OPTIMIZATION OF LAST-MILE DELIVERIES WITH SYNCHRONOUS TRUCK AND DRONES

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF

MASTER OF SCIENCE

IN

INDUSTRIAL ENGINEERING

By

Aysu Özel

July 2020

OPTIMIZATION OF LAST-MILE DELIVERIES WITH SYNCHRONOUS TRUCK AND DRONES

By Aysu Özel

July 2020

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Bahar Yetiş(Advisor)

Oya Karaşan

Amine Gizem Tiniç

Approved for the Graduate School of Engineering and Science:

Ezhan Karaşan
Director of the Graduate School

ABSTRACT

OPTIMIZATION OF LAST-MILE DELIVERIES WITH SYNCHRONOUS TRUCK AND DRONES

Aysu Özel
M.S. in Industrial Engineering
Advisor: Bahar Yetiş
July 2020

Deployment of drones in delivery operations has been attracting a growing interest from commercial sector due to its prospective advantages for the distribution systems. Motivated by the widespread adoption of drones in last-mile delivery, we introduce the minimum cost traveling salesman problem with multiple drones, where a truck and multiple drones work in synchronization to deliver parcels to customers. In this problem, we aim to find an optimal delivery plan for the truck and drones operating in tandem with the objective of minimizing the total operational cost including the vehicles' operating and waiting costs. We provide flow based and cut based mixed integer linear programming formulations along with valid inequalities. Since the connectivity constraints in the cut based formulation and the proposed valid inequalities are exponential in the size of the problem, we devise different branch-and-cut schemes to solve our problem. We also provide an alternative solution methodology using another cut based formulation with undirected route variables. To compare our formulations/algorithms and to demonstrate their competitiveness, we conduct computational experiments on a set of instances. The results indicate the superiority of utilizing branch-and-cut methodology over flow based formulation and good computational performance of the proposed algorithms in comparison to existing exact solution approaches in the literature. We also conduct sensitivity analyses on problem parameters and discuss their effects on the optimal solutions.

Keywords: Traveling salesman problem, delivery, drones, synchronization, branch-and-cut.

ÖZET

EŞZAMANLI KAMYON VE İNSANSIZ HAVA ARAÇLARI İLE SON MİL TESLİMATLARI ENİYİLEMESİ

Aysu Özel
Endüstri Mühendisliği, Yüksek Lisans
Tez Danışmanı: Bahar Yetiş
Temmuz 2020

Dağıtım sistemlerindeki olası avantajlarından dolayı, insansız hava araçlarının teslimat operasyonlarında görevlendirilmesi ticari sektör tarafından artan bir ilgi görmektedir. Bu çalışmada, insansız hava araçlarının son mil teslimatında yaygın kabul görmesi motivasyonuyla bir kamyon ve birden çok insansız hava aracının eşzamanlı şekilde müşterilere teslimat yaptığı, çoklu insansız hava aracı ile en düşük maliyetli gezgin satıcı problemi sunulmuştur. Bu problemde, araçların çalışma ve bekleme maliyetlerini içeren toplam işletme maliyetini enazlama amaç fonksiyonu ile, birlikte çalışan kamyon ve insansız hava araçları için en iyi teslimat planının bulunması hedeflenmiştir. Akış ve kesi tabanlı karma tamsayılı doğrusal programlama formülasyonları ve geçerli eşitsizlikler önerilmiştir. Kesi tabanlı formülasyondaki alt tur eleme kısıtları ve önerilen eşitsizlikler üstel sayıda olduğundan, problemi çözmek için dal ve kesi algoritmaları kullanılmıştır. Aynı zamanda, yönsüz rota değişkenleri içeren bir başka kesi tabanlı formülasyon kullanan alternatif bir çözüm yöntemi sunulmuştur. Geliştirilen algoritmaları karşılaştırmak ve performanslarını değerlendirmek amacıyla sayısal analizler yapılmıştır. Bunların sonuçları, dal ve kesi yönteminin akış modeline kıyasla problemi çok daha kısa sürede çözdüğünü ve sunulan algoritmaların literatürdeki kesin çözüm yöntemlerine kıyasla iyi performans sergilediğini göstermiştir. Yapılan duyarlılık analizleriyle ise problem parametrelerinin en iyi çözüm üzerindeki etkileri incelenmiştir.

Anahtar sözcükler: Gezgin satıcı problemi, teslimat, insansız hava aracı, eşzamanlama, dal ve kesi.

Acknowledgement

I would like to express my sincere gratitude to my advisor Prof. Bahar Yetiş for her continuous support and guidance during my study. I consider myself privileged to have the opportunity to work under her supervision.

I am genuinely grateful to Prof. Oya Karaşan and Asst. Prof. Amine Gizem Tiniç for accepting to read and review my thesis. Their remarks and suggestions have been very important and helpful.

I would like to thank all the faculty members of the Department of Industrial Engineering for preparing me for the following steps of my academic journey. I am thankful to all my fellow graduate students and my officemates, particularly Deniz Akkaya, for their precious friendships and many pleasant memories. I am also truly thankful to Çağla Dursunoğlu for being such a great companion during the last two years and always cheering me up when I am down.

I would like to extend my sincere thanks to my housemates Deniz Emre and İrem Aybala Gülen for standing by me all the time and becoming my family in Ankara. My dearest friends Merve Balıkçılar, Beyza İslamoğlu, Özge Çetiner and Görkem Ünlü also deserve heartfelt thanks for sticking with me since the first day we met.

I am deeply grateful to my parents Sevgi Özel and Kamil Özel, and my brother Alper Özel for their constant, unconditional love and support. I could not thank my family enough for all the sacrifices they made throughout my life, without them I would not be where I am today. I am also grateful to my grandmother Necibe Acar and my aunt Kamile Özel for always supporting me.

Last but not least, I would like to thank my best friend and my love Harun Avcı. Words are certainly not enough to express my gratitude for everything he has done for me. I am thankful that he has always been by my side through good and bad times, and believing in me more than I do.

Contents

1 Introduction **1**

2 Literature Review **5**

3 Problem Description and Flow Based Formulation **14**

4 Cut Based Formulation & Branch-and-Cut Algorithms **24**

4.1 Cut Based Formulation and Valid Inequalities 24

4.2 Branch-and-Cut Algorithms 26

4.2.1 Separation of connectivity constraints 27

4.2.2 Separation of lifted connectivity inequalities 28

5 An Alternative Cut Based Solution Methodology **30**

5.1 An Alternative Cut Based Formulation and Valid Inequalities . . 30

5.2 Branch-and-Cut Algorithms 37

5.2.1 Separation of connectivity constraints 38

5.2.2 Separation of lifted connectivity constraints 39

5.3 A Solution Approach Using (Undir) 40

6 Computational Experiments 45

6.1 Experimental Setting 45

6.2 Implementation details of branch-and-cut schemes 47

6.3 Comparison of the Proposed Algorithms 48

6.4 Sensitivity Analyses 53

6.5 Tests on Larger Instances 62

7 Conclusion 63

A Computational Results 72

List of Figures

3.1 Possible movements of the truck and drones. 16

3.2 Representation of the binary decision variables. 18

5.1 An example where (Undir) is unable to obtain consistency within the directions of drone sorties. 41

6.1 Comparison of standardized CPU times of the algorithms 52

6.2 The optimal solutions of uniform-62-n20 with $1/v = 1.5$, $c_o = 10$, $c_w = 7.5$. 55

6.3 The optimal solutions of uniform-63-n20 with $\alpha = 50^{th}$, $c_o = 4$, $c_w = 2$. 55

6.4 The optimal solutions of uniform-61-n20 with $\alpha = 25^{th}$, $1/v = 2$, $c_o = 10$. 56

List of Tables

2.1 Overview of the literature on truck-drone tandems requiring synchronization and proposing an exact approach 13

3.1 Parameters and decision variables. 17

6.1 Comparison of algorithms in terms of CPU times with $n = 11$ . . 49

6.2 Comparison of algorithms in terms of CPU times with $n = 12$ . . 50

6.3 Comparison of algorithms in terms of CPU times with $n = 13$ . . 51

6.4 Average standardized CPU times of the algorithms relative to DF 53

6.5 Sensitivity analyses on uniform-61-n20 with $n = 15$ 57

6.6 Sensitivity analyses on uniform-62-n20 with $n = 15$ 58

6.7 Sensitivity analyses on uniform-63-n20 with $n = 15$ 59

6.8 Sensitivity analyses on uniform-64-n20 with $n = 15$ 60

6.9 Sensitivity analyses on uniform-65-n20 with $n = 15$ 61

6.10 Experiments with $n = 19$ 62

A.1 Sensitivity analyses on uniform-61-n20 with $n = 14$ 73

A.2 Sensitivity analyses on uniform-62-n20 with $n = 14$ 74

A.3 Sensitivity analyses on uniform-63-n20 with $n = 14$ 75

A.4 Sensitivity analyses on uniform-64-n20 with $n = 14$ 76

A.5 Sensitivity analyses on uniform-65-n20 with $n = 14$ 77

Chapter 1

Introduction

In the past, unmanned aerial vehicles (UAVs), also known as drones, were mainly used in military domain for surveillance and intelligence purposes. During the last decade, due to their potential benefits for the logistics systems, they have attracted considerable attention from logistics service providers aiming for faster and more cost efficient last-mile deliveries.

Back in late 2013, Amazon's CEO Jeff Bezos unveiled an ongoing R&D project, "Amazon Prime Air", which aims to deliver lighter packages from warehouses to customers in at most 30 minutes using drones [1]. Then, the company successfully realized the first Prime Air delivery in December 2016. UPS was also said to be experimenting and evaluating different approaches with its own version of flying parcel carriers for delivering packages [2]. Later, in October 2019, with their "Flight Forward" drones, they received the U.S. Federal Aviation Administration's first full Part 135 Standard certification to operate a drone airline [3]. In August 2014, Google X revealed its own drone-based delivery project, "Project Wing", using a fixed-wing drone for parcel delivery [4]. Additionally, DHL launched a drone delivery service for the first time to deliver small parcels to a car-free German island, nine months after it announced its "Parcelcopter" research project in December 2013 [5].

The idea of using drones for the last-mile delivery is appealing due to drones' advantages over traditional delivery vehicles. For instance, drones can be operated without a human pilot and with lower transportation costs per kilometer [6]. Besides, drones are faster than conventional trucks and they can fly over road networks to avoid traffic congestion. Despite these advantages, because of drones' physical features, there are several restrictions regarding this new delivery approach. Since most drones on the market are battery-powered, their flight endurance, i.e., the maximum length of time that they can spend while they are airborne, is limited by their battery capacity. Additionally, most of the drones considered for delivery operations can carry packages up to a certain weight and size, and typically one package at a time. Thus, they need to turn back to the depot after each delivery. Contrarily, trucks can carry many packages, including heavier and larger ones, and have long range travel capability. However, trucks have higher operational costs and are slower compared to drones.

The drawbacks of both type of vehicles, namely drones and trucks, can be counteracted by the other one's advantages with their combined use in delivery networks, which leads to a novel last-mile delivery approach. More specifically, using delivery trucks and drones in tandem may alleviate drones' endurance limitation by allowing drones to be transported to the locations closer to customers. Thus, it enables to make faster deliveries at lower costs with drones [6]. To this end, different than a direct drone delivery from depot to customers, Workhorse Group in collaboration with the University of Cincinnati Department of Aerospace Engineering adopted a system where delivery trucks are paired with drones, called "HorseFly", which deliver parcels alongside the trucks [7]. In February 2017, Workhorse and UPS collaborated to test a residential package delivery using HorseFly launched from the roof of a delivery truck on actual UPS delivery routes [8]. In this test, while the truck driver continued along the route to make a delivery, the drone flew autonomously to a customer where it delivered a package and returned to the truck. In the mean time, many other logistics service providers have continued to work on integrating drones into conventional delivery approaches. For instance, Amazon has patented several technologies regarding the combined use of conventional delivery vehicles and drones [9].

Deploying drones in last-mile delivery has also received considerable attention from the scientific community in recent years. As pioneers of studying truck-drone tandem problems, Murray and Chu [10] introduced the flying sidekick traveling salesman problem (FSTSP). This problem is a variant of the traditional traveling salesman problem (TSP) where a delivery truck and a drone work in collaboration to deliver parcels. Specifically, the truck departs from the depot carrying a drone and all the parcels to be delivered. Then, the drone is launched from the truck to serve a nearby customer as the truck makes other deliveries simultaneously. After completing its service, the drone turns back to the truck at a customer location different from its launch point, and it is transported by the truck until its next launch location. The authors aim to find a routing plan with the objective of minimizing the time at which all deliveries are completed and both vehicles return to the depot (i.e., the completion time). While the FSTSP and some of its variants aim to minimize the completion time of the route, there are also several papers in the literature considering minimizing the total operational cost. Additionally, different settings including a single truck and multiple drones, and multiple trucks and multiple drones have been investigated as variants of truck-drone tandem delivery problems.

In this study, we consider a variant of the FSTSP where a single truck and multiple drones make deliveries in a synchronized manner. In this variant that we call the minimum cost traveling salesman problem with multiple drones (min-cost TSPMD), we aim to determine an optimal delivery plan such that all customers are served either by the truck or by a drone once, and the total operational cost is minimized. Here, the operational cost includes both the vehicles' operating (transportation) cost and the cost incurred while the truck and drones wait for each other to rendezvous.

Unlike the majority of the papers in the literature considering synchronization between trucks and drones (see, e.g., [10, 11, 12, 13, 14, 15]), we allow launch and retrieval locations of a drone to be the same. That is, the truck might wait the drones at their launch locations to retrieve them or it can travel to its next customer. Moreover, we do not limit the number of drones that can be launched/retrieved at a customer location. Additionally, we do not restrict the number of drones required per delivery plan. Considering these, we model a more flexible setting than existing variants of FSTSP. Our contributions can be summarized as follows. We introduce the min-cost TSPMD, and propose flow based and cut based mixed-integer linear programming (MILP) formulations and valid inequalities. We develop branch-and-cut algorithms to solve the cut based formulation. Additionally, we propose another solution methodology utilizing an alternative cut based formulation with undirected route variables. We conduct a computational study to compare the proposed solution methodologies and sensitivity analyses on problem parameters.

In Chapter 2, we review the related literature on routing a set of trucks and drones operating in coordination. In Chapter 3, we formally define our problem and provide a flow based MILP formulation. In Chapter 4, we provide a cut based formulation of the problem along with a class of valid inequalities. Also, branch-and-cut algorithms devised to solve the problem are presented. In Chapter 5, we propose an alternative solution methodology using another cut based formulation with undirected variables. Details of the formulation, related valid inequalities and branch-and-cut algorithms, and the solution methodology are presented. In Chapter 6, we report the results of our computational study including the comparison of the proposed solution methodologies and sensitivity analyses on problem parameters. Finally, we conclude our study with a summary of our findings and future work suggestions in Chapter 7.

Chapter 2

Literature Review

With the advancements in drone technology, distribution problems involving drones have attracted vast attention in recent years. Based on the presence of ground vehicles (e.g., trucks), those problems can be divided into two main categories. The problems in the first category consider using only drones as a means of transportation with no truck involved whereas the problems in the second category study combined use of drones and trucks. As the latter one is more relevant to our work, we focus on the problems of the second category. We refer the interested reader to the surveys on optimization approaches for civil applications of drones by Otto et al. [16] and a more recent one by Chung et al. [17].

The problems considering combined use of drones and trucks mainly differ based on the types of the vehicles making the deliveries, the number of trucks and drones present in the system, and the type of collaboration between the vehicles. There are several papers in the literature that consider combined use of drones and trucks in which only the drones make the last-mile deliveries. Mathew et al. [18] consider the heterogeneous delivery problem (HDP) in which the role of a single truck is to carry a shipment of packages as well as a drone responsible for performing single package deliveries. The authors propose a solution approach aiming to minimize the total delivery cost where the HDP reduces to the generalized traveling salesman problem that can be solved via existing heuristic methods in the literature. Bin Othman et al. [19] study a special case of the problem investigated by Mathew et al. [18] where a single truck follows a predetermined route. Several variants of this problem in which the truck is allowed or prohibited to wait at the launch location to retrieve the drone are examined. The authors propose an approximation algorithm to find the quickest way of delivering all parcels and study complexity of the variations examined. Similarly, Boysen et al. [20] study a drone scheduling problem for given truck routes. They differentiate the problem depending on the number of drones available and whether take-off and landing locations must be identical. The authors examine computational complexity for each variant and present mixed integer programs.

A problem in which a single truck may launch multiple drones from launch sites along the truck route is considered by Ferrandez et al. [21]. The drones return to the truck before the truck departs for the next launch site. The authors first determine the launch sites by using k-means clustering and then find the truck route as a TSP via a genetic algorithm to minimize the total delivery time. For the same problem, Chang and Lee [22] propose another solution approach based on k-means clustering and TSP modeling.

Peng et al. [23] consider a problem involving multiple drones carried by a single truck. They assume that each drone can deliver multiple parcels and has to be launched from and retrieved at the same location by the truck. They propose a genetic algorithm to minimize the total distance cost and time consumption. In the problem that Karak and Abdelghany [24] propose, a single truck may dispatch multiple drones at stations, whose locations are predetermined, and the drones may visit one or more customers to deliver or pick up their packages. The drones can return to any station along the truck route, which could be the same as or different from the dispatching one. The authors propose a solution methodology to minimize the routing cost of the vehicles. The k-multi-visit drone routing problem where a single truck acts as a mobile depot and recharging platform for the drones that can carry multiple heterogeneous packages is proposed by Poikonen and Golden [25]. In this problem, the drones dispatched from the same launch location need to be retrieved at the same landing location, which is allowed to be the launch location itself, by the truck. When the truck moves between drones' launch and landing locations, it is not allowed to stop or dispatch additional drones. The authors propose a heuristic approach to find minimum route completion time.

As a theoretical study, Carlsson and Song [26] define horsefly routing problem in which a truck serves as a mobile depot for the drone and launch/retrieval locations of the drone are not restricted to customer nodes. Some continuous approximation results in the Euclidean plane and potential benefits of using a drone in tandem with the truck are discussed.

Unlike the papers above, we allow the truck as well as the drones to make deliveries. The problem of combined use of drones and truck where both types of vehicles can make deliveries is first introduced by Murray and Chu [10]. They define two problems: the parallel drone scheduling traveling salesmen problem (PDSTSP) and the flying sidekick traveling salesmen problem (FSTSP). In the PDSTSP, multiple drones and a single truck make deliveries independently where the drones are launched from and retrieved at the depot and the truck serves the remaining customers along a TSP route without any synchronization between the vehicles. Murray and Chu [10] propose an MILP formulation minimizing the time at which both vehicles return to the depot after completing service. They also provide a greedy construction heuristic. Later, an iterative two-step heuristic is proposed by Mbiadou Saleu et al. [27] and a set of matheuristic methods is presented by Dell'Amico et al. [28] for the PDSTSP. Considering multi-truck and multi-depot variants of the PDSTSP, Ham et al. [29] let the drones perform both delivery and pick-up operations and propose a constraint programming approach. In another variant of the PDSTSP studied by Kim and Moon [30], drones are stored in and launched from an additional drone station instead of the depot.

In the FSTSP, parcel deliveries are performed by a single truck and a single drone operating in synchronization, and each customer is served either by the truck or by the drone with the objective of minimizing the time at which both vehicles complete the service and return to the depot. The drone can be launched from the truck at a customer node and the vehicles rendezvous at a different customer location after the drone completes its delivery. Murray and Chu [10] present an MILP formulation and greedy construction heuristics for the problem.

Variants of the FSTSP are studied after the initiating work of Murray and Chu [10]. Agatz et al. [31] consider a variant, which they call TSP with drone (TSP-D). In their setting, different from [10], the truck is allowed to revisit the nodes and the drone may rendezvous with the truck at the node where it is launched from. The authors propose an integer linear programming (ILP) formulation based on the concept of operations and aim to minimize the sum of the costs of the operations chosen which is equivalent to the completion time. They provide several route first-cluster second type heuristics. Yurek and Ozmutlu [32] present an iterative algorithm based on a decomposition approach to minimize delivery completion time for a variant of the FSTSP. Ponza [33] proposes a simulated annealing heuristic for the modified version of the FSTSP formulation and Dell'Amico et al. [34] also present an improved formulation of the FSTSP. In the variant presented by Ha et al. [11], rather than the completion time, the authors aim to minimize operational costs including the transportation expenses and the waiting penalties. As in the FSTSP, the launch and recovery locations for a drone are restricted to be different. They propose an MILP formulation which is an extension of the one proposed in [10] with the adapted objective as well as two heuristic methods. They, in [35], also present a hybrid genetic algorithm for the same problem considering both minimum operational cost and minimum completion time objectives, sequentially.

De Freitas and Penna [36] propose a neighborhood search metaheuristic to solve both the variants of the FSTSP and the TSP-D of [31]. Poikonen et al. [37] present four heuristics for the TSP-D based on the branch-and-bound algorithm. Wang et al. [38] propose an improved non-dominated sorting genetic algorithm to solve a multi-objective variant of the FSTSP considering both the operational cost and completion time. Jeong et al. [39] modify the FSTSP by focusing on two additional features: weight-dependent energy consumption and the resulting range limits for the drone, and circular no fly zones for drones. They propose a two-phase constructive and search heuristic algorithm.

In addition to the variants stated above, there are also several works presenting exact solution methodologies to the FSTSP. Bouman et al. [40] propose a dynamic programming (DP) approach for the TSP-D variant of [31] and a constraint programming approach is presented by Tang et al. [41]. Roberti and Rutmair [42] introduce an MILP formulation and branch-and-price approach for several variants of the TSP-D.

Including our work, several papers in the literature consider a setting in which a single truck and multiple drones are deployed. Yoon [43] studies a problem in which a single truck may launch multiple drones from different customer locations and retrieve them at different ones as well. An MILP formulation with the objective of minimizing the total cost consisting of the operational costs and a fixed cost associated with the drones is provided and tested on instances with up to 10 customers. Unlike our work, a drone is not allowed to be retrieved at the location from which it is launched. As an extension of the TSP-D, Tu et al. [12] introduce the TSP-mD in which a truck carries multiple drones. Aiming to minimize the total transportation cost, they extend the model proposed in [11] for the TSP-mD. The authors also assume that the launch and rendezvous nodes for a drone cannot be the same as in [11]. They propose an adaptive large neighborhood search heuristic. Murray and Raj [13] extend the FSTSP by considering multiple heterogeneous drones working in synchronization with the truck, whose flight endurance is modeled as a function of the drone's battery size, payload, travel distance, and flight phases. They assume that a drone cannot be retrieved at the same location where it is launched from. An MILP formulation minimizing the completion time is provided and only 8-customer instances can be solved to optimality due to computational complexity. They also propose a three-phased heuristic solution approach. With similar assumptions as the FSTSP, Moshref-Javadi et al. [44] consider the same setting with the objective of minimizing the waiting time of customers and call this problem as the simultaneous traveling repairman problem with drones. They propose an MILP formulation and solve the instances with up to 11 nodes to optimality.

In the presence of a single truck and multiple drones, Salama and Srinivas [45] consider partitioning customers into clusters, identifying a focal point for each cluster, and routing the truck through these points such that either a drone or the truck serve each customer. They examine two cases regarding a focal point selection: restricting it to customer locations and allowing it to be anywhere in the delivery area. While the truck waits at a focal point, each drone on the truck makes a single delivery, then turns back to the truck waiting at the same point. They propose mathematical programming models with the objectives of minimizing total costs and minimizing delivery completion time. They also determine the number of drones needed. As a theoretical study, Campbell et al. [46] provide a strategic analysis for the design of hybrid truck-drone delivery systems using continuous approximation techniques. They suggest that substantial cost savings can be achieved by using such a system especially with multiple drones per truck.

There are also some works in the literature considering a multi-truck multi-drone setting. Wang et al. [47] introduce the vehicle routing problem with drones (VRPD) which generalizes the TSP-D by deploying multiple trucks equipped with multiple drones. The authors assume that a drone has to rendezvous with the same truck that it is launched from. They present a theoretical study of the maximum savings that can be obtained by using the drones along with the trucks and provide several worst-case bounds. Poikonen et al. [48] extend these worst-case bounds by relaxing some assumptions such as unlimited endurance of the drones and the same distance metric for both vehicles.

Sacramento et al. [14] extend the FSTSP for multiple trucks, each carrying a single drone, with the objective of minimizing transportation cost of the vehicles. As in the FSTSP, they prohibit launch and retrieval locations of a drone to be identical. A mathematical formulation which is an extension of the FSTSP and a solution approach based on adaptive large neighborhood search metaheuristic are provided. Combining the approaches in the FSTSP and the PDSTSP, Wang et al. [49] consider a problem in which multiple trucks, each carrying a single drone, operate simultaneously with multiple independent drones that are launched from and retrieved at the depot. The drones carried by the trucks can visit multiple customers after being launched from the truck. The authors develop a three-step heuristic to minimize the total finish time.

Pugliese and Guerriero [50] consider the VRPD with time windows in which each truck is equipped with multiple drones and the objective is to minimize total transportation cost while respecting the time-window restrictions. They provide an MILP formulation and solve instances with 10 nodes to optimality. Schermer et al. [51] study a multi-truck problem where each truck may be equipped with multiple drones and the objective is minimizing the makespan. An MILP formulation and a matheuristic for larger-scale instances are proposed. Additionally, a variable neighborhood search algorithm for an extension of the problem in which drones' launches and retrievals may occur not only at customer locations but also at some discrete points along the arcs is proposed by Shermer et al. [52]. For a VRPD variant in which trucks can carry multiple drones, and a drone may visit multiple customers in a row before rendezvousing with the truck that it is launched from, a mixed integer nonlinear program (MINLP) and two heuristic methods are provided by Kitjacharoenchai et al. [53]. They assume that multiple drones are not allowed to be launched or retrieved at the same node.

Additionally, Kitjacharoenchai et al. [15] propose the multiple traveling salesman problem with drones (mTSPD) as an extension of the FSTSP without endurance limitations. In this problem, while the drones may be launched from and retrieved by different trucks, only one drone can be launched or retrieved at any customer location. Allowing drones to be dispatched and retrieved by different trucks, Daknama and Kraus [54] propose a nested-local search heuristic for solving the VRPD. Wang and Sheu [55] consider a problem in which multiple drones may be launched from multiple trucks at customer locations and drones can be retrieved by the trucks only at designated docking hubs after serving one or more customers on a sortie. The authors aim to minimize logistics cost including the fixed cost of trucks and transportation cost, and they provide arc and path based formulations. Also, a branch-and-price algorithm which solves problems up to 13 customers and 2 docking hubs to optimality within a five-hour time limit is proposed. Bakir and Tiniç [56] introduce the vehicle routing problem with flexible drones which aims to find a delivery plan for multiple trucks and drones operating in synchronization, with the objective of minimizing the makespan. In this problem, the trucks are allowed to visit the same location several times and drones are not dedicated to certain trucks. They propose an MILP on a time-space network, and present an algorithm based on a dynamic discretization discovery (DDD) approach.

In Table 2.1, we provide an overview of the literature on truck-drone tandems requiring synchronization between the vehicles and proposing an exact solution approach, since in our study we specifically consider synchronization and provide branch-and-cut algorithms which are exact solution methods. In this table, the columns represent the number of trucks (#T), the number of drones per truck (#D), whether launch and retrieval locations can be the same (L $\stackrel{?}{=}$ R), whether multiple drones can be launched and retrieved at a node (mL/R), the objectives, the approaches proposed and the sizes of the problems (including the depot) solved in the papers presented, respectively. In our work, we allow drone sorties to start and end at the same node as well as launching and retrieving multiple drones at a node. Besides, we consider the operational cost as the objective and do not restrict the number of drones used. We propose several MILP formulations and branch-and-cut algorithms to solve our problem. Considering these, our work differs from the existing literature.

Table 2.1: Overview of the literature on truck-drone tandems requiring synchronization and proposing an exact approach

Reference	#T	#D	$L \stackrel{?}{=} R$	mL/R	Objective	Approach	Size
Murray and Chu [10]	1	1			completion time	MILP	11
Agatz et al. [31]	1	1	✓		completion time	ILP	10
Yurek and Ozmutlu [32]	1	1			completion time	iterative decomposition	13
Ha et al. [11]	1	1			operational cost	MILP	11
Wang et al. [38]	1	1			completion time & operational cost	MILP	-
Jeong et al. [39]	1	1			completion time	MILP	11
Bouman et al. [40]	1	1	✓		completion time	DP	16
Tang et al. [41]	1	1			completion time	constraint programming	18
Roberti and Rutmair [42]	1	1	✓		completion time	MILP, branch-and-price	39
Murray and Raj [13]	1	m		✓	completion time	MILP	9
Moshref-Javadi et al. [44]	1	m		✓	waiting times	MILP	11
Sacramento et al. [14]	m	1			transportation cost	MILP	13
Pugliese and Guerriero [50]	m	m		✓	transportation cost	MILP	11
Schermer et al. [51]	m	m	✓	✓	completion time	MILP	11
Shermer et al. [52]	m	m		✓	completion time	MILP	10
Kitjacharoenchai et al. [53]	m	m			completion time	MINLP	9
Kitjacharoenchai et al. [15]	m	u			completion time	MILP	9
Wang and Sheu [55]	m	m			logistics cost	MILP, branch-and-price	14
Bakir and Tiniç [56]	m	m	✓	✓	completion time	MILP, DDD	10
This work	1	u	✓	✓	operational cost	MILP, branch-and-cut	19

m : multiple, u: unrestricted

Chapter 3

Problem Description and Flow Based Formulation

In this chapter, we provide a description of the min-cost TSPMD and propose a flow based formulation. Let $G = (V, A)$ be a complete directed graph with the set of nodes $V = \{0, 1, \ldots, n\}$ and the set of arcs $A = \{(i, j) : i, j \in V, i \neq j\}$. Assume that node 0 represents the depot and the remaining nodes correspond to the customer locations to be served. Consider a truck equipped with multiple drones. In the min-cost TSPMD, we aim to determine the route of the truck and the movements of the drones working in synchronization with the truck such that each customer $i \in V \backslash \{0\}$ is served either by the truck or by a drone once and the total cost including the vehicles' operating and waiting costs is minimized.

We denote the driving distance for the truck and the flight distance for a drone between $i \in V$ and $j \in V$ by d_{ij} and d'_{ij}, respectively. As the drones do not necessarily follow the road network, we can assume that $d'_{ij} \leq d_{ij}$, $\forall i, j \in V$. Letting v be the truck's speed, we obtain $t_{ij} = d_{ij}/v$, the time required for the truck to travel from $i \in V$ to $j \in V$. Similarly, $t'_{ij} = d'_{ij}/v'$ is the time required for a drone where v' is a drone's speed. We suppose that the operating costs of the truck and a drone per unit of time are c_o and c'_o, respectively. Also, the respective waiting costs of the truck and a drone per unit of time are c_w and c'_w.

The truck tour starts from and ends at the depot, and the drones can be launched from the depot or the truck at customer locations. While a drone launched from the depot needs to meet the truck at a customer location on the truck tour, a drone launched from a customer location can be retrieved at the same point or one of the following customer locations on the truck tour. The reason of this distinction is that the customers that are close enough to the depot can be assumed to get service by additional drones working directly from the depot, i.e., departing from and turning back to the depot after each delivery. Additionally, a drone completing its last delivery may directly fly to the depot. In the case that the launch and retrieval locations of a drone are the same, the truck waits for the drone at the launch location to complete its service. On the other hand, if the launch and retrieval locations are different, after launching the drone, the truck continues to visit its following customers until they meet at the drone's retrieval location. In both cases, after serving a single customer, a drone returns to the truck. Thus, a drone sortie can be identified by the launch, customer and retrieval nodes. While the truck travels along its route, each drone can be dispatched several times.

We assume that the truck is capable of launching and retrieving multiple drones at the same location and the drones are launched from a customer location as soon as the truck arrives at that location. While a drone is airborne, the truck traveling from the launch node to the retrieval node may serve no customer as well as several customers on the way. Figure 3.1 exhibits the possible movements of the truck and a collection of drones. The truck tour and the drones' movements are represented by solid and dashed lines, respectively. The truck can launch multiple drones at a customer node and might retrieve them at different nodes, conversely, the drones launched from different nodes can be retrieved at the same customer.

The drones have an endurance of T, in units of time, due to their limited battery capacity. That is, a drone's operating and waiting times per customer visit cannot exceed its endurance. Also, each drone can carry one parcel at a time, so that after each delivery, the drone needs to return to the truck to pick a new package before serving another customer as it is shown in Figure 3.1. When

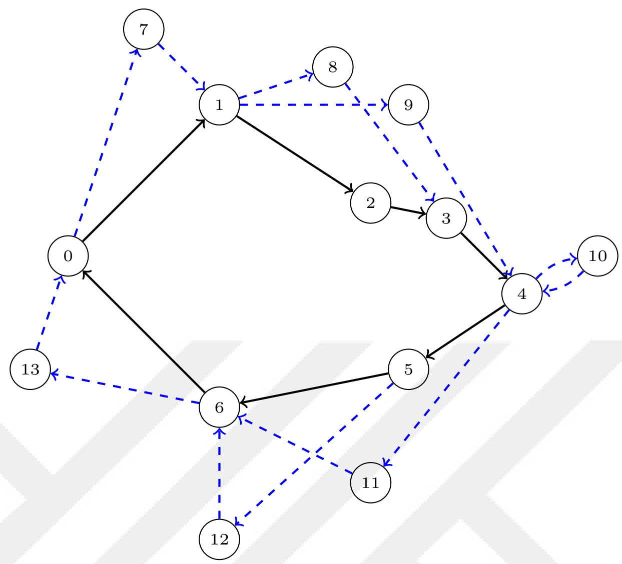


Figure 3.1: Possible movements of the truck and drones.

the drone returns to the truck, it may be loaded onto and transported by the truck to the next launch location or it can be dispatched from the truck directly for a new delivery. After making its last delivery, the drone may meet the truck to be transported to the depot or it may directly fly to the depot provided that its remaining battery level is enough.

At a drone's retrieval location, the vehicle arriving that location earlier waits the other one to rendezvous. If the drone arrives early, it waits the truck by hovering above incurring c'_w cost per unit of time. If the truck arrives early, it waits the drone presumably with a lower fuel consumption resulting in waiting cost c_w lower than its operating cost c_o per unit of time. Besides, the truck needs to rendezvous with all the drones to be retrieved at a location before proceeding to its next stop. In the case that the truck needs to wait at a customer location to meet the drones previously dispatched, launching drones as soon as it arrives at that location will provide the drones with more time (respecting the endurance limitation) before meeting the truck at the next rendezvous location.

We further assume that there are sufficiently large number of drones at the depot and on the truck as it is a common practice for a logistics company to purchase enough vehicles. Also, the truck has a sufficient capacity to carry all parcels and the drones required, and the number of batteries present at the truck is enough. For simplicity, we neglect the time required to launch/retrieve the drones, serve the customers and swap the batteries of the drones. However, as these are just parameters, they can be incorporated into our problem formulation easily. Lastly, it is assumed that the truck tour contains at least three nodes including the depot.

Table 3.1: Parameters and decision variables.

Parameters	**Description**
t_{ij}	The time required for the truck to travel from $i \in V$ to $j \in V$.
t'_{ij}	The time required for a drone to travel from $i \in V$ to $j \in V$.
c_o	Truck operating cost per unit of time.
c_w	Truck waiting cost per unit of time.
c'_o	Drone operating cost per unit of time.
c'_w	Drone waiting cost per unit of time.
T	Endurance of a drone, in units of time.

Decision variables	**Description**
$x_{ij} \in \{0,1\}$	$x_{ij} = 1$ if the truck travels from $i \in V$ to $j \in V_i$.
$y_{ij}^k \in \{0,1\}$	$y_{ij}^k = 1$ if a drone visits $k \in V_{i,j}$ when the truck travels from $i \in V$ to $j \in V_{i,k}$.
$s_i^k \in \{0,1\}$	$s_i^k = 1$ if a drone is launched from $i \in V$ to visit $k \in V_i$.
$e_j^k \in \{0,1\}$	$e_j^k = 1$ if a drone returning from $k \in V$ is retrieved at $j \in V_k$.
$w^k \geq 0$	The time that the drone returning from $k \in V$ waits for the truck at their meeting point, if the drone arrives earlier.
$r^k \geq 0$	The time that the truck waits for the drone returning from $k \in V$ at their meeting point, if the truck arrives earlier.
$R_i \geq 0$	The time that the truck waits at $i \in V$.
$h_i^k \geq 0$	The time that the truck waits at $i \in V$ while the drone visiting $k \in V_i$ is airborne unless the drone is retrieved at i.
$u_{ij} \geq 0$	The total travel time by the truck up through $j \in V$ if the truck travels from $i \in V$ to $j \in V_i$.

Let $V_S = V \setminus S$ for any set $S \subset V$. We use V_i and $V_{i,j}$ for $S = \{i\}$ and $S = \{i, j\}$, respectively. Table 3.1 shows the parameters and the decision variables and Figure 3.2 depicts the binary decision variables.

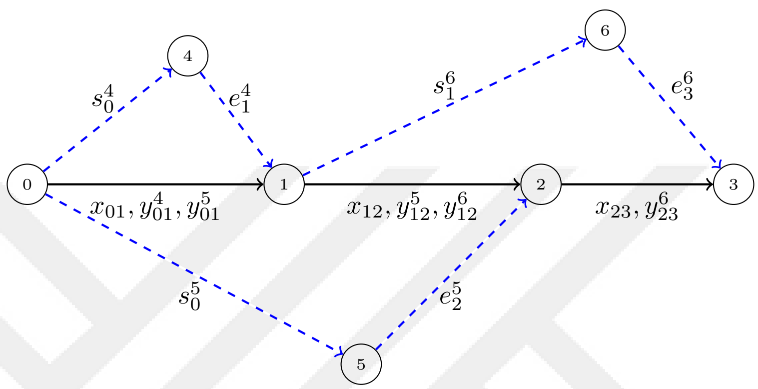


Figure 3.2: Representation of the binary decision variables.

The continuous decision variables can also be explained on Figure 3.2. For simplicity, here let drone-k represent the drone visiting node k. Suppose that drone-4 arrives at node 1 earlier than the truck. Then w^4 takes a positive value. Now, suppose that the truck arrives at node 1 earlier than drone-4. Then, both r^4 and R_1 take positive values. Also, since the truck launches drone-6 as soon as it arrives at node 1 and then waits for drone-4 while drone-6 is airborne, h_1^6 takes a positive value. Additionally, if drone-5 arrives at node 2 later than the truck, then the truck also have to wait at node 2, leading to a positive value for R_2. Since the truck waits at node 2 before arriving to retrieval node of drone-6, h_2^6 also takes a positive value. Thus, drone-6 has $h_1^6 + h_2^6$ units of more time (respecting the endurance limitation) before meeting the truck at node 3. Finally, as the truck travels from node 0 to 1, node 1 to 2 and node 2 to 3, u_{01}, u_{12}, u_{23} take positive values.

We introduce the following flow based directed problem formulation, denoted by (Dir-F).

$$\min \quad \sum_{i\in V}\sum_{j\in V_i} x_{ij}t_{ij}c_o + \sum_{k\in V} R_k c_w$$

$$+ \sum_{i\in V}\sum_{k\in V_i}\left(s_i^k t'_{ik} + e_i^k t'_{ki}\right) c'_o + \sum_{k\in V} w^k c'_w \tag{3.1}$$

$$\text{s.t.} \quad \sum_{i\in V_j}(x_{ij} + s_i^j) = 1 \qquad \forall j \in V \tag{3.2}$$

$$\sum_{i\in V_j} x_{ij} = \sum_{i\in V_j} x_{ji} \qquad \forall j \in V \tag{3.3}$$

$$\sum_{i\in V_k} s_i^k = \sum_{i\in V_k} e_i^k \qquad \forall k \in V \tag{3.4}$$

$$\sum_{j\in V_{i,k}} y_{ij}^k - \sum_{j\in V_{i,k}} y_{ji}^k = s_i^k - e_i^k \qquad \forall i \in V, k \in V_i \tag{3.5}$$

$$\sum_{k\in V_{i,j}} y_{ij}^k \leq n x_{ij} \qquad \forall i \in V, j \in V_i \tag{3.6}$$

$$s_i^k \leq \sum_{j\in V_i} x_{ij} \qquad \forall i \in V, k \in V_i \tag{3.7}$$

$$y_{i0}^k \leq e_0^k \qquad \forall i \in V_0, k \in V_{0,i} \tag{3.8}$$

$$s_0^k + e_0^k \leq 1 \qquad \forall k \in V_0 \tag{3.9}$$

$$\sum_{i\in V_k} s_i^k t'_{ik} + \sum_{j\in V_k} e_j^k t'_{kj} + w^k \leq T \qquad \forall k \in V \tag{3.10}$$

$$\sum_{i\in V_k} s_i^k t'_{ik} + \sum_{j\in V_k} e_j^k t'_{kj}$$

$$- \sum_{i\in V_k}\sum_{j\in V_k} y_{ij}^k t_{ij} - \sum_{i\in V_{0,k}} h_i^k \leq r^k \qquad \forall k \in V \tag{3.11}$$

$$\sum_{i\in V_k}\sum_{j\in V_k} y_{ij}^k t_{ij} + \sum_{i\in V_{0,k}} h_i^k$$

$$-\sum_{i\in V_k} s_i^k t'_{ik} - \sum_{j\in V_k} e_j^k t'_{kj} \leq w^k \qquad \forall k \in V \tag{3.12}$$

$$r^k - T(1 - e_i^k) \leq R_i \qquad \forall i \in V, k \in V_i \tag{3.13}$$

$$R_i \leq T \sum_{k\in V_i} e_i^k \qquad \forall i \in V \tag{3.14}$$

$$R_i - T(1 - \sum_{j\in V_k} y_{ij}^k) \leq h_i^k \qquad \forall i \in V, k \in V_i \tag{3.15}$$

$$h_i^k \leq R_i \qquad \forall i \in V, k \in V_i \tag{3.16}$$

$$h_i^k \leq T \sum_{j\in V_k} y_{ij}^k \qquad \forall i \in V, k \in V_i \tag{3.17}$$

$$\sum_{i\in V}\sum_{j\in V_i} x_{ij} \geq 3 \tag{3.18}$$

$$\sum_{j\in V_i} u_{ij} - \sum_{j\in V_i} u_{ji} - \sum_{j\in V_i} t_{ij}x_{ij} = 0 \qquad \forall i \in V_0 \tag{3.19}$$

$$u_{0i} = t_{0i}x_{0i} \qquad \forall i \in V_0 \tag{3.20}$$

$$u_{ij} \leq (M - t_{j0})x_{ij} \qquad \forall i \in V, j \in V_{0,i} \tag{3.21}$$

$$u_{i0} \leq Mx_{i0} \qquad \forall i \in V_0 \tag{3.22}$$

$$u_{ij} \geq (t_{0i} + t_{ij})x_{ij} \qquad \forall i \in V_0, j \in V_i \tag{3.23}$$

$$x_{ij}, s_i^j, e_i^j \in \{0, 1\} \qquad \forall i, j \in V \tag{3.24}$$

$$y_{ij}^k \in \{0, 1\} \qquad \forall i, j, k \in V \tag{3.25}$$

$$w^i, r^i, R_i \geq 0 \qquad \forall i \in V \tag{3.26}$$

$$h_i^k \geq 0 \qquad \forall i, k \in V \qquad (3.27)$$

The objective (3.1) is to minimize the total cost consisting of both the truck's and the drones' operating and waiting costs. Constraints (3.2) guarantee that all customers are visited either by the truck or a drone. Constraints (3.3)-(3.5) are flow balance constraints. That is, constraints (3.3) ensure that if the truck visits a customer $j \in V$, it also departs from j. Similarly, constraints (3.4) ensure that the drone visiting a customer $k \in V$ departs from k. Constraints (3.5) make sure that the truck departs from a launch node of a drone and arrives at the corresponding retrieval node of that drone by visiting zero or more customers in between. Constraints (3.6) guarantee that if the truck does not travel between two customer nodes $i, j \in V$, a drone cannot visit customer $k \in V$ when truck travels between i and j. Constraints (3.7) allow a drone to be launched at a customer node only if that node is on the truck tour. By constraints (3.8), the drones are enforced to return to the depot after their visits in order to meet the truck if the truck's next stop is the depot. The assumption that the depot cannot be both launch and retrieval nodes of a drone sortie is handled by constraints (3.9). The drones' limited endurance is imposed by constraints (3.10) such that a drone's total flight and waiting time cannot exceed its endurance. Constraints (3.11) determine the time that the truck waits for a drone departing from customer $k \in V$ at their rendezvous location, and conversely constraints (3.12) are to obtain the time that a drone departing from k waits the truck at its retrieval location. As the truck needs to wait all the drones to be retrieved at a node before proceeding to the next customer on its tour, constraints (3.13) determine the time that the truck needs to wait at a node for the drone arriving at the latest. By constraints (3.14), the truck waits at a customer node only to retrieve the drones. While a drone is on a sortie, we use h's to compute the total waiting time of the truck at the nodes visited except the retrieval node. Constraints (3.15) and (3.16) together make h equal to R if the drone visiting $k \in V$ is on the sortie while the truck waits at $i \in V$. Constraints (3.17) exclude the retrieval location of the drone visiting $k \in V$ from contributing to total waiting time of the truck associated with that drone sortie. The assumption that the truck tour contains at least three nodes is ensured by constraints (3.18). Constraints (3.19) are subtour elimination and constraints (3.20)-(3.23) are time bounding constraints proposed in [57] where M is a sufficiently large number. Finally, domain restrictions on the variables are given in constraints (3.24)-(3.27). The min-cost TSPMD turns out to be a classical TSP when $T = 0$ (i.e., when all the customers have to be visited by the truck). Thus, it is NP-hard, as the TSP has been proven to be NP-hard.

Notice that, as we neglect the time required to swap the batteries of the drones, once the truck retrieves a drone arriving earlier, the truck can supply that drone with a new parcel and launch it. On the other hand, when the truck arrives at a node earlier than all the drones to be retrieved at that location, it may dispatch other drones, as we do not limit the number of drones used per delivery plan. For instance, in Figure 3.1, if the drone coming from customer 7 arrives at customer 1 earlier than the truck, when the truck also arrives at customer 1, that particular drone can be supplied with a new parcel and relaunched to visit either customer 8 or 9. On the other hand, if the drone arrives later than the truck, the truck launches new drones to visit customers 8 and 9 as soon as it arrives at customer 1. Under this setting, using Algorithm 1, we can calculate the minimum number of drones required for a given solution obtained by (Dir-F). Algorithm 1 considers the number of outgoing/incoming drones at each node and checks whether the drones arrive earlier than the truck or not.

Algorithm 1 Calculate the minimum number of drones needed

function FIND NUMBER OF DRONES($\bar{x}, \bar{s}, \bar{e}, \bar{w}$)
    Input: $\bar{x}, \bar{s}, \bar{e}, \bar{w}$
    Output: The minimum number of drones needed.

    Initialize $dronesOut \leftarrow 0$ and $numDrones \leftarrow 0$.
    $i \leftarrow 0$. ▷ the truck tour starts from the depot
    while $TRUE$ **do**
        for $k \in V$ **do**
            if $\bar{s}_i^k = 1$ **then** ▷ a drone leaves i
                $out \leftarrow out + 1$.
            end if
            if $\bar{e}_i^k = 1$ **then** ▷ a drone arrives at i
                $in \leftarrow in + 1$.
            end if
            if $\bar{e}_i^k = 1$ and $\bar{w}^k > 0$ **then** ▷ a drone arrives at i before the truck
                $inBefore \leftarrow inBefore + 1$.
            end if
        end for
        if $i = 0$ **then**
            $inBefore \leftarrow 0$ and $in \leftarrow 0$.
        end if
        $dronesOut \leftarrow dronesOut + out - inBefore$. ▷ # of drones out when the truck arrives at i
        $numDrones \leftarrow \max\{dronesOut, numDrones\}$.
        $dronesOut \leftarrow dronesOut - (in - inBefore)$. ▷ # of drones out when the truck departs from i
        $i \leftarrow j$ such that $x_{ij} = 1$. ▷ the truck moves to the next node
        if $i = 0$ **then**
            Break loop. ▷ when the truck tour ends at the depot
        end if
    end while
    Return $numDrones$.
end function

Chapter 4

Cut Based Formulation & Branch-and-Cut Algorithms

In this chapter, we propose a cut based formulation and branch-and-cut algorithms to solve the min-cost TSPMD. Branch-and-cut algorithms are widely used in the literature considering the TSP variants for solving larger instances [58].

4.1 Cut Based Formulation and Valid Inequalities

We introduce the following cut based problem formulation, denoted by (Dir-C).

$$\begin{aligned} \text{min} \quad & (3.1) \\ \text{s.t.} \quad & (3.2)-(3.18), (3.24)-(3.27) \end{aligned}$$

$$\sum_{i \in V_0} x_{0i} = 1 \tag{4.1}$$

$$\sum_{m \in S} \sum_{k \in V \setminus S} x_{mk} \geq \sum_{i \in V_j} x_{ij} \qquad \forall S \subset V_0, 2 \leq |S| \leq n-2, \forall j \in S \tag{4.2}$$

Constraint (4.1) enforces the truck to depart from the depot. Constraints (4.2) are the connectivity cuts preventing subtours and they are exponential in the size of the problem.

For notational brevity, we introduce $x(S) = \sum_{m \in S} \sum_{k \in V \setminus S} x_{mk}$ for set $S \subset V_0$ and use it in the remainder of this chapter.

We let X denote the feasible set of (Dir-C) and define $\mathcal{N}_k$ as the set of nodes from where a drone can be launched to visit node $k \in V$ without violating its endurance limitation assuming that the drone will not wait the truck to rendezvous, i.e., $\mathcal{N}_k = \{i \in V_k : t'_{ik} + t'_{kj} \leq T \text{ for some } j \in V_k\}$.

The family of valid inequalities presented below is obtained by lifting the connectivity cuts (4.2) with variables s_i^j, $\forall i, j \in V$.

Theorem 1. *Let $S \subset V_0$ with $2 \leq |S| \leq n-2$ and $j \in S$ such that $\emptyset \neq \mathcal{N}_j \subset S$. The lifted connectivity inequality (LCI)*

$$x(S) \geq \sum_{i \in V_j} x_{ij} + \sum_{i \in V_j} s_i^j \tag{4.3}$$

is valid for X.

Proof. If $\sum_{i \in V_j} s_i^j = 0$, then constraint (4.3) reduces to (4.2). If $\sum_{i \in V_j} s_i^j = 1$, then $\sum_{i \in V_j} x_{ij} = 0$ by (3.2), and constraints (3.3),(3.7) and (3.10) together imply that $\sum_{l \in \mathcal{N}_j} \sum_{i \in V_l} x_{il} \geq 1$. Provided that $\mathcal{N}_j \subset S$, there exists $l \in S$ with $\sum_{i \in V_l} x_{il} = 1$. Constraint (4.2) for set S and node l implies $x(S) \geq 1$. Hence inequality (4.3) is satisfied in both cases. $\square$

If $S = \mathcal{N}_j \cup \{j\}$, then the resulting inequality (4.3) is called simple lifted connectivity inequality (SLCI). Note that these inequalities are polynomial in number.

These valid inequalities are adapted from the ones presented by Ozbaygin et al. [59] for a time constraint maximal covering salesman problem with weighted demands and partial coverage (TCMCSP). Our problem shares some similarities with this problem: In the TCMCSP, they utilize the notion of coverage (i.e., satisfying the demand of multiple customers by visiting a single customer location) and aim to find a tour visiting a subset of customers so that the amount of demand covered within a limited time is maximized. In our problem, we also find a truck tour visiting a subset of customers. Similar to the notion of coverage in the TCMCSP, we use the concept of endurance limitation for drones by which whether a drone can be launched from a customer location to visit a particular customer is determined.

These similarities encourage us to utilize the valid inequalities (4.3) which are the modified versions of the valid inequalities presented in [59]. These valid inequalities in [59] are proved to be strong when the connectivity cut that is lifted is strong for the polytope associated with the orienteering problem where every node is associated with a profit and the objective is to find a tour with maximum profit subject to a time restriction.

4.2 Branch-and-Cut Algorithms

Since (Dir-C) involves exponentially many constraints, we devise branch-and-cut algorithms to solve the problem. We propose three branch-and-cut schemes similar to the first three schemes described in [59] and follow similar separation procedures. The first scheme starts by solving the relaxation (3.1)-(3.18),(3.24)-(3.27) and (4.1). Then the violated connectivity constraints (4.2) are introduced only for integer solutions of the branch-and-cut tree. In the second scheme, connectivity constraints are separated also for fractional solutions at the root node of the branch-and-cut tree. In the last scheme, similar to the second one, violated constraints are separated also for fractional solutions only at the root node of the tree. First, we add the SLCIs for $j \in V_0$ with $0 \notin \mathcal{N}_j$ to the initial relaxation. Then, before checking whether a connectivity constraint $x(S) \geq \sum_{i \in V_j} x_{ij}$ is violated, we check the corresponding LCI given by $x(S) \geq \sum_{i \in V_j} x_{ij} + \sum_{i \in V_j} s_i^j$ if $0 \notin \mathcal{N}_j$ and $\mathcal{N}_j \subset S$. There might be some cases where $0 \notin \mathcal{N}_j$ holds, but $\mathcal{N}_j \backslash S \neq \emptyset$. In that case, we extend the set S by adding the nodes in $\mathcal{N}_j \backslash S$ to the set S and investigate whether the inequality $x(S \cup \mathcal{N}_j) \geq \sum_{i \in V_j} x_{ij} + \sum_{i \in V_j} s_i^j$ is satisfied. If one of these LCIs is violated, we introduce it instead of the connectivity constraint itself. Our separation procedures are explained in the following subsections.

Suppose that $\bar{G} = (\bar{V}, \bar{A})$ is the support graph induced by a given solution vector where $\bar{\nu} = (\bar{x}, \bar{y}, \bar{s}, \bar{e})$ constitutes the part of the solution associated with the binary variables. That is, $\bar{V} = \{j \in V : \sum_{i \in V_j} \bar{x}_{ij} > 0\}$ and $\bar{A} = \{(i, j) \in A : \bar{x}_{ij} > 0\}$. Let S_k be the connected components of $\bar{G}$, where $k = 0, 1, \ldots, t$ and $0 \in S_0$. Notice $t \geq 1$ indicates that the solution contains subtours. Also, we write $\bar{x}(S) = \sum_{m \in S} \sum_{k \in V \setminus S} \bar{x}_{mk}$ for set $S \subset V_0$.

4.2.1 Separation of connectivity constraints

There are four cases regarding $\bar{\nu} = (\bar{x}, \bar{y}, \bar{s}, \bar{e})$ and the corresponding support graph $\bar{G}$. In the first case, suppose that $\bar{\nu}$ is integral and $\bar{G}$ is connected, i.e., $t = 0$. Since the solution is feasible for the original problem, no connectivity constraint (4.2) is violated.

In the second case, suppose that $\bar{\nu}$ is integral but $\bar{G}$ is not connected, i.e., $t \geq 1$. In this case, for each S_k, $k = 1, \ldots, t$ and each $j \in S_k$, there is a violated connectivity constraint. Thus, the solution vector induces $\sum_{k=1}^{t} |S_k|$ violated constraints, where $|S_k|$ is the cardinality of set S_k. Introducing any one of these violated constraints to the model cuts off the current solution. However, we add the cuts (4.2) for every S_k, $k = 1, \ldots, t$ and for every $j \in S_k$ instead of adding a single cut, in order to speed up the solution procedure.

In the third case, consider that $\bar{\nu}$ has at least one fractional component and $\bar{G}$ is not connected, i.e., $t \geq 1$. As in the second case, a violated connectivity constraint is induced by each S_k, $k = 1, \ldots, t$ and each $j \in S_k$. Therefore, we add the cuts (4.2) for every S_k, $k = 1, \ldots, t$ and for every $j \in S_k$.

In the last case, now consider that $\bar{\nu}$ has at least one fractional component but $\bar{G}$ is connected, i.e., $t = 0$. In this case, violation of connectivity constraints can be investigated by solving a series of minimum cut problems on the graph $\bar{G}$ taking the capacity of each arc $(i,j) \in \bar{A}$ as $\bar{x}_{ij}$. We solve a minimum cut problem for each $j \in \bar{V}_0$ separating nodes j and 0. We obtain the node partition $S^*(j)$ containing j where $[S^*(j), V \backslash S^*(j)]$ defines a minimum cut with source j and sink 0, and the corresponding cut capacity, which is the sum of the weights of the arcs from source partition to sink partition. Let $cap(S)$ denote the capacity of the cut defined by the set S of nodes. Then, we compare $cap(S^*(j))$ with $\sum_{i \in \bar{V}_j} \bar{x}_{ij}$. If $cap(S^*(j))$ is greater than or equal to $\sum_{i \in \bar{V}_j} \bar{x}_{ij}$, then every $S \subset V_0$ containing j satisfies the inequality $\bar{x}(S) \geq \sum_{i \in \bar{V}_j} \bar{x}_{ij}$, i.e., the corresponding connectivity constraint (4.2). Because, we know that $cap(S) \geq cap(S^*(j))$ for any cut S containing j of the graph $\bar{G}$. Otherwise, a violated connectivity constraint corresponding to node j and the node partition $S^*(j)$ containing j where $[S^*(j), V \backslash S^*(j)]$ defines a minimum cut with source j and sink 0 is obtained and it is added to the model.

4.2.2 Separation of lifted connectivity inequalities

Our search for violated lifted connectivity inequalities is embedded into our connectivity constraint separation procedure. We separate LCIs prior to the corresponding connectivity constraint for a particular node j and set $S \subseteq V_0$ with $j \in S$. Similar to Section 4.2.1, our separation procedures for each four cases regarding $\bar{\nu}$ and corresponding support graph $\bar{G}$ are explained below.

Similar to the connectivity constraint separation, first consider the case where $\bar{\nu}$ is integral and $\bar{G}$ is connected, i.e., $t = 0$. There is no violated LCI, as the solution is feasible for the original problem.

In the second case, consider that $\bar{\nu}$ is integral but $\bar{G}$ is not connected, i.e., $t \geq 1$. In that case, we check whether the LCI induced by each node $j \in S_k$, $k = 1, \ldots, t$ and the set $S_k \cup \mathcal{N}_j$ is satisfied or not, provided that $0 \notin \mathcal{N}_j$. If $\mathcal{N}_j \subset S_k$, then we have a violated LCI induced by $j \in S_k$ and set S_k, since $\bar{x}(S_k) = 0$. Else, we investigate whether the inequality $\bar{x}(S_k \cup \mathcal{N}_j) \geq \sum_{i \in \bar{V}_j} \bar{x}_{ij} + \sum_{i \in \bar{V}_j} \bar{s}_i^j$ for each $j \in S_k$ such that $0 \notin \mathcal{N}_j$, for every $k = 1, \ldots, t$, and the set $S_k \cup \mathcal{N}_j$ is satisfied or not. If it is violated, we introduce it to the model. Then, for each $j \in S_k$ for every $k = 1, \ldots, t$, if $0 \in \mathcal{N}_j$ or $0 \notin \mathcal{N}_j$ but no LCI is violated, the connectivity constraint induced by j and S_k is introduced to the model in the same manner it is done in the second case of the connectivity constraint separation explained in the previous subsection.

In the third case, consider that $\bar{\nu}$ has at least one fractional component and $\bar{G}$ is not connected, i.e., $t \geq 1$. In that case, we separate LCIs and the connectivity constraints in the same manner as we do in the previous case.

Finally, consider that $\bar{\nu}$ has at least one fractional component but $\bar{G}$ is connected, i.e., $t = 0$. In this case, for each node $j \in \bar{V}_0$, minimum cut $S^*(j)$ separating node j and 0 is found as we do in the connectivity constraint separation. Provided that $0 \notin N_j$, the capacity of the cut $S^*(j) \cup N_j$ is compared with the value $(\sum_{i \in \bar{V}_j} \bar{x}_{ij} + \sum_{i \in \bar{V}_j} \bar{s}_i^j)$ to determine whether the LCI associated with j and $S^*(j) \cup N_j$ is satisfied or not. If it is violated, then we add it to the model. After our search for violated LCIs, for each node $j \in \bar{V}_0$, if $0 \in N_j$ or $0 \notin N_j$ and no LCI induced by j and $S^*(j) \cup N_j$ is found to be violated, we evaluate violation of the connectivity constraint regarding node j and $S^*(j) \cup N_j$ as we do in the last case of the connectivity constraint separation stated in the previous subsection.

Chapter 5

An Alternative Cut Based Solution Methodology

In this chapter, we present an alternative solution methodology using another cut based formulation with undirected decision variables.

5.1 An Alternative Cut Based Formulation and Valid Inequalities

Consider a complete undirected graph $G_U = (V, E)$ with the set of nodes $V = \{0, 1, \dots, n\}$ and the set of edges $E = \{\{i, j\} : i, j \in V, i < j\}$. As in the directed graph considered in Chapter 3, node 0 represents the depot and the remaining nodes correspond to the customer locations to be served. For $S \subset V$, we let $\delta(S) = \{e \in E : |e \cap S| = 1\}$. If $S = \{i\}$, we use $\delta(i)$ instead of $\delta(\{i\})$.

In this formulation, the same set of parameters and decision variables with Chapter 3 are used. As the graph is now undirected, instead of t_{ij} we use t_e to denote the time required for a truck to travel through $e = \{i, j\} \in E$. Also, we modify the decision variables x's and y's such that $x_e = 1$ if $e \in E$ is on the truck tour and $y_e^k = 1$ if a drone visits $k \in V$ when the truck travels through edge $e \in E$. Lastly, we introduce a new decision variable to keep track of the nodes visited by the truck when a drone is airborne to visit node $k \in V$, that is,

$$g_i^k = \begin{cases} 1, & \text{if } i \in V \text{ is visited by the truck when the drone visiting } k \in V_i \text{ is} \\ & \text{on the sortie} \\ 0, & \text{otherwise} \end{cases}$$

We introduce the following cut based problem formulation, denoted by (Undir).

$$\begin{aligned} \min \quad & \sum_{e \in E} x_e t_e c_o + \sum_{k \in V} R_k c_w \\ & + \sum_{i \in V} \sum_{k \in V_i} \left(s_i^k t'_{ik} + e_i^k t'_{ki} \right) c'_o + \sum_{k \in V} w^k c'_w \end{aligned} \tag{5.1}$$

$$\begin{aligned} \text{s.t.} \quad & (3.4), (3.9), (3.10), (3.13) \\ & (3.14), (3.16), (3.26), (3.27) \end{aligned}$$

$$\sum_{e \in \delta(j)} x_e + 2 \sum_{i \in V_j} s_i^j = 2 \qquad \forall j \in V \tag{5.2}$$

$$\sum_{e \in \delta(i)} y_e^k + s_i^k + e_i^k = 2 g_i^k \qquad \forall i \in V, k \in V_i \tag{5.3}$$

$$\sum_{k \in V} y_e^k \leq n x_e \qquad \forall e \in E \tag{5.4}$$

$$2 s_i^k \leq \sum_{e \in \delta(i)} x_e \qquad \forall i \in V, k \in V_i \tag{5.5}$$

$$s_0^k + e_0^k = \sum_{e \in \delta(0)} y_e^k \qquad \forall k \in V_0 \tag{5.6}$$

$$\sum_{i \in V_k} s_i^k t'_{ik} + \sum_{j \in V_k} e_j^k t'_{kj}$$

$$-\sum_{e\in E} y_e^k t_e - \sum_{i\in V_{0,k}} h_i^k \leq r^k \qquad \forall k \in V \tag{5.7}$$

$$\sum_{e\in E} y_e^k t_e + \sum_{i\in V_{0,k}} h_i^k$$

$$-\sum_{i\in V_k} s_i^k t'_{ik} - \sum_{j\in V_k} e_j^k t'_{kj} \leq w^k \qquad \forall k \in V \tag{5.8}$$

$$R_i - T(2 - \sum_{e\in\delta(i)} y_e^k - s_i^k) \leq h_i^k \qquad \forall i \in V, k \in V_i \tag{5.9}$$

$$2h_i^k \leq T(\sum_{e\in\delta(i)} y_e^k + s_i^k) \qquad \forall i \in V, k \in V_i \tag{5.10}$$

$$h_i^k \leq T(1 - e_i^k) \qquad \forall i \in V, k \in V_i \tag{5.11}$$

$$\sum_{e\in\delta(0)} x_e = 2 \tag{5.12}$$

$$\sum_{e\in\delta(S)} x_e \geq \sum_{e\in\delta(j)} x_e \qquad \forall S \subset V_0, \quad 3 \leq |S| \leq n-2, j \in S \tag{5.13}$$

$$s_i^k + s_i^m \leq 3 - y_{e_1}^k - y_{e_2}^m \qquad \forall i \in V, k \in V_i, m \in V_{i,k}, \quad e_1, e_2 \in \delta(i) : e_1 \neq e_2 \tag{5.14}$$

$$e_i^k + e_i^m \leq 3 - y_{e_1}^k - y_{e_2}^m \qquad \forall i \in V, k \in V_i, m \in V_{i,k}, \quad e_1, e_2 \in \delta(i) : e_1 \neq e_2 \tag{5.15}$$

$$e_i^k + s_i^m \leq 3 - y_e^k - y_e^m \qquad \forall i \in V, k \in V_i, m \in V_{i,k}, \quad e \in \delta(i) \tag{5.16}$$

$$x_e \in \{0,1\} \qquad \forall e \in E \tag{5.17}$$

$$g_i^k, s_i^k, e_i^k \in \{0,1\} \qquad \forall i \in V, k \in V_i \tag{5.18}$$

$$y_e^k \in \{0,1\} \qquad \forall k \in V, e \in E \tag{5.19}$$

As in the previous formulations (Dir-F) of Chapter 3 and (Dir-C) of Chapter 4, the objective (5.1) is to minimize the total cost consisting of both the truck's and the drones' operating and waiting costs. Constraints (5.2) guarantee that each node is visited either by the truck or by a drone, besides they ensure that a node visited by the truck has degree two in terms of variables x's. By constraints (5.3), when a drone is on the sortie to visit node k, the nodes visited by the truck are assigned as the launch and retrieval locations of that drone or intermediate stops for the truck. Constraints (5.4) are the undirected version of the constraints (3.6) that link variables y's and x's. Constraints (5.6) impose that the depot can either be the launch or retrieval node of a drone sortie, but not a node visited by the truck in between them. Constraints (5.7) and (5.8) are undirected versions of the constraints (3.11) and (3.12) determining the time that the truck or a drone waits the other one, respectively. Similarly, constraints (5.9) and (5.10) are undirected variants of the constraints (3.15) and (3.17), together with (5.11), they relate the variables R's and h, and set bounds on h's. Constraint (5.12) enforces the depot to be on the truck tour as in (4.1). Constraints (5.13) are connectivity cuts preventing subtours. Constraints (5.14)-(5.16) are to obtain consistency within the directions of drone sorties when two drones have a common launch/retrieval location. As the graph is undirected and there is no constraint determining the direction of the truck tour, the drones might fly through conflicting directions. That is, two drones might act as if the truck is traveling in opposite directions. Constraints (5.14) ensure that if two drones are launched from a node, they need to move as if the truck moves in a particular direction, i.e., truck traverses either e_1 or e_2. Similarly, constraints (5.15) ensure that if two drones are to be retrieved at a node, they need to come from the same direction. Lastly, constraints (5.16) deal with two drones: one to be retrieved at a node and the other to be launched from the same node. Finally, constraints (5.17)-(5.19) are domain constraints.

Let Y denote the feasible set of (Undir).

Theorem 2. *Let $S \subset V_0$ with $3 \leq |S| \leq n-2$ and $j \in S$ such that $\emptyset \neq \mathcal{N}_j \subset S$. The lifted connectivity inequality (LCI)*

$$\sum_{e \in \delta(S)} x_e \geq \sum_{e \in \delta(j)} x_e + 2 \sum_{i \in V_j} s_i^j \tag{5.20}$$

is valid for Y.

Proof. If $\sum_{j \in V} s_i^j = 0$, then (5.20) reduces to constraint (5.13). If $\sum_{j \in V} s_i^j = 1$, then $\sum_{e \in \delta(j)} x_e = 0$ by (5.2) and constraints (3.10) and (5.5) together imply that $\sum_{l \in \mathcal{N}_j} \sum_{e \in \delta(l)} x_e \geq 1$. Since $\mathcal{N}_j \subset S$ and $\sum_{l \in \mathcal{N}_j} \sum_{e \in \delta(l)} x_e \geq 1$, there exists $l \subset S$ with $\sum_{e \in \delta(l)} x_e = 1$. Constraint (5.13) for set S and node l implies $\sum_{e \in \delta(S)} x_e \geq 2$, hence inequality (5.20) is satisfied in both cases. □

Similar to Section 4.1, if $S = \mathcal{N}_j \cup \{j\}$, then the resulting inequality (5.20) is called simple lifted connectivity inequality (SLCI). Again, these inequalities are polynomial in number.

Recall that X denotes the feasible set of (Dir-C). Let $\tilde{X}$ be a set consisting of the feasible solutions in X such that each arc (i, j) traversed by the truck is replaced by the edge $\{i, j\}$. That is, the truck tour becomes undirected.

Theorem 3. *Any solution in $\tilde{X}$ is also in set Y.*

Proof. Consider a solution vector in $\tilde{X}$. With a slight abuse of notation, for any edge $e = \{i, j\} \in E$, note that $x_e = 0$ if $x_{ij} = x_{ji} = 0$ and $x_e = 1$ otherwise. Hence, and since $x_{ij} + x_{ji} \leq 1$, we have $x_e = x_{ij} + x_{ji}$. Similarly, for each $k \in V$, $y_e^k = y_{ij}^k + y_{ji}^k$. For any $j \in V$, suppose that $\sum_{i \in V_j} x_{ij} = 1$. Then, we have $\sum_{i \in V_j} s_i^j = 0$ from (3.2) and $\sum_{i \in V_j} x_{ji} = 1$ from (3.3). Thus, $\sum_{e \in \delta(j)} x_e = \sum_{i \in V_j} x_{ij} + \sum_{i \in V_j} x_{ji} = 2$. Hence, $\sum_{e \in \delta(j)} x_e + 2\sum_{i \in V_j} s_i^j = 2$. Similarly, suppose that $\sum_{i \in V_j} x_{ij} = 0$. Then, we have $\sum_{i \in V_j} s_i^j = 1$ from (3.2) and $\sum_{i \in V_j} x_{ji} = 0$ from (3.3). Thus, $\sum_{e \in \delta(j)} x_e = 0$. Hence, $\sum_{e \in \delta(j)} x_e + 2\sum_{i \in V_j} s_i^j = 2$.

We now show that for any $i \in V, k \in V_i$, if $e_i^k = s_i^k = 1$, then $\sum_{j \in V_k} y_{ij}^k = \sum_{j \in V_k} y_{ji}^k = 0$: Suppose that $e_i^k = s_i^k = 1$ and assume to the contrary that $\sum_{j \in V_k} y_{ij}^k = \sum_{j \in V_k} y_{ji}^k = 1$. First, note that $i \neq 0$ since it would lead a contradiction from (3.9) as $s_0^k + e_0^k = 2$ otherwise. Also, note that $s_j^k = 0$ and $e_j^k = 0$ for any $j \in V_{i,k}$ from (3.2) and (3.4). Thus, $\sum_{j \in V_k} y_{ij}^k - \sum_{j \in V_k} y_{ji}^k = 0$ for all $i \in V$ from (3.5). This implies that $y_{ij}^k = 1$ if $x_{ij} = 1$ for all $i \in V, j \in V_i$ since $y_{ij}^k = 1$ for some i, j. Hence, for some $i \in V_0$, we have $y_{i0}^k = 1$ since $\sum_{i \in V_0} x_{i0} = 1$.

However, it leads to a contradiction from (3.8) since $y_{i0} = 1$ and $e_0^k = 0$. Thus, $\sum_{j\in V_k} y_{ij}^k = \sum_{j\in V_k} y_{ji}^k = 0$ from (3.5). Following these results, in addition to (3.5), we know that $\sum_{e\in\delta(i)} y_e^k + s_i^k + e_i^k$ equals to either 0 or 2.

For any $e = \{i,j\} \in E$, from (3.6), $\sum_{k\in V} y_e^k = \sum_{k\in V_{i,j}} y_{ij}^k + \sum_{k\in V_{i,j}} y_{ji}^k \leq nx_{ij} + nx_{ji} = nx_e$.

For any $i \in V, k \in V_i$, from (3.7), we have $s_i^k \leq \sum_{j\in V_i} x_{ij}$. Then, $2s_i^k \leq \sum_{e\in\delta(i)} x_e = \sum_{j\in V_i} x_{ij} + \sum_{j\in V_i} x_{ji}$ since $\sum_{j\in V_i} x_{ij} = \sum_{j\in V_i} x_{ji}$ from (3.3).

For any $k \in V_0$, recall from (3.9) that $s_0^k + e_0^k \leq 1$. First, suppose that $s_0^k = e_0^k = 0$. Then, from (3.8), $y_{i0}^k = 0$ for all $i \in V_{0,k}$. Thus, $\sum_{i\in V_{0,k}} y_{i0}^k = 0$. Hence, we have $\sum_{i\in V_{0,k}} y_{0i}^k = 0$ from (3.5). Thus, $s_0^k + e_0^k = 0 = \sum_{i\in V_{0,k}} y_{0i}^k + \sum_{i\in V_{0,k}} y_{i0}^k = \sum_{e\in\delta(0)} y_e^k$. Now, suppose that $s_0^k = 1$ and $e_0^k = 0$. Then, from (3.8), $y_{i0}^k = 0$ for all $i \in V_{0,k}$. Thus, $\sum_{i\in V_{0,k}} y_{i0}^k = 0$. Hence, we have $\sum_{i\in V_{0,k}} y_{0i}^k = 1$ from (3.5). Thus, $s_0^k + e_0^k = 1 = \sum_{j\in V_{0,k}} y_{0i}^k + \sum_{i\in V_{0,k}} y_{i0}^k = \sum_{e\in\delta(0)} y_e^k$. Lastly, suppose that $s_0^k = 0$ and $e_0^k = 1$. Then, from (3.5), we have $\sum_{i\in V_{0,k}} y_{0i}^k - \sum_{i\in V_{0,k}} y_{i0}^k = -1$. Note that constraints (3.6) can be replaced with $y_{ij}^k \leq x_{ij}$ for all $i \in V, j \in V_i, k \in V_{i,j}$. Hence, $0 \leq \sum_{i\in V_{0,k}} y_{i0}^k \leq \sum_{i\in V_{0,k}} x_{i0} \leq 1$ and $0 \leq \sum_{i\in V_{0,k}} y_{0i}^k \leq \sum_{i\in V_{0,k}} x_{0i} \leq 1$. Thus, $\sum_{i\in V_{0,k}} y_{0i}^k = 0$ and $\sum_{i\in V_{0,k}} y_{i0}^k = 1$. Then, $s_0^k + e_0^k = 1 = \sum_{j\in V_{0,k}} y_{0i}^k + \sum_{i\in V_{0,k}} y_{i0}^k = \sum_{e\in\delta(0)} y_e^k$.

For any $k \in V$, note that $\sum_{e\in E} y_e^k t_e = \sum_{i\in V_k} \sum_{j\in V_k} y_{ij}^k t_{ij}$ since $y_e^k = y_{ij}^k + y_{ji}^k$ for all $e = \{i,j\} \in E$. Thus, (5.7) holds from (3.11). Similarly, (5.8) holds from (3.12).

For any $i \in V, k \in V_i$, we first show that $2 - 2\sum_{j\in V_k} y_{ij}^k - e_i^k \geq 1 - \sum_{j\in V_k} y_{ij}^k$: Suppose that $\sum_{j\in V_k} y_{ij}^k = 0$. Then, $2 - 2\sum_{j\in V_k} y_{ij}^k - e_i^k = 2 - e_i^k \geq 1 = 1 - \sum_{j\in V_k} y_{ij}^k$ since $e_i^k \leq 1$. Now, suppose that $\sum_{j\in V_k} y_{ij}^k = 1$. Assume to the contrary that $e_i^k = 1$. Then, from (3.5), we have $\sum_{j\in V_k} y_{ji}^k = 1$ and $s_i^k = 1$. However, it leads to a contradiction since $e_i^k = s_i^k = 1$ implies that $\sum_{j\in V_k} y_{ij}^k = 0$. Thus, $e_i^k = 0$. Hence, $2 - 2\sum_{j\in V_k} y_{ij}^k - e_i^k = 0 = 1 - \sum_{j\in V_k} y_{ij}^k$. Note that $\sum_{e\in\delta(i)} y_e^k + s_i^k = \sum_{j\in V_k} y_{ij}^k + \sum_{j\in V_k} y_{ji}^k + s_i^k = 2\sum_{j\in V_k} y_{ij}^k + e_i^k$ from (3.5). Thus, $2 - \sum_{e\in\delta(i)} y_e^k - s_i^k = 2 - 2\sum_{j\in V_k} y_{ij}^k - e_i^k \geq 1 - \sum_{j\in V_k} y_{ij}^k$. Hence, $R_i - T(2 - \sum_{e\in\delta(i)} y_e^k - s_i^k) \leq$

$R_i - T(1 - \sum_{j \in V_k} y_{ij}^k) \leq h_i^k$ from (3.15).

For any $i \in V, k \in V_i$, note from (3.5) and (3.17) that $T(\sum_{e \in \delta(i)} y_e^k + s_i^k) = T(2\sum_{j \in V_k} y_{ij}^k + e_i^k) \geq 2h_i^k + Te_i^k \geq 2h_i^k$.

For any $i \in V, k \in V_i$, first suppose that $e_i^k = 0$. Then, $h_i^k \leq T = T(1 - e_i^k)$ from (3.17) since $\sum_{j \in V_k} y_{ij}^k \leq 1$. Now suppose that $e_i^k = 1$. Then, from (3.5), we have $\sum_{j \in V_k} y_{ij}^k = 0$. Thus, from (3.17), $h_i^k = 0 = T(1 - e_i^k)$.

From (4.1) and (4.2), it is immediate that (5.12) and (5.13) hold, respectively.

For any $i \in V, k \in V_i, m \in V_{i,k}, e_1, e_2 \in \delta(i)$ with $e_1 \neq e_2$, (5.14) holds when $y_{e_1}^k = 0$ or $y_{e_2}^m = 0$ since $s_i^k + s_i^m \leq 2$. Thus, it is sufficient to show that it also holds when $y_{e_1}^k = y_{e_2}^m = 1$. To this end, suppose that $y_{e_1}^k = y_{e_2}^m = 1$ where $e_1 = \{i, j\}$ and $e_2 = \{i, j'\}$ for some $j, j' \in V$ with $j \neq j'$. Then, without loss of generality, $y_{ij}^k = y_{j'i}^m = 1$ since $y_{ij}^k = y_{ij'}^m = 1$ or $y_{ji}^k = y_{j'i}^m = 1$ are not possible. Assume to the contrary that $s_i^k = s_i^m = 1$. Then, from (3.5), $\sum_{j \in V_{i,k}} y_{ij}^k = \sum_{j \in V_{i,k}} y_{ij}^m = 1$. Moreover, $e_i^m = 1$ since $y_{j'i}^m = 1$, leading a contradiction as $s_i^m = 1$. Thus, $s_i^k = 0$ or $s_i^m = 0$, that is $s_i^k + s_i^m \leq 1 = 3 - y_{e_1}^k - y_{e_2}^m$.

Since (5.14) holds, (5.15) also holds from (3.4).

For any $i \in V, k \in V_i, m \in V_{i,k}, e \in \delta(i)$, (5.14) holds when $y_e^k = 0$ or $y_e^m = 0$ since $e_i^k + s_i^m \leq 2$. Thus, it is sufficient to show that it also holds when $y_e^k = y_e^m = 1$. To this end, suppose that $y_e^k = y_e^m = 1$ where $e = \{i, j\}$ for some $j \in V$. If $y_{ij}^k = 1$, then $e_i^k = 0$ from (3.5). Similarly, if $y_{ji}^m = 1$, then $s_i^m = 0$ from (3.5). In both cases, (5.16) holds. Consider the case when $y_{ji}^k = y_{ij}^m = 1$, which implies that $x_{ji} = x_{ij} = 1$. This case is not possible. Thus, (5.16) always holds.

Since the solution vector in $\tilde{X}$ satisfies all the constraints of (Undir), it is also in Y. □

It is easy to see that a solution in $\tilde{X}$ yields the same objective value in (Dir-C) and (Undir). Thus, following Theorem 3, the optimal objective value of (Undir) provides a lower bound to that of (Dir-C).

5.2 Branch-and-Cut Algorithms

Similar to the branch-and-cut algorithms described in Section 4.2, we use three branch-and-cut schemes for solving (Undir). The first scheme starts by solving the relaxation of (Undir) which excludes the connectivity constraints (5.13) and direction consistency constraints (5.14)-(5.16) from the whole formulation. Then the violated connectivity constraints and direction consistency constraints are introduced only for integer solutions of the branch-and-cut tree. In the second scheme, while the direction consistency constraints are continued to be separated only for integer solutions, connectivity constraints are separated also for fractional solutions at the root node of the tree. In the last scheme, similar to the previous one, the direction consistency constraints are separated only for integer solutions and other violated constraints are separated also for fractional solutions only at the root node of the tree. First, we add the SLCIs for $j \in V_0$ with $0 \notin \mathcal{N}_j$ to the initial relaxation. Then, before checking whether a connectivity constraint $\sum_{e\in\delta(S)} x_e \geq \sum_{e\in\delta(j)} x_e$ is violated, we check the corresponding LCI given by $\sum_{e\in\delta(S)} x_e \geq \sum_{e\in\delta(j)} x_e + \sum_{i\in V_j} s_i^j$ if $0 \notin \mathcal{N}_j$ and $\mathcal{N}_j \subset S$. There might be some cases where $0 \notin \mathcal{N}_j$ holds, but $\mathcal{N}_j \backslash S \neq \emptyset$. In that case, we extend the set S by adding the nodes in $\mathcal{N}_j \backslash S$ to the set S and investigate whether the inequality $\sum_{e\in\delta(S\cup\mathcal{N}_j)} x_e \geq \sum_{e\in\delta(j)} x_e + \sum_{i\in V_j} s_i^j$ is satisfied. If one of these LCIs is violated, we introduce it instead of the connectivity constraint itself. Our separation procedures are explained in the following subsections.

Suppose that $\bar{G}_U = (\bar{V}, \bar{E})$ is the support graph induced by a given solution vector where $\bar{\nu} = (\bar{x}, \bar{y}, \bar{s}, \bar{e})$ constitutes the part of the solution associated with the binary variables. That is, $\bar{V} = \{j \in V : \sum_{e\in\delta(j)} \bar{x}_e > 0\}$ and $\bar{E} = \{e \in E : \bar{x}_e > 0\}$. Let S_k be the connected components of $\bar{G}_U$, where $k = 0, 1, \ldots, t$ and $0 \in S_0$. Notice $t \geq 1$ indicates that the solution contains subtours.

5.2.1 Separation of connectivity constraints

Our separation procedure for the connectivity constraints (5.13) is the same with the procedure followed for the separation of the connectivity constraints (4.2) of (Dir-C) explained in Section 4.2.1. Consider the same four cases regarding $\bar{\nu} = (\bar{x}, \bar{y}, \bar{s}, \bar{e})$ and the corresponding support graph $\bar{G}_U$. In the first three cases, we investigate violation of connectivity constraints (5.13) instead of (4.2) in the same manner as we do in Section 4.2.1 and we introduce the cuts (5.13) if they are violated.

Only in the last case, we use a more efficient way to identify violated connectivity cuts (5.13). Rather than solving $|\bar{V}_0|$ minimum cut problems, we first find a global minimum cut of the graph $\bar{G}_U$, which partitions $\bar{G}_U$ into two subsets such that the number of edges between these subsets is minimized. Suppose that $S^* \subset \bar{V}_0$ is the set of nodes inducing minimum cut of the graph $\bar{G}_U$. Then, for every $j \in S^*$, we have $cap(S^*) = cap(S^*(j))$ and thus, $S^* = S^*(j)$. That is, S^* is the minimum cut separating any $j \in S^*$ from node 0. Thus, we do not need to solve separate minimum cut problems for each node in S^*. If $cap(S^*)$ is greater than or equal to $\sum_{e\in\delta(j)} x_e$, then every $S \subset V_0$ containing j satisfies the inequality $\sum_{e\in\delta(S)} x_e \geq \sum_{e\in\delta(j)} x_e$, i.e., connectivity constraint (5.13). Otherwise, we add the violated connectivity cut to the model. Additionally, we know that $cap(S) \geq cap(S^*)$ for any cut S of the graph $\bar{G}_U$. This implies that the connectivity constraints induced by nodes $m \in \bar{V}_0 \backslash S^*$ such that $cap(S^*) \geq \sum_{e\in\delta(m)} x_e$ and the sets $S \subset V_0$ containing m are satisfied. For each of the remaining node l that the associated connectivity constraints are not investigated yet, we solve a minimum cut problem to find the minimum cut $S^*(l)$. Then, we check whether the connectivity constraint induced by l and $S^*(l)$ is satisfied in the same way as we do in the last case explained in Chapter 4.2.1.

5.2.2 Separation of lifted connectivity constraints

Our separation procedure for the lifted connectivity constraints (5.20) is the same with the procedure followed for separation of the lifted connectivity constraints (4.3) of (Dir-C) explained in Section 4.2.2. Similarly, we embed our search for violated LCIs into our connectivity constraint separation procedure.

Consider the same four cases regarding $\bar{\nu} = (\bar{x}, \bar{y}, \bar{s}, \bar{e})$ and the corresponding support graph $\bar{G}_U$. In the first three cases, we investigate violation of LCIs (5.20) instead of (4.3) in the same manner as we do in Section 4.2.2 and we introduce the cuts (5.20) if they are violated.

Only in the last case, as we do in the connectivity constraint separation explained in Section 5.2.1, we first find a global minimum cut of the graph $\bar{G}_U$. Once a global minimum cut S^* is determined, we check violation of LCIs for each $j \in S^*$. Provided that $0 \notin N_j$, the capacity of the cut $S^* \cup N_j$ is compared with the value $(\sum_{e \in \delta(j)} x_e + \sum_{i \in \bar{V}_j} \bar{s}_i^j)$ to determine whether the LCI associated with j and $S^* \cup N_j$ is satisfied or not. If it is violated, then we add it to the model. Upon completing the search for LCIs for each $j \in S^*$, if $0 \in \mathcal{N}_j$ or $0 \notin \mathcal{N}_j$ but no LCI is violated, we investigate the violation of the connectivity constraint regarding node $j \in S^*$ and set S^*.

Additionally, as the capacity of any cut of $\bar{G}_U$ is at least as large as that of S^*, LCIs induced by nodes $m \in \bar{V}_0 \backslash S^*$ such that $cap(S^*) \geq \sum_{e \in \delta(m)} x_e + \sum_{i \in \bar{V}_j} \bar{s}_i^j$ and the sets $S \subset V_0$ containing m are satisfied. For each of the remaining node l, we solve a minimum cut problem to find the minimum cut $S^*(l)$ and check whether the LCI induced by l and $S^*(l) \cup N_l$ is violated, provided that $0 \notin N_l$. If $0 \in \mathcal{N}_l$ or the LCI associated with the node l and $S^*(l)$ is found to be satisfied, then we investigate the corresponding connectivity inequality induced by l and $S^*(l)$ as we do in Section 5.2.1.

5.3 A Solution Approach Using (Undir)

Consider a solution to our problem. In this solution, if all customers are visited by the truck, then reversing the direction of the truck tour yields the same objective value, because it only involves the operation cost of the truck. On the other hand, if some customers are visited by the drones, the objective value may also involve the waiting times of both types of vehicles at rendezvous points. Consequently, reversing the direction of the tour does not necessarily lead to the same waiting costs and thus the same objective value. Hence, the direction of the truck tour has a considerable impact on the objective value.

Recall that (Undir) does not consider the direction of the truck tour. Therefore, its solution may include drone sorties assuming the opposite directions for the truck tour, which is not feasible for the original problem. Although the direction consistency constraints (5.14)-(5.16) make the directions of two drone sorties sharing a common launch/retrieval node the same, (Undir) is unable to do so when the sorties do not have a common launch/retrieval node. In this case, we can obtain a feasible solution to our problem by selecting a direction for the truck tour and making the directions of all the drone sorties consistent with that particular direction (by assigning $s_i^k = e_j^k$ and vice versa, if necessary). Since some adjustments to sortie directions might be needed, the objective value of this feasible solution may differ from the objective value given by (Undir).

We exemplify this situation in Figure 5.1. The left subfigure is a solution of (Undir) whereas the right one is a solution of (Dir-F) for the same problem. Since the constraints (5.14)-(5.16) are only able to correct the directions of two drone sorties having a common launch/retrieval node, (Undir) is unable to make the directions of sorties 1-2-3 and 5-6-7 consistent.

Thus, we develop Algorithm 2 that makes use of (Undir) and investigates both directions for the truck tour along with the corresponding drone sorties. In Algorithm 2, first we obtain a solution with (Undir). Then, using this solution, Algorithm 3 constructs two truck tours in which the truck moves in the opposite

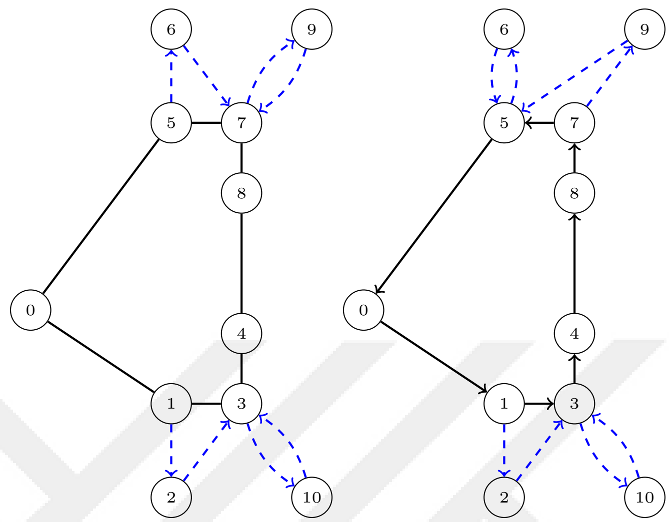


Figure 5.1: An example where (Undir) is unable to obtain consistency within the directions of drone sorties.

directions. Following that, with Algorithm 4, we evaluate the costs of the tours in which all the drone sorties are in line with the direction of the truck tour.

For a particular tour, Algorithm 4 changes the launch and retrieval nodes of a drone sortie with each other if the direction of the sortie conflicts with the truck's travel direction, then, it calculates the cost of the tour. To this end, Algorithm 4 first provides all the binary variables corresponding to both the truck's movements in a particular direction and corrected launch/retrieval nodes of the drone sorties to (Dir-F). Note that providing these binary variables reduces (Dir-F) to a linear program denoted by (Dir-LP) that can be solved quickly. By solving (Dir-LP), the cost associated with the tour, which is the objective value returned by (Dir-LP), is obtained. If (Dir-LP) returns a feasible solution, then the tour obtained by Algorithm 4 is feasible for the original problem. Thus, the cost associated with the tour provides an upper bound for the optimal objective value of the original problem. Note that it is also possible that (Dir-LP) is unable to provide a feasible solution. The reason is that when the launch and retrieval nodes of a drone sortie are exchanged, the associated drone waiting time might change and violate the drone endurance limitation. In that case, the tour obtained by correcting some drone sorties is not feasible for the original problem.

With the following lemma, we show the relations between the optimal objective values of the proposed formulations.

Lemma 1. *Suppose that (Dir-LP) has a feasible solution. Then,*

$$z(\textit{Undir}) \leq z(\textit{Dir-C}) = z(\textit{Dir-F}) \leq z(\textit{Dir-LP})$$

where $z(\cdot)$ is the optimal objective value of the corresponding formulation.

Proof. The first inequality is immediate from Theorem 3 and the second inequality holds since the optimal solution obtained by (Dir-LP) is a feasible solution for (Dir-C) and (Dir-F). □

In practice, the first inequality in Lemma 1 is due to the fact that (Undir) allows drone sorties in the opposite directions in favor of the objective function.

Suppose that (Dir-LP) in Algorithm 4 has a feasible solution for at least one of the two tours obtained by Algorithms 3, i.e., at least one tour is feasible for the original problem. Consider the tour with lower cost if both of them are feasible, otherwise consider the feasible one. If the cost of this tour, z(Dir-LP), is equal to z(Undir), then the tour is optimal for the original problem from Lemma 1. Otherwise, the tour is not guaranteed to be optimal, which we rarely observed (15 out of 480) in our computational experiments in Section 6. In this case, to obtain the optimal solution, as a part of Algorithm 2, we solve (Dir-C). In this stage of the algorithm, we prefer (Dir-C) to (Dir-F) based on our initial computational experiments. At this point, without solving (Dir-C) in Algorithm 2, it is possible to report an optimality gap, which is computed as [z(Dir-LP)$-z$(Dir-Undir)]/z(Dir-LP), by utilizing bounds for the optimal objective value from Lemma 1.

Secondly, suppose that (Dir-LP) in Algorithm 4 does not have a feasible solution for the two tours obtained by Algorithms 3, i.e., none of the tours is feasible for the original problem. In our computational experiments in Section 6, we seldom observed this case (1 out of 480). To obtain the optimal solution, as a part of Algorithm 2, we solve (Dir-C). Again, z(Undir) provides a lower bound for the optimal objective value of the problem. However, we do not have an upper bound since there is no feasible tour on hand in this case.

Algorithm 2 Solution methodology using (Undir).

Solve (Undir) to obtain $(\bar{x}, \bar{y}, \bar{s}, \bar{e})$ and its objective $z(\text{Undir})$.
$tour_1$, $tour_2 \leftarrow$ FIND TRUCK TOURS$(\bar{x})$
$cost_1 \leftarrow$ TOUR COST CALCULATION$(tour_1, \bar{y}, \bar{s}, \bar{e})$
$cost_2 \leftarrow$ TOUR COST CALCULATION$(tour_2, \bar{y}, \bar{s}, \bar{e})$
if $z(\text{Undir}) \neq \min\{cost_1, cost_2\}$ **then**
    Solve (Dir-C).
else
    The tour with lower cost is optimal.
end if

Algorithm 3 Finding the truck tours.

function FIND TRUCK TOURS$(\bar{x})$
    Input: $\bar{x}$
    Output: $tour_1, tour_2$

    Initialize $tour_1 \leftarrow (0)$ and $tour_2 \leftarrow (0)$, $currentNode \leftarrow 0$.
    for $e \in \delta(0)$ **do**
        if $\bar{x}_e = 1$ **then**
            $nextNode \leftarrow i$ s.t. $i \in e$.
            Break loop.
        end if
    end for
    $tour_1 \leftarrow (tour_1, nextNode)$ and $tour_2 \leftarrow (nextNode, tour_2)$.
    while $nextNode \neq 0$ **do** ▷ Terminate the tour at 0
        for $e \in \delta(nextNode)$ **do**
            if $\bar{x}_e = 1$ and $previousNode \notin e$ **then**
                $currentNode \leftarrow nextNode$.
                $nextNode \leftarrow i$ s.t. $i \in e$.
                Break loop.
            end if
        end for
        $tour_1 \leftarrow (tour_1, nextNode)$ and $tour_2 \leftarrow (nextNode, tour_2)$.
    end while
    Return $tour_1$ and $tour_2$.
end function

Algorithm 4 Cost calculation of a tour.

function TOUR COST CALCULATION($tour$, $\bar{y}$, $\bar{s}$, $\bar{e}$)
    Input: $tour$, $\bar{y}$, $\bar{s}$, $\bar{e}$
    Output: $cost$

    Initialize for all $i, j \in V$, $\hat{x}_{ij} \leftarrow 0$, $\hat{s}_i^j \leftarrow 0$, $\hat{e}_i^j \leftarrow 0$.
    for $i \in tour$ **do**
        $e \leftarrow \{i, j\}$ such that j is the next node on the tour after i.
        $\hat{x}_{ij} \leftarrow 1$.
        for $k \in V$ **do**
            if $\bar{y}_e^k = 1$ and ($\bar{s}_i^k = 1$ or $\bar{e}_i^k = 1$) **then**
                $\hat{s}_i^k \leftarrow 1$.   ▷ Drone is launched from i to visit k
            end if
            if $\bar{y}_e^k = 1$ and ($\bar{s}_j^k = 1$ or $\bar{e}_j^k = 1$) **then**
                $\hat{e}_j^k \leftarrow 1$.   ▷ Drone returns to j after visiting k
            end if
            if $\bar{s}_i^k = 1$ and $\bar{e}_i^k = 1$ **then**
                $\hat{s}_i^k \leftarrow 1$ and $\hat{e}_i^k \leftarrow 1$.   ▷ Drone is launched & retrieved at i
            end if
        end for
    end for
    Reduce (Dir-F) to (Dir-LP) by assigning the values of x, y, s, e as $\hat{x}, \hat{y}, \hat{s}, \hat{e}$.
    Solve (Dir-LP).
    if (Dir-LP) is feasible **then**
        $cost \leftarrow z$(Dir-LP), the objective value returned by (Dir-LP).
    else
        $cost \leftarrow \infty$.
    end if
    Return $cost$.
end function

Chapter 6

Computational Experiments

In this chapter, we present the results of our computational experiments. We begin with a description of the experimental setting and problem instances in Section 6.1. Then, we provide some implementation details of the branch-and-cut schemes in Section 6.2 before presenting a comparison of the proposed algorithms in Section 6.3. We conduct sensitivity analyses using different parameter settings in Section 6.4 and present results for some larger-scale instances in Section 6.5.

All experiments are performed on a Linux OS with Dual Intel Xeon E5-2690 v4 14 Core 2.6 GHz processor with 128 GB of RAM. The models and algorithms are implemented in Java and solved using IBM ILOG CPLEX 12.7.1.

6.1 Experimental Setting

In our experiments, for the locations of the nodes, we use some of the instances provided by Agatz et al. [31] for the TSP-D, namely, the uniform instances where coordinates for the locations are drawn independently and uniformly from $\{0, 1, \ldots, 100\}$. These instances are generated in different sizes. For our experiments in Section 6.3 and Section 6.4, as our problem size varies between 11 to

15 nodes, we use the next largest available set of uniform instances: *uniform-61-n20, uniform-62-n20, uniform-63-n20, uniform-64-n20, uniform-65-n20.* We assign the first location as the depot and the remaining locations up to problem size as the customer nodes. Based on these locations, we compute Euclidean distances and round them to the nearest integer. Then, to ensure that the triangular inequality holds, we perform a shortest path algorithm within all pairs to correct any violation. We assume that both the truck and drones follow the Euclidean distance metric in our computational study.

In order to compare the performances of proposed formulations and algorithms, we use different parameter settings. For the drone/truck speed ratio, i.e., v'/v, we consider the values 1.5 and 2, where v'/v represents that a drone is v'/v times as fast as the truck. Hence, the time required for a drone to travel from i to j, t'_{ij}, is v/v' times the time required for the truck to travel, t_{ij}.

Similarly for the truck/drone operating cost ratio, i.e., c_o/c'_o, we consider the values 2 and 4 in Section 6.3, then we also consider 10 for this ratio in Section 6.4 for sensitivity analyses. Those values are in line with the parameters considered in [46], [14], [50] and [55].

We also set the drone waiting/operating cost ratio, i.e., c'_w/c'_o, to 1 in all experiments, as we assume that the drone hovers while waiting for the truck, thus incurs the same cost. For the truck waiting/operating cost ratio, i.e., c_w/c_o, in Section 6.3, we use 0.75. This is because, while there are other components of the operating cost such as lease/purchase, repair & maintenance, insurance, permit costs and also driver wage, American Transportation Research Institute [60] points out that approximately 24% of the hourly average marginal cost of a truck is comprised of fuel cost and we assume that while the truck is waiting, it either stops or consumes considerably less fuel. Later, in Section 6.4 for sensitivity analyses, we also consider 0.5 for this ratio. In our experiments, without loss of generality, we set the values of the drone speed, waiting and operating costs to 1 unit and based on the ratios stated above, parameters regarding the truck are derived.

We also consider different drone endurance, T, values corresponding to relatively shorter and longer endurance times in our computational experiments. In Section 6.3, we use T values such that only 25^{th} and 50^{th} percentiles of all possible drone sorties, $t'_{ik} + t'_{kj} \leq T, i, j \in V, k \in V_0$, are allowed. In Section 6.4, we also allow 75^{th} percentile of all possible drone sorties. We denote the percentile allowed by α.

6.2 Implementation details of branch-and-cut schemes

In the branch-and-cut schemes for both (Dir-C) and (Undir), we invoke EdmondsKarpMFImpl procedure from the Java graph theory (JGraphT) library to find a minimum cut between a given pair of source and sick nodes in graph. It uses the algorithm in [61] and implements MinimumSTCutAlgorithm from JGraphT. In the branch-and-cut schemes for (Undir), we additionally invoke StoerWagnerMinimumCut procedure from JGraphT, which implements the algorithm in [62], to find a global minimum cut of a given undirected graph.

The separation routines for LCIs and connectivity constraints are implemented in callbacks. In all our branch-and-cut schemes, the separation for the solutions with an integral $\bar{z}$, which constitutes the part of the given solution vector associated with the binary variables, is conducted whenever they are identified by using LazyConstraintCallback in CPLEX Java Library. In the schemes involving separation for solutions with fractional $\bar{z}$, LCIs and connectivity constraints are separated only at the root node of the branch-and-cut tree by using UserCutCallback in CPLEX Java library.

Whenever we investigate whether a LCI or connectivity constraint is satisfied or not, the inequality is considered as violated if its violation exceeds 5%. The violation of an inequality $LHS \geq RHS$, where LHS and RHS represent the left- and right-hand-sides, respectively, is determined by $(RHS{-}LHS)/RHS$.

6.3 Comparison of the Proposed Algorithms

In this section, we compare the performances of proposed formulations and algorithms in terms of CPU times using a single thread and setting the time limit to one hour. That is, (Dir-F), (Dir-C) with three branch-and-cut schemes described in Section 4.2 and Algorithm 2 utilizing (Undir) with three branch-and-cut schemes described in Section 5.2, in total seven formulations/algorithms, are compared. For this comparison, we consider three different problem sizes, n, where instances consist of 11, 12, 13 nodes including the depot.

The results of our experiments are provided in Tables 6.1 through 6.3, each table corresponding to a different problem size and demonstrating the CPU times in seconds. In these tables, names of the instances where locations of the nodes are obtained are represented by N61 - N65 corresponding to the instances from uniform-61-n20 to uniform-65-n20. While DF represents (Dir-F), D1-D3 represent (Dir-C) solved with three schemes explained in Section 4.2 sequentially. Lastly, U1-U3 represent Algorithm 2 using (Undir) with three schemes explained Section 5.2, respectively. For each instance size, using each network N61 - N65 with 8 different sets of α, v'/v and c_o/c_o' values (by setting v' and c' to 1) explained in Section 6.1, we solve 40 instances. In total, the results regarding 120 instances with all sizes are shown.

In these tables, the shortest time associated with each instance is written in bold. If an instance cannot be solved to optimality within the time limit, we report the final gap, i.e., the percentage gap between the objective function value of the best integer solution and the best lower bound, in parenthesis. As it can be seen in Table 6.3, only two instances cannot be solved to optimality within one hour. Also, for U1-U3, when using Algorithm 2, the instances where the tours returned by Algorithms 3 and 4 are not optimal and (Dir-C) is solved as a part of Algorithm 2 are denoted by an asterisk (*). In those instances, when solving (Dir-C), the same scheme number with (Undir) is used. Among 120 instances, in only two instances, such a case occurs. Thus, although (Undir) and Algorithms 3 and 4 do not guarantee the optimal solution, they yield the optimal solution in almost

Table 6.1: Comparison of algorithms in terms of CPU times with $n = 11$

Loc.	α	$1/v$	c_o	DF	D1	D2	D3	U1	U2	U3
N61	0.25	1.5	2.0	42	8	10	5	4	**3**	**3**
			4.0	56	31	13	8	6	5	**2**
		2.0	2.0	20	9	8	4	**2**	3	**2**
			4.0	42	12	15	6	3	3	**2**
	0.5	1.5	2.0	82	68	104	**60**	92*	129*	73*
			4.0	132	70	103	60	41	**33**	34
		2.0	2.0	43	32	32	24	10	7	**6**
			4.0	39	28	24	13	**6**	7	9
N62	0.25	1.5	2.0	71	17	14	10	5	**3**	**3**
			4.0	103	36	40	49	19	17	**14**
		2.0	2.0	73	26	33	15	7	9	**6**
			4.0	60	41	22	18	9	9	**6**
	0.5	1.5	2.0	25	16	18	18	6	6	**4**
			4.0	126	52	66	66	65	40	**39**
		2.0	2.0	57	56	48	48	31	22	**21**
			4.0	71	54	31	31	17	20	**10**
N63	0.25	1.5	2.0	111	25	24	19	**8**	12	**8**
			4.0	106	70	79	34	18	22	**16**
		2.0	2.0	19	14	10	8	5	5	**3**
			4.0	34	27	19	10	5	7	**4**
	0.5	1.5	2.0	61	59	59	34	19	17	**16**
			4.0	117	84	78	78	**35**	98	70
		2.0	2.0	42	25	43	45	12	**10**	14
			4.0	198	49	86	63	35	**31**	56
N64	0.25	1.5	2.0	20	9	6	6	**2**	**2**	**2**
			4.0	31	17	12	9	**3**	4	4
		2.0	2.0	15	4	4	5	3	**2**	**2**
			4.0	20	8	6	4	**2**	**2**	**2**
	0.5	1.5	2.0	81	61	76	53	**32**	38	**32**
			4.0	159	94	55	60	**28**	29	32
		2.0	2.0	115	55	82	51	17	**15**	17
			4.0	98	40	26	39	13	10	**9**
N65	0.25	1.5	2.0	14	5	5	3	**2**	**2**	**2**
			4.0	42	16	15	9	9	10	**7**
		2.0	2.0	22	7	4	**2**	**2**	**2**	**2**
			4.0	31	11	10	8	6	6	**4**
	0.5	1.5	2.0	85	44	28	22	11	11	**10**
			4.0	161	65	71	38	36	50	**34**
		2.0	2.0	68	44	44	23	29	25	**14**
			4.0	53	49	50	**22**	**22**	27	27

Table 6.2: Comparison of algorithms in terms of CPU times with $n = 12$

Loc.	α	$1/v$	c_o	DF	D1	D2	D3	U1	U2	U3
N61	0.25	1.5	2.0	151	45	36	30	**6**	7	8
			4.0	216	111	126	74	16	**14**	16
		2.0	2.0	112	25	38	24	8	6	**4**
			4.0	118	56	75	42	12	12	**6**
	0.5	1.5	2.0	163	366	189	155	115	82	**35**
			4.0	1108	373	303	417	414	**246**	306
		2.0	2.0	715	127	239	121	85	92	**69**
			4.0	208	237	130	65	51	75	**28**
N62	0.25	1.5	2.0	111	45	36	28	16	16	**10**
			4.0	296	116	87	129	41	31	**29**
		2.0	2.0	153	113	61	38	23	**15**	16
			4.0	236	65	71	71	17	16	**13**
	0.5	1.5	2.0	116	61	79	55	31	28	**23**
			4.0	413	306	304	277	268	**126**	197
		2.0	2.0	142	124	121	89	**55**	69	63
			4.0	106	136	107	84	61	47	**44**
N63	0.25	1.5	2.0	54	31	40	19	16	7	**5**
			4.0	267	149	168	74	60	**48**	67
		2.0	2.0	71	34	27	19	10	20	**9**
			4.0	111	73	69	52	**12**	18	39
	0.5	1.5	2.0	51	44	43	22	11	11	**9**
			4.0	251	304	191	215	242	**122**	182
		2.0	2.0	121	91	115	48	58	**41**	**41**
			4.0	511	269	223	98	127	**81**	150
N64	0.25	1.5	2.0	181	61	32	32	12	**11**	**11**
			4.0	453	148	147	112	31	38	**22**
		2.0	2.0	163	49	38	28	11	**9**	12
			4.0	144	48	34	39	12	15	**11**
	0.5	1.5	2.0	305	236	332	344	**126**	210	162
			4.0	438	262	377	201	155	169	**102**
		2.0	2.0	835	241	145	134	62	**61**	78
			4.0	416	150	182	148	79	54	**38**
N65	0.25	1.5	2.0	89	26	19	21	14	**9**	13
			4.0	128	96	46	44	**28**	36	31
		2.0	2.0	54	21	11	14	11	**6**	**6**
			4.0	63	31	19	23	21	16	**15**
	0.5	1.5	2.0	175	107	151	155	102	**34**	58
			4.0	347	249	194	210	249	**183**	228
		2.0	2.0	196	128	170	171	157	**116**	189
			4.0	121	100	102	113	122	**68**	102

Table 6.3: Comparison of algorithms in terms of CPU times with $n = 13$

Loc.	α	$1/v$	c_o	DF	D1	D2	D3	U1	U2	U3
N61	0.25	1.5	2.0	111	36	79	33	20	16	**7**
			4.0	268	212	148	97	**34**	39	54
		2.0	2.0	146	56	42	29	17	15	**14**
			4.0	616	82	84	54	39	25	**22**
	0.5	1.5	2.0	1020	323	264	**241**	416	313	281
			4.0	1063	2455	1192	1061	1396	**919**	1318
		2.0	2.0	861	638	796	431	213	437	**171**
			4.0	641	337	180	423	166	**99**	430
N62	0.25	1.5	2.0	626	257	170	111	**19**	26	20
			4.0	761	474	203	292	125	**54**	78
		2.0	2.0	1153	307	174	125	62	**47**	51
			4.0	574	211	206	**156**	253*	239*	181*
	0.5	1.5	2.0	460	426	177	216	137	110	**101**
			4.0	1698	1292	1138	1426	**806**	813	1119
		2.0	2.0	465	322	384	251	158	**103**	105
			4.0	384	517	233	218	290	**122**	174
N63	0.25	1.5	2.0	367	85	55	17	30	24	**9**
			4.0	955	360	360	189	368	146	**89**
		2.0	2.0	226	69	77	47	26	19	**13**
			4.0	1255	245	167	122	55	46	**21**
	0.5	1.5	2.0	841	108	74	63	56	24	**16**
			4.0	2222	903	727	**513**	713	926	731
		2.0	2.0	665	202	120	81	133	100	**58**
			4.0	1409	1049	1084	369	578	613	**342**
N64	0.25	1.5	2.0	320	117	152	259	48	**26**	46
			4.0	1011	412	247	271	**77**	86	94
		2.0	2.0	514	118	94	117	43	33	**32**
			4.0	617	152	153	81	37	39	**27**
	0.5	1.5	2.0	(5.55)	1838	1894	2184	1095	**642**	1424
			4.0	(21.94)	(7.42)	(3.50)	(5.50)	**892**	2101	2681
		2.0	2.0	2869	2032	748	1412	828	320	**175**
			4.0	2094	2860	1092	1800	**382**	424	713
N65	0.25	1.5	2.0	296	204	162	161	**43**	94	55
			4.0	673	295	224	224	100	89	**53**
		2.0	2.0	214	113	63	68	39	39	**22**
			4.0	279	175	134	115	86	**42**	71
	0.5	1.5	2.0	1314	600	466	201	351	312	**194**
			4.0	1428	1187	999	599	**325**	1228	632
		2.0	2.0	779	606	641	426	353	791	**162**
			4.0	314	422	**144**	170	357	250	286

all of the instances. It is noteworthy that using Algorithm 2 without solving (Dir-C) in Step 6 returns a solution with 0.0053% gap on the average of 120 instances, where the gap is computed as $[z(\text{Dir-LP}) - z(\text{Dir-Undir})]/z(\text{Dir-LP})$ by utilizing bounds from Lemma 1 in Section 5.3.

In Table 6.4, we summarize the results of our experiments for each problem size and use the same abbreviations for the algorithms with the previous tables. For the instances solved to optimality within time limit using (Dir-F), we standardize the computation times obtained through each algorithm dividing them by the CPU time of (Dir-F) and then multiplying by 100. We report the average standardized CPU times of the algorithms for each instance size as well as the overall average in Table 6.4. Also, Figure 6.1 displays the distribution of standardized CPU times of the algorithms for all these instances. The results show that Algorithm 2 using (Undir) with branch-and-cut scheme 3 outperforms all the other algorithms with respect to computation time. Specifically, it outperforms (Dir-F) by up to 98.3% with an average of 77.4%. Therefore, in our sensitivity analyses in the following section, we report the results of Algorithm 2 using (Undir) with branch-and-cut scheme 3.

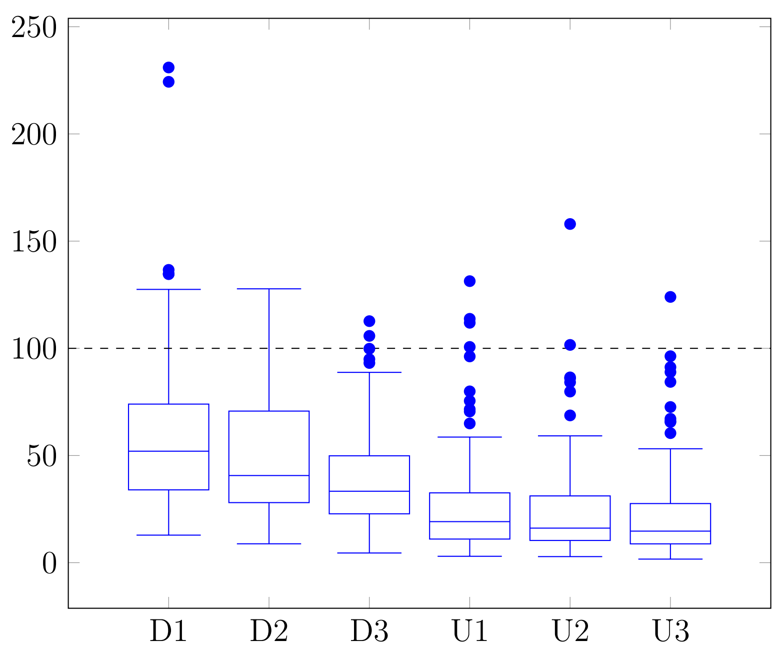


Figure 6.1: Comparison of standardized CPU times of the algorithms

Table 6.4: Average standardized CPU times of the algorithms relative to DF

n	DF	D1	D2	D3	U1	U2	U3
11	100	53.3	51.7	37.9	22.8	24.7	19.9
12	100	60.8	54.7	43.5	30.3	23.8	24.7
13	100	59.5	42.4	36.8	27.9	25.7	23.1
Avg.	100	57.9	49.6	39.4	27	24.7	22.6

6.4 Sensitivity Analyses

In this section, we analyze the effects of drone endurance, drone/truck speed and operating cost ratios, and also truck waiting/operating cost ratio on shares of the components of total cost, drone usage and solution time. Based on our results in previous section, in our experiments for sensitivity analyses, we solve the problems with Algorithm 2 using (Undir) with branch-and-cut scheme 3. Different from the previous section, we allow parallel processing with up to 10 threads and do not impose a time limit. For these analyses, we consider two problem sizes of 14 and 15 nodes including the depot.

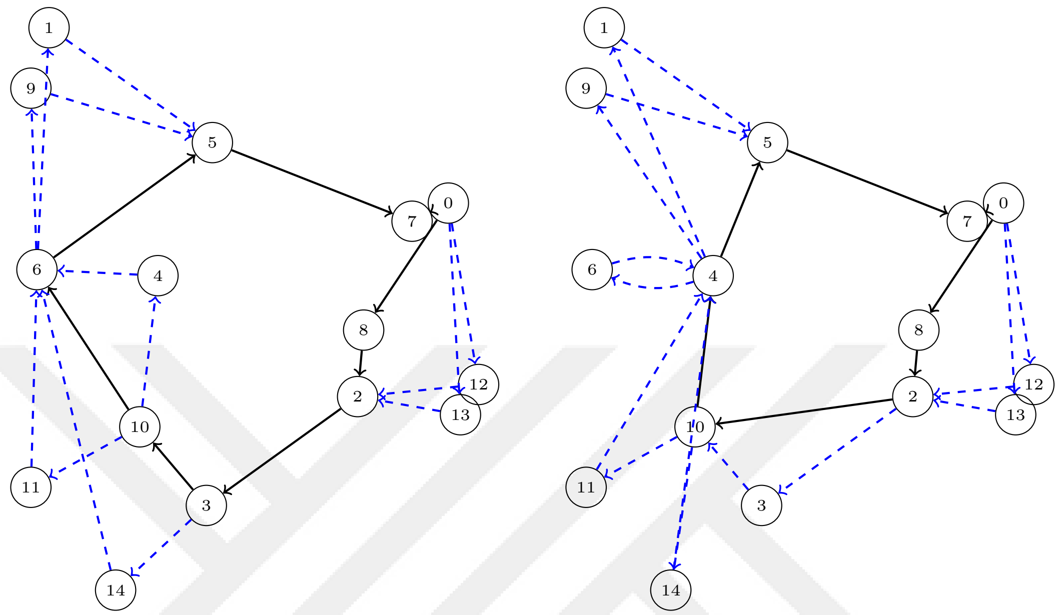
While the results of our experiments for $n = 14$ are provided in Appendix A Table A.1-Table A.5, the results for $n = 15$ are provided in Table 6.5-Table 6.9. Each table corresponds to a different layout of nodes from uniform-61-n20 to uniform-65-n20. In these tables, the first four columns represent the problem parameters, namely, percentile of allowed sorties reflecting drone endurance, drone/truck speed, truck/drone operating cost, and lastly truck waiting cost, where drone speed and operating cost are set to 1. The first one of the following five columns shows the objective values (Obj.) which are the total operational costs and the remaining ones present the shares of the cost components, that is, the truck's total transportation (C_o) and waiting (C_w), drones' total transportation (C'_o) and waiting (C'_w) costs, in sequence. In the next two columns, total number of customers served by drones is denoted by #dv and the minimum number of drones needed for a delivery plan is denoted by #dn. Here, #dn is computed by Algorithm 1 in Section 3. Finally, the last column shows the solution times in seconds. For each instance size, using each layout of nodes with 36 different parameter configurations explained thoroughly in Section 6.1, we solve 360 instances in total for sensitivity analyses.

Similar to the previous section, the instances where Algorithms 3 and 4 could not return the optimal solution and thus (Dir-C) with branch-and-cut scheme 3 is solved as a part of Algorithm 2 are denoted by an asterisk (*) next to their solution times. Among 360 instances, in 13 instances this case occurs. However, it is worth noting that using Algorithm 2 without solving (Dir-C) returns a solution with 0.0056% gap for $n = 14$ and 0.0049% gap for $n = 15$ on the average of all instances of the corresponding problem size. Recall that the gap is computed as $[z(\text{Dir-LP})-z(\text{Dir-Undir})]/z(\text{Dir-LP})$. Additionally, in 1 out of 480 instances solved so far, we observed that Algorithms 3 and 4 could not provide a feasible solution, thus (Dir-C) in Algorithm 2 is solved to obtain the optimal solution.

It can be observed from Tables 6.5 through Table 6.9 that the shares of total waiting cost of the drones are considerably low compared to other cost components. This indicates that making the drones wait is not preferable although their waiting cost is lower than that of the truck. The reason is that it is more advantageous to operate the drones instead of making them wait, as operating a drone is cheaper than operating the truck. Moreover, for the drones to wait, the truck needs to travel more so that it arrives at the meeting point later than the drones, which is not desirable in this case.

Endurance. For the instances having the same set of parameters, when the number of allowed drone sorties increases (i.e., when α increases), more customers may be served by the drones as it is shown in Figure 6.2, and this leads to lower objective values. The impact of increasing α on the objective value is relatively small when the operational cost of the truck is lower. On the other hand, for larger α values, the solution times are higher, since the number of possible drone deliveries are greater.

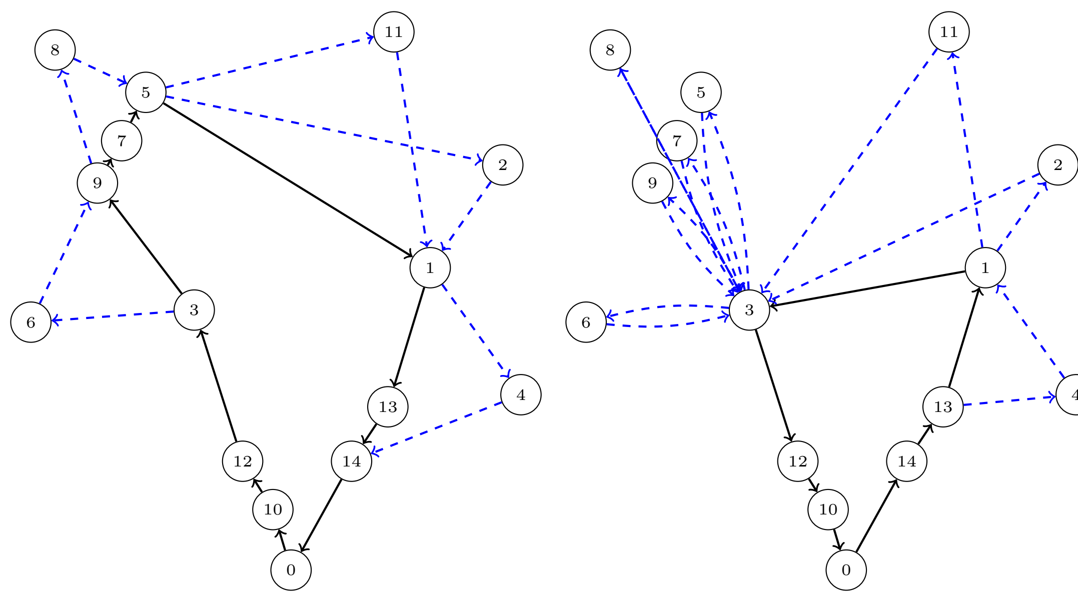
Speed. A decrease in the truck's speed makes using the truck less advantageous for deliveries. Thus, with lower speed for the truck, the number of customers



(a) When $\alpha = 25^{th}$

(b) When $\alpha = 50^{th}$

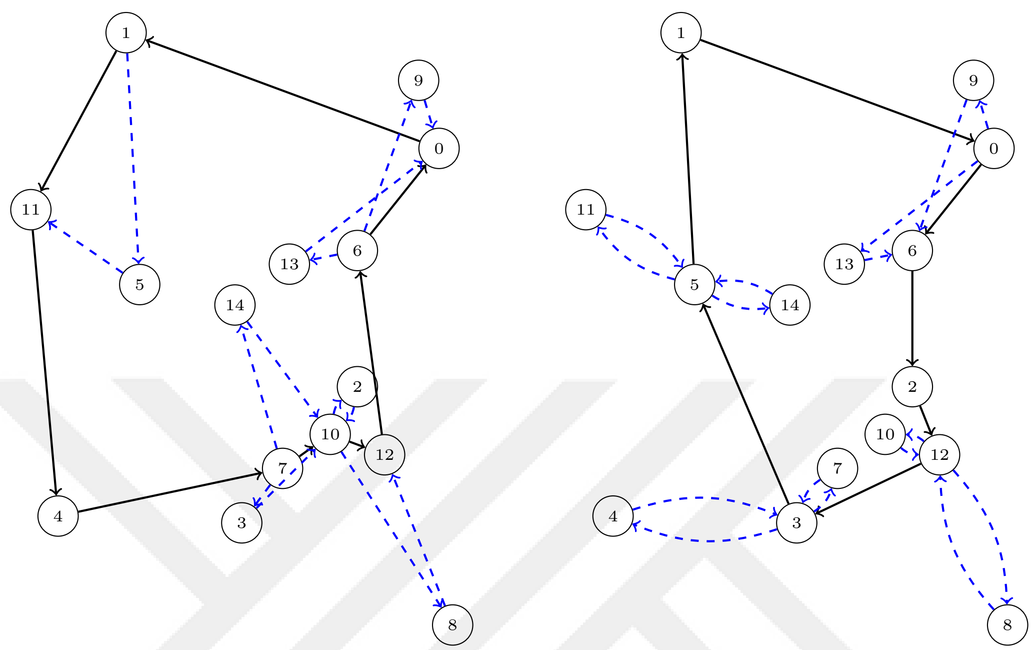
Figure 6.2: The optimal solutions of uniform-62-n20 with $1/v = 1.5$, $c_o = 10$, $c_w = 7.5$.



(a) When $1/v = 1.5$

(b) When $1/v = 2$

Figure 6.3: The optimal solutions of uniform-63-n20 with $\alpha = 50^{th}$, $c_o = 4$, $c_w = 2$.



(a) When $c_w = 7.5$ (b) When $c_w = 5$

Figure 6.4: The optimal solutions of uniform-61-n20 with $\alpha = 25^{th}$, $1/v = 2$, $c_o = 10$.

visited by the drones is higher. In those cases, as the truck makes less deliveries, instead of moving to new customers, the truck waits for the drones to complete their deliveries and turn back as it is shown in Figure 6.3.

Cost. Similar to the effects of lower speed for the truck, an increase in the truck's operational cost makes using the truck less preferable. Thus, with higher costs for the truck, the drones tend to visit more customers which leads the truck to wait for the drones longer and more frequently. Additionally, when the truck's waiting cost is lower, it waits more. In some cases, the truck may even act like a mobile depot, that is, it launches several drones at a customer location and waits for them to return before proceeding to its next stop as it is demonstrated in Figure 6.4. In that case, as the truck waits more rather than moving, the share of the total operating cost of the truck is less and conversely the share of the total waiting cost of the truck is more compared to the instances where the truck makes more deliveries.

Table 6.5: Sensitivity analyses on uniform-61-n20 with $n = 15$

α	$1/v$	c_o	c_w	Obj.	C_o	C_w	C'_o	C'_w	#dv	#dn	Time
25^{th}	1.5	2.0	1.0	904	84.6%	5.2%	10.2%	0.0%	2	1	28
			1.5	926	88.4%	5.1%	6.5%	0.0%	1	1	41
		4.0	2.0	1679	82.9%	5.7%	11.1%	0.4%	4	2	57
			3.0	1727	80.6%	8.3%	10.8%	0.4%	4	2	75
		10.0	5.0	3841	76.9%	14.2%	8.6%	0.2%	8	3	47
			7.5	3999	84.0%	8.9%	6.8%	0.3%	6	3	923*
	2.0	2.0	1.0	1140	77.2%	8.9%	13.8%	0.1%	3	1	13
			1.5	1182	82.6%	5.6%	11.8%	0.0%	3	2	23
		4.0	2.0	2122	82.9%	9.6%	7.4%	0.0%	3	1	12
			3.0	2200	83.3%	5.7%	10.8%	0.2%	5	2	36
		10.0	5.0	4958	79.5%	14.7%	5.8%	0.0%	8	2	39
			7.5	5104	88.2%	6.2%	5.6%	0.1%	7	4	37
50^{th}	1.5	2.0	1.0	847	74.4%	4.1%	21.5%	0.0%	4	3	650
			1.5	862	79.7%	1.1%	18.0%	1.2%	3	2	822
		4.0	2.0	1366	61.9%	13.1%	24.9%	0.1%	7	3	317
			3.0	1455	58.1%	18.4%	23.4%	0.1%	7	3	964
		10.0	5.0	2904	72.8%	15.4%	11.7%	0.1%	7	3	288
			7.5	3127	67.6%	21.5%	10.9%	0.0%	7	3	1265
	2.0	2.0	1.0	991	56.9%	8.7%	34.3%	0.1%	7	3	189
			1.5	1026	62.3%	5.1%	32.1%	0.5%	6	3	932†
		4.0	2.0	1631	69.2%	8.5%	22.3%	0.1%	7	3	123
			3.0	1700	66.4%	12.2%	21.4%	0.1%	7	3	172
		10.0	5.0	3521	75.0%	12.8%	11.8%	0.5%	9	4	167
			7.5	3702	76.2%	14.0%	9.8%	0.0%	7	3	227
75^{th}	1.5	2.0	1.0	846	74.4%	4.1%	21.5%	0.0%	4	3	11566
			1.5	861	63.8%	3.3%	32.4%	0.5%	4	3	18918
		4.0	2.0	1262	51.8%	6.7%	41.4%	0.2%	9	7	5785
			3.0	1301	54.8%	6.0%	38.6%	0.5%	8	6	25342
		10.0	5.0	2298	57.4%	14.5%	28.0%	0.1%	11	8	641
			7.5	2441	67.0%	10.4%	22.1%	0.5%	9	7	2874
	2.0	2.0	1.0	962	49.1%	5.6%	45.2%	0.1%	8	4	2526
			1.5	982	48.1%	4.3%	47.4%	0.3%	8	5	4395
		4.0	2.0	1447	54.2%	6.6%	39.0%	0.2%	10	6	1105
			3.0	1476	58.5%	5.1%	35.6%	0.7%	9	6	2028
		10.0	5.0	2678	65.0%	10.5%	23.8%	0.7%	11	7	155
			7.5	2818	61.7%	14.9%	22.6%	0.7%	11	7	433

† indicates that (Dir-LP) in Algorithm 4 does not have a feasible solution.

Table 6.6: Sensitivity analyses on uniform-62-n20 with $n = 15$

α	$1/v$	c_o	c_w	Obj.	C_o	C_w	C'_o	C'_w	#dv	#dn	Time
25^{th}	1.5	2.0	1.0	940	83.3%	2.6%	13.8%	0.3%	3	2	108
			1.5	952	82.2%	3.9%	13.7%	0.3%	3	2	195
		4.0	2.0	1612	74.8%	4.3%	20.5%	0.3%	6	2	185
			3.0	1626	74.2%	1.3%	23.2%	1.4%	6	3	94
		10.0	5.0	3415	84.8%	2.8%	12.1%	0.3%	7	4	133
			7.5	3462	83.6%	4.1%	12.0%	0.3%	7	4	146
	2.0	2.0	1.0	1157	60.5%	8.7%	30.8%	0.0%	7	3	40
			1.5	1198	74.8%	2.1%	22.0%	1.1%	5	2	164
		4.0	2.0	1953	63.9%	12.5%	23.4%	0.2%	9	4	135*
			3.0	2017	76.5%	2.8%	19.0%	1.6%	7	2	82
		10.0	5.0	4191	74.4%	14.6%	10.9%	0.1%	9	4	71*
			7.5	4383	79.4%	9.8%	10.5%	0.3%	9	4	53
50^{th}	1.5	2.0	1.0	940	83.3%	2.6%	13.8%	0.3%	3	2	136
			1.5	952	82.2%	3.9%	13.7%	0.3%	3	2	439
		4.0	2.0	1560	62.7%	5.8%	30.9%	0.6%	8	6	937
			3.0	1602	61.0%	7.9%	29.8%	1.3%	8	5	1691
		10.0	5.0	3153	77.5%	6.7%	15.1%	0.7%	8	5	436
			7.5	3258	75.0%	9.7%	14.6%	0.7%	8	5	702
	2.0	2.0	1.0	1138	61.5%	6.6%	31.9%	0.0%	7	3	166
			1.5	1170	65.6%	7.2%	27.0%	0.2%	6	3	170
		4.0	2.0	1866	66.9%	3.4%	29.0%	0.6%	9	6	148
			3.0	1898	65.8%	5.1%	28.6%	0.6%	9	6	462
		10.0	5.0	3834	81.4%	4.2%	14.1%	0.3%	9	6	103
			7.5	3914	79.7%	6.1%	13.8%	0.3%	9	6	98
75^{th}	1.5	2.0	1.0	940	83.3%	2.6%	13.8%	0.3%	3	2	215
			1.5	952	82.2%	3.9%	13.7%	0.3%	3	2	534
		4.0	2.0	1559	67.4%	4.0%	28.2%	0.5%	7	5	4149
			3.0	1586	66.2%	4.6%	28.8%	0.4%	7	5	9517
		10.0	5.0	2783	59.8%	11.0%	29.2%	0.0%	11	9	274
			7.5	2934	56.7%	15.0%	28.1%	0.2%	11	10	2523
	2.0	2.0	1.0	1138	61.5%	6.6%	31.9%	0.0%	7	3	304
			1.5	1169	65.6%	7.2%	27.0%	0.2%	6	3	840
		4.0	2.0	1790	49.6%	5.8%	44.4%	0.2%	11	8	501
			3.0	1842	48.2%	8.5%	43.2%	0.2%	11	8	2050
		10.0	5.0	3269	67.3%	7.2%	25.2%	0.3%	11	9	199
			7.5	3379	65.7%	9.3%	24.4%	0.5%	11	9	190

Table 6.7: Sensitivity analyses on uniform-63-n20 with $n = 15$

α	$1/v$	c_o	c_w	Obj.	C_o	C_w	C'_o	C'_w	#dv	#dn	Time
25^{th}	1.5	2.0	1.0	871	80.9%	2.1%	17.0%	0.0%	3	2	9
			1.5	880	80.1%	3.1%	16.8%	0.0%	3	2	10
		4.0	2.0	1592	84.8%	2.3%	12.4%	0.6%	4	2	52
			3.0	1610	83.8%	3.4%	12.2%	0.6%	4	2	53
		10.0	5.0	3628	76.9%	13.9%	9.1%	0.1%	7	4	199*
			7.5	3670	92.0%	0.7%	6.7%	0.6%	4	1	102
	2.0	2.0	1.0	1086	86.6%	0.6%	12.9%	0.0%	3	2	9
			1.5	1089	86.3%	0.8%	12.9%	0.0%	3	2	8
		4.0	2.0	2008	75.7%	10.4%	13.9%	0.0%	6	3	21
			3.0	2036	88.4%	0.9%	9.3%	1.4%	4	2	47
		10.0	5.0	4534	82.0%	10.6%	7.4%	0.0%	7	4	54*
			7.5	4741	80.2%	12.3%	7.0%	0.5%	6	3	23
50^{th}	1.5	2.0	1.0	871	80.9%	2.1%	17.0%	0.0%	3	2	13
			1.5	880	80.1%	3.1%	16.8%	0.0%	3	2	18
		4.0	2.0	1543	77.4%	2.3%	20.1%	0.2%	5	3	365
			3.0	1561	76.5%	3.5%	19.9%	0.2%	5	3	501
		10.0	5.0	3189	66.3%	15.9%	17.8%	0.0%	8	7	130
			7.5	3334	85.5%	0.8%	13.3%	0.5%	6	3	534
	2.0	2.0	1.0	1086	86.6%	0.6%	12.9%	0.0%	3	2	25
			1.5	1089	86.3%	0.8%	12.9%	0.0%	3	2	17
		4.0	2.0	1886	59.8%	10.4%	29.6%	0.2%	8	6	64
			3.0	1964	81.1%	2.9%	16.0%	0.0%	5	3	209
		10.0	5.0	3872	72.8%	12.7%	14.4%	0.1%	8	6	37
			7.5	4117	68.5%	17.9%	13.6%	0.1%	8	6	64
75^{th}	1.5	2.0	1.0	871	80.9%	2.1%	17.0%	0.0%	3	2	14
			1.5	880	80.1%	3.1%	16.8%	0.0%	3	2	23
		4.0	2.0	1543	77.4%	2.3%	20.1%	0.2%	5	3	1128
			3.0	1561	76.5%	3.5%	19.9%	0.2%	5	3	1924
		10.0	5.0	2778	48.6%	20.5%	30.8%	0.1%	11	10	89
			7.5	3057	55.4%	20.9%	23.7%	0.0%	9	9	693
	2.0	2.0	1.0	1086	86.6%	0.6%	12.9%	0.0%	3	2	30
			1.5	1089	86.3%	0.8%	12.9%	0.0%	3	2	34
		4.0	2.0	1796	40.1%	12.8%	47.1%	0.0%	11	10	232
			3.0	1882	48.0%	11.5%	40.1%	0.4%	9	8	612
		10.0	5.0	3221	55.9%	17.9%	26.3%	0.0%	11	10	52
			7.5	3508	51.3%	24.6%	24.1%	0.0%	11	10	114

Table 6.8: Sensitivity analyses on uniform-64-n20 with $n = 15$

α	$1/v$	c_o	c_w	Obj.	C_o	C_w	C'_o	C'_w	#dv	#dn	Time
25^{th}	1.5	2.0	1.0	999	83.1%	5.2%	11.6%	0.1%	2	1	57
			1.5	1025	81.0%	7.6%	11.3%	0.0%	2	1	57
		4.0	2.0	1828	73.2%	12.4%	14.0%	0.5%	6	2	177
			3.0	1904	80.0%	8.2%	10.9%	0.8%	4	2	168
		10.0	5.0	4110	80.3%	11.6%	7.6%	0.5%	7	2	48
			7.5	4349	75.9%	16.5%	7.2%	0.5%	7	2	63
	2.0	2.0	1.0	1252	71.2%	11.3%	17.4%	0.0%	6	2	35
			1.5	1293	80.7%	6.3%	13.0%	0.0%	3	2	49
		4.0	2.0	2236	78.7%	8.9%	12.3%	0.0%	7	2	20
			3.0	2334	76.4%	12.1%	11.5%	0.0%	6	2	35
		10.0	5.0	5176	85.0%	9.7%	5.3%	0.0%	7	2	21
			7.5	5413	81.3%	13.2%	5.6%	0.0%	7	3	28
50^{th}	1.5	2.0	1.0	980	82.7%	3.7%	13.7%	0.0%	3	3	352
			1.5	988	84.0%	1.7%	14.2%	0.1%	2	2	375
		4.0	2.0	1714	78.4%	3.7%	17.5%	0.4%	6	3	1670*
			3.0	1733	79.6%	2.2%	17.6%	0.5%	5	2	1021
		10.0	5.0	3793	87.0%	4.2%	8.7%	0.1%	7	4	667
			7.5	3853	86.8%	4.6%	8.3%	0.3%	6	3	854
	2.0	2.0	1.0	1220	73.1%	7.4%	19.5%	0.0%	6	3	262
			1.5	1238	84.3%	0.5%	15.2%	0.0%	3	2	381
		4.0	2.0	2137	82.4%	1.2%	15.7%	0.7%	7	3	286
			3.0	2148	83.1%	1.0%	15.2%	0.7%	6	3	336
		10.0	5.0	4816	91.4%	1.3%	7.0%	0.3%	7	3	294
			7.5	4848	90.7%	2.0%	6.9%	0.3%	7	3	426
75^{th}	1.5	2.0	1.0	980	82.7%	3.7%	13.7%	0.0%	3	3	867
			1.5	988	84.0%	1.7%	14.2%	0.1%	2	2	814
		4.0	2.0	1584	48.1%	9.9%	42.0%	0.0%	8	7	279
			3.0	1662	45.8%	14.2%	40.0%	0.0%	8	7	1695
		10.0	5.0	2945	63.1%	13.3%	23.5%	0.0%	9	8	371
			7.5	3141	59.2%	18.7%	22.1%	0.0%	9	8	509
	2.0	2.0	1.0	1220	73.1%	7.4%	19.5%	0.0%	6	3	656
			1.5	1235	74.5%	1.7%	23.6%	0.2%	5	3	2611
		4.0	2.0	1813	56.0%	7.3%	36.7%	0.0%	8	7	469
			3.0	1879	54.1%	10.5%	35.4%	0.0%	8	7	778
		10.0	5.0	3513	70.6%	9.4%	20.0%	0.0%	9	8	105
			7.5	3678	67.4%	13.5%	19.1%	0.0%	9	8	306

Table 6.9: Sensitivity analyses on uniform-65-n20 with $n = 15$

α	$1/v$	c_o	c_w	Obj.	C_o	C_w	C'_o	C'_w	#dv	#dn	Time
25^{th}	1.5	2.0	1.0	815	72.5%	6.6%	21.0%	0.0%	4	2	179
			1.5	834	86.7%	2.1%	11.3%	0.0%	2	1	589
		4.0	2.0	1359	67.5%	9.3%	22.6%	0.6%	7	3	141
			3.0	1422	64.6%	13.3%	21.6%	0.6%	7	3	1514
		10.0	5.0	2820	73.9%	11.3%	14.7%	0.1%	9	5	145
			7.5	2976	71.1%	15.2%	13.6%	0.1%	8	5	357
	2.0	2.0	1.0	959	63.8%	5.1%	30.9%	0.2%	7	3	49
			1.5	978	68.3%	5.8%	25.7%	0.2%	6	2	50
		4.0	2.0	1604	69.3%	6.1%	23.9%	0.7%	9	4	55
			3.0	1652	68.3%	7.6%	23.4%	0.7%	8	4	42
		10.0	5.0	3417	81.4%	6.7%	11.6%	0.3%	9	4	31
			7.5	3532	78.7%	9.8%	11.2%	0.3%	9	4	44
50^{th}	1.5	2.0	1.0	802	73.6%	2.7%	23.6%	0.1%	4	3	800
			1.5	813	72.6%	4.1%	23.2%	0.1%	4	3	1136
		4.0	2.0	1300	70.6%	4.6%	24.5%	0.3%	7	5	1650
			3.0	1330	69.0%	6.8%	23.9%	0.3%	7	5	4547
		10.0	5.0	2486	65.8%	10.7%	23.3%	0.2%	11	8	107
			7.5	2618	62.4%	15.2%	22.2%	0.2%	11	8	282
	2.0	2.0	1.0	950	64.4%	1.5%	33.3%	0.8%	7	3	181
			1.5	957	63.9%	2.2%	33.0%	0.8%	7	3	248
		4.0	2.0	1513	58.7%	6.2%	35.0%	0.1%	10	7	190
			3.0	1560	56.9%	9.0%	33.9%	0.1%	10	7	417
		10.0	5.0	2984	74.4%	7.5%	18.0%	0.1%	10	7	92
			7.5	3096	71.7%	10.9%	17.3%	0.1%	10	7	120
75^{th}	1.5	2.0	1.0	802	73.6%	2.7%	23.6%	0.1%	4	3	1154
			1.5	813	72.6%	4.1%	23.2%	0.1%	4	3	3095
		4.0	2.0	1299	70.6%	4.6%	24.5%	0.3%	7	5	1553
			3.0	1329	69.0%	6.8%	23.9%	0.3%	7	5	8796
		10.0	5.0	2342	50.0%	19.2%	30.7%	0.1%	12	11	390
			7.5	2567	45.6%	26.3%	28.0%	0.1%	12	11	939
	2.0	2.0	1.0	950	64.4%	1.5%	33.3%	0.8%	7	3	358
			1.5	957	63.9%	2.2%	33.0%	0.8%	7	3	492
		4.0	2.0	1501	41.5%	12.3%	46.1%	0.1%	12	10	347
			3.0	1560	56.9%	9.0%	33.9%	0.1%	10	7	905
		10.0	5.0	2712	57.5%	16.6%	25.8%	0.1%	12	11	42
			7.5	2937	53.1%	23.0%	23.8%	0.1%	12	11	405

6.5 Tests on Larger Instances

In this section, we present the results of our experiments on larger problem instances. We solve these problems with Algorithm 2 using (Undir) with branch-and-cut scheme 3 as we do in the previous section. Similarly, we allow parallel processing with up to 10 threads, but this time we set the time limit to six hours.

For this section, we consider the problem size of 19 nodes. By using 4 parameter configurations for each layout from uniform-61-n20 to uniform-65-n20, we solve 20 instances in total. Specifically, we fix v'/v as 2 and c_o/c'_o as 4, where v' and c'_o is set to 1. We consider 25^{th} and 50^{th} percentiles for α, and 0.5 and 0.75 for c_w/c_o.

Table 6.10 demonstrates the solution times (in seconds) of the instances. If an instance cannot be solved to optimality within the time limit, we report the final gap, i.e., the percentage gap between the objective function value of the best integer solution and the best lower bound, in parenthesis. While we are able to solve all the instances with $\alpha = 25^{th}$ to optimality within the time limit, for the instances with $\alpha = 50^{th}$ we observe varying final gaps. Similar to our observation in sensitivity analyses, these results show that our problem becomes harder to solve when α increases, as there are more possible drone sorties with greater α values.

Table 6.10: Experiments with $n = 19$

		Locations				
α	c_w	N61	N62	N63	N64	N65
25^{th}	2.0	4053	8466*	1040	587	466
	3.0	1892	7699	1375	1034	1101
50^{th}	2.0	(2.97)	(0.89)	(3.11)	15451	17137
	3.0	(6.84)	(6.74)	(3.09)	20290	(3.94)

Chapter 7

Conclusion

In this study, we introduce the minimum cost traveling salesman problem with multiple drones, where a single truck and multiple drones make deliveries in a synchronized manner. While the truck partly alleviates the drones' endurance limitation by transporting as well as launching/retrieving them, the drones allow faster deliveries at lower costs. Different from the majority of the studies in the literature, we allow launch and retrieval locations of a drone to be the same and do not restrict the number of drones used in a tour.

We present flow based and cut based MILP formulations with objective functions taking into account both the vehicles' operating and waiting costs. Alongside the branch-and-cut algorithms to solve the problem with the cut based formulation, we present an alternative solution algorithm utilizing another cut based MILP formulation with undirected variables. In order to find the delivery plan with minimum cost, we propose seven different solution approaches using three formulations that we provided.

In our computational study, we compare the proposed solution approaches and conduct sensitivity analyses on the problem parameters. Computational experiments demonstrate the superiority of the branch-and-cut solution approach over the flow based formulation. Moreover, the alternative solution methodology utilizing the cut based formulation with undirected variables is shown to outperform the formulations with directed variables.

Future extensions of this study could consider using multiple trucks with the same objective function and under the same assumptions. In the literature, there are several studies considering multiple truck settings, however their objectives and/or assumptions, as well as solution methodologies differ from ours. In fact, our formulations can easily be adapted to a multiple truck version of our problem with slight modifications on some constraints, e.g., constraints (4.1) and (5.12), which control the number of trucks in the system. Additionally, limiting the number of drones in a truck can be another future extension. Lastly, due to the fact that our problem is NP-hard, and thus the problem size that could be solved to optimality is limited, heuristic solution approaches can be developed to solve larger instances.

Bibliography

[1] C. Rose, "Amazon's Jeff Bezos looks to the future." https://www.cbsnews.com/news/amazons-jeff-bezos-looks-to-the-future/, 2013. Accessed: 2020-06-06.

[2] B. Popper, "Ups researching delivery drones that could compete with amazon's prime air." https://www.theverge.com/2013/12/3/5169878/ups-is-researching-its-own-delivery-drones-to-compete-with-amazons, 2013. Accessed: 2020-06-06.

[3] "Ups flight forward attains faa's first full approval for drone airline." https://pressroom.ups.com/pressroom/ContentDetailsViewer.page?ConceptType=PressReleases\&id=1569933965476-404, 2019. Accessed: 2020-06-06.

[4] A. C. Madrigal, "Inside google's secret drone-delivery program." https://www.theatlantic.com/technology/archive/2014/08/inside-googles-secret-drone-delivery-program/379306/, 2014. Accessed: 2020-06-06.

[5] A. Hern, "Dhl launches first commercial drone 'parcelcopter' delivery service." https://www.theguardian.com/technology/2014/sep/25/german-dhl-launches-first-commercial-drone-delivery-service, 2014. Accessed: 2020-06-06.

[6] M. Wohlsen, "The next big thing you missed: Amazon's delivery drones could work—they just need trucks." https://www.wired.com/2014/06/

the-next-big-thing-you-missed-delivery-drones-launched-from-trucks-are-the-future-of-shipping/, 2014. Accessed: 2020-06-06.

[7] "Unmanned aerial vehicle in logistics." https://www.dhl.com/content/dam/downloads/g0/about_us/logistics_insights/DHL_TrendReport_UAV.pdf, 2014. Accessed: 2020-06-06.

[8] S. Burns, "Drone meets delivery truck." https://www.ups.com/us/en/services/knowledge-center/article.page?kid=cd18bdc2, 2017. Accessed: 2020-06-06.

[9] A. H. Michel, "Amazon's drone patents." https://dronecenter.bard.edu/files/2017/09/CSD-Amazons-Drone-Patents-1.pdf, 2017. Accessed: 2020-06-06.

[10] C. C. Murray and A. G. Chu, "The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery," *Transportation Research Part C: Emerging Technologies*, vol. 54, pp. 86–109, 2015.

[11] Q. M. Ha, Y. Deville, Q. D. Pham, and M. H. Hà, "On the min-cost traveling salesman problem with drone," *Transportation Research Part C: Emerging Technologies*, vol. 86, pp. 597–621, 2018.

[12] P. A. Tu, N. T. Dat, and P. Q. Dung, "Traveling salesman problem with multiple drones," in *Proceedings of the Ninth International Symposium on Information and Communication Technology*, pp. 46–53, 2018.

[13] C. C. Murray and R. Raj, "The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones," *Transportation Research Part C: Emerging Technologies*, vol. 110, pp. 368–398, 2020.

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

[15] P. Kitjacharoenchai, M. Ventresca, M. Moshref-Javadi, S. Lee, J. M. Tanchoco, and P. A. Brunese, "Multiple traveling salesman problem with drones:

Mathematical model and heuristic approach," *Computers & Industrial Engineering*, vol. 129, pp. 14–30, 2019.

[16] A. Otto, N. Agatz, J. Campbell, B. Golden, and E. Pesch, "Optimization approaches for civil applications of unmanned aerial vehicles (uavs) or aerial drones: A survey," *Networks*, vol. 72, no. 4, pp. 411–458, 2018.

[17] S. H. Chung, B. Sah, and J. Lee, "Optimization for drone and drone-truck combined operations: A review of the state of the art and future directions," *Computers & Operations Research*, p. 105004, 2020.

[18] N. Mathew, S. L. Smith, and S. L. Waslander, "Planning paths for package delivery in heterogeneous multirobot teams," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 4, pp. 1298–1308, 2015.

[19] M. S. bin Othman, A. Shurbevski, Y. Karuno, and H. Nagamochi, "Routing of carrier-vehicle systems with dedicated last-stretch delivery vehicle and fixed carrier route," *Journal of Information Processing*, vol. 25, pp. 655–666, 2017.

[20] N. Boysen, D. Briskorn, S. Fedtke, and S. Schwerdfeger, "Drone delivery from trucks: Drone scheduling for given truck routes," *Networks*, vol. 72, no. 4, pp. 506–527, 2018.

[21] S. M. Ferrandez, T. Harbison, T. Weber, R. Sturges, and R. Rich, "Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm," *Journal of Industrial Engineering and Management (JIEM)*, vol. 9, no. 2, pp. 374–388, 2016.

[22] Y. S. Chang and H. J. Lee, "Optimal delivery routing with wider drone-delivery areas along a shorter truck-route," *Expert Systems with Applications*, vol. 104, pp. 307–317, 2018.

[23] K. Peng, J. Du, F. Lu, Q. Sun, Y. Dong, P. Zhou, and M. Hu, "A hybrid genetic algorithm on routing and scheduling for vehicle-assisted multi-drone parcel delivery," *IEEE Access*, vol. 7, pp. 49191–49200, 2019.

[24] A. Karak and K. Abdelghany, "The hybrid vehicle-drone routing problem for pick-up and delivery services," *Transportation Research Part C: Emerging Technologies*, vol. 102, pp. 427–449, 2019.

[25] S. Poikonen and B. Golden, "Multi-visit drone routing problem," *Computers & Operations Research*, vol. 113, p. 104802, 2020.

[26] J. G. Carlsson and S. Song, "Coordinated logistics with a truck and a drone," *Management Science*, vol. 64, no. 9, pp. 4052–4069, 2018.

[27] R. G. Mbiadou Saleu, L. Deroussi, D. Feillet, N. Grangeon, and A. Quilliot, "An iterative two-step heuristic for the parallel drone scheduling traveling salesman problem," *Networks*, vol. 72, no. 4, pp. 459–474, 2018.

[28] M. Dell'Amico, R. Montemanni, and S. Novellani, "Matheuristic algorithms for the parallel drone scheduling traveling salesman problem," *Annals of Operations Research*, pp. 1–16, 2020.

[29] A. M. Ham, "Integrated scheduling of m-truck, m-drone, and m-depot constrained by time-window, drop-pickup, and m-visit using constraint programming," *Transportation Research Part C: Emerging Technologies*, vol. 91, pp. 1–14, 2018.

[30] S. Kim and I. Moon, "Traveling salesman problem with a drone station," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 1, pp. 42–52, 2018.

[31] N. Agatz, P. Bouman, and M. Schmidt, "Optimization approaches for the traveling salesman problem with drone," *Transportation Science*, vol. 52, no. 4, pp. 965–981, 2018.

[32] E. E. Yurek and H. C. Ozmutlu, "A decomposition-based iterative optimization algorithm for traveling salesman problem with drone," *Transportation Research Part C: Emerging Technologies*, vol. 91, pp. 249–262, 2018.

[33] A. Ponza, "Optimization of drone-assisted parcel delivery," Master's thesis, Università Degli Studi Di Podova, 2016.

[34] M. Dell'Amico, R. Montemanni, and S. Novellani, "Drone-assisted deliveries: new formulations for the flying sidekick traveling salesman problem," *Optimization Letters*, pp. 1–32, 2019.

[35] Q. M. Ha, Y. Deville, Q. D. Pham, and M. H. Hà, "A hybrid genetic algorithm for the traveling salesman problem with drone," *Journal of Heuristics*, vol. 26, no. 2, pp. 219–247, 2020.

[36] J. C. de Freitas and P. H. V. Penna, "A variable neighborhood search for flying sidekick traveling salesman problem," *International Transactions in Operational Research*, vol. 27, no. 1, pp. 267–290, 2020.

[37] S. Poikonen, B. Golden, and E. A. Wasil, "A branch-and-bound approach to the traveling salesman problem with a drone," *Informs Journal on Computing*, vol. 31, no. 2, pp. 335–346, 2019.

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

[39] H. Y. Jeong, B. D. Song, and S. Lee, "Truck-drone hybrid delivery routing: Payload-energy dependency and no-fly zones," *International Journal of Production Economics*, vol. 214, pp. 220–233, 2019.

[40] P. Bouman, N. Agatz, and M. Schmidt, "Dynamic programming approaches for the traveling salesman problem with drone," *Networks*, vol. 72, no. 4, pp. 528–542, 2018.

[41] Z. Tang, W.-J. van Hoeve, and P. Shaw, "A study on the traveling salesman problem with a drone," in *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pp. 557–564, Springer, 2019.

[42] R. Roberti and M. Ruthmair, "Exact methods for the traveling salesman problem with drone." submitted, 2019.

[43] J. J. Yoon, *The traveling salesman problem with multiple drones: an optimization model for last-mile delivery.* PhD thesis, Massachusetts Institute of Technology, 2018.

[44] M. Moshref-Javadi, A. Hemmati, and M. Winkenbach, "A truck and drones model for last-mile delivery: A mathematical model and heuristic approach," *Applied Mathematical Modelling*, vol. 80, pp. 290–318, 2020.

[45] M. Salama and S. Srinivas, "Joint optimization of customer location clustering and drone-based routing for last-mile deliveries," *Transportation Research Part C: Emerging Technologies*, vol. 114, pp. 620–642, 2020.

[46] J. F. Campbell, D. Sweeney, and J. Zhang, "Strategic design for delivery with trucks and drones," *Supply Chain Analytics Report SCMA*, 2017.

[47] X. Wang, S. Poikonen, and B. Golden, "The vehicle routing problem with drones: Several worst-case results," *Optimization Letters*, vol. 11, no. 4, pp. 679–697, 2017.

[48] S. Poikonen, X. Wang, and B. Golden, "The vehicle routing problem with drones: Extended models and connections," *Networks*, vol. 70, no. 1, pp. 34–43, 2017.

[49] D. Wang, P. Hu, J. Du, P. Zhou, T. Deng, and M. Hu, "Routing and scheduling for hybrid truck-drone collaborative parcel delivery with independent and truck-carried drones," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10483–10495, 2019.

[50] L. D. P. Pugliese and F. Guerriero, "Last-mile deliveries by using drones and classical vehicles," in *International Conference on Optimization and Decision Science*, pp. 557–565, Springer, 2017.

[51] D. Schermer, M. Moeini, and O. Wendt, "A matheuristic for the vehicle routing problem with drones and its variants," *Transportation Research Part C: Emerging Technologies*, vol. 106, pp. 166–204, 2019.

[52] D. Schermer, M. Moeini, and O. Wendt, “A hybrid vns/tabu search algorithm for solving the vehicle routing problem with drones and en route operations,” *Computers & Operations Research*, vol. 109, pp. 134–158, 2019.

[53] P. Kitjacharoenchai, B.-C. Min, and S. Lee, “Two echelon vehicle routing problem with drones in last mile delivery,” *International Journal of Production Economics*, p. 107598, 2019.

[54] R. Daknama and E. Kraus, “Vehicle routing with drones,” *arXiv preprint arXiv:1705.06431*, 2017.

[55] Z. Wang and J.-B. Sheu, “Vehicle routing problem with drones,” *Transportation research part B: methodological*, vol. 122, pp. 350–364, 2019.

[56] I. Bakir and G. Ö. Tiniç, “Optimizing drone-assisted last-mile deliveries: The vehicle routing problem with flexible drones,” 2020.

[57] I. Kara, “Arc based integer programming formulations for the distance constrained vehicle routing problem,” in *3rd IEEE International Symposium on Logistics and Industrial informatics*, pp. 33–38, IEEE, 2011.

[58] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The traveling salesman problem: a computational study*. Princeton university press, 2006.

[59] G. Ozbaygin, H. Yaman, and O. E. Karasan, “Time constrained maximal covering salesman problem with weighted demands and partial coverage,” *Computers & Operations Research*, vol. 76, pp. 226–237, 2016.

[60] D. Murray and S. Glidewell, “An analysis of the operational costs of trucking: 2019 update,” 2019.

[61] J. Edmonds and R. M. Karp, “Theoretical improvements in algorithmic efficiency for network flow problems,” *Journal of the ACM (JACM)*, vol. 19, no. 2, pp. 248–264, 1972.

[62] M. Stoer and F. Wagner, “A simple min-cut algorithm,” *Journal of the ACM (JACM)*, vol. 44, no. 4, pp. 585–591, 1997.

Appendix A

Computational Results

Table A.1: Sensitivity analyses on uniform-61-n20 with $n = 14$

α	$1/v$	c_o	c_w	Obj.	C_o	C_w	C'_o	C'_w	#dv	#dn	Time
25^{th}	1.5	2.0	1.0	832	79.0%	2.4%	18.6%	0.0%	3	2	9
			1.5	842	78.0%	3.6%	18.4%	0.0%	3	2	12
		4.0	2.0	1492	83.2%	2.7%	13.7%	0.4%	4	2	21
			3.0	1512	82.1%	4.0%	13.5%	0.4%	4	2	20
		10.0	5.0	3394	87.9%	3.6%	8.2%	0.3%	6	2	28
			7.5	3450	87.8%	4.3%	7.5%	0.3%	5	2	90*
	2.0	2.0	1.0	1077	81.3%	3.8%	14.9%	0.0%	3	2	6
			1.5	1082	90.2%	0.7%	9.1%	0.0%	2	1	8
		4.0	2.0	1986	85.0%	4.1%	10.8%	0.1%	5	3	15
			3.0	2027	83.3%	6.1%	10.6%	0.0%	5	3	17
		10.0	5.0	4629	86.0%	8.5%	5.5%	0.0%	6	3	19
			7.5	4743	89.0%	6.5%	4.5%	0.0%	5	3	20
50^{th}	1.5	2.0	1.0	832	73.9%	4.2%	21.9%	0.0%	4	3	242
			1.5	839	80.0%	1.2%	18.5%	0.4%	3	2	235
		4.0	2.0	1357	68.5%	6.9%	24.5%	0.1%	6	4	226
			3.0	1404	66.2%	9.9%	23.7%	0.1%	6	4	472
		10.0	5.0	2884	78.5%	8.8%	12.6%	0.0%	7	4	113
			7.5	3008	77.3%	11.6%	11.1%	0.0%	6	4	447
	2.0	2.0	1.0	986	61.3%	6.4%	32.2%	0.2%	6	3	197*
			1.5	1017	59.4%	9.3%	31.2%	0.2%	6	3	395*
		4.0	2.0	1618	68.7%	9.4%	21.6%	0.2%	7	3	37
			3.0	1694	65.6%	13.5%	20.7%	0.2%	7	3	83
		10.0	5.0	3514	79.1%	10.8%	10.0%	0.1%	7	3	85
			7.5	3698	75.2%	14.8%	10.0%	0.0%	7	3	82
75^{th}	1.5	2.0	1.0	832	73.9%	4.2%	21.9%	0.0%	4	3	731
			1.5	839	80.0%	1.2%	18.5%	0.4%	3	2	1837
		4.0	2.0	1254	47.8%	9.4%	42.6%	0.2%	9	8	1434
			3.0	1306	51.9%	7.7%	40.1%	0.3%	8	6	8877
		10.0	5.0	2278	57.9%	14.6%	27.3%	0.1%	10	8	152
			7.5	2444	54.0%	20.4%	25.5%	0.1%	10	8	528
	2.0	2.0	1.0	981	56.7%	4.9%	38.0%	0.4%	7	4	1540
			1.5	1000	55.6%	4.5%	39.8%	0.1%	7	4	3798
		4.0	2.0	1427	48.8%	8.7%	42.5%	0.1%	10	9	215
			3.0	1469	54.5%	7.4%	38.1%	0.1%	9	8	477
		10.0	5.0	2657	65.5%	11.7%	22.8%	0.0%	10	9	75
			7.5	2798	62.2%	15.0%	22.1%	0.7%	10	7	170

Table A.2: Sensitivity analyses on uniform-62-n20 with $n = 14$

α	$1/v$	c_o	c_w	Obj.	C_o	C_w	C'_o	C'_w	#dv	#dn	Time
25^{th}	1.5	2.0	1.0	885	88.5%	0.0%	10.1%	1.5%	2	1	39
25^{th}	1.5	2.0	1.5	885	88.5%	0.0%	10.1%	1.5%	2	1	33
25^{th}	1.5	4.0	2.0	1511	75.0%	1.9%	22.4%	0.7%	6	2	81
25^{th}	1.5	4.0	3.0	1523	74.4%	2.1%	22.4%	1.1%	6	2	72
25^{th}	1.5	10.0	5.0	3213	84.5%	3.0%	12.0%	0.5%	7	3	45
25^{th}	1.5	10.0	7.5	3260	83.3%	4.4%	11.8%	0.5%	7	3	51
25^{th}	2.0	2.0	1.0	1108	63.1%	9.1%	27.8%	0.0%	6	2	31
25^{th}	2.0	2.0	1.5	1144	86.4%	0.0%	12.8%	0.9%	3	1	55
25^{th}	2.0	4.0	2.0	1856	67.2%	8.6%	24.0%	0.2%	8	3	20
25^{th}	2.0	4.0	3.0	1895	73.5%	0.8%	23.7%	2.1%	8	4	106*
25^{th}	2.0	10.0	5.0	3968	78.6%	10.1%	11.2%	0.1%	8	3	19
25^{th}	2.0	10.0	7.5	4005	86.9%	0.9%	11.2%	1.0%	8	4	57*
50^{th}	1.5	2.0	1.0	885	88.5%	0.0%	10.1%	1.5%	2	1	59
50^{th}	1.5	2.0	1.5	885	88.5%	0.0%	10.1%	1.5%	2	1	65
50^{th}	1.5	4.0	2.0	1475	44.7%	15.2%	40.1%	0.0%	10	7	216
50^{th}	1.5	4.0	3.0	1522	64.3%	8.3%	26.1%	1.3%	7	4	540
50^{th}	1.5	10.0	5.0	2703	54.9%	16.6%	28.1%	0.4%	11	10	37
50^{th}	1.5	10.0	7.5	2928	50.7%	23.0%	25.9%	0.3%	11	10	71
50^{th}	2.0	2.0	1.0	1087	64.4%	4.5%	31.1%	0.0%	6	4	52
50^{th}	2.0	2.0	1.5	1106	69.4%	4.1%	26.3%	0.2%	5	4	121
50^{th}	2.0	4.0	2.0	1685	52.7%	10.9%	36.2%	0.2%	10	7	30
50^{th}	2.0	4.0	3.0	1777	50.0%	15.5%	34.3%	0.2%	10	7	95
50^{th}	2.0	10.0	5.0	3214	61.6%	14.0%	24.4%	0.0%	11	10	30
50^{th}	2.0	10.0	7.5	3439	57.6%	19.6%	22.8%	0.0%	11	10	31
75^{th}	1.5	2.0	1.0	885	88.5%	0.0%	10.1%	1.5%	2	1	61
75^{th}	1.5	2.0	1.5	885	88.5%	0.0%	10.1%	1.5%	2	1	67
75^{th}	1.5	4.0	2.0	1459	58.0%	6.2%	35.2%	0.6%	8	5	1107
75^{th}	1.5	4.0	3.0	1502	56.3%	8.4%	34.4%	0.9%	8	6	2301
75^{th}	1.5	10.0	5.0	2596	64.1%	6.8%	28.6%	0.5%	10	10	128
75^{th}	1.5	10.0	7.5	2685	62.0%	9.9%	27.6%	0.4%	10	10	140
75^{th}	2.0	2.0	1.0	1087	64.4%	4.5%	31.1%	0.0%	6	4	156
75^{th}	2.0	2.0	1.5	1106	69.4%	4.1%	26.3%	0.2%	5	4	313
75^{th}	2.0	4.0	2.0	1679	52.9%	6.2%	40.7%	0.2%	10	7	194
75^{th}	2.0	4.0	3.0	1703	52.1%	1.4%	44.9%	1.6%	10	9	285
75^{th}	2.0	10.0	5.0	3051	72.8%	1.3%	25.0%	0.9%	10	9	109
75^{th}	2.0	10.0	7.5	3071	72.3%	2.0%	24.9%	0.9%	10	9	174

Table A.3: Sensitivity analyses on uniform-63-n20 with $n = 14$

α	$1/v$	c_o	c_w	Obj.	C_o	C_w	C'_o	C'_w	#dv	#dn	Time
25^{th}	1.5	2.0	1.0	868	80.8%	3.1%	16.1%	0.0%	3	2	5
			1.5	881	79.6%	4.5%	15.9%	0.0%	3	2	9
		4.0	2.0	1592	88.2%	1.8%	9.5%	0.5%	3	2	31
			3.0	1607	87.4%	2.7%	9.5%	0.5%	3	2	31
		10.0	5.0	3613	76.8%	14.0%	9.1%	0.1%	7	4	150*
			7.5	3706	92.3%	0.0%	7.1%	0.7%	4	1	76
	2.0	2.0	1.0	1082	86.5%	0.6%	12.9%	0.0%	3	2	5
			1.5	1085	86.3%	0.8%	12.9%	0.0%	3	2	8
		4.0	2.0	2000	75.6%	10.4%	14.0%	0.0%	6	3	25
			3.0	2028	88.4%	0.9%	9.3%	1.4%	4	2	23
		10.0	5.0	4514	82.0%	10.6%	7.4%	0.0%	7	4	53*
			7.5	4721	80.1%	12.4%	7.1%	0.5%	6	3	16
50^{th}	1.5	2.0	1.0	868	80.8%	3.1%	16.1%	0.0%	3	2	13
			1.5	881	79.6%	4.5%	15.9%	0.0%	3	2	17
		4.0	2.0	1546	76.8%	3.4%	19.5%	0.2%	5	3	157
			3.0	1571	75.6%	2.8%	20.9%	0.7%	5	2	383
		10.0	5.0	3183	66.0%	15.4%	18.6%	0.0%	8	8	108
			7.5	3332	85.1%	0.3%	13.8%	0.8%	6	3	221
	2.0	2.0	1.0	1082	86.5%	0.6%	12.9%	0.0%	3	2	12
			1.5	1085	86.3%	0.8%	12.9%	0.0%	3	2	17
		4.0	2.0	1878	59.6%	10.4%	29.8%	0.2%	8	6	68
			3.0	1976	56.7%	14.9%	28.3%	0.2%	8	6	114
		10.0	5.0	3852	72.7%	12.7%	14.5%	0.1%	8	6	29
			7.5	4096	68.3%	17.9%	13.6%	0.1%	8	6	39
75^{th}	1.5	2.0	1.0	868	80.8%	3.1%	16.1%	0.0%	3	2	12
			1.5	881	79.6%	4.5%	15.9%	0.0%	3	2	20
		4.0	2.0	1546	76.8%	3.4%	19.5%	0.2%	5	3	509
			3.0	1571	75.6%	2.8%	20.9%	0.7%	5	2	1574
		10.0	5.0	2726	49.5%	20.9%	29.6%	0.0%	10	10	49
			7.5	3011	44.8%	28.4%	26.8%	0.0%	10	10	120
	2.0	2.0	1.0	1082	86.5%	0.6%	12.9%	0.0%	3	2	22
			1.5	1085	86.3%	0.8%	12.9%	0.0%	3	2	35
		4.0	2.0	1755	41.0%	13.0%	46.0%	0.0%	10	10	59
			3.0	1869	38.5%	18.3%	43.2%	0.0%	10	10	264
		10.0	5.0	3177	56.7%	17.9%	25.4%	0.0%	10	10	34
			7.5	3462	52.0%	24.7%	23.3%	0.0%	10	10	76

Table A.4: Sensitivity analyses on uniform-64-n20 with $n = 14$

α	$1/v$	c_o	c_w	Obj.	C_o	C_w	C'_o	C'_w	#dv	#dn	Time
25^{th}	1.5	2.0	1.0	1028	90.5%	3.5%	6.0%	0.0%	2	2	44
			1.5	1029	100.0%	0.0%	0.0%	0.0%	0	0	40
		4.0	2.0	1919	82.2%	6.4%	10.9%	0.5%	5	2	73
			3.0	1945	91.0%	0.0%	8.2%	0.8%	3	2	50
		10.0	5.0	4428	88.1%	5.9%	5.7%	0.3%	6	2	43
			7.5	4540	87.2%	7.2%	5.2%	0.4%	5	2	43
	2.0	2.0	1.0	1308	80.4%	6.9%	12.7%	0.0%	5	2	13
			1.5	1334	92.1%	0.9%	7.0%	0.0%	2	1	40
		4.0	2.0	2425	85.8%	6.2%	8.0%	0.0%	6	2	17
			3.0	2481	84.2%	5.1%	10.8%	0.0%	6	4	83
		10.0	5.0	5679	90.9%	4.2%	4.9%	0.0%	7	5	13
			7.5	5799	89.0%	6.2%	4.8%	0.0%	7	5	28
50^{th}	1.5	2.0	1.0	997	86.4%	4.1%	9.5%	0.0%	1	1	431
			1.5	1017	84.6%	6.0%	9.3%	0.0%	1	1	867
		4.0	2.0	1764	71.4%	10.7%	17.6%	0.3%	6	3	315
			3.0	1836	78.4%	6.9%	14.2%	0.5%	4	2	418
		10.0	5.0	3846	77.6%	12.3%	9.8%	0.3%	7	3	64
			7.5	4073	73.3%	15.6%	10.4%	0.8%	7	3	108
	2.0	2.0	1.0	1228	65.8%	11.0%	23.2%	0.0%	6	3	168
			1.5	1262	85.9%	3.0%	11.2%	0.0%	2	1	162
		4.0	2.0	2167	73.5%	13.0%	13.5%	0.0%	7	3	141
			3.0	2282	70.8%	12.2%	15.1%	1.9%	6	3	173
		10.0	5.0	4870	81.7%	10.2%	7.2%	0.9%	7	3	34
			7.5	5097	78.1%	13.4%	7.7%	0.8%	7	4	77
75^{th}	1.5	2.0	1.0	962	81.1%	3.7%	15.2%	0.0%	2	2	336
			1.5	976	88.2%	0.0%	10.0%	1.8%	1	1	716
		4.0	2.0	1528	49.9%	10.3%	39.9%	0.0%	7	6	645
			3.0	1606	47.4%	14.7%	37.9%	0.0%	7	6	1126
		10.0	5.0	2842	60.7%	15.4%	23.9%	0.0%	10	8	153
			7.5	3061	56.3%	21.4%	22.2%	0.0%	10	8	295
	2.0	2.0	1.0	1183	42.9%	5.6%	51.5%	0.0%	7	6	325
			1.5	1212	80.5%	0.5%	17.7%	1.3%	3	1	1134
		4.0	2.0	1754	52.5%	8.8%	38.8%	0.0%	10	8	213
			3.0	1822	55.8%	8.7%	35.5%	0.0%	9	7	217
		10.0	5.0	3365	68.4%	11.4%	20.2%	0.0%	10	8	46
			7.5	3557	64.7%	16.2%	19.1%	0.0%	10	8	77

Table A.5: Sensitivity analyses on uniform-65-n20 with $n = 14$

α	$1/v$	c_o	c_w	Obj.	C_o	C_w	C'_o	C'_w	#dv	#dn	Time
25^{th}	1.5	2.0	1.0	803	72.1%	6.7%	21.3%	0.0%	4	2	41
			1.5	822	86.5%	2.1%	11.4%	0.0%	2	1	81
		4.0	2.0	1358	66.7%	7.7%	25.0%	0.6%	7	3	64
			3.0	1410	64.2%	11.2%	24.0%	0.6%	7	3	98
		10.0	5.0	2796	73.5%	11.4%	15.2%	0.0%	9	5	34
			7.5	2952	70.6%	15.4%	14.0%	0.0%	8	5	74
	2.0	2.0	1.0	975	71.7%	4.9%	23.4%	0.0%	6	3	22
			1.5	997	77.4%	5.4%	17.2%	0.0%	4	2	27
		4.0	2.0	1696	73.1%	5.5%	19.8%	1.5%	8	4	30
			3.0	1743	71.1%	8.1%	19.3%	1.5%	8	4	30
		10.0	5.0	3697	83.9%	6.4%	9.1%	0.7%	8	4	18
			7.5	3814	81.3%	9.2%	8.8%	0.7%	8	4	24
50^{th}	1.5	2.0	1.0	790	73.2%	2.8%	23.9%	0.1%	4	3	68
			1.5	801	72.2%	4.1%	23.6%	0.1%	4	3	106
		4.0	2.0	1302	69.6%	2.9%	27.2%	0.3%	7	5	156
			3.0	1321	68.6%	4.3%	26.8%	0.3%	7	5	305
		10.0	5.0	2453	65.4%	10.8%	23.6%	0.1%	11	8	65
			7.5	2585	62.1%	15.4%	22.4%	0.1%	11	8	81
	2.0	2.0	1.0	969	72.2%	3.9%	23.6%	0.2%	6	3	105
			1.5	976	73.4%	1.8%	24.2%	0.6%	5	3	82
		4.0	2.0	1564	55.8%	5.8%	35.7%	2.7%	10	7	68
			3.0	1609	54.2%	8.4%	34.7%	2.7%	10	7	131
		10.0	5.0	3007	72.5%	7.5%	18.6%	1.4%	10	7	23
			7.5	3119	69.9%	10.8%	17.9%	1.4%	10	7	51
75^{th}	1.5	2.0	1.0	790	73.2%	2.8%	23.9%	0.1%	4	3	94
			1.5	801	72.2%	4.1%	23.6%	0.1%	4	3	296
		4.0	2.0	1302	69.6%	2.9%	27.2%	0.3%	7	5	1107
			3.0	1321	68.6%	4.3%	26.8%	0.3%	7	5	1755
		10.0	5.0	2388	53.4%	17.3%	29.3%	0.0%	11	9	182
			7.5	2585	62.1%	15.4%	22.4%	0.1%	11	8	306
	2.0	2.0	1.0	969	72.2%	3.9%	23.6%	0.2%	6	3	150
			1.5	976	73.4%	1.8%	24.2%	0.6%	5	3	176
		4.0	2.0	1564	55.8%	5.8%	35.7%	2.7%	10	7	268
			3.0	1609	54.2%	8.4%	34.7%	2.7%	10	7	394
		10.0	5.0	2828	60.1%	14.1%	25.1%	0.6%	11	10	21
			7.5	3028	56.1%	19.8%	23.4%	0.6%	11	10	74