764.46	15.12	22.97	4.17	3.28	4.15	0.49
	80	3396502.48	3996079	258893.28	13.83	20.95	3.94	3.73	4.74	0.47
CMT5 (199,9,8)	10	1419540.92	1643841	108970.19	22.94	33.33	3.94	1.58	2.13	0.20
	20	2711808.24	3096235	195440.91	21.49	28.73	2.84	2.37	3.63	0.37
	30	3929397.12	4468400	284471.40	20.36	27.59	2.73	3.22	4.36	0.47
	40	5063679.84	5739811	348592.79	19.70	26.52	2.43	4.37	6.08	0.67
	50	6142347.12	6936879	427684.42	18.71	27.31	2.53	5.80	8.99	1.16
	60	7177526.08	8181177	486045.38	17.96	24.35	2.04	7.20	10.71	1.36
	70	8146948.64	9243250	567250.70	16.91	24.14	2.94	9.24	15.42	2.08
	80	9029270.76	10223006	635457.72	15.66	23.80	3.18	10.57	16.40	2.21

Table 5.2 Impact of parameter $\mathcal{T}$ on a set of five representative instances

HMb. The reconstruction of the giant tour is modified (Line 11 of Algorithm 6). We consider a parameter X. The idea of this new giant tour reconstruction is to concatenate the first X vehicle tours of the best solution in a random order and then insert the customers assigned to the drones and the customers of the remaining K-X vehicles via best insertion. Initially the value of X is set to X=K. This value changes and is updated as follows: if the current solution is better than bestSol (Line 8 of Algorithm 6), the value of X is set to K (we intensify the search around the new best solution) otherwise the value of X is set to max (0, X -1) (we decrement X if X>0 to diversify).

MS. The giant tour (Line 11 of Algorithm 6) is generated with a randomized nearest-neighbor heuristic. In our implementation, one of the three nearest neighbors is randomly chosen. This randomization is also introduced at the initialization of the algorithm (Line 1 of Algorithm 6). Compared to **HM**, the memory component is lost, and the iterative mechanism is that of a standard multi-start instead of an ILS (see Algorithm 9).

HM(LL), HMb(LL), MS(LL). The three heuristics **HM**, **HMb** and **MS**, are declined in a second version. In these new versions, labels are limited using the technique presented in Section 5.3.1. Every elementary iteration of the algorithm should then be less efficient but faster.

HM(UB), HMb(UB), MS(UB). The three heuristics **HM**, **HMb** and **MS**, are declined in a third version. In these new versions, the decoding procedure is restricted to the computation of the upper bounds: both the lower bound and the labeling algorithm are deactivated. Every elementary iteration of the algorithm should then be even less efficient but faster.

Algorithm 9 Multi-start heuristic **MS**

1: $bestSol \leftarrow (\tau_1^{rec} = \emptyset, \tau_2^{rec} = \emptyset, \ldots, \tau_K^{rec} = \emptyset, \pi_1^{rec} = \emptyset, \ldots, \pi_M^{rec} = \emptyset)$
2: **while** $explorationTime \leq timeLimit$ **do**
3: $\quad \tau \leftarrow solveTSP()$
4: $\quad (\tau_1, \tau_2, \ldots, \tau_K, \pi_{drones}) \leftarrow split(\tau)$
5: $\quad (\tau_1^{opt}, \tau_2^{opt}, \ldots, \tau_K^{opt}) \leftarrow reoptimizeTSP(\tau_1, \tau_2, \ldots, \tau_K)$
6: $\quad (\pi_1^{opt}, \ldots, \pi_M^{opt}) \leftarrow optimizePMS(\pi_{drones})$
7: $\quad (\tau_1^{imp}, \tau_2^{imp}, \ldots, \tau_K^{imp}, \pi_1^{imp}, \ldots, \pi_M^{imp}) \leftarrow improveSol(\tau_1^{opt}, \tau_2^{opt}, \ldots, \tau_K^{opt}, \pi_1^{opt}, \ldots, \pi_M^{opt})$
8: $\quad$ **if** solution $(\tau_1^{imp}, \tau_2^{imp}, \ldots, \tau_K^{imp}, \pi_1^{imp}, \ldots, \pi_M^{imp})$ is better than $bestSol$ **then**
9: $\quad\quad bestSol \leftarrow (\tau_1^{imp}, \tau_2^{imp}, \ldots, \tau_K^{imp}, \pi_1^{imp}, \ldots, \pi_M^{imp})$
10: $\quad$ **end if**
11: **end while**

Tables 5.3 and 5.4 present the completion times (C.T.) obtained with the 9 heuristics, *i.e.,* **HM** and the 8 variants, and for the 20 instances. Furthermore, a synthetic table showing the gaps of the different heuritic versions is provided (Table 5.5). This gap uses the best completion time found by all the variants as a reference value. $gap = 100 \times \frac{c_{variant} - c_{reference}}{c_{reference}}$. In all experiments, the computing time limit is set to 1000 seconds. Best solutions are highlighted in bold.

Tables 5.6 to 5.8 give more details on the execution of the heuristics. In Table 5.6 details are provided for methods **HM**, **HMb** and **MS**. Table 5.7 is about variants with limited labels in procedure $split(\tau)$: **HM(LL)**, **HMb(LL)** and **MS(LL)**. Table 5.8 considers variants with upper bounds only: **HM(UB)**, **HMb(UB)** and **MS(UB)**.

In Tables 5.6 and 5.7, the following information is reported. Column #Split reports the average number of calls to procedure $split(\tau)$. #Lb gives the average number of labels generated in procedure $split(\tau)$ (summed over all nodes and including labels eventually deleted by the different labels pruning mechanisms). Del is the percentage of labels deleted with the label pruning mechanisms. Column gD gives the average gap between the drone completion time found by procedure $split(\tau)$ (assuming a single drone with a speed multiplied by M) and the completion time obtained after having applied the LPT heuristic. $\mathtt{T}_{Split}$, $\mathtt{T}_{Opt}$ and $\mathtt{T}_{LS}$ indicate the computing time spent in procedure $split(\tau)$, reoptimization ($reoptimizeTSP(\tau_1, \tau_2, \ldots, \tau_K)$ and $optimizePMS(\pi_{drones})$) and local search ($improveSol(\tau_1^{opt}, \tau_2^{opt}, \ldots, \tau_K^{opt}, \pi_1^{opt}, \ldots, \pi_M^{opt})$), respectively. #D.C. is the number of customers assigned to the drones. C.T. gives the completion time.

In Table 5.8, column #Iter represents the number of iterations of the main loop, $\mathtt{T}_{UB}$ is the time spent to compute the upper bound. Other columns are labeled as in tables 5.6 and 5.7.

	Method	C.T.	Method	C.T.	Method	C.T.
CMT1 (50,3,2)	**HM**	168	**HM(LL)**	**166**	**HM(UB)**	174
	HMb	168	**HMb(LL)**	168	**HMb(UB)**	174
	MS	188	**MS(LL)**	196	**MS(UB)**	204
CMT2 (75,5,5)	**HM**	**130.23**	**HM(LL)**	132	**HM(UB)**	140
	HMb	133.60	**HMb(LL)**	133.41	**HMb(UB)**	140
	MS	148	**MS(LL)**	152	**MS(UB)**	152
CMT3 (100,4,4)	**HM**	**184**	**HM(LL)**	187.04	**HM(UB)**	195.42
	HMb	186	**HMb(LL)**	186.17	**HMb(UB)**	197.24
	MS	208	**MS(LL)**	204	**MS(UB)**	216
CMT4 (150,6,6)	**HM**	160.38	**HM(LL)**	162	**HM(UB)**	166
	HMb	**150**	**HMb(LL)**	162	**HMb(UB)**	164
	MS	184	**MS(LL)**	180	**MS(UB)**	192
CMT5 (199,9,8)	**HM**	**138**	**HM(LL)**	140	**HM(UB)**	142.04
	HMb	139.29	**HMb(LL)**	**138**	**HMb(UB)**	140
	MS	152	**MS(LL)**	154	**MS(UB)**	152
E-n51-k5 (50,3,2)	**HM**	**168**	**HM(LL)**	**168**	**HM(UB)**	168.86
	HMb	**168**	**HMb(LL)**	**168**	**HMb(UB)**	174
	MS	182	**MS(LL)**	180	**MS(UB)**	196
E-n76-k8 (75,4,4)	**HM**	**154**	**HM(LL)**	156	**HM(UB)**	161.86
	HMb	**154**	**HMb(LL)**	156	**HMb(UB)**	162
	MS	168	**MS(LL)**	182	**MS(UB)**	186
E-n101-k8 (100,4,4)	**HM**	186	**HM(LL)**	188	**HM(UB)**	196
	HMb	**184**	**HMb(LL)**	190.17	**HMb(UB)**	196
	MS	208	**MS(LL)**	224	**MS(UB)**	216
M-n151-k12 (150,6,6)	**HM**	**154**	**HM(LL)**	164	**HM(UB)**	168
	HMb	158.96	**HMb(LL)**	162	**HMb(UB)**	169.88
	MS	186	**MS(LL)**	182	**MS(UB)**	182
M-n200-k16 (199,8,8)	**HM**	**144**	**HM(LL)**	148	**HM(UB)**	152
	HMb	148	**HMb(LL)**	146	**HMb(UB)**	152
	MS	162	**MS(LL)**	156	**MS(UB)**	168

Table 5.3 Solution values (10 first instances)

	Method	C.T.	Method	C.T.	Method	C.T.
P-n51-k10 (50,5,5)	**HM**	**111.07**	**HM(LL)**	112.69	**HM(UB)**	118
	HMb	114	**HMb(LL)**	114	**HMb(UB)**	118
	MS	118	**MS(LL)**	122	**MS(UB)**	133.25
P-n55-k10 (54,5,5)	**HM**	128	**HM(LL)**	128	**HM(UB)**	130
	HMb	128	**HMb(LL)**	**126**	**HMb(UB)**	132
	MS	138	**MS(LL)**	142	**MS(UB)**	148
P-n60-k10 (59,5,5)	**HM**	**114**	**HM(LL)**	114.86	**HM(UB)**	122
	HMb	116	**HMb(LL)**	116	**HMb(UB)**	120
	MS	124	**MS(LL)**	124	**MS(UB)**	124
P-n65-k10 (64,5,5)	**HM**	**126**	**HM(LL)**	128	**HM(UB)**	134
	HMb	**126**	**HMb(LL)**	**126**	**HMb(UB)**	131.36
	MS	138	**MS(LL)**	142	**MS(UB)**	154
P-n70-k10 (69,5,5)	**HM**	129.29	**HM(LL)**	136	**HM(UB)**	138
	HMb	**128**	**HMb(LL)**	132	**I3Sb(UB)**	136.56
	MS	138	**MS(LL)**	146	**MS(UB)**	158
P-n76-k5 (75,3,2)	**HM**	202	**HM(LL)**	202	**HM(UB)**	210
	HMb	**200**	**HMb(LL)**	202	**HMb(UB)**	210
	MS	214	**MS(LL)**	243.44	**MS(UB)**	258
P-n101-k4 (100,2,2)	**HM**	342.69	**HM(LL)**	346	**HM(UB)**	353.26
	HMb	**342**	**HMb(LL)**	348	**HMb(UB)**	354
	MS	396	**MS(LL)**	388	**MS(UB)**	422
X-n110-k13 (109,7,6)	**HM**	**1864**	**HM(LL)**	1898	**HM(UB)**	1926
	HMb	1898	**HMb(LL)**	1898	**HMb(UB)**	1960
	MS	2080	**MS(LL)**	2044	**MS(UB)**	1970
X-n115-k10 (114,5,5)	**HM**	**2258**	**HM(LL)**	2262	**HM(UB)**	2316
	HMb	2300	**HMb(LL)**	2274	**HMb(UB)**	2332
	MS	2658	**MS(LL)**	2504	**MS(UB)**	2862
X-n139-k10 (138,5,5)	**HM**	2928.64	**HM(LL)**	2534	**HM(UB)**	2594
	HMb	2740	**HMb(LL)**	**2492**	**HMb(UB)**	2550
	MS	3144	**MS(LL)**	2696	**MS(UB)**	3022

Table 5.4 Solution values (10 last instances)

Table 5.5 Gaps based on the best completion time found by the 9 heuristics

	HM	**HMb**	**MS**	**HM(LL)**	**HMb(LL)**	**MS(LL)**	**HM(UB)**	**HMb(UB)**	**MS(UB)**
CMT1 (50,3,2)	1.20	1.20	13.25	**0.00**	1.20	18.07	4.82	4.82	22.89
CMT2 (75,5,5)	**0.00**	2.59	13.64	1.36	0.03	16.72	7.50	7.50	16.72
CMT3 (100,4,4)	**0.00**	1.09	13.04	1.65	1.18	10.87	6.21	7.19	17.39
CMT4 (150,6,6)	6.92	**0.00**	22.67	8.00	8.00	20.00	10.67	9.33	28.00
CMT5 (199,9,8)	**0.00**	0.93	10.14	1.45	**0.00**	11.59	2.93	1.45	10.14
E-n51-k5 (50,3,2)	**0.00**	**0.00**	8.33	**0.00**	**0.00**	7.14	0.51	3.57	16.67
E-n76-k8 (75,4,4)	**0.00**	**0.00**	9.09	1.30	1.30	18.18	5.10	5.19	20.78
E-n101-k8 (100,4,4)	1.09	**0.00**	13.04	2.17	3.35	21.73	6.52	6.52	17.39
M-n151-k12 (150,6,6)	**0.00**	3.22	20.78	6.49	5.19	18.18	9.09	10.31	18.18
M-n200-k16 (199,8,8)	**0.00**	2.78	12.50	2.78	1.39	8.33	5.55	5.55	16.67
P-n51-k10 (50,5,5)	**0.00**	2.64	6.24	1.46	2.64	9.84	6.24	6.24	19.97
P-n55-k7 (54,4,3)	1.59	1.59	9.52	1.59	**0.00**	12.70	3.17	4.76	17.46
P-n60-k10 (59,5,5)	**0.00**	1.75	8.77	0.75	1.75	8.77	7.02	5.26	8.77
P-n65-k10 (64,5,5)	**0.00**	**0.00**	9.52	1.59	**0.00**	12.70	5.35	4.25	22.22
P-n70-k10 (69,5,5)	1.00	**0.00**	7.81	6.25	3.12	14.06	4.69	6.69	23.44
P-n76-k5 (75,3,2)	1.00	**0.00**	7.00	1.00	1.00	21.72	5.00	5.00	29.00
P-n101-k4 (100,2,2)	0.20	**0.00**	15.79	1.17	1.75	13.45	3.29	3.51	23.39
X-n110-k13 (109,7,6)	**0.00**	1.82	11.59	1.82	1.82	9.66	3.33	5.15	5.69
X-n115-k10 (114,5,5)	**0.00**	1.86	17.71	0.18	0.71	10.89	2.57	3.51	12.93
X-n139-k10 (138,5,5)	17.52	9.95	26.16	1.68	**0.00**	8.19	4.09	2.33	21.27

Table 5.6 Decoding with procedure $split(\tau)$ (complete decoding)

Instance		Execution details							Solution details	
		#Split	#Lb	Del(%)	gD(%)	T_{Split}	T_{Opt}	T_{LS}	#D.C.	C.T.
CMT1 (50,3,2)	**HM**	1920	169688	96.27	2.16	290.45	606.22	18.14	8	168
	HMb	1755	203593	96.26	2.03	340.73	566.92	19.48	8	168
	MS	2548	881	97.69	0.001	70.51	852.54	14.81	10	188
CMT2 (75,5,5)	**HM**	527	865676	97.78	3.89	651.69	279.33	8.85	11	130.23
	HMb	453	1009351	97.88	3.68	723.42	242.75	6.23	13	133.60
	MS	1375	401662	98.72	2.62	323.19	626.44	6.30	16	148
CMT3 (100,4,4)	**HM**	195	2501797	98.20	2.49	881.77	105.99	2.04	17	184
	HMb	135	3089482	98.22	2.55	930.39	72.42	1.72	15	186
	MS	1696	22831	99.07	0.01	156.16	772.92	8.07	16	208

Continued on next page

Table 5.6 – continued from previous page

Instance		Execution details							Solution details	
		#Split	#Lb	Del(%)	gD(%)	T_{Split}	T_{Opt}	T_{LS}	#D.C.	C.T.
CMT4 (150,6,6)	**HM**	51	9172075	98.97	3.03	955.19	42.57	1.00	17	160.38
	HMb	40	9611465	98.97	3.72	966.96	32.58	0.73	18	150
	MS	234	3990575	99.31	0.05	848.35	138.38	1.32	24	184
CMT5 (199,9,8)	**HM**	33	17588386	99.29	4.60	959.96	44.89	0.78	19	138
	HMb	23	19889825	99.29	5.77	987.09	28.56	0.57	14	139.29
	MS	25	15430506	99.45	0.28	978.73	27.86	0.37	27	152
E-n51-k5 (50,3,2)	**HM**	1927	167681	96.32	2.08	277.97	616.89	19.09	9	168
	HMb	1709	204593	96.27	2.02	348.56	560.89	19.15	9	168
	MS	2864	1109	97.98	0.001	76.31	846.46	9.44	8	182
E-n76-k8 (75,4,4)	**HM**	555	876503	97.61	2.82	713.09	247.92	6.01	13	154
	HMb	454	1020952	97.69	2.56	766.15	201.69	5.90	12	154
	MS	1983	174313	98.56	3.87	215.29	719.07	4.89	14	168
E-n101-k8 (100,4,4)	**HM**	162	2706778	98.17	2.56	897.93	87.58	1.81	16	186
	HMb	145	2890459	98.21	2.43	904.11	82.45	2.01	15	184
	MS	1758	25273	99.06	0.004	165.52	760.12	8.55	16	208
M-n151-k12 (150,6,6)	**HM**	45	9457220	98.95	3.25	958.30	37.46	0.80	25	154
	HMb	41	10503649	98.96	2.67	965.91	34.80	0.65	17	158.96
	MS	193	4395453	99.29	0.34	844.65	145.07	1.34	24	186
M-n200-k16 (199,8,8)	**HM**	25	18639347	99.26	3.97	983.16	30.20	0.62	20	144
	HMb	16	22132150	99.28	4.61	1010.45	20.36	0.42	19	148
	MS	24	16384011	99.45	1.01	991.39	22.97	0.41	25	162
P-n51-k10 (50,5,5)	**HM**	1221	223455	96.83	6.16	317.52	607.45	12.10	11	111.07
	HMb	1095	254528	96.94	5.98	352.82	579.69	11.86	10	114
	MS	1825	118732	98.16	0.02	154.68	790.60	7.95	11	118
P-n55-k10 (54,5,5)	**HM**	1624	183464	96.99	3.68	232.50	672.05	15.44	7	128
	HMb	1459	229487	96.99	3.55	293.84	621.46	16.16	7	128
	MS	2168	3839	98.60	0.005	68.51	862.47	10.54	8	138
P-n60-k10 (59,5,5)	**HM**	973	361918	97.23	5.18	424.83	509.61	10.46	11	114
	HMb	932	403321	97.39	4.99	457.95	481.91	10.16	11	116
	MS	1442	160015	98.42	0.01	174.40	768.30	9.02	14	124
P-n65-k10 (64,5,5)	**HM**	783	504060	97.43	5.09	509.63	399.71	11.12	11	126
	HMb	709	582851	97.51	4.81	579.55	371.07	8.39	12	126
	MS	1554	276576	98.44	0.01	282.48	661.43	8.03	14	138

Continued on next page

Table 5.6 – continued from previous page

Instance		Execution details							Solution details	
		#Split	#Lb	Del(%)	gD(%)	T_{Split}	T_{Opt}	T_{LS}	#D.C.	C.T.
P-n70-k10 (69,5,5)	**HM**	646	694435	97.61	4.21	626.20	327.40	7.47	11	129.29
	HMb	569	778133	97.69	3.98	666.11	295.18	7.12	13	128
	MS	1439	310237	98.59	0.01	293.67	655.65	5.33	15	138
P-n76-k5 (75,3,2)	**HM**	883	696083	97.37	1.20	628.64	309.95	9.73	11	202
	HMb	657	842403	97.35	1.11	707.73	246.43	9.25	10	200
	MS	2747	1367	96.81	0.40	123.94	781.10	12.49	11	214
P-n101-k4 (100,2,2)	**HM**	149	2403670	97.37	0.69	923.65	68.45	1.22	17	342.69
	HMb	96	2757296	97.29	0.59	945.61	50.45	1.07	17	342
	MS	1818	5678	97.45	0.002	154.54	767.80	9.78	20	396
X-n110-k13 (109,7,6)	**HM**	71	6129616	98.48	5.68	945.97	52.87	1.42	16	1864
	HMb	56	6810276	98.54	6.80	952.17	42.57	1.49	12	1898
	MS	254	3235309	99.09	0.05	831.01	156.30	1.31	17	2080
X-n115-k10 (114,5,5)	**HM**	26	8970734	98.37	3.39	986.31	14.70	0.45	15	2258
	HMb	24	8953906	98.44	4.31	1009.97	13.59	0.61	17	2300
	MS	295	2263434	99.04	0.04	854.58	131.20	1.52	20	2658
X-n139-k10 (138,5,5)	**HM**	1	77374287	98.56	1.18	3056.25	0.70	0.08	23	2928.64
	HMb	1	82009638	98.55	5.85	3726.85	0.68	0.13	23	2740
	MS	1	47270494	98.07	2.18	1798.23	0.56	0.05	24	3144

Table 5.7 Decoding with limited procedure $split(\tau)$ (label elimination)

Instance		Execution details							Solution details	
		#Split	#Lb	Del(%)	gD(%)	T_{Split}	T_{Opt}	T_{LS}	#D.C.	C.T.
CMT1 (50,3,2)	**HM(LL)**	2231	102388	94.05	2.90	164.58	706.18	32.57	8	166
	HMb(LL)	1853	107130	93.57	2.88	158.58	653.23	43.40	7	168
	MS(LL)	2696	3114	98.39	0.001	72.44	853.68	12.66	7	196
CMT2 (75,5,5)	**HM(LL)**	1071	419731	96.24	3.57	338.56	565.05	26.75	14	132
	HMb(LL)	902	428728	96.45	3.34	310.76	558.85	33.29	13	133.41
	MS(LL)	1284	241493	97.87	0.007	192.13	760.15	5.93	15	152
CMT3 (100,4,4)	**HM(LL)**	920	630959	93.05	4.23	407.14	496.82	27.10	18	187.04
	HMb(LL)	883	647251	93.08	3.59	410.27	457.01	26.71	17	186.17
	MS(LL)	1692	26008	97.89	0.01	135.04	794.69	8.17	17	204
CMT4 (150,6,6)	**HM(LL)**	401	2209683	96.24	3.59	616.58	322.68	16.54	18	162
	HMb(LL)	405	2238685	96.87	3.03	594.03	314.67	17.48	20	162

Continued on next page

Table 5.7 – continued from previous page

Instance		Execution details							Solution details	
		#Split	#Lb	Del(%)	gD(%)	T_{Split}	T_{Opt}	T_{LS}	#D.C.	C.T.
	MS(LL)	778	817197	98.24	0.01	376.63	574.26	4.85	28	180
CMT5 (199,9,8)	**HM(LL)**	180	5645118	98.02	4.21	743.80	223.56	7.97	23	140
	HMb(LL)	188	5711948	98.4297	3.25	720.29	229.33	8.68	21	138
	MS(LL)	304	3507043	98.82	0.09	647.20	332.00	1.70	35	154
E-n51-k5 (50,3,2)	**HM(LL)**	2242	100780	94.29	3.12	160.78	705.68	33.02	7	168
	HMb(LL)	2141	108046	93.49	2.77	172.09	649.58	34.40	8	168
	MS(LL)	2323	2001	97.86	0.50	67.30	868.55	5.66	10	180
E-n76-k8(75,4,4)	**HM(LL)**	1313	345500	94.57	3.14	319.37	564.96	34.84	15	156
	HMb(LL)	1250	349315	94.66	2.96	317.68	523.16	33.00	16	156
	MS(LL)	1573	130334	98.21	5.33	146.87	791.73	7.77	20	182
E-n101-k8 (100,4,4)	**HM(LL)**	921	629648	93.06	4.15	405.20	497.80	26.59	18	188
	HMb(LL)	826	654237	93.05	3.87	415.56	453.82	25.66	18	190.17
	MS(LL)	1490	247410	98.53	0.004	211.90	725.76	6.92	17	224
M-n151-k12 (150,6,6)	**HM(LL)**	401	2228103	96.43	3.46	613.81	323.70	18.07	23	164
	HMb(LL)	410	2258814	96.84	2.63	600.45	310.26	17.16	20	162
	MS(LL)	849	888681	98.09	0.01	368.20	583.89	5.08	26	182
M-n200-k16 (199,8,8)	**HM(LL)**	202	5140018	97.76	3.75	724.95	237.31	10.03	24	148
	HMb(LL)	208	5258553	98.30	3.06	705.40	236.05	9.69	20	146
	MS(LL)	434	2347860	98.69	1.37	544.48	421.49	2.76	31	156
P-n51-k10 (50,5,5)	**HM(LL)**	1495	160311	95.84	5.92	206.31	694.00	23.35	10	112.69
	HMb(LL)	1425	171177	95.99	5.72	212.75	648.58	21.99	10	114
	MS(LL)	1826	112113	97.75	1.50	135.88	812.67	6.47	12	122
P-n55-k10 (54,5,5)	**HM(LL)**	1747	139775	95.84	4.75	168.45	720.44	25.52	8	128
	HMb(LL)	1690	150711	95.52	4.39	178.89	667.40	25.66	8	126
	MS(LL)	2389	14359	98.48	0.01	78.90	852.62	10.24	11	142
P-n60-k10 (59,5,5)	**HM(LL)**	1311	235830	96.25	5.19	253.27	648.90	23.03	12	114.86
	HMb(LL)	1303	250511	96.37	4.79	258.56	605.59	21.65	10	116
	MS(LL)	1787	116243	98.14	0.01	132.04	809.75	8.28	12	124
P-n65-k10 (64,5,5)	**HM(LL)**	1234	299012	96.11	4.81	296.03	606.55	24.45	11	128
	HMb(LL)	1188	306392	96.45	4.56	285.63	583.53	22.02	9	126
	MS(LL)	1594	189110	97.84	0.63	175.28	769.47	7.49	15	142
P-n70-k10 (69,5,5)	**HM(LL)**	1179	354052	95.99	3.81	326.20	574.87	26.53	12	136
	HMb(LL)	1058	361613	96.42	3.65	313.53	562.61	23.10	10	132

Continued on next page

Table 5.7 – continued from previous page

Instance		Execution details							Solution details	
		#Split	#Lb	Del(%)	gD(%)	T_{Split}	T_{Opt}	T_{LS}	#D.C.	C.T.
	MS(LL)	1603	208460	97.86	0.16	188.34	754.99	7.18	15	146
P-n76-k5 (75,3,2)	**HM(LL)**	1655	262675	94.42	2.09	259.27	613.38	31.95	11	202
	HMb(LL)	1440	275681	93.89	1.81	276.87	569.41	28.21	10	202
	MS(LL)	2236	1101	96.51	0.002	90.35	839.46	4.58	9	243.44
P-n101-k4 (100,2,2)	**HM(LL)**	1179	439302	93.36	0.78	292.53	604.22	18.60	18	346
	HMb(LL)	1138	463067	93.09	0.78	299.16	559.76	19.74	16	348
	MS(LL)	1276	1366	96.43	0.75	64.43	882.19	6.85	19	388
X-n110-k13 (109,7,6)	**HM(LL)**	508	1502768	96.31	7.16	583.05	350.33	21.81	19	1898
	HMb(LL)	510	1491711	96.47	6.06	577.05	328.85	21.97	15	1898
	MS(LL)	933	734380	98.14	0.01	342.45	612.55	4.78	18	2044
X-n115-k10 (114,5,5)	**HM(LL)**	616	1207908	93.83	5.92	592.19	327.68	29.34	17	2262
	HMb(LL)	653	1204199	94.22	4.02	568.99	310.53	31.50	18	2274
	MS(LL)	1376	284800	97.98	0.006	251.63	683.53	6.86	17	2504
X-n139-k10 (138,5,5)	**HM(LL)**	385	2069330	92.49	7.73	690.26	207.10	34.70	26	2534
	HMb(LL)	296	1986166	92.49	7.61	670.71	218.65	40.42	19	2492
	MS(LL)	1409	48399	98.81	0.006	208.41	719.99	7.14	25	2696

Table 5.8 Decoding with upper bounds only

Instance		Execution details					Solution details	
		#Iter	gD(%)	T_{UB}	T_{Opt}	T_{LS}	#D.C.	C.T.
CMT1 (50,3,2)	**HM(UB)**	2299	1.80	58.74	784.98	51.70	10	174
	HMb(UB)	2275	1.72	51.63	830.11	37.41	10	174
	MS(UB)	2228	0.003	54.62	891.56	10.13	9	204
CMT2 (75,5,5)	**HM(UB)**	1318	2.85	67.49	800.17	42.48	14	140
	HMb(UB)	1371	2.81	64.33	827.78	33.03	16	140
	MS(UB)	1367	0.01	70.08	885.80	6.89	15	152
CMT3 (100,4,4)	**HM(UB)**	1068	2.17	86.60	780.01	38.91	17	195.42
	HMb(UB)	1307	2.18	88.23	787.96	34.99	17	197.24
	MS(UB)	1621	0.005	114.62	824.06	7.81	21	216
CMT4 (150,6,6)	**HM(UB)**	710	2.52	166.51	696.51	46.37	25	166
	HMb(UB)	718	2.51	172.37	713.95	34.37	22	164
	MS(UB)	992	0.02	211.78	728.63	5.40	27	192

Continued on next page

Table 5.8 – continued from previous page

Instance		Execution details					Solution details	
		#Iter	gD(%)	T_{UB}	T_{Opt}	T_{LS}	#D.C.	C.T.
CMT5 (199,9,8)	**HM(UB)**	430	3.19	297.64	611.46	24.33	26	142.04
	HMb(UB)	424	3.05	310.84	613.63	16.58	27	140
	MS(UB)	615	0.02	378.96	576.42	3.56	34	152
E-n51-k5 (50,3,2)	**HM(UB)**	2250	1.80	54.27	813.95	39.84	10	168.82
	HMb(UB)	2153	1.68	50.34	833.54	37.28	9	174
	MS(UB)	3177	0.17	79.70	835.66	13.94	13	196
E-n76-k8(75,4,4)	**HM(UB)**	1604	2.23	66.60	807.95	37.60	14	161.86
	HMb(UB)	1382	2.21	61.48	829.50	34.21	14	162
	MS(UB)	2077	0.005	95.73	828.64	9.55	16	186
E-n101-k8 (100,4,4)	**HM(UB)**	1317	2.20	90.14	782.18	36.48	19	196
	HMb(UB)	1211	2.13	86.73	797.36	33.91	17	196
	MS(UB)	1604	0.01	121.34	811.58	7.8	20	216
M-n151-k12 (150,6,6)	**HM(UB)**	698	2.66	174.18	710.21	34.59	26	168
	HMb(UB)	671	2.53	172.67	731.71	25.73	25	169.88
	MS(UB)	861	0.01	204.83	746.78	4.27	27	182
M-n200-k16 (199,8,8)	**HM(UB)**	423	2.85	289.41	625.17	20.70	29	152
	HMb(UB)	407	2.88	289.58	626.62	21.08	29	152
	MS(UB)	545	0.77	307.89	651.94	3.68	30	168
P-n51-k10 (50,5,5)	**HM(UB)**	1342	5.04	41.89	862.75	27.59	14	118
	HMb(UB)	1195	4.91	45.17	826.71	55.80	13	118
	MS(UB)	1882	0.43	56.55	889.16	7.15	14	133.25
P-n55-k10 (54,5,5)	**HM(UB)**	1262	2.87	52.25	831.88	35.11	10	130
	HMb(UB)	1678	2.91	51.91	795.34	67.06	7	132
	MS(UB)	2413	0.002	74.88	853.80	11.24	11	148
P-n60-k10 (59,5,5)	**HM(UB)**	1381	4.42	51.95	841.48	30.17	13	122
	HMb(UB)	1564	4.39	56.32	822.68	36.12	14	120
	MS(UB)	1722	0.02	63.50	879.45	7.93	12	124
P-n65-k10 (64,5,5)	**HM(UB)**	1369	4.09	56.85	817.76	39.89	12	134
	HMb(UB)	1489	3.96	59.04	815.87	39.92	15	131.36
	MS(UB)	1510	0.007	63.82	883.64	6.59	17	154
P-n70-k10 (69,5,5)	**HM(UB)**	1314	3.12	62.40	792.90	53.37	15	138
	HMb(UB)	1414	3.12	62.92	809.69	41.53	13	136.56
	MS(UB)	1358	0.004	65.22	884.86	6.18	20	158

Continued on next page

Table 5.8 – continued from previous page

Instance		Execution details					Solution details	
		#Iter	gD(%)	$\mathtt{T}_{UB}$	$\mathtt{T}_{Opt}$	$\mathtt{T}_{LS}$	#D.C.	C.T.
P-n76-k5 (75,3,2)	**HM(UB)**	1782	1.09	74.51	749.87	54.73	11	210
	HMb(UB)	1808	1.07	72.22	778.88	47.46	14	210
	MS(UB)	1547	1.23	73.03	868.26	7.40	13	258
P-n101-k4 (100,2,2)	**HM(UB)**	1471	0.65	75.15	775.19	34.40	19	353.26
	HMb(UB)	1491	0.62	71.99	791.42	34.22	17	354
	MS(UB)	2369	1.39	119.92	784.19	11.01	20	422
X-n110-k13 (109,7,6)	**HM(UB)**	922	5.35	116.61	763.39	38.94	17	1926
	HMb(UB)	751	5.29	111.39	783.22	35.63	18	1960
	MS(UB)	963	0.02	138.06	816.56	4.69	18	1970
X-n115-k10 (114,5,5)	**HM(UB)**	1005	2.79	116.02	730.93	55.49	20	2316
	HMb(UB)	961	2.68	116.86	752.25	47.41	19	2332
	MS(UB)	1121	0.67	136.63	808.88	5.44	24	2862
X-n139-k10 (138,5,5)	**HM(UB)**	1029	3.03	152.88	696.88	54.09	25	2594
	HMb(UB)	806	2.96	152.67	711.63	51.821	22	2550
	MS(UB)	1144	0.1	208.01	731.57	5.71	29	3022

In these tables, we can see that the two giant tour construction methods show very similar behaviors (first line and second line of the tables for each instance). Both solution values and the set of execution parameters shown in the tables are very similar. On the contrary, the multi-start variant (third line) is not able to find solutions that are as good as those of these two methods. Not a single best solution is found with this approach. It demonstrates the gains provided by the memory technique introduced in the ILS.

Eliminating labels in the decoding procedure does not have a clear impact on solution values. In most cases results are very similar, even if most best values are found with the unrestricted $split(\tau)$ procedure. It is however interesting to see that eliminating labels provide some robustness. For some difficult instances, like X-n139-k10, it avoid getting stuck in a difficult decoding. On our set of instances, methods with label elimination always reach at least 25 iterations in the imparted 1000 seconds. On the contrary, when the number of labels is not limited in procedure $split(\tau)$, several cases with very few iterations, sometimes a single, can be observed.

Completely avoiding exploration is however not enough. When decoding is only based on upper bounds, solution quality significantly decreases. Again, no best values are found with this setting.

Going into more details, with table 5.6 we can see that the number of calls to procedure $split(\tau)$ decreases when the size of the instance increases. This was expected because the

larger the size of the instance, the larger the number of labels generated in the decoding procedure and the longer the computing time of procedure $split(\tau)$. Table 5.6 also shows that dominance and bounding procedures are essential. They enable to prune more than 95% of the labels in most cases. Unfortunately, for some instances, up to millions of labels remain. For this reason, on large-sized instances ($n > 60$), about 70 to 90% of the computing time is spent in the decoding procedure, 10 to 30% in the optimization step and 1 to 4% only in the local search. On the contrary, when instances are smaller ($n \leq 60$), the heuristics spend more time in optimization step (50 to 80% of the computing time). Finally, for almost all instances, gD has a value of less than 5% except for a few cases with **HM** and **HMb**. It shows that aggregating drones in the decoding procedure has a limited impact.

From Table 5.7, we can make the following additional observations. For instances of size $n < 100$, the time spent in the decoding procedure is less than 35% of the running time. On the other hand, it takes about 40% to 70% of the running time for the decoding procedure to be completed on larger instances ($n \geq 100$). It shows that eliminating labels in the decoding procedure surely accelerates the procedure. When the larger part of the computing time was initially spent in procedure $split(\tau)$, it enables a large increase in the number of iterations. Having an accelerated decoding with labels limitation leads to a lost of efficiency of this decoding and an increase in the time spent in the optimization step.

Finally Table 5.8 shows that, for the 3 variants, approximately 80% of the execution time is spent in the optimization step. The computation of the upper bound is very fast (less than 30% of the execution time in most cases). Most of the running time is spent in the optimization step. This was expected due to the simplicity of the upper bound computation scheme which may lead to solution of low quality.

From Tables 5.6, 5.7 and 5.8, another important observation to be made is the few time spent in local search. This shows that decoding and optimization components together are quite good enough to lead to solutions of good quality. However, local search still helps to improve the solution.

5.4 Conclusion

In this chapter, we have proposed a hybrid metaheuristic to solve the PDSMTSP which is adapted from the iterative two-step heuristic descibed in the previous chaper for the PDSTSP. Firstly, a giant tour visiting all customers is built. A second step uses dynamic programming for efficiently partitioning the customers of the giant tour between the set of vehicles and the fleet of drones with the restriction that each vehicle route follows the order defined by the giant tour. Introducing several vehicles has a huge computational impact on this step. To circumvent this difficulty, we introduced several upper bounding and lower bounding techniques. In addition, the restriction imposed in the order of vehicle routes can be very detrimental on effectiveness. In a thrid step, we apply some local search operators to relax this constraint and converge towards better solutions. The general scheme of the hybrid metaheuristic can be interpreted as an Iterated Local Search.

Experiments conducted on instances taken from CVRPLIB allowed us to assess the performance of the proposed heuristic. They demonstrated the importance of the decoding procedure, the bounding techniques in this procedure and the memory mechanism of ILS in the heuristic. Unfortunately, this work being the first on the PDSMTSP, no comparison with the literature was possible. We implemented a simple branch-and-cut algorithm (in chapter 3) in an attempt of obtaining a comparison basis, but, unfortunately, no competitive results could be obtained with this branch-and-cut. However, from the optimality gap obtained by CPLEX (for the instances that could have been resolved), we can say that the results obtained by the hybrid metaheuristic confirm the effectiveness of the proposed approach.

The content of this chapter is the subject of a paper submitted to the European Journal of Operational Research (EJOR) [MS+20].

Conclusion and perspectives

Drone deliveries look like the future: unmanned aerial vehicles rapidly delivering packages to our doors, eliminating both waiting times and the cost of human labor. The idea of parcel delivery by drones is gaining ground and gradually becoming a reality all over the world. Time saving, road risk reduction, less pollution, in addition to being practical, drones are also an effective solution. However, many technical and legal obstacles complicate its application on a large scale. Indeed, the use of drones is difficult to generalize in urban areas. Along with a larger population, city landscapes present their ownchallenges: more obstacles to detect and avoid (UAVs are restricted to fly at low altitude), specific weather and wind conditions, reduced lines of sight and fewer safe landing locations. Delivery by drones therefore remains subject to strict regulations for the moment and the authorities are reluctant to give authorizations.

From the literature review we could observe that most of the drone research has been focusing on the scenario where drones are deployed from a delivery truck. Very less research focused on the model where drones perform deliveries independently of other vehicles. Therefore, it was necessary to explore this different configuration.

In this thesis, we studied the *Parallel Drone Scheduling Traveling Salesman Problem* (PDSTSP) initially introduced by Murray and Chu [MC15]. In this problem, one or several drones deliver packages concurrently with a single vehicle. Some customers are served by drones directly from a depot (with the drone making back and forth trips between de depot and the customer location), while the remaining customers are served by the vehicle along its route. The objective is to minimize the completion time, that is, the time when all drones and the vehicle returned to the depot with all deliveries carried out. We proposed an iterative two-steps heuristic that uses dynamic programming for efficiently partitioning the customers between the vehicle and the drone fleet. The heuristic is composed of a coding step which transforms a solution into a customer sequence and a decoding step which decomposes this sequence into a tour for the vehicle and series of trips for the drones. The decomposition made in the decoding step is optimal when a single vehicle and a single drone are considered. A MILP formulation of the problem and a simple Branch-and-Cut procedure have also been proposed. Experiments conducted on benchmark instances confirmed the efficiency of the approach. We obtained very promising results with a clear improvement over the existing literature.

We also studied an extended version of the problem by considering several vehicles. We called it *Parallel Drone Scheduling Multiple Traveling Salesman Problem* (PDSMTSP). We proposed a hybrid metaheuristic adapted from the iterative two-step heuristic proposed for the PDSTSP. Firstly, a giant tour visiting all customers is built. Then, dynamic programming is used for efficiently partitioning the customers of the giant tour between the set of vehicles and the fleet of drones with the restriction that each vehicle route follows the order defined by the giant tour. Since considering several vehicles greatly impacted the computational

time of the dynamic programming procedure, we introduced several upper bounding and lower bounding techniques to manage this difficulty. After partitioning the customers of the giant tour between the set of vehicles and the fleet of drones, a thrid step applies local search to improve the solution. The proposed approach could be validated via an experimental campaign on instances taken from the CVRPLIB library. Experiments conducted on instances taken from CVRPLIB allowed us to assess the performance of the proposed heuristic. Not being aware of any work in the litterature on the PDSMTSP, no comparison with the literature was possible. However, computational experiments comparing several variants of the hybrid metaheuristic give some insights on this drone delivery system. We also provided a MILP formulation of the problem and we implemented a simple Branch-and-Cut algorithm in an attempt of obtaining a comparison basis, but, this Branch-and-Cut only allows to solve tiny instances, and further developments are still necessary in order to obtain better results.

Hereinafter, some ideas regarding future work are suggested: First, we mainly focused in this study on the split procedure. We have decided to favor the optimization of vehicle tours to the detriment of drones because it allows the most significant gains. However, in some cases when the load of drones is greater than that of vehicles, clearly the quality of the results could be slightly improved by solving better the PMS problems that are repeatedly solved in the solution scheme. In the current implementations, the algorithms used for solving these problems are very simple. They have the advantage to be very quick, but it is certainly possible to gain in effectiveness with a limited increase in computing times. Using a population based meta-heuristic scheme rather than a local search based metaheuristic can also be explored.

Another perspective of this work is to develop an exact solution framework by implementing a more robust Branch-and-Cut procedure. Branch-and-price could also be a promising approach. The partitioning between vehicles and drones would easily be managed, as it would only affect the master problem, but the completion time objective would certainly be more challenging.

A better controlling of the combinatorial explosion in *split* procedure is certainly an important direction to follow. Better compromises between decoding quality and number of labels generated can certainly be achieved.

Additionally, capacity contraints and time window constraints could be added to the PDSTSP and the PDSMTSP to analyse the benefits of drones in the delivery of time sensitive orders.

Another interesting perspective would be to explore the work proposed by Ulmer and Thomas [UT18] on the SDDPHF which is a dynamic variant of the PDSTSP. Indeed, this problem is closer to reality by taking into account customers time window contraints, the loading time for both vehicles and drones, the time to drop off a package from a vehicle or a drone as well as the recharging or battery swap time for drones. Designing heuristics and metaheuristics for this problem would be interesting.

Furthermore, building a new delivery model based on Amazon's idea of using virtually any vertical structure, like street lights, cellphone towers, and even church steeples, as a

recharging and docking station for drone delivery seems a good direction to follow. Indeed, it would extend the range of drones by enabling their recharge without returning to the depot. The new model could then consider the possibility for a drone to carry multiple packages.

We close this section with an important note related to the current health context in the world and its impact on delivery by drones. During the health crisis due to the coronavirus pandemic, deliveries by drones emerged as ideal contactless sales solutions. In a world where social distancing is the rule, this sector is doing well and obtaining many exemptions to which it was not previously entitled. Drone deliveries have helped to fight the spread of the virus, allowing the economy to continue in periods of lockdown. For example, *Wing*, the Alphabet subsidiary, has seen deliveries double in the US and Australia during the confinement period. The current period should also offer very good prospects for the future. According to a study published by analyst firm Gartner, more than a million drones could make deliveries in 2026, compared to just 20.000 today.

Bibliography

[ABS18] Niels Agatz, Paul Bouman, and Marie Schmidt. "Optimization approaches for the traveling salesman problem with drone." *Transportation Science* 52.4 (2018), pp. 965–981 (cit. on pp. 44, 45, 54).

[AD03] Majid Aldaihani and Maged M Dessouky. "Hybrid scheduling methods for paratransit operations." *Computers & Industrial Engineering* 45.1 (2003), pp. 75–96 (cit. on p. 30).

[AF11] Stuart M Adams and Carol J Friedland. "A survey of unmanned aerial vehicle (UAV) usage for imagery collection in disaster research and management." *9th International Workshop on Remote Sensing for Disaster Response*. 2011, p. 8 (cit. on p. 11).

[Ali] *Alibaba Begins Drone Delivery Test Program In China*. Available: `https://www.fastcompany.com/3041988/alibaba-begins-drone-delivery-test-program-in-china`. 2015 (cit. on p. 20).

[Ama] *Amazon.com, Inc. Amazon Prime Air. [Online].* Available :`http://www.amazon.com/primeair`. 2013 (cit. on pp. 1, 10, 18).

[Amb+04] Marco Ambrosini, Thomas Caruso, Sara Foresti, and Giovanni Righini. "A GRASP for the pickup and delivery problem with rear loading." *Information Technology Department, Technical Report* 65 (2004) (cit. on p. 30).

[App+98] David Applegate, Robert Bixby, William Cook, and Vasek Chvátal. "On the solution of traveling salesman problems" (1998) (cit. on p. 72).

[ASV14] Claudia Archetti, M Grazia Speranza, and Daniele Vigo. "Chapter 10: Vehicle routing problems with profits." In: *Vehicle Routing: Problems, Methods, and Applications, Second Edition*. SIAM, 2014, pp. 273–297 (cit. on p. 31).

[Aus] *Australia Post tests drones for parcel delivery. [Online].* Available: `http://www.smh.com.au/technology/innovation/australia-post-tests-drones-for-parcel-delivery-20160415-go77a4.html`. 15 April 2016 (cit. on p. 22).

[Bam15] Dane Bamburry. "Drones: Designed for product delivery." *Design Management Review* 26.1 (2015), pp. 40–48 (cit. on p. 2).

[Bar+98] Cynthia Barnhart, Ellis L Johnson, George L Nemhauser, Martin WP Savelsbergh, and Pamela H Vance. "Branch-and-price: Column generation for solving huge integer programs." *Operations research* 46.3 (1998), pp. 316–329 (cit. on p. 33).

[BAS18] Paul Bouman, Niels Agatz, and Marie Schmidt. "Dynamic programming approaches for the traveling salesman problem with drone." *Networks* 72.4 (2018), pp. 528–542 (cit. on pp. 44, 54).

[BCM08] Roberto Baldacci, Nicos Christofides, and Aristide Mingozzi. "An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts." *Mathematical Programming* 115.2 (2008), pp. 351–385 (cit. on p. 33).

[Bea83] John E Beasley. "Route first—cluster second methods for vehicle routing." *Omega* 11.4 (1983), pp. 403–408 (cit. on pp. 34, 35).

[BMR11] Roberto Baldacci, Aristide Mingozzi, and Roberto Roberti. "New route relaxation and pricing strategies for the vehicle routing problem." *Operations research* 59.5 (2011), pp. 1269–1283 (cit. on p. 45).

[BMR12] Roberto Baldacci, Aristide Mingozzi, and Roberto Roberti. "Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints." *European Journal of Operational Research* 218.1 (2012), pp. 1–6 (cit. on p. 32).

[Bol03] Willard Bolton. "Operational experience with UAV payloads for climate research applications." *2nd AIAA" Unmanned Unlimited" Conf. and Workshop & Exhibit*. 2003, p. 6620 (cit. on p. 12).

[Boy+18] Nils Boysen, Dirk Briskorn, Stefan Fedtke, and Stefan Schwerdfeger. "Drone delivery from trucks: Drone scheduling for given truck routes." *Networks* 72.4 (2018), pp. 506–527 (cit. on p. 48).

[CAR18] Chun Cheng, Yossiri Adulyasak, and Louis-Martin Rousseau. *Formulations and exact algorithms for drone routing problem*. CIRRELT, Centre interuniversitaire de recherche sur les réseaux d'entreprise ..., 2018 (cit. on pp. 2, 53, 56).

[Cdi] *Cdiscount lance son projet Pelican de livraison par drones en milieu urbain.* Available: `https://www.lsa-conso.fr/cdiscount-lance-son-projet-pelican-de-livraison-par-drones-en-milieu-urbain,261514`. 2017 (cit. on p. 22).

[Che15] Eric Cheng. *Aerial photography and videography using drones*. Peachpit Press, 2015 (cit. on p. 10).

[CL18] Yong Sik Chang and Hyun Jung Lee. "Optimal delivery routing with wider drone-delivery areas along a shorter truck-route." *Expert Systems with Applications* 104 (2018), pp. 307–317 (cit. on pp. 46, 55).

[CM82] João Carlos Namorado Climaco and Ernesto Queirós Vieira Martins. "A bicriterion shortest path algorithm." *European Journal of Operational Research* 11.4 (1982), pp. 399–404 (cit. on pp. 92, 113).

[CM99] Prasad Chalasani and Rajeev Motwani. "Approximating capacitated routing and delivery problems." *SIAM Journal on Computing* 28.6 (1999), pp. 2133–2149 (cit. on p. 29).

[Con] *A Short History of Unmanned Aerial Vehicles (UAV).* Available: `https://consortiq.com/short-history-unmanned-aerial-vehicles-uavs/`. 2012 (cit. on p. 7).

[Cor+07] Jean-François Cordeau, Gilbert Laporte, Martin WP Savelsbergh, and Daniele Vigo. "Vehicle routing." *Handbooks in operations research and management science* 14 (2007), pp. 367–428 (cit. on p. 32).

[Cra+10] Teodor Gabriel Crainic, Guido Perboli, Simona Mancini, and Roberto Tadei. "Two-echelon vehicle routing problem: a satellite location analysis." *Procedia-Social and Behavioral Sciences* 2.3 (2010), pp. 5944–5955 (cit. on p. 30).

[Cra17] Arthur P Cracknell. "UAVs: regulations and law enforcement." *International journal of remote sensing* 38.8-10 (2017), pp. 3054–3067 (cit. on p. 12).

[CS17] Youngmin Choi and Paul M Schonfeld. "Optimization of multi-package drone deliveries considering battery capacity." *Proceedings of the 96th Annual Meeting of the Transportation Research Board, Washington, DC, USA*. 2017, pp. 8–12 (cit. on pp. 53, 56).

[CS18] John Gunnar Carlsson and Siyuan Song. "Coordinated logistics with a truck and a drone." *Management Science* 64.9 (2018), pp. 4052–4069 (cit. on pp. 44, 54).

[CSZ17] James F Campbell, Don Sweeney, and Juan Zhang. "Strategic design for delivery with trucks and drones." *Supply Chain Analytics Report SCMA (04 2017)* (2017) (cit. on pp. 47, 55).

[CW64] Geoff Clarke and John W Wright. "Scheduling of vehicles from a central depot to a number of delivery points." *Operations research* 12.4 (1964), pp. 568–581 (cit. on p. 34).

[CYL17] Jie Chen, Fang Ye, and Yibing Li. "Travelling salesman problem for uav path planning with two parallel optimization algorithms." *2017 progress in electromagnetics research symposium-fall (PIERS-FALL)*. IEEE. 2017, pp. 832–837 (cit. on p. 53).

[CZH18] Zhongkai Cai, Zizhen Zhang, and Huang He. "Solving the last mile delivery problem using iterated local search approach." *2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC)*. IEEE. 2018, pp. 1–6 (cit. on p. 37).

[Del+07] Mauro Dell'Amico, Michele Monaci, Corrado Pagani, and Daniele Vigo. "Heuristic approaches for the fleet size and mix vehicle routing problem with time windows." *Transportation Science* 41.4 (2007), pp. 516–526 (cit. on p. 29).

[DH93] G Dueck and New Optimization Heuristics. "The great deluge algorithm and the record-to-record travel." *Journal of Computational Physics* 104.1 (1993), pp. 86–92 (cit. on p. 36).

[Dhl] *DHL International GmbH. (2014, Sep.) DHL parcelcopter launches initial operations for research purposes. [Online].* Available: `http://www.dhl.com/en/press/releases/releases_2014/group/dhl_parcelcopter_launches_initial_operations_for_research_purposes.html`. 2014 (cit. on p. 19).

[DK17] Rami Daknama and Elisabeth Kraus. "Vehicle routing with drones." *arXiv preprint arXiv:1705.06431* (2017) (cit. on p. 47).

[DMC96] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. "Ant system: optimization by a colony of cooperating agents." *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 26.1 (1996), pp. 29–41 (cit. on p. 37).

[DMN19] Mauro Dell'Amico, Roberto Montemanni, and Stefano Novellani. "Drone-assisted deliveries: new formulations for the flying sidekick traveling salesman problem." *Optimization Letters* (2019), pp. 1–32 (cit. on pp. 44, 54).

[DMN20] Mauro Dell'Amico, Roberto Montemanni, and Stefano Novellani. "Matheuristic algorithms for the parallel drone scheduling traveling salesman problem." *Annals of Operations Research* (2020), pp. 1–16 (cit. on pp. 51, 56).

[Dom] *Domino's delivers world's first ever pizza by drone.* Available: `https://www.cnbc.com/2016/11/16/dominos-has-delivered-the-worlds-first-ever-pizza-by-drone-to-a-new-zealand-couple.html`. 2016 (cit. on p. 21).

[Dor+17] Kevin Dorling, Jordan Heinrichs, Geoffrey G Messier, and Sebastian Magierowski. "Vehicle routing problems for drone delivery." *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 47.1 (2017), pp. 70–85 (cit. on pp. 2, 53, 56).

[Dor92] Marco Dorigo. "Optimization, learning and natural algorithms." *PhD Thesis, Politecnico di Milano* (1992) (cit. on p. 37).

[Dpd] *DPDGroup Drone Delivery.* Available: `https://www.dpd.com/home/insights/drone_delivery`. 2014 (cit. on p. 21).

[DR59] George B Dantzig and John H Ramser. "The truck dispatching problem." *Management science* 6.1 (1959), pp. 80–91 (cit. on p. 28).

[Dro] *Drone delivery services growing at extraordinary rate - new Unmanned Airspace survey.* Available: `https://www.unmannedairspace.info/uncategorized/drone-delivery-services-growing-at-extraordinary-rate-new-unmanned-airspace-survey/`. 11 September 2019 (cit. on p. 24).

[DS90] Gunter Dueck and Tobias Scheuer. "Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing." *Journal of computational physics* 90.1 (1990), pp. 161–175 (cit. on p. 36).

[DSC20] Iman Dayarian, Martin Savelsbergh, and John-Paul Clarke. "Same-day delivery with drone resupply." *Transportation Science* 54.1 (2020), pp. 229–249 (cit. on pp. 48, 55).

[DW60] George B Dantzig and Philip Wolfe. "Decomposition principle for linear programs." *Operations research* 8.1 (1960), pp. 101–111 (cit. on p. 33).

[DYKB19] Okan Dukkanci, Bahar Y Kara, and Tolga Bektas. "The drone delivery problem." *Tolga, The Drone Delivery Problem (January 10, 2019)* (2019) (cit. on pp. 49, 50, 55).

[EC19] Tomislav Erdelić and Tonči Carić. "A survey on the electric vehicle routing problem: variants and solution approaches." *Journal of Advanced Transportation* 2019 (2019) (cit. on p. 31).

[EM09] Cao Erbao and Lai Mingyong. "A hybrid differential evolution algorithm to vehicle routing problem with fuzzy demands." *Journal of computational and applied mathematics* 231.1 (2009), pp. 302–310 (cit. on p. 38).

[Esc+12] Christian Eschmann, Chen-Ming Kuo, Chung-Hsin Kuo, and Christian Boller. "Unmanned aircraft systems for remote building inspection and monitoring." *Proceedings of the 6th European Workshop on Structural Health Monitoring, Dresden, Germany*. Vol. 36. 2012, p. 13 (cit. on p. 11).

[Eud] *EU wide rules on drones*. Available: `https://www.easa.europa.eu/newsroom-and-events/press-releases/eu-wide-rules-drones-published`. 11 June 2019 (cit. on p. 16).

[FDG05] Dominique Feillet, Pierre Dejax, and Michel Gendreau. "Traveling salesman problems with profits." *Transportation science* 39.2 (2005), pp. 188–205 (cit. on p. 31).

[Fei10] Dominique Feillet. "A tutorial on column generation and branch-and-price for vehicle routing problems." *4or* 8.4 (2010), pp. 407–424 (cit. on p. 33).

[FJ81] Marshall L Fisher and Ramchandran Jaikumar. "A generalized assignment heuristic for vehicle routing." *Networks* 11.2 (1981), pp. 109–124 (cit. on p. 34).

[FJF15] Lester Randolph Ford Jr and Delbert Ray Fulkerson. *Flows in networks*. Princeton university press, 2015 (cit. on p. 72).

[FP20] Júlia Cária de Freitas and Puca Huachi Vaz Penna. "A variable neighborhood search for flying sidekick traveling salesman problem." *International Transactions in Operational Research* 27.1 (2020), pp. 267–290 (cit. on pp. 44, 54).

[FR89] Thomas A Feo and Mauricio GC Resende. "A probabilistic heuristic for a computationally difficult set covering problem." *Operations research letters* 8.2 (1989), pp. 67–71 (cit. on p. 37).

[Fuk+06] Ricardo Fukasawa, Humberto Longo, Jens Lysgaard, Marcus Poggi De Aragão, Marcelo Reis, Eduardo Uchoa, and Renato F Werneck. "Robust branch-and-cut-and-price for the capacitated vehicle routing problem." *Mathematical programming* 106.3 (2006), pp. 491–511 (cit. on p. 33).

[FW12] Rachel L Finn and David Wright. "Unmanned aircraft systems: Surveillance, ethics and privacy in civil applications." *Computer Law & Security Review* 28.2 (2012), pp. 184–194 (cit. on p. 12).

[GH88] David E Goldberg and John H Holland. "Genetic algorithms and machine learning." *Machine learning* 3.2 (1988), pp. 95–99 (cit. on p. 37).

[Gha+18] Hakim Ghazzai, Abdullah Kadri, Mahdi Ben Ghorbel, and Hamid Menouar. "Optimal sequential and parallel UAV scheduling for multi-event applications." *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*. IEEE. 2018, pp. 1–6 (cit. on p. 54).

[Glo77] Fred Glover. "Heuristics for integer programming using surrogate constraints." *Decision sciences* 8.1 (1977), pp. 156–166 (cit. on p. 38).

[Glo86] Fred Glover. "Future paths for integer programming and links to artificial intelligence." *Computers & operations research* 13.5 (1986), pp. 533–549 (cit. on p. 36).

[Glo89] Fred Glover. "Tabu search—part I." *ORSA Journal on computing* 1.3 (1989), pp. 190–206 (cit. on p. 36).

[Glo90] Fred Glover. "Tabu search—part II." *ORSA Journal on computing* 2.1 (1990), pp. 4–32 (cit. on p. 36).

[GLV16] Aldy Gunawan, Hoong Chuin Lau, and Pieter Vansteenwegen. "Orienteering problem: A survey of recent variants, solution approaches and applications." *European Journal of Operational Research* 255.2 (2016), pp. 315–332 (cit. on p. 31).

[GM16] Shriram Ganesh and Jose Roberto Menendez. *Methods, systems and devices for delivery drone security.* US Patent 9,359,074. 2016 (cit. on p. 2).

[GM74] Billy E Gillett and Leland R Miller. "A heuristic algorithm for the vehicle-dispatch problem." *Operations research* 22.2 (1974), pp. 340–349 (cit. on p. 34).

[Gol+84] Bruce Golden, Arjang Assad, Larry Levy, and Filip Gheysens. "The fleet size and mix vehicle routing problem." *Computers & Operations Research* 11.1 (1984), pp. 49–66 (cit. on p. 30).

[GRW08] Bruce L Golden, Subramanian Raghavan, and Edward A Wasil. *The vehicle routing problem: latest advances and new challenges.* Vol. 43. Springer Science & Business Media, 2008 (cit. on p. 32).

[GT18] Anne Goodchild and Jordan Toy. "Delivery by drone: An evaluation of unmanned aerial vehicle technology in reducing CO2 emissions in the delivery service industry." *Transportation Research Part D: Transport and Environment* 61 (2018), pp. 58–67 (cit. on p. 2).

[Gul17] Timothy Gulden. "The Energy Implications of Drones for Package Delivery" (2017) (cit. on p. 2).

[Ha+15] Quang Minh Ha, Yves Deville, Quang Dung Pham, and Minh Hoang Ha. "Heuristic methods for the traveling salesman problem with drone." *Comput. Sci* (2015) (cit. on pp. 44, 45, 54).

[Ha+18] Quang Minh Ha, Yves Deville, Quang Dung Pham, and Minh Hoàng Hà. "On the min-cost traveling salesman problem with drone." *Transportation Research Part C: Emerging Technologies* 86 (2018), pp. 597–621 (cit. on pp. 45, 47, 54).

[Ham18] Andy M Ham. "Integrated scheduling of m-truck, m-drone, and m-depot constrained by time-window, drop-pickup, and m-visit using constraint programming." *Transportation Research Part C: Emerging Technologies* 91 (2018), pp. 1–14 (cit. on pp. 52, 56).

[Hel00] Keld Helsgaun. "An effective implementation of the Lin–Kernighan traveling salesman heuristic." *European Journal of Operational Research* 126.1 (2000), pp. 106–130 (cit. on pp. 89, 92, 110, 112).

[Hol+92] John Henry Holland et al. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992 (cit. on p. 37).

[Int] *A Brief History of Drones: The Remote Controlled Unmanned Aerial Vehicles (UAVs)*. Available: `https://consortiq.com/short-history-unmanned-aerial-vehicles-uavs/`. 20 June 2020 (cit. on p. 7).

[Jon17] Therese Jones. *International commercial drone regulation and drone delivery services*. Tech. rep. RAND Corporation, 2017 (cit. on p. 16).

[Kar72] Richard M Karp. "Reducibility among combinatorial problems." In: *Complexity of computer computations*. Springer, 1972, pp. 85–103 (cit. on p. 61).

[KLA19] Ines Khoufi, Anis Laouiti, and Cedric Adjih. "A Survey of Recent Extended Variants of the Traveling Salesman and Vehicle Routing Problems for Unmanned Aerial Vehicles." *Drones* 3.3 (2019), p. 66 (cit. on p. 40).

[KM17] Abhishake Kundu and Timothy I Matis. "A delivery time reduction heuristic using drones under windy conditions." *IIE Annual Conference. Proceedings*. Institute of Industrial and Systems Engineers (IISE). 2017, pp. 1864–1869 (cit. on p. 44).

[KP12] Suresh Nanda Kumar and Ramasamy Panneerselvam. "A Survey on the Vehicle Routing Problem and Its Variants." *Intelligent Information Management* 4.3 (2012), pp. 66–74 (cit. on p. 28).

[KXX03] Ikou Kaku, Yiyong Xiao, and Guoping Xia. "The deterministic annealing algorithms for vehicle routing problems." *International Journal of Smart Engineering System Design* 5.4 (2003), pp. 327–339 (cit. on p. 36).

[Kyt+07] Jari Kytöjoki, Teemu Nuortio, Olli Bräysy, and Michel Gendreau. "An efficient variable neighborhood search heuristic for very large scale vehicle routing problems." *Computers & operations research* 34.9 (2007), pp. 2743–2757 (cit. on p. 37).

[Lap+00] Gilbert Laporte, Michel Gendreau, J-Y Potvin, and Frédéric Semet. "Classical and modern heuristics for the vehicle routing problem." *International transactions in operational research* 7.4-5 (2000), pp. 285–300 (cit. on p. 34).

[Lap09] Gilbert Laporte. "Fifty years of vehicle routing." *Transportation science* 43.4 (2009), pp. 408–416 (cit. on p. 32).

[Lap92] Gilbert Laporte. "The vehicle routing problem: An overview of exact and approximate algorithms." *European journal of operational research* 59.3 (1992), pp. 345–358 (cit. on p. 28).

[LD60] Ailsa H Land and Alison G Doig. "An automatic method of solving discrete programming problems." *Econometrica: Journal of the Econometric Society* (1960), pp. 497–520 (cit. on p. 32).

[Lin+15] Julie Linchant, Jonathan Lisein, Jean Semeki, Philippe Lejeune, and Cédric Vermeulen. "Are unmanned aircraft systems (UASs) the future of wildlife monitoring? A review of accomplishments and challenges." *Mammal Review* 45.4 (2015), pp. 239–252 (cit. on p. 12).

[Lin65] Shen Lin. "Computer solutions of the traveling salesman problem." *Bell System Technical Journal* 44.10 (1965), pp. 2245–2269 (cit. on pp. 28, 35).

[Lip] Andrew Liptak. *7-Eleven just made the first commercial delivery by drone. [Online].* Available: `http://www.theverge.com/2016/7/23/12262468/7-11-first-retailer-deliver-food-drone`. 23 July 2016 (cit. on p. 21).

[Liu+20] Yao Liu, Zhong Liu, Jianmai Shi, Guohua Wu, and Witold Pedrycz. "Two-Echelon Routing Problem for Parcel Delivery by Cooperated Truck and Drone." *IEEE Transactions on Systems, Man, and Cybernetics: Systems* (2020) (cit. on pp. 45, 54).

[LK73] Shen Lin and Brian W Kernighan. "An effective heuristic algorithm for the traveling-salesman problem." *Operations research* 21.2 (1973), pp. 498–516 (cit. on p. 35).

[LLS17] Zhihao Luo, Zhong Liu, and Jianmai Shi. "A two-echelon cooperated routing problem for a ground vehicle and its carried unmanned aerial vehicle." *Sensors* 17.5 (2017), p. 1144 (cit. on pp. 50, 55).

[LMS03] Helena R Lourenço, Olivier C Martin, and Thomas Stützle. "Iterated local search." In: *Handbook of metaheuristics*. Springer, 2003, pp. 320–353 (cit. on p. 111).

[LMS19] Helena R Lourenço, Olivier C Martin, and Thomas Stützle. "Iterated local search: Framework and applications." In: *Handbook of metaheuristics*. Springer, 2019, pp. 129–168 (cit. on p. 37).

[Loh17] Andrew J Lohn. "What's the Buzz?: The City-Scale Impacts of Drone Delivery" (2017) (cit. on p. 2).

[LPRC01] Philippe Lacomme, Christian Prins, and Wahiba Ramdane-Chérif. "A genetic algorithm for the capacitated arc routing problem and its extensions." *Workshops on Applications of Evolutionary Computation*. Springer. 2001, pp. 473–483 (cit. on p. 35).

[Lys97] Jens Lysgaard. "Clarke & Wright's savings algorithm." *Department of Management Science and Logistics, The Aarhus School of Business* 44 (1997) (cit. on p. 98).

[LYY99] Sushil J Louis, Xiangying Yin, and Zhen Ya Yuan. "Multiple vehicle routing with time windows using genetic algorithms." *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*. Vol. 3. IEEE. 1999, pp. 1804–1808 (cit. on p. 29).

[Mar+17] Mario Marinelli, Leonardo Caggiani, Michele Ottomanelli, and Mauro Dell'Orco. "En-route truck–drone parcel delivery for optimal vehicle routing strategies." *IET Intelligent Transport Systems* 12.4 (2017), pp. 253–261 (cit. on pp. 44, 54).

[MC15] Chase C Murray and Amanda G Chu. "The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery." *Transportation Research Part C: Emerging Technologies* 54 (2015), pp. 86–109 (cit. on pp. 2, 3, 40, 43, 45, 51, 54, 56, 59, 77, 78, 86, 90, 97–99, 106, 137).

[MF+16] Sergio Mourelo Ferrandez, Timothy Harbison, Troy Webwer, Robert Sturges, and Robert Rich. "Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm." *Journal of Industrial Engineering and Management* 9.2 (2016), pp. 374–388 (cit. on pp. 44, 46, 54).

[MH97] Nenad Mladenović and Pierre Hansen. "Variable neighborhood search." *Computers & operations research* 24.11 (1997), pp. 1097–1100 (cit. on p. 37).

[Min05] Aristide Mingozzi. "The multi-depot periodic vehicle routing problem." *International Symposium on Abstraction, Reformulation, and Approximation*. Springer. 2005, pp. 347–350 (cit. on p. 30).

[MJ76] RH Mole and SR Jameson. "A sequential route-building algorithm employing a generalised savings criterion." *Journal of the Operational Research Society* 27.2 (1976), pp. 503–511 (cit. on p. 34).

[ML04] Silvia Mazzeo and Irene Loiseau. "An ant colony algorithm for the capacitated vehicle routing." *Electronic Notes in Discrete Mathematics* 18 (2004), pp. 181–186 (cit. on p. 38).

[Mon16] Jose-Alejandro Montoya. "Electric Vehicle Routing Problems: models and solution approaches." PhD thesis. Angers, 2016 (cit. on p. 31).

[MR20] Chase C Murray and Ritwik Raj. "The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones." *Transportation Research Part C: Emerging Technologies* 110 (2020), pp. 368–398 (cit. on pp. 47, 55).

[MS+17] Raïssa G Mbiadou Saleu, Laurent Deroussi, Dominique Feillet, Nathalie Grangeon, and Alain Quilliot. "Optimization of urban delivery systems with drones." in: 6th meeting of the EURO Working Group on Vehicle Routing and Logistics optimization (VEROLOG). 2017 (cit. on p. 5).

[MS+18] Raïssa G Mbiadou Saleu, Laurent Deroussi, Dominique Feillet, Nathalie Grangeon, and Alain Quilliot. "An iterative two-step heuristic for the parallel drone scheduling traveling salesman problem." *Networks* 72.4 (2018), pp. 459–474 (cit. on pp. 4, 56, 107).

[MS+19] Raïssa G Mbiadou Saleu, Laurent Deroussi, Dominique Feillet, Nathalie Grangeon, and Alain Quilliot. "Heuristic and dynamic programming for Parallel Drone Scheduling with Multiple Drones and Vehicles." in: 7th meeting of the EURO Working Group on Vehicle Routing and Logistics optimization (VEROLOG). 2019 (cit. on p. 4).

[MS+20] Raïssa G Mbiadou Saleu, Laurent Deroussi, Dominique Feillet, Nathalie Grangeon, and Alain Quilliot. "The Parallel Drone Scheduling Problem with Multiple Drones and Vehicles." Submitted to the European Journal of Operational Research (EJOR). 2020 (cit. on pp. 4, 136).

[MSW15] Neil Mathew, Stephen L Smith, and Steven L Waslander. "Planning paths for package delivery in heterogeneous multirobot teams." *IEEE Transactions on Automation Science and Engineering* 12.4 (2015), pp. 1298–1308 (cit. on pp. 48, 55).

[Or77] Ilhan Or. "Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking." PhD thesis. Northwestern University, Evanston, IL, 1977 (cit. on p. 35).

[Osm93] Ibrahim Hassan Osman. "Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem." *Annals of operations research* 41.4 (1993), pp. 421–451 (cit. on p. 36).

[Oth+17] Mohd Shahrizan bin Othman, Aleksandar Shurbevski, Yoshiyuki Karuno, and Hiroshi Nagamochi. "Routing of carrier-vehicle systems with dedicated last-stretch delivery vehicle and fixed carrier route." *Journal of Information Processing* 25 (2017), pp. 655–666 (cit. on pp. 49, 55).

[Ott+18] Alena Otto, Niels Agatz, James Campbell, Bruce Golden, and Erwin Pesch. "Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey." *Networks* 72.4 (2018), pp. 411–458 (cit. on p. 40).

[PC77] Christos H. Papadimitriou and PAPADIMITRIOU CH. "The Euclidean traveling salesman problem is NP-complete." (1977) (cit. on p. 61).

[PDH08] Sophie N Parragh, Karl F Doerner, and Richard F Hartl. "A survey on pickup and delivery models part ii: Transportation between pickup and delivery locations." *Journal für Betriebswirtschaft* 58.2 (2008), pp. 81–117 (cit. on p. 29).

[Per+02] Francisco B Pereira, Jorge Tavares, Penousal Machado, and Ernesto Costa. "GVR: a new genetic representation for the vehicle routing problem." *Irish Conference on Artificial Intelligence and Cognitive Science*. Springer. 2002, pp. 95–102 (cit. on p. 29).

[PG17] Luigi Di Puglia Pugliese and Francesca Guerriero. "Last-Mile Deliveries by Using Drones and Classical Vehicles." *International Conference on Optimization and Decision Science*. Springer. 2017, pp. 557–565 (cit. on pp. 48, 55).

[PG20] Stefan Poikonen and Bruce Golden. "Multi-visit drone routing problem." *Computers & Operations Research* 113 (2020), p. 104802 (cit. on pp. 48, 55).

[PGW19] Stefan Poikonen, Bruce Golden, and Edward A Wasil. "A branch-and-bound approach to the traveling salesman problem with a drone." *INFORMS Journal on Computing* 31.2 (2019), pp. 335–346 (cit. on pp. 44, 54).

[PKS18] Jiyoon Park, Solhee Kim, and Kyo Suh. "A Comparative Analysis of the Environmental Benefits of Drone-Based Delivery Services in Urban and Rural Areas." *Sustainability* 10.3 (2018), p. 888 (cit. on p. 2).

[Pon16] Andrea Ponza. "Optimization of drone-assisted parcel delivery" (2016) (cit. on pp. 44, 54).

[Pot09] Jean-Yves Potvin. "State-of-the art review—Evolutionary algorithms for vehicle routing." *INFORMS Journal on computing* 21.4 (2009), pp. 518–548 (cit. on p. 37).

[PR87] Manfred Padberg and Giovanni Rinaldi. "Optimization of a 532-city symmetric traveling salesman problem by branch and cut." *Operations Research Letters* 6.1 (1987), pp. 1–7 (cit. on p. 33).

[PR91] Manfred Padberg and Giovanni Rinaldi. "A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems." *SIAM review* 33.1 (1991), pp. 60–100 (cit. on p. 72).

[Pri04] Christian Prins. "A simple and effective evolutionary algorithm for the vehicle routing problem." *Computers & operations research* 31.12 (2004), pp. 1985–2002 (cit. on pp. 35, 37, 38).

[PZC16] Sangyoung Park, Licong Zhang, and Samarjit Chakraborty. "Design space exploration of drone infrastructure for large-scale delivery services." *Proceedings of the 35th International Conference on Computer-Aided Design*. ACM. 2016, p. 72 (cit. on p. 2).

[Ral03] Ted K Ralphs. "Parallel branch and cut for capacitated vehicle routing." *Parallel Computing* 29.5 (2003), pp. 607–629 (cit. on p. 29).

[RBL96] Jacques Renaud, Fayez F Boctor, and Gilbert Laporte. "An improved petal heuristic for the vehicle routeing problem." *Journal of the operational Research Society* 47.2 (1996), pp. 329–336 (cit. on p. 34).

[Reg] *Drone Regulations worldwide*. Available: `https://drone-traveller.com/drone-regulations-worldwide/`. 16 August 2019 (cit. on p. 16).

[Res+15] Agoston Restas et al. "Drone applications for supporting disaster management." *World Journal of Engineering and Technology* 3.03 (2015), p. 316 (cit. on p. 11).

[RR19] Roberto Roberti and Mario Ruthmair. "Exact methods for the traveling salesman problem with drone." *Optimization Online* (2019) (cit. on pp. 44, 54).

[RR94] César Rego and Catherine Roucairol. "Le problème de tournées de vehicules: Étude et résolution approchée." PhD thesis. INRIA, 1994 (cit. on p. 36).

[San15] Chris Sandbrook. "The social implications of using drones for biodiversity conservation." *Ambio* 44.4 (2015), pp. 636–647 (cit. on p. 12).

[Sav97] Martin Savelsbergh. "A branch-and-price algorithm for the generalized assignment problem." *Operations research* 45.6 (1997), pp. 831–841 (cit. on p. 33).

[Sch16] Fred Schenkelberg. "How reliable does a delivery drone have to be?" *2016 annual reliability and maintainability symposium (RAMS)*. IEEE. 2016, pp. 1–5 (cit. on p. 2).

[Sha+19] Hazim Shakhatreh, Ahmad H Sawalmeh, Ala Al-Fuqaha, Zuochao Dou, Eyad Almaita, Issa Khalil, Noor Shamsiah Othman, Abdallah Khreishah, and Mohsen Guizani. "Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges." *IEEE Access* 7 (2019), pp. 48572–48634 (cit. on p. 10).

[Sin] *SingPost Collaborates with IDA to Develop First Drone for Mail Delivery*. Available: `https://www.singpost.com/about-us/news-releases/singpost-collaborates-ida-develop-first-drone-mail-delivery`. 2015 (cit. on p. 23).

[SLC16] Khin Thida San, Eun Young Lee, and Yoon Seok Chang. "The delivery assignment solution for swarms of UAVs dealing with multi-dimensional chromosome representation of genetic algorithm." *Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), IEEE Annual*. IEEE. 2016, pp. 1–7 (cit. on pp. 54, 56).

[SMW18] Daniel Schermer, Mahdi Moeini, and Oliver Wendt. "Algorithms for solving the vehicle routing problem with drones." *Asian Conference on Intelligent Information and Database Systems*. Springer. 2018, pp. 352–361 (cit. on p. 47).

[SR14] Kaarthik Sundar and Sivakumar Rathinam. "Algorithms for Routing an Unmanned Aerial Vehicle in the Presence of Refueling Depots." *IEEE Trans. Automation Science and Engineering* 11.1 (2014), pp. 287–294 (cit. on pp. 52, 56).

[SSW18] Maximilian Schiffer, Sebastian Stütz, and Grit Walther. "Electric commercial vehicles in mid-haul logistics networks." In: *Behaviour of Lithium-Ion Batteries in Electric Vehicles*. Springer, 2018, pp. 153–173 (cit. on p. 31).

[SW06] Stephan Scheuerer and Rolf Wendolsky. "A scatter search heuristic for the capacitated clustering problem." *European Journal of Operational Research* 169.2 (2006), pp. 533–547 (cit. on p. 38).

[Swi] *Further drone flights in Lugano.* Available: `https://www.post.ch/en/about-us/company/media/press-releases/2017/further-drone-flights-in-lugano`. 2017 (cit. on p. 23).

[Tah+12] Khairul Nizam Tahar, Anuar Ahmad, Wan Abdul Aziz Wan Mohd Akib, and Wan Mohd Naim Wan Mohd. "Aerial mapping using autonomous fixed-wing unmanned aerial vehicle." *Signal Processing and its Applications (CSPA), 2012 IEEE 8th International Colloquium on*. IEEE. 2012, pp. 164–168 (cit. on p. 11).

[Tai99] Éric D Taillard. "A heuristic column generation method for the heterogeneous fleet VRP." *RAIRO-Operations Research-Recherche Opérationnelle* 33.1 (1999), pp. 1–14 (cit. on p. 30).

[Tan+01] Kay Chen Tan, Loo Hay Lee, QL Zhu, and Ke Ou. "Heuristic methods for vehicle routing problem with time windows." *Artificial intelligence in Engineering* 15.3 (2001), pp. 281–295 (cit. on p. 36).

[TDD18] Phan Anh Tu, Nguyen Tuan Dat, and Pham Quang Dung. "Traveling salesman problem with multiple drones." *Proceedings of the Ninth International Symposium on Information and Communication Technology*. ACM. 2018, pp. 46–53 (cit. on pp. 46, 55).

[THG97] EG Talbi, Z Hafidi, and JM Geib. "Parallel adaptive tabu search for large optimization problems" (1997) (cit. on p. 38).

[Thi09] Dirk Thierens. "Adaptive operator selection for iterated local search." *International Workshop on Engineering Stochastic Local Search Algorithms*. Springer. 2009, pp. 140–144 (cit. on p. 37).

[TLK18] Maryam Torabbeigi, Gino J Lim, and Seon Jin Kim. "Drone delivery schedule optimization considering the reliability of drones." *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE. 2018, pp. 1048–1053 (cit. on p. 54).

[TP93] Paul M Thompson and Harilaos N Psaraftis. "Cyclic transfer algorithm for multivehicle routing and scheduling problems." *Operations research* 41.5 (1993), pp. 935–946 (cit. on p. 36).

[Tro+18] Asma Troudi, Sid-Ali Addouche, Sofiene Dellagi, and Abderrahman Mhamedi. "Sizing of the Drone Delivery Fleet Considering Energy Autonomy." *Sustainability* 10.9 (2018), p. 3344 (cit. on p. 53).

[TV02] Paolo Toth and Daniele Vigo. *The vehicle routing problem*. SIAM, 2002 (cit. on pp. 32–34).

[TV14] Paolo Toth and Daniele Vigo. *Vehicle routing: problems, methods, and applications*. SIAM, 2014 (cit. on pp. 28, 32).

[TZP10] Jiafu Tang, Jun Zhang, and Zhendong Pan. "A scatter search algorithm for solving vehicle routing problem with loading cost." *Expert Systems with Applications* 37.6 (2010), pp. 4073–4083 (cit. on p. 38).

[UDO16] Gheorghe Udeanu, Alexandra Dobrescu, and Mihaela Oltean. "Unmanned aerial vehicle in military operations." *Scientific Research & Education in the Air Force-AFASES* 1 (2016), pp. 199–205 (cit. on p. 9).

[Ukr] *Ukrainian post successfully tests drone delivery.* Available: `https://112.international/economics/ukrainian-post-successfully-tests-drone-delivery-5482.html`. 2016 (cit. on p. 23).

[Ups] *UPS tests launching drones from trucks equipped with battery chargers*. Available: `https://abcnews.go.com/Business/ups-tests-launching-drones-trucks-equipped-battery-chargers/story?id=45650029`. 2017 (cit. on p. 20).

[UT18] Marlin W Ulmer and Barrett W Thomas. "Same-day delivery with heterogeneous fleets of drones and vehicles." *Networks* 72.4 (2018), pp. 475–505 (cit. on pp. 52, 56, 138).

[VA95] Marcus Gerardus Aldegonda Verhoeven and Emile HL Aarts. "Parallel local search." *Journal of Heuristics* 1.1 (1995), pp. 43–65 (cit. on p. 38).

[VB94] Alex Van Breedam. *An Analysis of the Behavior of Heuristics for the Vehicle Routing Problem for a Selectrion of Problems with Vehicle-related, Customer-related, and Time-related Constraints*. RUCA, 1994 (cit. on p. 35).

[VB96] Alex Van Breedam. *An analysis of the effect of local improvement operators in genetic algorithms and simulated annealing for the vehicle routing problem*. RUCA Belgium, 1996 (cit. on p. 35).

[Vil+18] Juan G Villegas, Christelle Guéret, Jorge E Mendoza, and Alejandro Montoya. "The technician routing and scheduling problem with conventional and electric vehicle." In: *Technical Report*. 2018 (cit. on p. 31).

[Vos+19] Reza Vosooghi, Jakob Puchinger, Marija Jankovic, and Anthony Vouillon. "Shared autonomous vehicle simulation and service design." *Transportation Research Part C: Emerging Technologies* 107 (2019), pp. 15–33 (cit. on pp. 50, 55).

[VSVO11] Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. "The orienteering problem: A survey." *European Journal of Operational Research* 209.1 (2011), pp. 1–10 (cit. on p. 31).

[Wei95] Gideon Weiss. *Scheduling: Theory, algorithms, and systems*. 1995 (cit. on pp. 97, 98, 112).

[Win] *Transforming the way goods are transported*. Available: `https://x.company/projects/wing/`. 2014 (cit. on p. 19).

[WPG17] Xingyin Wang, Stefan Poikonen, and Bruce Golden. "The vehicle routing problem with drones: several worst-case results." *Optimization Letters* 11.4 (2017), pp. 679–697 (cit. on pp. 47, 55).

[WS19] Zheng Wang and Jiuh-Biing Sheu. "Vehicle routing problem with drones." *Transportation research part B: methodological* 122 (2019), pp. 350–364 (cit. on pp. 47, 55).

[WT10] Sonia Waharte and Niki Trigoni. "Supporting search and rescue operations with UAVs." *Emerging Security Technologies (EST), 2010 International Conference on*. IEEE. 2010, pp. 142–147 (cit. on p. 11).

[YO18] Emine Es Yurek and H Cenk Ozmutlu. "A decomposition-based iterative optimization algorithm for traveling salesman problem with drone." *Transportation Research Part C: Emerging Technologies* 91 (2018), pp. 249–262 (cit. on pp. 44, 45, 54).

[Yoo18] Justin J Yoon. "The traveling salesman problem with multiple drones: an optimization model for last-mile delivery." PhD thesis. Massachusetts Institute of Technology, 2018 (cit. on pp. 46, 55).

[Zha+18] Shuai Zhang, Yuvraj Gajpal, SS Appadoo, and MMS Abdulkader. "Electric vehicle routing problem with recharging stations for minimizing energy consumption." *International Journal of Production Economics* 203 (2018), pp. 404–413 (cit. on p. 31).

[Zhu00] Kenny Qili Zhu. "A new genetic algorithm for VRPTW." *Proceedings of the international conference on artificial intelligence*. Citeseer. 2000 (cit. on p. 29).

[Zip] *The Future of Healthcare is Out for Delivery. [Online]*. Available: `http://flyzipline.com/product/`. 2016 (cit. on p. 21).

[ZK10] Emmanouil E Zachariadis and Chris T Kiranoudis. "A strategy for reducing the computational complexity of local search-based methods for the vehicle routing problem." *Computers & operations research* 37.12 (2010), pp. 2089–2105 (cit. on p. 36).

[ZK12] Chunhua Zhang and John M Kovacs. "The application of small unmanned aerial systems for precision agriculture: a review." *Precision agriculture* 13.6 (2012), pp. 693–712 (cit. on p. 11).

[ZZ09] Tong Zhen and Qiuwen Zhang. "A hybrid metaheuristic algorithm for the multi-depot vehicle routing problem with time windows." *2009 International Conference on Networks Security, Wireless Communications and Trusted Computing*. Vol. 2. IEEE. 2009, pp. 798–801 (cit. on p. 38).