Analytics-Based Optimization for the Integration of Drones into Last-Mile Logistics

Amro M. El-Adle

University of Massachusetts Amherst

Follow this and additional works at: https://scholarworks.umass.edu/dissertations_2

Part of the Business Administration, Management, and Operations Commons, Business Analytics Commons, and the Operations and Supply Chain Management Commons

Recommended Citation
https://doi.org/10.7275/24602740 https://scholarworks.umass.edu/dissertations_2/2309

This Open Access Dissertation is brought to you for free and open access by the Dissertations and Theses at ScholarWorks@UMass Amherst. It has been accepted for inclusion in Doctoral Dissertations by an authorized administrator of ScholarWorks@UMass Amherst. For more information, please contact scholarworks@library.umass.edu.
Analytics-Based Optimization for the Integration of Drones into Last-Mile Logistics

Amro M. El-Adle

Follow this and additional works at: https://scholarworks.umass.edu/dissertations_2

Part of the Business Administration, Management, and Operations Commons, Business Analytics Commons, and the Operations and Supply Chain Management Commons
ANALYTICS-BASED OPTIMIZATION FOR
THE INTEGRATION OF DRONES
INTO LAST-MILE LOGISTICS

A Dissertation Presented
by
AMRO MOHAMED EL-ADLE

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2021

Isenberg School of Management
ANALYTICS-BASED OPTIMIZATION FOR THE INTEGRATION OF DRONES INTO LAST-MILE LOGISTICS

A Dissertation Presented

by

AMRO MOHAMED EL-ADLE

Approved as to style and content by:

________________________________________
Ahmed Ghoniem, Chair

________________________________________
Senay Solak, Member

________________________________________
Hari Balasubramanian, Member

________________________________________
Anna Nagurney, PhD Coordinator
Isenberg School of Management
DEDICATION

In the name of the All-Merciful, the All-Beneficent, the One who made me, and who
gifted me opportunities, curiosity, and love without measure. To Mohamed and
Amal, the hardest-working people I know. And to Amira, Hany, and Kaream, with
whom I share the best of who I am.
ACKNOWLEDGMENTS

“Give everything the time it deserves.” This was some of the advice Professor Ghoniem shared with me at the start of my journey at Isenberg, and it’s the same advice of which we have so often availed during my time here. When everyone rushes about to do everything, day by day Professor Ghoniem spent time with me before and after class, on walks to the parking lot, and in almost every corridor of Isenberg to slow things down and to teach me to appreciate the details of our work. This was all the more true when inevitably life did not unfold as I had planned. And in those trying moments, Professor Ghoniem taught me to balance trade-offs far more subtle than even mathematics can capture. For his kindness, his enduring patience and forbearance, and for his brilliance as more than my advisor and mentor but also as my friend, Professor Ghoniem deserves far more praise and gratitude than the space and time I have devoted here. Learning is a most special type of light, and I feel honored and privileged to reflect Professor Ghoniem’s radiance.

I’m most grateful to Professors Solak and Balasubramanian, who graciously served as members of my dissertation committee. Over many lectures and many more conversations that bled from class into office hours and beyond, Senay and Hari taught me the value of patience, of asking the right questions, and of perseverance. To their credit, the questions they posed during my comprehensive exams were later asked of me as interview questions during my job search. For all of my questions they have answered, all the thoughtful questions they have asked of me, and for their care and attention to my work and development, I am thankful.

I owe tremendous thanks to Professor Mohamed Haouari, who has been a pleasant, patient, and most encouraging co-author and companion on this journey. I have
benefited tremendously not only from his vast knowledge and experience in our field, but also from his warm demeanor and humility. I learned to accept constructive criticism from referees, and to reply with only sweetness when squeezed thanks to si Mohamed. For this, and many more lifelong lessons, I feel especially fortunate to have benefited from his company.

I also want to make special mention of Professor Iqbal Agha, who warmly welcomed me to Isenberg and generously provided me opportunities to be a better student, teacher, and researcher. Academia can be a challenging environment, but thanks to Iqbal’s advice I found it easier to navigate. For being available throughout my journey, and particularly for his insights in key moments, I am most grateful to have spent time with Iqbal. And my special thanks are also due to Audrey Kieras, whose effervescence about my work and my teaching assignments helped uplift my spirits.

More broadly, Professors Milne, Nakosteen, and Nagurney have supported my doctoral pursuits and provided funding for many trips from which I benefited, and for which I am grateful. I am also deeply indebted to Mike Korza, and to Priscilla Mahar before him, both of whom helped me navigate confusing turns of my journey.

Long before graduate school, I benefited from the knowledge and the patience of tremendous teachers. They believed in me, went above and beyond their duties to grant me opportunities for success, and their kind words motivated me to be my best. To name just a few: David Cox; Christine Wasmuth; Jeremy Stanton; Stephen Selby; Virginia Finn; Thomas O’Shaughnessy; Sara Solberg; Fatema Musabeh; Donald Delo; Holly Smith; Sister Fatimah; Imam Farghal; Imam Wissam; and Tanya Leise, who first suggested the idea of joining a PhD program to me.

As a doctoral student, I sometimes worked at my home where I fed myself, and in the homes of so many people who graciously made time to prepare meals on my behalf. For all those people who tended to my needs, and who helped me in ways I may not have fully appreciated, I am most grateful. For their excellent company,
and their sincere desire to help me succeed, I would like to thank Auntie Naz, Tant Madiha, and Tant Jeehan. There are also friends who made to visit and keep me sane, including Stephanie (whose care packages have brought me untold relief), Sylvia, and Shenali. And I would also like to thank my lovely neighbors in Granby, who made my life immeasurably richer: Maria; Maureen; Charles and Kathy; Hendro and his family; Kenny and his family; and Sean and Amy; Rhonda and her family; and Jeremy and his family.

Far and away the coolest part of being a PhD is not what I’ve shared in these pages, but what I shared with the friends who enveloped me in their lives. In that special group I was fortunate to meet friends who became my eating partners, who listened to my late-night rants on mathematical modeling, and who helped make sure I prayed on time. Baboucarr, Hassaan, Nazmul, Samy, Haitham, and Abdullah all became my brothers, and made these few years some of the best of my life. Rammah deserves special mention as an amazing older brother, and an exceptionally calm person who kept me positive. There are also too many Ahamid for me to count: Gameel; Salah; and Aly el Din, each of whom took me under his wing. From the masjid, I was blessed to be in the company of Hassan, Eesah, Musa, Brahim, and Patrick. I shared laughs, occasionally tremendous food, and soccer matches in which I did not belong with Boimin, Noman, Hassan, Suhaib, Soleiman, Waleed, Gamal, and many more brothers. My success is the direct result of all of these people making the time and the effort to care for me in so many ways.

Finally, this list is far from being exhaustive. For anyone I may not have mentioned by name, I remain grateful and I hope that I can pass along to others all the goodness and knowledge from which I have been fortunate to receive.
The growing volume and consistency of online ordering has renewed interest in technology-driven supply chain operations, with a focus on logistics optimization. Recent trials have demonstrated the viability of using unmanned aerial vehicles (UAVs), known colloquially as drones, in last-mile deliveries. In minimizing delivery times for customers, managing the load of logistics workers, and reducing congestion and pollution per delivery, the integration of drones into traditional delivery networks presents transformative potential to benefit consumers, firms, and society at large. This dissertation investigates operational and tactical problems that inform routing and assignment decisions for vehicle-drone delivery systems. In three essays that follow, mathematical programming models and heuristic algorithms are developed for these problems, and their computational tractability enhanced via optimization constructs spanning preprocessing, valid inequalities, and reformulation techniques. Moreover, insights are provided into the network topographies that stand to benefit from the
integration of drones into last-mile deliveries. The impact of these optimization techniques is assessed using benchmark instances from the growing literature.

Chapter 2, based on the work of El-Adle et al. [2021], investigates the operational problem of assigning customers to be visited either by a delivery vehicle or by a portable companion drone launched from the vehicle, recently known in the literature as a Traveling Salesman Problem with Drone (TSP-D). A novel 0-1 mixed-integer program (MIP) is proposed for the TSP-D that synchronizes vehicle and drone operations with the objective of minimizing the total duration of the joint tour. Using a combination of valid inequalities, pre-processing, and other bound tightening strategies, the tractability of the proposed MIP formulation is enhanced to produce exact solutions to benchmark instances having up to 24 customers, the largest in the literature at the time of publication. Our work further enabled exact solutions for certain networks involving 32 customers, twice the size of instances solved in the extant literature at that point.

Since commercial parcel delivery typically involve up to 100 packages per route, Chapter 3 builds on the breakthrough in Chapter 2 and proposes an optimization-based variable neighborhood search heuristic. Starting with a relaxed problem structure, the heuristic uses two distinct MIP formulations to progressively integrate constraints relating to synchronous travel between the vehicle and drone as well as cyclic flights. By termination, the heuristic restores all problem assumptions to yield a high-quality feasible solution. On a set of benchmark instances from the literature, the heuristic improves upon the best-known results for 113/120 instances having up to 100 nodes, with comparable computational effort to approaches in the literature [Schermer et al., 2019]. We also propose several pre-processing techniques that simplify decision-making associated with drone cycles, and analytically investigate the conditions under which they are optimal.
Building upon the previous work, Chapter 4 examines the idea of simplifying operational last-mile delivery problems that involve vehicles and portable drones. By optimizing a multi-period last-mile delivery problem, it is possible to identify, at a more tactical level, customers that should be served by drone and those who should receive their packages by vehicle in a given season. This tactical assignment is informed by geospatial and demand data analytics and an optimization model that tackles the underlying operational vehicle routing and drone flights over the multi-period horizon under investigation. We propose a novel mixed-integer formulation to this challenging last-mile delivery problem with drone eligibility, which is also embedded in an optimization-based variable neighborhood search that effectively and consistently yields near-optimal solutions (within 0.5% optimal) for networks involving 200 customers in manageable computational times.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGMENTS</td>
</tr>
<tr>
<td>ABSTRACT</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
</tr>
</tbody>
</table>

## CHAPTER

1. INTRODUCTION AND PROBLEM BACKGROUND/MOTIVATION ................................. 1

   1.1 UAV Developments in Last-Mile Deliveries ................................. 4

       1.1.1 Drone-Vehicle Integration ................................. 5

           1.1.1.1 Flight Conditions/Capacity ......................... 6

           1.1.1.2 Civilian Attitude/Perception of Drone Delivery ........ 7

   1.2 Diversified Drone Delivery Applications and Related Modeling Assumptions 9

   1.3 Contributions ......................................................... 12

2. ENHANCED MIP FOR PARCEL DELIVERY BY VEHICLE AND DRONE ............................. 15

   2.1 Introduction ........................................................... 15

   2.2 Literature Review .................................................... 16

       2.2.1 Related Literature .............................................. 16

       2.2.2 Extant Operational Assumptions ............................... 20

   2.3 Optimization Model ................................................... 23

   2.4 Bound Improvement Strategies .......................................... 29
2.4.1 Lower Bound Improvements .............................................. 29
   2.4.1.1 Valid Inequalities .................................................. 29
   2.4.1.2 Scaled “Big M” Values .......................................... 31
   2.4.1.3 A Heuristic for Generating an Initial Feasible TSP-D
           Solution ................................................................. 33
2.4.2 Upper Bound Improvements ............................................. 34
2.4.3 Instance Pre-processing ............................................... 36
2.5 Computational Study ...................................................... 36
   2.5.1 Problem Instances .................................................... 37
   2.5.2 Summary of Results ................................................... 38
   2.5.3 Bound Improvement Assessment ...................................... 39
   2.5.4 Solving Instances of Size $N \geq 24$ ................................. 41
   2.5.5 Sensitivity Analysis of Maximum Drone Flight Duration .......... 44
2.6 Conclusion ................................................................. 45

3. AN OPTIMIZATION-BASED HEURISTIC FOR ASSESSING
   CYCLIC DRONE FLIGHTS ON JOINT VEHICLE-DRONE
   PARCEL DELIVERY .............................................................. 47
   3.1 Introduction .............................................................. 47
   3.2 Literature Review ........................................................ 49
      3.2.1 MIP-Based Heuristic Approaches .................................. 49
      3.2.2 Other Optimization-Based Approaches ................................ 51
   3.3 Problem Statement & MIP Formulations ................................. 52
      3.3.1 Restricted MIP Formulation ........................................ 53
         3.3.1.1 Formulation ..................................................... 54
      3.3.2 Conditions for Drone Cycles ....................................... 58
      3.3.3 Unrestricted MIP Formulation ...................................... 59
   3.4 Description of VNS Approach ............................................ 63
      3.4.1 DIP Procedure ....................................................... 64
      3.4.2 Phase 1: Rapid Improvement ....................................... 67
      3.4.3 Phase 2: Deeper Exploration ....................................... 67
      3.4.4 Diversification/Perturbation Schemes & Termination
         Criteria ................................................................. 68
3.5 Computational Study .................................................. 69
  3.5.1 Problem Instances ............................................... 69
  3.5.2 Performance on Benchmark Instances ...................... 70
  3.5.3 Sensitivity Analysis of Drone Cycles ....................... 73

3.6 Conclusion ............................................................. 75

4. LAST-MILE DELIVERY WITH DRONE ELIGIBILITY ......... 77
  4.1 Introduction ......................................................... 77
  4.2 Literature Review ................................................ 80
    4.2.1 Single-Period Variants ..................................... 80
    4.2.2 Periodic & Consistent Routing Problems ............... 81
    4.2.3 Tactical Planning ........................................... 82
  4.3 Problem Statement & MIP Formulation ......................... 83
    4.3.1 Problem Statement .......................................... 83
    4.3.2 MIP Formulation ............................................ 86
  4.4 Matheuristic for TSP-DE ......................................... 90
    4.4.1 Initialization ............................................... 91
    4.4.2 Period-Specific Demand Improvement .................... 95
    4.4.3 Geo-spatial Improvement ................................... 97
  4.5 Computational Study .............................................. 100
    4.5.1 Description of Problem Instances ....................... 100
    4.5.2 Base Results ............................................... 103
    4.5.3 Matheuristic Progression & Parameter Tuning .......... 107
  4.6 Conclusion .......................................................... 108

5. CONCLUSIONS AND FUTURE RESEARCH ......................... 111
  5.1 Managerial Insights .............................................. 113
  5.2 Future Research .................................................. 115

APPENDICES
  A. SUPPORTING MATERIAL FOR CHAPTER 2 ...................... 118
  B. SUPPORTING MATERIAL FOR CHAPTER 3 ...................... 134
  C. SUPPORTING MATERIAL FOR CHAPTER 4 ...................... 138
<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 A summary of key operational assumptions in the literature</td>
<td>21</td>
</tr>
<tr>
<td>2.2 Summary of Computational Results</td>
<td>38</td>
</tr>
<tr>
<td>2.3 Objective, Computational Effort Sensitivity to Flight Time for $N = 16$</td>
<td>44</td>
</tr>
<tr>
<td>3.1 Summary of DIP’s Performance on Benchmark Instances</td>
<td>71</td>
</tr>
<tr>
<td>3.2 Exact Solutions for $n = 12$ Using Model TSPD3</td>
<td>74</td>
</tr>
<tr>
<td>4.1 2019 Parcel Delivery Statistics</td>
<td>101</td>
</tr>
<tr>
<td>4.2 Dataset S</td>
<td>104</td>
</tr>
<tr>
<td>4.3 Dataset M</td>
<td>105</td>
</tr>
<tr>
<td>4.4 Dataset L</td>
<td>107</td>
</tr>
<tr>
<td>A.1 Iteration 0, Routing Variable Values</td>
<td>118</td>
</tr>
<tr>
<td>A.2 Iteration 0, Minimum Cut Generation</td>
<td>119</td>
</tr>
<tr>
<td>A.3 Iteration 1, Routing Variable Values</td>
<td>120</td>
</tr>
<tr>
<td>A.4 Computational Results for $N = 16, 20$, and 24 for SC instances</td>
<td>129</td>
</tr>
<tr>
<td>A.5 Computational Results for $N = 28$ and 32 for SC instances</td>
<td>130</td>
</tr>
<tr>
<td>A.6 Computational Results for $N = 16, 20$, and 24 for DC instances</td>
<td>130</td>
</tr>
<tr>
<td>A.7 Computational Results for $N = 28$ and 32 for DC instances</td>
<td>131</td>
</tr>
<tr>
<td>A.8 Computational Results for $N = 16, 20$, and 24 for U instances</td>
<td>132</td>
</tr>
</tbody>
</table>
A.9 Computational Results for $N = 28$ and $32$ for $U$ instances . . . . . . . . . . . 133
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2 Tandem vehicle-drone delivery systems tested by UPS [Burns, 2017] and DHL [Forde, 2020]</td>
<td>6</td>
</tr>
<tr>
<td>1.3 Patents granted to Amazon show a battery swapping station atop a street light (top) and multi-level fulfillment centers for drones [Gentry et al., 2016, Curnlander et al., 2015]</td>
<td>11</td>
</tr>
<tr>
<td>2.1 A toy-sized example of timing variables in Model TSPD1</td>
<td>27</td>
</tr>
<tr>
<td>2.2 Allowable drone flight paths for service and rendezvous under $F = 30$</td>
<td>29</td>
</tr>
<tr>
<td>2.3 A <em>Double Greedy</em> initial solution (right) improves upon a greedy one (left)</td>
<td>34</td>
</tr>
<tr>
<td>2.4 Bound Quality Assessment, Average Deviations</td>
<td>40</td>
</tr>
<tr>
<td>2.5 Lower Bound Degradation, Average Deviations</td>
<td>43</td>
</tr>
<tr>
<td>3.1 A step-by-step illustration of DIP.</td>
<td>66</td>
</tr>
<tr>
<td>3.2 DIP Improvements by Flight Ratio, Instance Size</td>
<td>72</td>
</tr>
<tr>
<td>3.3 Relative Improvement and Computational Effort, DIP Phases 1 &amp; 2</td>
<td>73</td>
</tr>
<tr>
<td>4.1 Demand Inconsistency Across Two Periods in a Small TSP-DE Instance</td>
<td>85</td>
</tr>
<tr>
<td>4.2 Polar Visualization of TSP-DE Instance</td>
<td>86</td>
</tr>
<tr>
<td>4.3 An Overview of the Matheuristic</td>
<td>91</td>
</tr>
</tbody>
</table>
4.4 Drone-Eligible Nodes in a Densely-Populated Neighborhood ............ 93
4.5 Matheuristic Solution Quality by Stage .................................. 108
5.1 Breakdown of National Airspace Devoted to Delivery Drones in the United States Authority [2020] .................................................. 116
B.1 Drone Flight Scenario 1 ......................................................... 134
B.2 Drone Flight Scenario 2 ......................................................... 134
CHAPTER 1
INTRODUCTION AND PROBLEM
BACKGROUND/MOTIVATION

In the United States, the volume of parcels delivered to customers has grown rapidly: from 11 billion parcels in 2018 to 15 billion in 2020 [Laseter et al., 2018, Spadafora, 2020]. E-commerce sales, which had comprised just under 10% of retail spending in the United States in 2019, accounted for more than 25% of such sales in 2020 [Orendorff, 2021]. The requirements for social distancing and the avoidance of large gatherings at retail locations as a result of the Covid-19 pandemic have only accelerated this trend [Palmer, 2020]. Indeed, the increasing volume of deliveries has also been accompanied by a rise in the frequency of deliveries. In 2005, Amazon introduced a subscription-based delivery program called Prime that promised two-day delivery on a selection of items; the delivery time frame was eventually reduced to one day, and even two hours in certain cities [Del Ray, 2020]. In 2016, a Walmart executive scrapped plans by his company to introduce a similar two-day subscription model because “...two-day shipping had become ‘table stakes’ — a consumer expectation single-handedly created by Amazon Prime” [Del Ray, 2020]. Moreover, more than one quarter of all retail sales in the United States are now online [Khalid, 2019]. By 2020, Amazon became the leader for all online orders with 40% of the market; Walmart ranks second with a 5% share [Del Ray, 2020]. This explosion in online ordering has fundamentally altered the landscape of logistics, particularly those of the last mile.

Whereas traditional delivery networks are designed to optimize long-distance shipping, the shipping necessitated by e-commerce orders is typically much shorter - the number of trips shorter than 50 miles are growing by 25% annually [Laseter et al.,
The volume of deliveries is also far more variable - peak shipping volume in December, during holiday season in the United States, may be 25% greater than in September [Laseter et al., 2018]. This dynamic has spurred tremendous change in the logistics industry: Figure 1.1 tracks the growth in daily parcel delivery volume during the US holiday season by UPS, Fedex, and the United States Postal Service (USPS) [Fedex Public Relations, 2017, Frum, 2017, 2018, Ziobro, 2020a].

The increased volume of deliveries has led to higher staffing levels at logistics providers, and inevitably to increasing costs. The four largest carriers in the United States – Amazon, USPS, UPS, and Fedex – collectively employ more than 2.5 million people [Pellas, 2019]. Surges in demand due to seasonal shopping patterns, or more recently due to the Covid-19 pandemic, can lead to the hiring of 200,000 additional temporary laborers [Pellas, 2019]. This type of logistics planning model is inherently risky: fluctuations in demand can overwhelm the firm, and the additional labor
may further deplete profit margins [Berthiaume, 2020]. The risk is even greater for merchants, that unlike Walmart and Amazon, have little control over logistics cost. According to the CEO of Wing Aviation, a drone delivery firm that is owned by the same parent company as Google, “...Today a recipient is charged a delivery fee, but so is the merchant...our aim [with drone delivery] is to provide a service at a cost lower for both. We think single numbers of dollars will be the likely amount an order will cost when it is commercially live” [Lunden, 2018]. In pursuit of lower delivery costs, Walmart has invested in robots that may pick out groceries autonomously, and Amazon began development of delivery drones in 2013 [Del Ray, 2020, Murray and Chu, 2015]. Beyond convenience, service requirements may also incentivize logistics providers to invest in automation. The USPS, for example, is mandated by law to deliver mail and parcels to all residents of the United States; for certain remote areas, the marginal cost of delivery far exceeds marginal revenue.

According to a recent report by McKinsey, manned delivery over networks with long inter-customer distances and sparse drop density, such that drivers travel considerable distances between deliveries of small items, will become increasingly cost-prohibitive [Joerss et al., 2016]. Rising demand volume, according to the report, is likely to adversely affect traditional manned delivery. In addition to the expected rise in labor costs, rising parcel volume is likely to lead to increased usage of vehicles, leading in turn to traffic congestion and pollution. Traditional delivery vehicles are largely powered by diesel fuel, which deteriorates air quality; in a number of European countries, regulations have been passed to effectively ban diesel vehicles from the city centers, or to allow their entrance only during off-peak hours to relieve traffic congestion [Petzinger, 2018]. The report concludes that traditional diesel-powered vehicles used for last-mile deliveries are ill-suited for the parcels of the future [Joerss et al., 2016]. In their autonomy and reliance on electric batteries that may be recharged or exchanged, drones present a viable path to the future of last-mile logistics.
1.1 UAV Developments in Last-Mile Deliveries

Amazon’s 2013 initiatives for autonomous delivery technology have proven prescient, as the adoption of drones for commercial purposes has been accelerating. According to the Federal Aviation Authority (FAA), over 360,000 commercial drones were registered to operate in the United States as of May 2021, a figure expected to double by 2023 [Schaufele and Lukacs, 2019]. By comparison, the entire U.S. general aviation fleet had about 220,000 aircraft in 2017 [Pasztor, 2018, Schaufele and Lukacs, 2019]. One market highlighted by the FAA for vast potential is parcel delivery.

Starting in 2017, the FAA established several Integration Pilot Programs (IPPs) for unmanned aerial systems, or drones, across the United States with the goal of gathering data and feedback to inform future regulation of drones and accelerate their adoption in the National Airspace [Schaufele and Lukacs, 2019]. In a collaboration between federal and state governments as well as private firms, lead IPPs were awarded to the North Carolina Department of Transportation, and the city of San Diego, where trials are ongoing for parcel delivery via drone [Schaufele and Lukacs, 2019]. Beyond the United States, several firms including Amazon and Wing had demonstrated delivery by drone concepts in the United Kingdom and Australia, respectively, for the goal of testing the operational challenges of the technology.

In 2019, Wing Aviation, a subsidiary of the same company which operates Google, became the first drone operator to register with the FAA as a small-sized air carrier [Pasztor, 2019]. The designation allowed Wing to charge customers for its drone delivery service in Virginia [Pasztor, 2019]. Retailers like Walmart and pharmacies like CVS have also successfully partnered with logistics providers like UPS and startups like Flytrex to use drones for orders placed via app [Leonard, 2021].

While the commercial market for drone delivery develops, the FAA has been careful to regulate all aspects of commercial deployment in American airspace. As part of preliminary FAA regulations, commercial drone pilots have a designated flight
altitude and range: drones may not fly above 400 hundred feet, and must remain within the visual line of sight of their pilots at all times [Schaufele and Lukacs, 2019]. More recently, the FAA has designated a number of IPPs with diverse geographic conditions where drones may be piloted beyond the visual line of sight of their pilots [Pasztor, 2019]. For example, in North Carolina, the FAA approved Flytrex to deliver Walmart orders by flying drones remotely over civilians and their homes, with more than 2,000 households in range of the drone depot [Leonard, 2021].

1.1.1 Drone-Vehicle Integration

While several modes of drone delivery have been proposed, and several others may yet emerge, two such modes have become pre-eminent: a drone departing a depot for a customer directly; or the drone being carried aboard a traditional vehicle with the two carriers conducting deliveries in tandem. For several firms whose market share dominates the current commercial delivery industry, including UPS and Fedex, the latter model has gained traction. This may be driven by the firms’ existing investments in their traditional fleets. It may also be motivated by an interest in expediting the roll-out of drone deliveries, which may be managed more expediently through the firm’s existing fleet rather than in a drone-only setting where a lack of regulatory clarity may hamper further developments. In May 2018, the Workhorse Group began deliveries via drone near Cincinnati, Ohio as part of a joint project with the Federal Aviation Administration [Straight, 2018]. In the trial, packages of up to 10 lbs were carried autonomously by the drone for a flight range of up to 30 minutes; the company estimates the drone deliveries cost $0.03 per mile. In contrast, one third-party logistics firm in Ontario estimates that local dense delivery via a one-person box truck can cost $1.50 per mile [Hochfelder, 2017].

Under the vehicle-drone model, a single manned vehicle may be deployed with one or more UAVs aboard. The majority of the vehicle’s cargo capacity is reserved for
Figure 1.2: Tandem vehicle-drone delivery systems tested by UPS [Burns, 2017] and DHL [Forde, 2020]

parcels, with some space devoted to a hub for launching and receiving UAVs. Under this model, drones are assumed to have the capacity for carrying a single parcel (likely weighing 5 lbs or fewer) per flight. After delivery, the drone(s) return to the vehicle to be reloaded with new parcels. Prototype models developed by UPS and Mercedes Benz are shown in Figure 1.2.

1.1.1.1 Flight Conditions/Capacity

The engines permitting drone flight are typically powered by electric batteries located on board the drone, alongside antennas and cameras which allow the UAV to be piloted remotely while transmitting essential data back to its pilot. Fundamentally, these batteries represent two trade-offs for drone deliveries: first, the drone’s flight
time; and second, its carrying capacity. Batteries located on board the drone typically permit flights of up to 30 minutes at a speed of up to 60 miles per hour [Straight, 2018]. Conducting such a flight, however, exhausts the battery and require the drone to recharge before resuming flight. Thus the cumulative flight time of a drone, over many flights, is unconstrained. But individual flights are limited by the flight range allowed by the battery. More generally, drones have a limited carrying capacity: publicly-available prototypes are limited to a single parcel weighing at most 10 pounds [Straight, 2018]. Thus the weight of the parcel inherently limits the drone’s capacity for flight range and speed. Note that for both of these trade-offs, the drone-vehicle system presents advantages over the drone-only approach.

Future developments in the energy density of drones may mitigate both of these limitations. Improved energy storage may allow for longer drone flights. Alternatively, batteries with greater capacity at the same relative weight may allow for larger drones that can carry more parcels to be commercially-viable. Nevertheless, at the moment there remain two key trade-offs inherent in drone delivery systems as a result of their limited battery and flight capacity: drones need to recharge their batteries; and to reload parcels after each delivery.

1.1.1.2 Civilian Attitude/Perception of Drone Delivery

As both the development and the exposure of drone delivery platforms has progressed, a critical benchmark for the adoption of the technology is the acceptance of the general populace. The potential ubiquity of drones in the national airspace, their interactions with people and their property, as well as the potential safety and security risks associated with the technology have all led to reservations amongst many people about drone deliveries. Specifically, in a 2016 national survey conducted by the USPS, 44% of Americans liked the idea of drone delivery, with 23% remaining undecided [Soffronoff et al., 2016]. Amongst the 34% of users who did not favor
drone delivery, drone malfunctions, theft, and intentional misuse comprised 75% of the top concerns cited [Soffronoff et al., 2016]. Delivery within one hour was cited as the most important benefit amongst supporters of drone delivery [Soffronoff et al., 2016]. As trials have unfolded, however, these perceived customer concerns have been unfounded in the early going. Indeed, many customers who have experienced drone delivery generally reported positive experiences.

For example, after testing commercial drone deliveries in Christiansburg, Virginia starting in mid-2020, the Virginia Tech Mid-Atlantic Aviation Partnership (MAAP) surveyed local residents about their experience with drone deliveries. More than 89% reported positive experiences, saying that they had or planned to use the service [Knight, 2021]. The advent of the program during the Covid-19 pandemic and the ability to support local businesses with contactless delivery during the pandemic may have aided in the positive perception of drone delivery [Knight, 2021]. Additionally, in partnership with Wing Aviation, MAAP had spent months prior to the start of the trial educating members of the community about the technology and allowing them to ask questions [Knight, 2021]. Although a number of participants in the Christiansburg trial (17%) still complained of the noise caused by the drones, 75% of them remained positive on the outlook of the drone delivery program [Knight, 2021].

More generally, the nascent development of drone delivery platforms and the importance of customer acceptance makes the technology adaptable to customer concerns. Wing Aviation had faced similar complaints about noise pollution in an early trial in Canberra, a suburb of the Australian capital [Crumley, 2021]. That trial involved about 100 households receiving deliveries from local cafes and restaurants via drone [Crumley, 2021]. Residents complained of the incessant whirring and humming of drones hovering overhead [Crumley, 2021]. As a result, Wing re-designed its drone’s flight components to limit their audible perception at ground level [Crumley,
2021]. And in a second Australian trial in Logan, the company reported an increase of 500% in sales volume after installing the reduced-sound design [Crumley, 2021].

1.2 Diversified Drone Delivery Applications and Related Modeling Assumptions

The nascent development of drone technology and related research has yielded a variety of considerations both for vehicle-drone routing problems, and for drone applications more broadly. These considerations include engineering elements of the drone’s design, such as speed and flight range, which are analogous to vehicular operational assumptions and may be incorporated into mathematical models in a straightforward manner. For example, Agatz et al. [2018] explored the sensitivity of vehicle-drone routing solutions to drone speed and flight range, both of which were deterministic parameters in the study. Some engineering considerations for drones, however, require more subtle modeling adjustments.

For instance, although the flight range of the drone may be calculated based on the drone’s battery capacity, the range may also change dynamically, with the battery being exhausted more or less quickly depending on the speed at which the drone is flown. For example, Raj and Murray [2020] investigate a vehicle-drone routing problem in which the drone’s speed is represented by a decision variable, and may be slowed in case of carrying a relatively heavy parcel, or accelerated in case of incoming inclement weather. Indeed, the rate of ascent or descent by the drone, also known in the literature as trajectory optimization, during a flight or the drop-off of a parcel may also affect the flight range [Coutinho et al., 2018]. Similarly, drone deliveries powered by rechargeable electric batteries may reduce harmful carbon emissions as compared with diesel-powered vehicles. Goodchild and Toy [2018], however, showed that the reduction in emissions was dependent on the density of the customer network being served. These operational assumptions may affect routing decisions in subtle
ways continue to evolve with drone technology, and hearken to a need for modeling approaches that capture new details of this novel technology. For example, Chauhan et al. [2019] investigate a problem in which customers are assigned to localized facilities from which drones may be launched; Hong et al. [2018] consider path planning for drone deliveries in which flight paths must be planned around airspace associated with airports or military bases; and Escribano Macias et al. [2020] consider the deployment of facilities for rapid delivery of medical supplies via drone.

In the specific context of vehicle-drone routing optimization problems on which this work is focused, the academic literature features a variety of use-cases and operational assumptions that impact both modeling and solution approaches. Several studies, including Dayarian et al. [2020], Pina-Pardo et al. [2021], consider problems in which a vehicle delivers parcels and is dynamically re-supplied with new parcels by a drone launched from the central depot. Other studies, including Ham [2018], Schermer et al. [2019], Murray and Raj [2020], Luo et al. [2021] among others, investigate the implications of using multiple drones (whether ground-based or airborne) to aid with parcel delivery from a single vehicle. Moreover, Reed et al. [2021], investigate the impact of using an autonomous vehicle that allows a deliveryperson to drop off parcels while the vehicle drives itself to avoid congestion and parking in dense urban environments. In the same vein, Baloch and Gzara [2020] investigate the impact of drone-only deliveries on an e-tailer’s distribution network, using estimated costs from New York City as a case study. More broadly, Poikonen and Golden [2020] incorporate a parcel-specific drone energy function as well as specialized launch/landing nodes for the drone (distinct from customer locations) into their formulation. Collectively, these studies demonstrate the variety of assumptions and approaches underpinning research for vehicle-drone routing applications. The coming chapters focus on a particular stream of this literature, in which some consensus on
modeling assumptions has emerged, and highlight facets of the problem in which the presented work diverges from the literature.

Figure 1.3: Patents granted to Amazon show a battery swapping station atop a street light (top) and multi-level fulfillment centers for drones [Gentry et al., 2016, Curnlander et al., 2015].

Both well-established firms and start-ups are racing to develop new applications for drones, including in the context of logistics. Drones have been used to deliver blood supplies between depots in Rwanda, to transport cargo weighing up to 500 pounds, and to carry testing samples across a medical campus in North Carolina [Kolodny, 2019, Davies, 2018, Banker, 2020]. Start-ups are exploring the use of drones to carry multiple parcels at the same time, and using ground-based drones that navigate public roads and sidewalks to deliver customer orders [Singh, 2021, Steinberg, 2021]. As
governmental regulations unfold in the drone delivery space, firms may tailor technological developments to suit these new applications. For example, Figure 1.3 shows two of more than 50 patents filed by Amazon relating to potential innovations for drone delivery service [Michel, 2017].

1.3 Contributions

While synchronous travel between a vehicle and a companion drone allows the latter to recharge and to be re-loaded with parcels after deliveries, the inclusion of both carriers generates a number of challenges relating to minimizing the duration of the carriers’ tour, including:

- the assignment of customers to each carrier;
- the sequence of customer deliveries;
- the designation of rendezvous sites between the carriers in order to synchronize their operations;

Essay 1 investigates the so-called Traveling Salesman Problem with Drone (TSP-D), the objective of which is to synchronize operations between a single vehicle and its companion drone so as to minimize the duration of a tour serving a network of customers from a central depot. The essay contributes a novel MIP formulation enhanced by objective reformulation, valid inequalities, and pre-processing procedures that permit exact solutions to previously-unsolved benchmark instances from the literature. Whereas the largest instances solved optimally in the literature contained no more than 16 nodes (at the time of publication), the proposed formulation consistently yields provably optimal solutions for benchmark instances having 24 nodes as well as making a breach into instances having as many as 32 nodes.

Essay 2 proposes a variable neighborhood search (VNS) heuristic designed to solve industrial-scale instances of the TSP-D that incorporate cyclic drone flights. Starting
from restricted problem assumptions, the heuristic uses two distinct mixed-integer programming formulations to progressively permit synchronous travel between the vehicle and drone, as well as cyclic drone flights that launch and land, upon delivery, at the same node. By termination, the heuristic restores the full problem structure to yield high-quality feasible solutions. On a set of benchmark instances from the literature, the VNS improves upon the best-known results for 113/120 instances having up to 100 nodes, with comparable computational effort to existing approaches. We also propose pre-processing techniques to simplify the solution space associated with drone cycles, and investigate the conditions under which such cycles are optimal. Using the pre-processing procedures, the VNS yields solutions featuring drone cycles over roughly the same computational effort required for solving instances where cycles are disallowed. Furthermore, permitting drone cycles yields an average improvement of about 5% in both solution quality and the number of customers served via drone.

Essay 3 investigates a notion of carrier consistency, in which customers place recurring orders fulfilled across single or multiple weekly deliveries. A retailer uses a vehicle and its companion drone in tandem to fulfill orders, seeking to assign customers durably for delivery by vehicle or by drone and thereby to simplify the synchronized operations of both carriers throughout the entire planning horizon. The problem is amenable to a multi-period Traveling Salesman Problem with Drone Eligibility considerations for which we develop a mixed-integer program (MIP) formulation that yields provably optimal solutions for instances having up to 40 customers over a 6-period horizon. Moreover, for large-scale instances involving up to 200 customers over the same horizon, a matheuristic is proposed that generates assignments based on customers’ geographical locations and frequency of demand. By orchestrating neighborhood searches along these two features, the proposed heuristic effectively identifies in manageable times customers who are best served by drone, thereby consistently yielding solutions that exhibit optimality gaps within 0.5% on average.
In addressing these open questions in the literature, the succeeding chapters of this dissertation discuss in detail the related contributions and position the present work accordingly. In essence, the dissertation introduces and/or advances the tractability of computationally challenging operational and tactical optimization problems that arise in the deployment of drones into last-mile logistics.
CHAPTER 2

ENHANCED MIP FOR PARCEL DELIVERY BY VEHICLE AND DRONE

This chapter investigates the Traveling Salesman Problem with Drone (TSP-D) as a 0-1 mixed-integer program (MIP) that synchronizes vehicle and drone operations with the objective of minimizing the duration of the joint tour. Using a combination of valid inequalities, pre-processing, and other bound tightening strategies, we enhance the tractability of the proposed MIP formulation. In a computational study exploring benchmark instances, the proposed formulation demonstrates a computational advantage over extant formulations to produce optimal solutions to large instances in the literature.

2.1 Introduction

Advancements in drone-based deliveries have prompted the development of the TSP-D in the academic literature, which inherits the traditional computational challenges posed by the (notoriously difficult) Traveling Salesman Problem (TSP), with the additional challenges of specifying and synchronizing the delivery efforts of the vehicle and its companion drone. There is a growing, recent literature presenting modeling approaches [Murray and Chu, 2015, Agatz et al., 2018, Bouman et al., 2018, Es Yurek and Ozmutlu, 2018] as well as worst-case results for the problem [Wang et al., 2017, Poikonen et al., 2017] under different settings. This chapter contributes to the literature of exact solution approaches a novel MIP formulation which generates provably optimal solutions for the largest benchmark instances in the literature.
The remainder of this chapter is organized as follows. §2.2 presents a review of the literature for the TSP-D. §2.3 provides a formal problem statement and introduces our modeling approach. §2.4 details our model enhancement strategies. §2.5 presents our computational study, in which we solve benchmark instances from the recent literature, present sensitivity analysis with respect to the maximum flight duration of the drone, and highlight some limitations of the proposed methodology for larger problem instances. We present a conclusion and future research avenues in §2.6.

2.2 Literature Review

This section highlights two aspects of the related literature for the TSP-D: a review of exact solutions; and an overview of the operational assumptions therein. To position our work within the extant literature, we also highlight key assumptions in the literature about joint vehicle-drone operations.

2.2.1 Related Literature

According to an editorial by Agatz and Campbell [2018], there is a growing academic interest in the optimization of drone delivery systems. In addition to the aforementioned references, a review by Otto et al. [2018] cites exact solution approaches due to Ha et al. [2018], and Boysen et al. [2018]; Carlsson and Song [2017] and Campbell et al. [2017] rely on continuous approximation techniques. Ha et al. [2018] investigate minimizing the operational costs, which encompass transportation and waiting costs for each vehicle, which they term the min-cost TSP-D. Their model extends the formulation first introduced in Murray and Chu [2015]. Boysen et al. [2018] assume that the sequence of customers to be visited by the vehicle is fixed; they present approaches for generating drone tours along a given vehicle path. In this chapter, we focus on an exact optimization approach with the objective of minimizing the duration of carrier routes. In what follows, we use the terms drone and
vehicle whenever a specific reference is required; the term carrier refers to either mode of delivery.

In their “flying sidekick” traveling salesman problem (FSTSP), Murray and Chu [2015] presented a minimax MIP formulation that optimizes the latest carrier’s arrival at the depot. Over a collection of 72 instances, each with 10 nodes, they generated feasible solutions using their exact formulation, as well as heuristic solutions based on a variety of strategies. In their most effective heuristic implementation that improves upon an initial TSP tour, they report a mean improvement of 1.16% over the best-known solution produced by the MIP formulation in time-limited computational runs.

Agatz et al. [2018] introduced the notion of “operations” for the drone and vehicle that motivates the development of a set-covering type model. Each operation is a sequence of node visits beginning and ending with a node visited by both the vehicle and drone. Each sequence has at most one drone delivery and some non-negative number of vehicle deliveries. Each operation pairs partial tours for the drone and vehicle; summing the duration of all operations represents the length of a complete tour of the network. They reported exact solution times under 40 seconds for instances having up to 10 nodes; for instances having 12 nodes, however, they reported exceeding their machine’s memory after two hours of computational effort.

Es Yurek and Ozmutlu [2018] proposed a two-stage algorithm, in which the vehicle route (and resulting assignment of customers to carriers) is first determined dynamically, then an optimal completion of drone flights is determined using an MIP formulation. In the first stage, they generate an optimal TSP solution for the instance. Next, they determine a priori the maximum number of nodes that will be served by the drone, and proceed to enumerate vehicle routes that tour through the remaining nodes. In the second stage, an MIP formulation is used to generate optimal drone flights based on the routes generated in the first stage. Whenever this completed solution improves the best-known objective value for the instance, the al-
algorithm continues to explore vehicle routes with fewer drone assignments. In this way, the authors solved optimal instances having up to 12 nodes with a computational advantage over the formulations of Murray and Chu [2015] and Agatz et al. [2018]. For instances having more than 12 nodes, they suggest an optimization-based heuristic.

Bouman et al. [2018] retained the concept of operations from Agatz et al. [2018], with an additional parameter controlling the number of customers the vehicle may visit while the drone is in flight. Using a 3-pass dynamic programming approach, they investigated the impact of varying this parameter on solution quality and computation time. Under their most effective implementation, they solved instances having up to 16 nodes without restrictions on the vehicle visits (i.e. using an exact solution approach) within an average of 2.5 hours. With up to 12 hours of computational effort, they reported heuristic solutions for larger instances.

Poikonen et al. [2019] developed an approximate branch-and-bound (B&B) algorithm in which each B&B node is associated with a tour sequence. Starting at the root node (with an arbitrary 3-cycle), the farthest location from any of those already visited in the given sequence is chosen to be visited next. A full enumeration of all tour sequences that visit the new location forms the children nodes in the next level of the B&B tree. For each child node, the authors deploy an exact partitioning (EP) technique to determine the assignment of locations to each carrier. The EP of a complete tour (i.e. a sequence which visits all locations in the instance) forms a valid upper bound, while the EP of a partial tour (i.e. a sequence which does not visit all locations in the instance) generally yields a lower bound. To avoid the computational burden associated with a full enumeration of tours, the authors introduce a Tree Exploration Ratio (TER): the B&B search terminates whenever the ratio between the best-known upper and lower bounds is greater than or equal to TER. Thus if $\text{TER} = \infty$, all feasible solutions in the B&B tree are explored to yield an exact solution;
smaller values of TER yield heuristic solutions. The study reports exact solutions to benchmark instances with up to 10 nodes.

In addition to proving the $\mathcal{NP}$-hardness of the TSP-D, Tang et al. [2019] deployed constraint programming to model the problem. On randomly-generated instances having up to 18 nodes, the authors demonstrate that their approach has a computational advantage over the DP-based approach of Agatz et al. [2018]. Tang et al. [2019] also demonstrated that for instances having up to 50 nodes, their approach produced superior solutions to that of Poikonen et al. [2019]. The latter approach, however, was more effective for solving larger instances, and exhibited a computational advantage across instance size.

Schermer et al. [2019] introduced both exact MIP formulation and a heuristic approach for a multi-vehicle variant of the TSP-D, known in the literature as the vehicle routing problem with drones (VRP-D). After enhancing the formulation with a series of valid inequalities, the authors compared its performance with a matheuristic which partitions the VRP-D into sub-problems: allocation and sequencing; then assignment and scheduling. The first set of sub-problems assign customers to a particular vehicle-drone tandem; the latter sub-problems, termed the drone assignment and scheduling problem (DASP) by the authors, find an optimal path for a single vehicle and its companion drone(s) through a set of customers. The DASP does not assume a given vehicle route, but instead requires the vehicle and drone(s) to be routed simultaneously. As such, a single-vehicle DASP with a single drone is equivalent to the TSP-D. For large instances having up to 100 nodes, for which solving the DASP optimally was considerably more challenging, the authors proposed a scheme wherein a starting feasible solution may be “partitioned” into several overlapping sections, each of which may be individually optimized.
Considering the aforementioned literature, our approach relies on a novel MIP formulation that is enhanced with mathematical programming constructs to improve the tractability of the TSP-D and to enable exact solutions to larger instances.

2.2.2 Extant Operational Assumptions

Table 2.1 highlights six key vehicle-drone operational assumptions in extant computational studies. For each publication, after listing the authors’ approach and the size of the largest instance solved optimally, the next two columns of Table 2.1 show associated assumptions about the maximum flight time of the drone and its speed relative to the vehicle. While Table 2.1 indicates that Agatz et al. [2018] and Bouman et al. [2018] assumed the drone has unlimited flight time for deliveries, Murray and Chu [2015] and Es Yurek and Ozmutlu [2018] assumed time-restricted drone flights, which reflect current and foreseeable industry practices. Workhorse, which is currently conducting trials of drone deliveries in the United States, and Amazon Prime Air have both developed drones intended to deliver parcels within a range of 30 minutes [McFarland, 2017, Straight, 2018]. Indeed, Agatz et al. [2018] include a parameter in their formulation to “...take into account other practical restrictions on the path of the drone...”. Although longer drone flights may eventually become technically possible, to ignore the constraints associated with limited flight times may produce solutions that are of limited practical use due to the current battery capacity of drones. As a consequence, we investigate the trade-off between logistical gains and computational expense for gradually relaxing this parameter in §2.5.5.

As other studies have noted, flight time is not the sole determinant of the number of deliveries made by the drone [Agatz et al., 2018]. Indeed, the flight range of the drone (which is informed by both flight time and speed) determines the consequent range of customers within reach for drone delivery. As highlighted in Table 2.1,
Table 2.1: A summary of key operational assumptions in the literature

<table>
<thead>
<tr>
<th>Publication</th>
<th>Approach</th>
<th>Largest Instance Solved Optimally (Nodes)</th>
<th>Drone Flight Time (Minutes)</th>
<th>Ratio of Drone to Vehicle Speed</th>
<th>Repeated Visits</th>
<th>Nodes Eligible for Drone Delivery (%)</th>
<th>Service Time</th>
<th>Drone Depot Arrival</th>
</tr>
</thead>
<tbody>
<tr>
<td>Murray and Chu [2015]</td>
<td>MIP</td>
<td>10</td>
<td>20/40</td>
<td>0.6-1.4</td>
<td>N</td>
<td>80-90</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Agatz et al. [2018]</td>
<td>IP</td>
<td>10</td>
<td>Unlimited</td>
<td>2</td>
<td>Y</td>
<td>100</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Es Yurek and Ozmutlu [2018]</td>
<td>Dynamic Programming + MIP</td>
<td>12</td>
<td>20</td>
<td>1.4</td>
<td>N</td>
<td>100</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Bouman et al. [2018]</td>
<td>Dynamic Programming</td>
<td>16</td>
<td>Unlimited</td>
<td>2</td>
<td>Y</td>
<td>100</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Poikonen et al. [2019]</td>
<td>MIP</td>
<td>10</td>
<td>10/20/30</td>
<td>0.5-3.0</td>
<td>Y</td>
<td>100</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Tang et al. [2019]</td>
<td>MIP</td>
<td>10</td>
<td>20</td>
<td>2.0</td>
<td>Y</td>
<td>100</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Schermer et al. [2019]</td>
<td>MIP</td>
<td>10</td>
<td>Variable</td>
<td>1.0-3.0</td>
<td>N</td>
<td>100</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>This chapter</td>
<td>MIP</td>
<td>32</td>
<td>30</td>
<td>2</td>
<td>N</td>
<td>100</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>

Our assumptions are as conducive to drone deliveries as comparable studies in the emerging TSP-D literature.

Table 2.1 also highlights an operational assumption regarding repeated carrier visits to a single node: Agatz et al. [2018] and Bouman et al. [2018] allow both carriers to visit the same node multiple times, while Murray and Chu [2015] and Es Yurek and Ozmutlu [2018] allow only a single visit by each carrier to any node. The former studies motivate this choice by illustrating in Agatz et al. [2018] an example of an optimal solution in which the vehicle’s optimal path visits one customer twice: once for delivery; then again to re-supply the drone. As other studies have, this chapter allows at most a single visit by each carrier to a node.

The next two columns in Table 2.1 show the ratio of customers eligible for drone delivery, and whether the authors allowed for service times associated with each customer. Although service times are typical of parcel delivery applications, the
service time associated with a drone delivery is uncharted territory. According to a patent recently filed by Amazon, several options exist for the drone to leave a parcel at a customer location, including: vertically descending to deliver; using a tether to lower the parcel; or allowing the parcel to parachute into a designated landing zone [Michel, 2017]. While these systems are likely to be faster than a delivery person, the totality of service times associated with drone deliveries is more complex. After each delivery, a drone requires a new parcel to be loaded into its delivery bay. The drone may also require a battery swap to replenish its flight capacity. As of this writing, only Murray and Chu [2015] explicitly incorporated service times into their computational study. This work conforms to the predominant choice made in the academic literature of not accounting for service times, knowing that they may be incorporated in the future.

The final column denotes whether the drone may arrive at the depot before the vehicle. Murray and Chu [2015] and Es Yurek and Ozmutlu [2018] allow this feature, while Agatz et al. [2018] and Bouman et al. [2018] do not. None of the aforementioned authors expound on this assumption. Logistically, allowing the drone to return to the depot prior to the vehicle is justifiable since the drone may await the vehicle on the ground just as it would at any customer node. Moreover, the duration of the tour is determined by the arrival of the latest carrier at the depot. Thus even if the drone were to arrive at the depot before the vehicle, the arrival time at the depot would be determined by the vehicle’s arrival at a later time.

Finally, several of the aforementioned recent studies have investigated general advantages of joint vehicle-drone over vehicle-only routing, as well as the sensitivity of vehicle-drone operations to parameters such as drone speed and maximum flight time. For example, Wang et al. [2017] present a theorem which shows that a lower bound on the optimal objective value of the TSP-D may be expressed as a function of the optimal TSP objective value for the same instance; in computational results,
Agatz et al. [2018] report an average improvement of about 30% for heuristic TSP-D solutions over optimal TSP solutions.

2.3 Optimization Model

Formally, the TSP-D may be defined as the problem of routing one ground vehicle and one companion drone through a network of nodes such that each node is served by exactly one carrier. The objective is to minimize the return time of both carriers to the depot. It is assumed that the drone and vehicle may depart the depot exactly once. The vehicle may visit any node only once. The drone may deliver only once per flight before returning to the vehicle or to the depot. If the drone returns to the depot, it may not depart again. If the drone is not in flight or else waiting for the vehicle to arrive at a customer node where the drone is located, then the drone must be transported aboard the vehicle. The duration of individual drone sorties is capped, in order to account for limited battery life, but the cumulative flight time of the drone is not. That is, we assume that the drone may fly for a limited amount of time before returning to the vehicle to swap batteries and instantly recharge to fly again. All items are assumed to be eligible for delivery by either carrier.

The drone and vehicle may wait for one another at a node. It is assumed that if the drone waits, it has flown to the customer and waits on the ground. Further, if the vehicle arrives at a node after the drone, the customer at that node is served before the drone is launched. At nodes where they meet, the drone and vehicle depart at the same time. Thus the drone will always wait for the vehicle to complete delivery at a rendezvous node before they both depart. As discussed in §2.2, service times associated with either carrier may be added to the formulation, and the nodes eligible for drone delivery may also be specified under alternate assumptions. Our notation is summarized as follows:
Input Parameters

- \( V = \{0, 1, \ldots, n\} \): Set of all nodes in the network.
- \( V^* = V \setminus \{0\} \): Set of customer nodes, which excludes the central depot.
- \( N \): The total number of nodes in the network, i.e. \( N = n + 1 \).
- \( \alpha \): Ratio of drone to vehicle speed.
- \( t_{ij} \): Vehicle travel time from \( i \) to \( j \), \( \forall i, j \in V, i \neq j \).
- \( f_{ij} \): Drone travel time from \( i \) to \( j \), \( \forall i, j \in V, i \neq j \).
- \( M \): A “large enough” scalar.
- \( F \): Maximum flight duration for the drone to make a single delivery then return to the vehicle.

Decision Variables

- \( x_{ij} \in \{0, 1\} \): \( x_{ij} = 1 \iff \) the vehicle travels from \( i \) to \( j \), \( \forall i, j \in V, i \neq j \).
- \( y_{ij} \in \{0, 1\} \): \( y_{ij} = 1 \iff \) the drone travels from \( i \) to \( j \), \( \forall i, j \in V, i \neq j \).
- \( z_{ij} \in [0, 1] \): \( z_{ij} = 1 \iff \) the drone flies from \( i \) to \( j \), \( \forall i, j \in V, i \neq j \). Note if the drone is carried aboard the vehicle, \( y_{ij} = 1 \) but \( z_{ij} = 0 \).
- \( w_j \in [0, 1] \): \( w_j = 1 \iff \) the vehicle visits node \( j \), \( \forall j \in V \).
- \( a_j \in [0, M] \): The arrival time of the delivery carrier at node \( j \), \( \forall j \in V \).
- \( d_j \in [0, M] \): The departure time of the delivery carrier at node \( j \), \( \forall j \in V \).
- \( e_j \in [0, f_{j}^{max}] \): The time spent by the vehicle waiting for the drone at node \( j \),
  \[ \forall j \in V, where f_{j}^{max} = \max\{\max_{i,k \in V, f_{ik} + f_{kj} \leq F, f_{ik} + f_{kj} - t_{ij}}\}, 0\} \]

Our MIP formulation, denoted by \textbf{TSPD1} is as follows:
TSPD1: Minimize $\sum_{i \in V} \sum_{j \in V} t_{ij}x_{ij} + \sum_{k \in V} e_k$  \hspace{1cm} (2.1a)

s.t. $\sum_{j \in V^*} x_{0j} = 1$,  \hspace{1cm} (2.1b)

$\sum_{j \in V} x_{ij} = w_i$, \hspace{1cm} $\forall i \in V$,  \hspace{1cm} (2.1c)

$\sum_{j \in V} x_{ij} - \sum_{j \in V} x_{ji} = 0$, \hspace{1cm} $\forall i \in V$,  \hspace{1cm} (2.1d)

$\sum_{j \in V^*} y_{0j} = 1$,  \hspace{1cm} (2.1e)

$\sum_{j \in V} y_{ij} - \sum_{j \in V} y_{ji} = 0$, \hspace{1cm} $\forall i \in V$,  \hspace{1cm} (2.1f)

$1 - w_i \leq \sum_{j \in V} y_{ij} \leq 1$, \hspace{1cm} $\forall i \in V$,  \hspace{1cm} (2.1g)

$w_i + w_j \geq y_{ij}$, \hspace{1cm} $\forall i, j \in V | i \neq j$,  \hspace{1cm} (2.1h)

$y_{ij} + w_i + w_j \leq x_{ij} + 2$, \hspace{1cm} $\forall i, j \in V | i \neq j$,  \hspace{1cm} (2.1i)

$y_{ij} - x_{ij} \leq z_{ij} \leq 1 - x_{ij}$, \hspace{1cm} $\forall i, j \in V | i \neq j$,  \hspace{1cm} (2.1j)

$z_{ij} \leq y_{ij}$, \hspace{1cm} $\forall i, j \in V | i \neq j$,  \hspace{1cm} (2.1k)

$\sum_{i \in V} f_{ij}z_{ij} + \sum_{k \in V} f_{jk}z_{jk} \leq F + Mw_j$, \hspace{1cm} $\forall j \in V^*$,  \hspace{1cm} (2.1l)

$a_j \geq d_i + t_{ij} - M(1 - x_{ij})$, \hspace{1cm} $\forall i, j \in V | i \neq j$,  \hspace{1cm} (2.1m)

$d_j \geq a_j$, \hspace{1cm} $\forall j \in V^*$,  \hspace{1cm} (2.1n)

$a_j \geq d_i + f_{ij} - M(1 - z_{ij}) - Mw_j$, \hspace{1cm} $\forall i, j \in V | i \neq j$,  \hspace{1cm} (2.1o)

$d_j \geq d_i + f_{ij} - M(1 - z_{ij})$, \hspace{1cm} $\forall i \in V, j \in V^*$,  \hspace{1cm} (2.1p)

$e_j \geq d_j - (d_i + t_{ij}) - M(1 - x_{ij})$, \hspace{1cm} $\forall i \in V, j \in V^* | i \neq j$,  \hspace{1cm} (2.1q)
\[ e_0 \geq (d_j + f_{j0}) - (d_i + t_i0) - M(1 - x_{i0}), \quad \forall i, j \in V^*|i \neq j, \quad (2.1r) \]

\[ a_0 \leq \sum_{i \in V} \sum_{j \in V} t_{ij}x_{ij} + \sum_{k \in V} e_k, \quad (2.1s) \]

\[ a, d, w, z, e \geq 0, \quad w \leq 1, \quad (2.1t) \]

\[ a, d \leq M, \quad e \leq f^{\text{max}}, \quad x, y \text{ binary}. \quad (2.1u) \]

**Remark.** Although it is possible to eliminate Constraint (2.1l) for certain nodes which are ineligible for drone delivery, doing so had a negligible impact on the computational performance of the formulation in our experience.

The objective function (2.1a) minimizes the duration of the carriers’ tour. Specifically, the summations of \(x\)-variables and \(e\)-variables minimize the travel and waiting time of the vehicle, respectively. Combined, these terms account for the makespan of a tour in which both carriers have returned to the depot.

**Remark.** Note that objective (2.1a) is mathematically equivalent to minimizing \(a_0\), the arrival time at the depot. But the former objective drastically reduces the computational effort of the solver by strengthening the underlying continuous relaxation.

Constraints (2.1b) and (2.1d) ensure that the vehicle departs the depot for exactly one destination, and thereafter travels from exactly one predecessor node to exactly one successor node. Constraints (2.1e) and (2.1f) serve the same purpose for the drone, whereas Constraint (2.1c) sets \(w_i = 1\) for any node \(i\) visited by the vehicle. Constraint (2.1g) enforces a visit by the drone in the absence of a vehicle visit. More generally, if the drone travels from \(i\) to \(j\), Constraint (2.1h) ensures that at least one of \(i\) or \(j\) must be visited by the vehicle. Constraint (2.1i) requires that if the drone travels from \(i\) to \(j\) where both \(i\) and \(j\) are served by the vehicle, then the vehicle must also travel from \(i\) to \(j\). Relating to the drone’s flight arcs, Constraint (2.1j)
requires that the drone fly from $i$ to $j$ if it was not transported aboard the vehicle. Constraint (2.1k) ensures that if the drone flies from $i$ to $j$, then it must be traveling from $i$ to $j$. Constraint (2.1l) guarantees that for a given flight, the drone can travel for at most a duration of $F$ before rendezvousing with the vehicle. Constraint (2.1t) enforces nonnegativity and binary restrictions on the decision variables. Given the binariness of the $x$- and $y$-variables and the related constraints, the $w$- and $z$-variables are guaranteed to be binary.

Figure 2.1: A toy-sized example of timing variables in Model TSPD1

Constraints (2.1m) through (2.1p) relate to arrival and departure times in the spirit of Miller-Tucker-Zemlin-type subtour elimination constraints (MTZ SECs). Depending on which carrier serves a node, several of (2.1m) through (2.1p) may define the arrival and departure times at that node. Figure 2.1 illustrates a toy-sized example, in which the arrival time at node 1 is set according to the movement of both
vehicles. At node 2, the arrival time is determined by the drone (since it delivers the parcel there), and the arrival time at node 3 is set according to the vehicle’s travel time. But since the carriers rendezvous at node 3, the departure time there is determined by the latter carrier to arrive. Specifically, the departure time at node 3 takes into account any time spent waiting by the vehicle for the drone’s arrival.

Due to the construction of the objective function, constraints to determine vehicle waiting time are required. Constraint (2.1q) sets the vehicle waiting time equal to the difference of the actual and expected (i.e. the departure time with no waiting associated) departure times from a node. At the depot, Constraint (2.1r) ensures that whenever the drone departs a node later than the vehicle, the vehicle’s waiting time is equal to the difference. Finally, although Constraints (2.1m) and (2.1o) set lower bounds for the value of the arrival time at the depot, the upper bound in the formulation is not tight. That is, any scalar larger than $a_0$ is a suitable value for $M$, such that $a_0 > \sum_{i \in V} \sum_{j \in V} t_{ij} x_{ij} + \sum_{k \in V} e_k$ is possible. To ensure $a_0$ represents the latest arrival time of either carrier at the depot, Constraint (2.1s) is deployed.

Note the inclusion of $M$ in Constraint (2.1l), which limits the drone’s flight duration while simultaneously allowing it to rendezvous with the vehicle. Figure 2.2 illustrates both scenarios. Shaded circles correspond to rendezvous nodes between the vehicle and drone, unshaded circles correspond to nodes served by the drone, and italicized numbers above the dotted arcs show the drone’s flight time. Suppose $F = 30$; then Constraint (2.1l) ensures that a drone flight originating at node 1, serving node 2, and rendezvousing at node 3 is allowed. Using the same constraint without the $Mw_j$ term, however, the drone would be unable to launch from node 3 to serve node 4 after having landed from node 2, since $f_{23} + f_{34} > F$. Thus the term $Mw_j$ resets drone flight time whenever the carriers rendezvous.
2.4 Bound Improvement Strategies

In this section, we propose several model enhancements that aim at further strengthening the underlying continuous relaxation of Model TSPD1, generating tighter upper bounds on the optimal objective value, as well as pre-processing procedures that simplify the formulation.

2.4.1 Lower Bound Improvements

We begin with a focus on improving the objective value of the continuous relaxation of Model TSPD1. This is achieved by introducing a series of valid inequalities as described next and by tightening “M” scalars in the formulation.

2.4.1.1 Valid Inequalities

A feasible solution to a TSP may be characterized by the connectivity of a sub-graph describing the movement of the vehicle. That is, since the vehicle tour must begin and terminate at the depot while visiting every other node in the network, an inequality which ensures the connectivity of that vehicle tour would be valid for the TSP formulation. Further details may be found in Laporte [1986]. The same principle applies for the TSP-D. In the latter problem, the existence of two carriers – each of which traverses a subgraph through the network – ensures that a broader series of valid inequalities inducing connectivity may be derived, as follows:
\[
\sum_{(k,l) \in \delta^+(S)} x_{kl} \geq w_i, \quad i \in V^*, S \subset V : 0 \in S, i \notin S, \quad (2.3a)
\]
\[
\sum_{(k,l) \in \delta^+(S)} y_{kl} \geq \sum_{j \in V} y_{ij}, \quad i \in V^*, S \subset V : 0 \in S, i \notin S. \quad (2.3b)
\]

In Model TSPD1, the \(x\)-variables define a tour through node 0 as well as all nodes satisfying \(w_i = 1\). Constraint (2.3a) ensures the connectivity of that subgraph, where \(\delta^+(S)\) is the set of arcs having their origin nodes in \(S\) and destination nodes in \(V \setminus S\). Similarly, Constraint (2.3b) guarantees that if \(i \in V^*\) is visited by the drone (i.e, \(\sum_{j \in V} y_{ij} > 0\)), then the incidence vector of the \(y\)-variables induces a dipath from the depot through node \(i\). The introduction of these cut-set constraints can improve the objective value of the LP relaxation (LPR) of Model TSPD1 by eliminating a specific fractional LP solution. In practice, it is computationally onerous to generate all such cut-set constraints, the number of which grows exponentially. Instead, we use a constraint generation procedure that introduces rounds of such cuts dynamically to strengthen the model’s LPR at the root node of B&B/C algorithm.

A summary of the procedure for cut-set generation is below; let \(Z^*_{LPR}\) be an optimal LPR objective value, let \((\bar{x}, \bar{y}, \bar{w})\) denote variable values associated with \(Z^*_{LPR}\), let \(\epsilon\) be a scalar used for precision, let \(\text{MF-X}\) and \(\text{MF-Y}\) denote maximum flow problems for the vehicle and drone tours, respectively, and let \(NII\) be the number of non-improving iterations. Appendix A.1 features a small example of the procedure.

1. Given \(Z^*_{LPR}\) with fractional \((\bar{x}, \bar{y}, \bar{w})\), cut-set constraints may be constructed for both the vehicle and drone paths (\(\bar{x}\)- and \(\bar{y}\)-variables, respectively). For \(i \in V^*\), solve \(\text{MF-X}\) and \(\text{MF-Y}\). Retain the associated violations and minimum cut-sets.

We focus on cut-sets which maximize the violation of the constraint to be added by the fractional solution at hand. These are known as minimum cuts,
which may be generated by solving a maximum flow linear program [Ford and Fulkerson, 1956].

2. Append to the formulation only the largest of either the most violated vehicle or drone cut-set constraints.

   This procedure introduces a single deep cut, rather than a number of implied cuts.

3. Re-solve for $Z_{LPR}^*$; the previous solution and associated variable values will be cut off. New values of $(\bar{x}, \bar{y}, \bar{w})$ will be obtained.

4. If $Z_{LPR}^*$ improves by less than $\epsilon$, let $NII \leftarrow NII + 1$.

5. Repeat Steps 1-5 until a sufficient number of $NII$ is observed.

   The most effective parameter settings in our experience were $\epsilon = 0.005$ and $NII = 3$.

2.4.1.2 Scaled “Big M” Values

MIP formulations that involve Big-M-type constraints are known to produce rather weak LP relaxations. Using a single valid $M$ value throughout the formulation results in further weakening the underlying continuous relaxation. Instead, we introduce in this section a set of $M$ parameters, each tailored to satisfy a specific constraint in Model TSPD1. In what follows, let $D_{\text{max}}$ be the duration of a feasible TSP-D tour.

Each of the affected constraints from Model TSPD1 is reproduced below with updated notation and tightened $M$ values; derivations are available in Appendix A.2:
\[ \sum_{i \in V} f_{ij}z_{ij} + \sum_{k \in V} f_{jk}z_{jk} \leq F + M^1_jw_j, \quad \forall j \in V^*, \quad (2.4a) \]
\[ a_j \geq d_i + t_{ij} - M^2_{ij}(1 - x_{ij}), \quad \forall i, j \in V | i \neq j, \quad (2.4b) \]
\[ a_j \geq d_i + f_{ij} - M^3_{ij}(1 - z_{ij}) - M^4_jw_j, \quad \forall i, j \in V | i \neq j, \quad (2.4c) \]
\[ d_j \geq d_i + f_{ij} - M^5_{ij}(1 - z_{ij}), \quad \forall i \in V, j \in V^*, \quad (2.4d) \]
\[ e_j \geq d_j - (d_i + t_{ij}) - M^6_{ij}(1 - x_{ij}), \quad \forall i \in V, j \in V^* | i \neq j \quad (2.4e) \]
\[ e_0 \geq (d_j + f_{j0}) - (d_i + t_{i0}) - M^7_i(1 - x_{i0}), \quad \forall i, j \in V^* | i \neq j, \quad (2.4f) \]
\[ d_j \leq M, \quad \forall j \in V^*. \quad (2.4g) \]

where

\[ M^1_j = \max(0, \max_{i \in V} f_{ij}, \max_{j \in V} f_{jk}), \quad \] when \[ f_{ij} \leq F, f_{jk} \leq F \]

\[ M^2_{ij} = \begin{cases} 
    t_{0j} - f_{0j}, & i = 0, \\
    t_{i0} - f_{i0}, & j = 0, \\
    (D_{\max} - f_{i0} + t_{ij}) - f_{0j}, & \text{otherwise},
\end{cases} \]

\[ M^3_{ij} = \begin{cases} 
    0, & i = 0 \text{ or } j = 0, \\
    (D_{\max} - f_{j0} + f_{ij}) - f_{0j}, & \text{otherwise},
\end{cases} \]
\[ M_{ij}^4 = \begin{cases} (f_{0j} - t_{0j}), & i = 0, \\ 0, & j = 0, \\ (D_{\text{max}} - f_{j0} + f_{ij}) - t_{0j}, & \text{otherwise}, \end{cases} \]

\[ M_{ij}^5 = \begin{cases} 0, & i = 0, \\ (D_{\text{max}} - f_{i0} + f_{ij}) - f_{0j}, & \text{otherwise}, \end{cases} \]

\[ M_{ij}^6 = \begin{cases} (D_{\text{max}} - f_{j0}) - t_{0j}, & i = 0, \\ (D_{\text{max}} - f_{j0}) & \text{otherwise}, \end{cases} \]

\[ M_i^7 = D_{\text{max}} - t_{i0}, \quad \forall i \in V^*, \]

\[ M = D_{\text{max}} - \min_{i,j \in V^*, f_{ij} + f_{j0} \leq F} \{f_{j0}, t_{j0}\}. \]

2.4.1.3 A Heuristic for Generating an Initial Feasible TSP-D Solution

Due to the role of \( D_{\text{max}} \) in scaling \( M \) parameters, it is important generate a high-quality initial solution. To this end, we developed a Double Greedy algorithm. Starting at the depot, the Double Greedy algorithm finds the two nearest nodes that have yet to be served. From the last node visited, the algorithm determines the time required by the vehicle to visit and serve the nearest and next-nearest node in that order, respectively. Starting at the last node visited, the algorithm also determines the flight time required for the drone to serve the nearest node and fly to the next-nearest node for a rendezvous with the vehicle. If such a drone flight is judicious (i.e., time-saving as compared with the alternative of delivery via vehicle), then the algorithm assigns the nearest node to delivery via drone, and the next-nearest node to vehicle delivery. If the drone flight is irrelevant due to excessive waiting time for the vehicle or flight time restrictions, then the vehicle is routed to its nearest neighbor with the drone on board, and the algorithm begins anew. In the worst case, the Double Greedy algorithm guarantees a heuristic solution equivalent to that of a
greedy vehicle tour. In our experience, however, *Double Greedy* frequently improved upon greedy vehicle tours, as illustrated in Figure 2.3.

Figure 2.3: A *Double Greedy* initial solution (right) improves upon a greedy one (left)

2.4.2 Upper Bound Improvements

In our experience with the TSP-D, commercial solvers are often encumbered by the difficulty of generating good feasible solutions in early stages of the B&B/C algorithm. Initializing the solver with a so-called “warm” start, in which a feasible objective value with associated integral variable values are specified, may be an effective strategy to overcome this difficulty. Warm starts improve computational performance in at least two ways: firstly, a feasible solution provides a cutoff value which is useful in pruning the B&B tree; secondly, the availability of a feasible solution provides a more meaningful value of $D_{\text{max}}$ to better scale the values of $M$ discussed in §2.4.1.2. Moreover, commercial solvers deploy certain procedures (e.g., the Relaxation Induced Neighborhood Search) for discovering improving solutions in the vicinity of feasible solutions, provided that such feasible/incumbent solutions are available in the first place. In summary, there is a pass-through effect associated with warm starts: they immediately improve the best-known upper bound, and they can even improve the best-known lower bound by scaling $M$ values. To maximize the benefit of a warm start
while avoiding excessive computational effort in the generation thereof, we devised
the following multi-step exact solution procedure:

- **Step 0:** Append cut-set constraints to Model TSPD1 using Algorithm 1 \( (\epsilon = 0.005, N, N, N = 3) \). Hereafter, let the augmented formulation be Model TSPD1A.

- **Step 1:** Solve an MIP relaxation of Model TSPD1A, in which \( x \)-variables are binary, while the \( y \)-variables are continuous; let the resulting solution be \( (\bar{x}, \bar{w}) \).

- **Step 2:** Set \( w_i = 1 \) \( \forall i \in V^* | \bar{w}_i = 1 \) from Step 1. Using the Double Greedy procedure, generate an initial feasible solution using the fixed \( w \)-variables. Let \( Z_{\text{warm}} \) be the solution generated with the fixed \( w \)-values, and let \( D_{\text{max}} = Z_{\text{warm}} \).

- **Step 3:** Unfix all variables, and declare \( x \)- and \( y \)-variables as binary. Solve Model TSPD1A with a focus on generating feasible solutions. Let \( Z_{\text{warm}}^2 \) be the best-known solution, and let \( D_{\text{max}} = Z_{\text{warm}}^2 \).

- **Step 4:** Solve Model TSPD1A optimally, with \( Z_{\text{warm}}^2 \) as a warm start.

The main benefits of this approach are that it allows the TSP-D to be solved pro-
gressively, where relaxations of the problem progressively generate useful information
for solving the full MIP model. Specifically, solving Step 0 generates a warm start for
Step 1; solving Step 1 generates a feasible assignment of \( w \)-variables for Step 2, and
a valid lower bound for Step 4; solving Step 2 generates a warm start for Step 3; and
solving Step 3 focuses the solver on improving the upper bound bridging the dual gap
for an optimal solution in Step 4. Various solver settings are adjusted through the
course of this approach; for Steps 1 and 3, a time limit of 1,000 seconds is imposed;
the solver’s emphasis is on finding feasible solutions; the solver may terminate with
a relative MIP gap of 5%; and the solver is limited to 25 MIP solutions. In Step 4,
we limit the solver to 10,000 seconds of computation time.
2.4.3 Instance Pre-processing

To further improve the performance of our formulation, the customer network may be sparsified a priori. Since the drone is limited to a flight time of $F$, for every $f_{ij} \geq F$, fix $z_{ij} = 0 \ \forall i,j \in V$. Certain nodes may also be deemed ineligible for delivery by drone. Let $f_{i}^{\min} = \min_{j,k \in V} \{ f_{ji} + f_{ik} \} \ \forall i \in V^*$, such that $f_{i}^{\min}$ is the minimal duration of a drone’s flight to serve customer $i$. If $f_{i}^{\min} > F$, then $i$ requires a flight time greater than the drone’s flight duration; thus, we fix $w_i = 1$ in order for the vehicle to serve such customers. We may also eliminate certain flight arcs from the network that may never be part of a feasible delivery flight. As Constraint (2.4a) shows, in order for arc $f_{ij}$ to be part of a flight path, it must be used by the drone either to launch from a rendezvous node to a customer, or to land at a rendezvous node after serving a customer. If no such flights are possible, then the flight arc is irrelevant. More precisely, $\forall i,j \in V$, find $\min_{k \in V} \{ f_{jk}, f_{ki} \}$, where $i \neq j, j \neq k, k \neq i$. If $f_{ij} + f_{jk} > F$ and $f_{ki} + f_{ij} > F$, then fix $z_{ij} = 0$. Finally, during a commercial solver’s search for an optimal solution, a higher priority may be assigned to the $x$-variables. As discussed in §2.3, the $x$-variables play a key role in determining the length of both carriers’ tours; giving those variables precedence in the branch-and-bound search significantly improved the solver’s performance.

2.5 Computational Study

This section presents computational results using a machine with an Intel i7-7700K processor and 32 GB of RAM, on which models were implemented via AMPL, and solved using Gurobi version 8.1.0. We have given consideration to CPLEX, XPRESS, and Gurobi, but found the latter to yield overall better results in our experience. For all instances, we assume: the drone travels at twice the vehicle’s speed (i.e., $\alpha = 2$); all nodes are eligible for service by the drone; and the drone’s maximum travel duration per flight, $F$, is limited to 30 minutes. As highlighted in §2.2, these
assumptions permit a comparable variety of drone operations as those allowed in the extant literature, and conform to foreseeable industry implementations.

2.5.1 Problem Instances

We use instances by Bouman et al. [2018], designed for the TSP-D. Specifically, we focus on the single-center (SC); dual-center (DC); and uniform (U) topographies, where nodes are likely to be clustered around a single depot to mimic a city center; dual depots; or uniformly about the map, respectively. Details of the generation scheme are available in Agatz et al. [2018]. These instances were published with \( N = 8, 9, 10, 20, \) and 50. For the other values of \( N \) discussed below, customers were chosen in order from the next-largest available data set (e.g., the first 16 nodes from \( N = 20 \) instances are chosen for \( N = 16 \)). We note:

- All nodes are connected with symmetrical travel distances;
- We assume Euclidean travel distances, where the triangle inequality holds;
- All distances were rounded to 6 decimal places. The integrality threshold of the solver was also set to 6 decimal places.

Finally, as noted in §2.2, some authors have solved instances as large as \( N = 16 \). Using instances of \( N = 8, 9, \) and 10, the preliminary results of our MIP formulation drastically outperformed that of Murray and Chu [2015] with an improvement in computation time of over 99% (both formulations were implemented and solved using our computational platform for comparative purposes). Although Agatz et al. [2018] and Es Yurek and Ozmutlu [2018] published comparisons of their results against those of other authors, they provided exact solutions to instances of similar size, namely \( N \leq 12 \). In contrast, Bouman et al. [2018] did not compare their computational results to those of other authors when solving instances of size \( N = 16 \). Given that our results were obtained using a computational platform comparable to those used
by the aforementioned authors, in what follows, we also report the results of only our formulation for values of $N \geq 16$.

### 2.5.2 Summary of Results

Table 2.2 summarizes the results of our study; detailed tables which provide information for each instance are available in Appendix A.3. In Table 2.2, beneath the header identifying the topography of the benchmark instances, we report: Opt., the number of instances solved optimally; B&B, the average number of branch-and-bound nodes explored by the solver; CPU (s), the average of the total computational effort of the multi-step solution procedure outlined in §2.4.2; and Gap, the average MIP gap reported by the solver at termination. The **Summary** row shows the total number of instances solved optimally, as well as the average of B&B, CPU (s), and Gap for all instances.

<table>
<thead>
<tr>
<th>$N$</th>
<th>SC</th>
<th>DC</th>
<th>U</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>10/10</td>
<td>7,594</td>
<td>24.3</td>
</tr>
<tr>
<td>20</td>
<td>10/10</td>
<td>651,610</td>
<td>1,221.0</td>
</tr>
<tr>
<td>24</td>
<td>7/10</td>
<td>3,990,267</td>
<td>10,650.2</td>
</tr>
<tr>
<td>28</td>
<td>1/10</td>
<td>1,802,544</td>
<td>10,794.6</td>
</tr>
<tr>
<td>32</td>
<td>0/10</td>
<td>880,703</td>
<td>11,146.3</td>
</tr>
<tr>
<td><strong>Summary</strong></td>
<td>28/50</td>
<td>1,406,544</td>
<td>6,767.3</td>
</tr>
</tbody>
</table>

Model **TSPD2** yields optimal solutions for instances having up to 24 nodes and occasionally for those having up to 32 nodes. Across the three instance topographies (SC, DC, and U), 58/60 instances of size $N \leq 20$ were solved to optimality. The formulation also showed success for instances of size $N = 24$ and limited success beyond, which we discuss further in §2.5.4. To the best of our knowledge, this contributes new optimal results to the literature for instances of this size and under the operational settings discussed in §2.2. Our results also demonstrate differences in the computational challenge associated with topographies: for the same instance size, the solver
consistently required more time (and typically terminated with wider gaps) for U instances, and to a lesser extent, the SC instances, as compared with DC instances.

2.5.3 Bound Improvement Assessment

As discussed in §2.4, generating high-quality upper and lower bounds early in the B&B/C process can tightly encapsulate the search for an optimal solution and thus aid convergence. Figure 2.4 below plots the quality of both upper- and lower-bounds produced during the multi-step solution procedure for DC instances with $N = 16$ to 32. More specifically, the WS line charts the quality of the warm start generated in Step 3 of the procedure delineated in §2.4.2; LP1 and LP2 represent the LP relaxations prior to Step 0, and after Step 0, respectively. That is, LP1 represents the LP relaxation of Model $\text{TSPD1}$, whereas LP2 represents that of Model $\text{TSPD2}$, after the cut-set constraints have been appended. In all cases, the vertical axis represents the deviation of the bound from the best-known solution for the instance. Note that the lower bounds are presented with negative deviations (i.e. the plotted deviations are $Z^{*}_{LB} - Z^{*} \over Z^{*}$, where $Z^{*}$ is the best-known solution for the instance, and $Z^{*}_{LB}$ is the relevant lower bound.).

The quality of the upper bounds used as a warm start in the exact solution step was relatively consistent across instances of different sizes and topographies. Across $N = 16$ and 20, the warm start generated in Step 3 was within 1% of the best-known upper bound generated by the exact solution procedure in Step 4 for 51/60 instances across the three topographies. Over the same set of instances, the maximum gap between a warm start and the best-known solution was 3.8%. This can be attributed to the progressive structure of the proposed multi-step solution procedure. As noted in §2.4.2, each step in this procedure generates information (e.g., a warm start or an improved lower bound) that facilitates a solution in both the succeeding steps of the procedure. Just as important, these results justify focused computational effort.
across multiple solution steps in order to generate improved bounds for the instance. For larger instances in our test-bed, investing a considerable amount of computational effort to generate high-quality bounds paid off with 17/30 instances solved optimally for $N = 24$ in the exact solution stage.

![Figure 2.4: Bound Quality Assessment, Average Deviations](image)

In contrast to the upper bounds, Figure 2.4 shows a consistent degradation in the quality of the formulation’s lower bounds. In general, as $N$ increases, the gaps associated with LP1 and LP2 tend to get larger. More specifically, however, Figure 2.4 illustrates that the LP2 gap is substantially tighter than that of the LP1 gap, and the degradation in the former is slower than the degradation in the latter, even as $N$ increases. This can be attributed to the dynamic generation of cut-set constraints outlined in §2.4.1.1, which are appended so as to maximize the improvement in the LP2 gap at minimal computational expense. That said, there is a decreasing return
in the difference between the LP1 and LP2 gaps for larger values of $N$. Although adding more cut-set constraints might have improved the LP2 Gap, doing so would also burden the solver with additional constraints in the exact solution step. In our experience, the parameters outlined in §2.4.1.1 provide the best computational results.

The confluence of strategies from §2.4, rather than any single strategy, enabled the success of our formulation. As an example, consider instance $SingleCenter−77−n50$ under $N = 24$. The warm start generated therein was only 0.3% sub-optimal as compared with the best-known solution. Nevertheless, the solver spent more than 10,000 CPU seconds improving the lower bound, starting at an LP2 Gap of 40.6% before terminating with an MIP Gap of 6.3%. Absent the strategies deployed to improve the LP1 Gap, the MIP Gap at termination would have been considerably larger. Taken together, the strategies outlined in §2.4 reduce the computational burden of the solver far more significantly than they do when considered separately.

### 2.5.4 Solving Instances of Size $N \geq 24$

Our results also demonstrate the difficulty of solving larger TSP-D instances to optimality. In spite of the strategies outlined in §2.4 and considerable computational effort, closing the MIP Gaps associated with $N \geq 24$ remains a challenge. For example, with a time limit of 10,000 CPU seconds in the exact solution stage, instance $SingleCenter−77−n50$ terminated with an MIP Gap of 6.3%. When allotted up to 50,000 seconds of computation time on our machine, the same instance terminated with an MIP Gap of 1.31%. As highlighted in §2.5.3, the warm starts generated in the solution procedure were generally of very high quality: for $N = 24$, there was an average gap of 3.2%, 1.1%, and 3.8% between the warm start and the best-known solution for the SC, DC, and U instances, respectively. In contrast, the average LP2 gaps at the start of the solver’s search for an optimal solution were typically much larger, averaging 39.2%, 35.3%, and 57.9% for the same class of instances. In our
experience, improving the lower bound in the course of the B&B/C algorithm was the solver’s principal challenge in obtaining optimal solutions.

To mitigate the deteriorating quality of the LP2 Gap, we introduced the following constraint for instances of $N \geq 24$:

$$\sum_{i \in V} \sum_{j \in V} t_{ij}x_{ij} + \sum_{k \in V} (e_k + w_k) \geq Z_{\text{RMIP}}. \quad (2.5a)$$

$Z_{\text{RMIP}}$ denotes the dual bound associated with $Z_{\text{warm}}$, the optimal solution to the relaxed MIP (RMIP) obtained in Step 2 of the multi-step solution procedure. We note firstly the validity of $Z_{\text{RMIP}}$ as a lower bound: Step 2 is a relaxed version of Model TSPD2 in which the $x$-variables remain binary, but the drone routing variables ($y, z$) are declared continuous. By virtue of Constraint (2.1b), the binariness of the $w$-variables is implied by the binariness of the $x$-variables. Note that both the $x$- and $w$-variables feature in the objective function (3.3a), such that their binariness produces lower bounds which are both valid and high-quality (in terms of their objective values). Thus the optimal solution to RMIP must have an objective value no higher than that of the exact version of the MIP in Step 4 (i.e., $Z_{\text{RMIP}}^* \leq Z^*$). It is important to note that the solution procedure we outlined may terminate with an MIP Gap as high as 5% for $Z_{\text{warm}}$; in other words, the procedure is guaranteed to find a feasible value of $Z_{\text{warm}}$, rather than the optimal value of $Z_{\text{RMIP}}$. Therefore, let $Z_{\text{RMIP}} = Z_{\text{warm}}(1 - g)$, where $g$ is the MIP Gap in Step 2. Introducing Constraint (2.5a) allows the solver to retain the lower bound information from Step 2. In the absence of this constraint, the solver would rely solely on LP2 as the best-known lower bound in Steps 3 and 4 of the solution procedure.

Even with the improvement from LP1 to LP2 as a result of the cut-set constraints, RMIP is a substantially tighter lower bound, as summarized in Figure 2.5. Moreover, as $N$ increases, the degradation in the quality of the RMIP Gap is much more tightly controlled as compared with the degradation in LP1 and LP2. As noted in §2.3,
the reliance of the objective function (3.3a) on the vehicle routing variables is the foundation for the strength of the RMIP lower bounds. For example, at $N = 28$, Figure 2.5 shows that both the LP1 and LP2 Gaps averaged nearly 40%, whereas the RMIP Gap was just under 20%. In fact, it is likely that the RMIP Gap is even tighter than what these results show. As discussed in §2.4.2, we limited the solver to 1,000 CPU seconds in the search for an optimal objective value in the RMIP step. In many cases, particularly for $N \geq 24$, the solver terminated its search due to exhausting that time limit, rather than solving for the optimal value of RMIP. With greater computational resources, the RMIP Gaps would be even tighter for large values of $N$.

Using a constraint to explicitly introduce a lower bound yielded encouraging results in our experience: absent Constraint (2.5a), our formulation failed to prove an optimal solution for any instance of $N = 24$. But with its inclusion, a few instances of size $N = 28$ and 32 were solved to optimality.

Figure 2.5: Lower Bound Degradation, Average Deviations
2.5.5 Sensitivity Analysis of Maximum Drone Flight Duration

Table 2.3 explores the impact of the drone’s flight capacity, $F$, on the objective value and computational time associated with the formulation. Starting with the baseline value of $F = 30$ reported previously in our computational study, $F$ is increased for instances with $N = 16$, the largest solved in the literature.

Table 2.3: Objective, Computational Effort Sensitivity to Flight Time for $N = 16$

<table>
<thead>
<tr>
<th>$F$ = 45 minutes</th>
<th>SC</th>
<th></th>
<th></th>
<th>DC</th>
<th></th>
<th></th>
<th>U</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg.</td>
<td>Z* (%)</td>
<td>CPU (%)</td>
<td>Z* (%)</td>
<td>CPU (%)</td>
<td>Z* (%)</td>
<td>CPU (%)</td>
<td></td>
</tr>
<tr>
<td>-8.2</td>
<td>1,979.3</td>
<td>-5.0</td>
<td>3,151.8</td>
<td>-11.5</td>
<td>1,859.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max</td>
<td>-20.9</td>
<td>13,023.6</td>
<td>-10.9</td>
<td>9,784.0</td>
<td>-24.0</td>
<td>10,573.5</td>
<td></td>
</tr>
<tr>
<td>Opt.</td>
<td>10/10</td>
<td>10/10</td>
<td>10/10</td>
<td>10/10</td>
<td>10/10</td>
<td>10/10</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$F$ = 60 minutes</th>
<th>SC</th>
<th></th>
<th></th>
<th>DC</th>
<th></th>
<th></th>
<th>U</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg.</td>
<td>Z* (%)</td>
<td>CPU (%)</td>
<td>Z* (%)</td>
<td>CPU (%)</td>
<td>Z* (%)</td>
<td>CPU (%)</td>
<td></td>
</tr>
<tr>
<td>-16.7</td>
<td>30,300.1</td>
<td>-8.0</td>
<td>4,583.6</td>
<td>-16.5</td>
<td>11,232.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max</td>
<td>-31.6</td>
<td>154,572.3</td>
<td>-13.8</td>
<td>16,500.8</td>
<td>-27.0</td>
<td>38,188.8</td>
<td></td>
</tr>
<tr>
<td>Opt.</td>
<td>10/10</td>
<td>10/10</td>
<td>8/10</td>
<td>8/10</td>
<td>8/10</td>
<td>8/10</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$F$ = Unlimited</th>
<th>SC</th>
<th></th>
<th></th>
<th>DC</th>
<th></th>
<th></th>
<th>U</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg.</td>
<td>Z* (%)</td>
<td>CPU (%)</td>
<td>Z* (%)</td>
<td>CPU (%)</td>
<td>Z* (%)</td>
<td>CPU (%)</td>
<td></td>
</tr>
<tr>
<td>-37.6</td>
<td>68,171.3</td>
<td>-25.5</td>
<td>79,887.9</td>
<td>-16.4</td>
<td>19,105.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max</td>
<td>-45.2</td>
<td>154,743.5</td>
<td>-35.8</td>
<td>238,169.8</td>
<td>-27.6</td>
<td>61,565.1</td>
<td></td>
</tr>
<tr>
<td>Opt.</td>
<td>0/10</td>
<td>3/10</td>
<td>0/10</td>
<td>0/10</td>
<td>0/10</td>
<td>0/10</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.3 indicates the trade-off between the logistical benefits and considerable computational expense of increasing the value of $F$. Using a value of $F = 45$ rather than $F = 30$ minutes increased the computational effort from one or two minutes to over half an hour on average across topographies, while the improvement in objective values of the instances averaged $5 - 11.5\%$. Letting $F = 60$ minutes, an increase of 100\% over the baseline flight time, yielded practically prohibitive computational effort (of several hours on average) for a modest improvement of $8 - 16.5\%$ in objective values. Under unlimited flight time settings, our formulation produced results in line with those reported in Agatz et al. [2018]. Specifically, Agatz et al. [2018] summarized their heuristic results of $N = 20, 50$ and $100$ under unlimited flight time, reporting on the deviation between the heuristic TSP-D solutions and an optimal TSP solution for the same $U$ instances of about 30\% on average. Under the same operational settings
for U instances of size \( N = 16 \), our best-known TSP-D solutions had an average improvement of 48.0\% over the optimal TSP solutions.

The increased computational burden associated with larger values of \( F \) in our formulation is due fundamentally to the deterioration of lower bounds. Larger values of \( F \) typically render increasingly weak LP relaxations of the formulation. The RMIP Gaps (the tightest lower bounds as established in §2.5.3) for U instances averaged: 11.6\%; 18.4\%; 25.9\%; and 28.2\%, for \( F = 30, 45, 60 \) minutes, and unlimited flight time, respectively. Nevertheless, the proposed solution approach yielded high-quality upper bounds. For U instances, the warm start generated in Step 3 of the solution procedure was within 2\% of the best-known solution for: 10/10; 9/10; 8/10; and 7/10 instances (in order of increasing \( F \) values). These results were achieved under the same computational time limits described in §2.4.2, which limit the exact solution step to 10,000 CPU seconds. With more time or better computational resources, it may be possible to solve more instances optimally.

2.6 Conclusion

Beginning with a novel MIP formulation for the TSP-D, we employed cut generation and bound improvement strategies in order to enhance our model. Our results demonstrate a computational advantage to the proposed approach over alternative MIP formulations in the literature. Our computational study reports provably optimal solutions for instances involving up to 24 nodes and encouraging breaches for instances having 28 and 32 nodes, thereby expanding the range of solved instances in the existing literature. In particular, our approach brings to bear for the TSP-D several techniques of integer programming which have been successful for the TSP. The modeling strategies deployed in this setting point to the pertinence of optimized delivery services via vehicle and drone under manageable computational effort. Our formulation also exhibited computational limitations for larger instances under less
constrained maximum drone flight time values which exceed the current capability of drone delivery systems. Despite some encouraging results, relaxing the flight time parameter results in a weakening of the continuous relaxation of the proposed model and impedes the overall branch-and-cut solution effort for this challenging optimization problem. Although some instances remain unsolved from an exact point of view, the proposed model and enhancement techniques can be judiciously employed to yield useful bounds for optimization-based heuristic approaches in future research. We anticipate that further exact and heuristic strategies, involving MIP constructs, dynamic programming, decomposition techniques, and sophisticated enumeration techniques, will continue to advance the computational solvability of the TSP-D under various operational settings. The latter may include networks with multiple depots, a fleet of vehicles, or the possibility for a vehicle to operate several drones, which we recommend for future research. It may be also interesting to analyze in the future the logistical impact of having ineligible customers for drone service, based on their geographical location, the weight-bulk ratio of the products ordered, or simply if they have not subscribed to this service.
CHAPTER 3
AN OPTIMIZATION-BASED HEURISTIC FOR ASSESSING CYCLIC DRONE FLIGHTS ON JOINT VEHICLE-DRONE PARCEL DELIVERY

We propose a variable neighborhood search (VNS) heuristic for the Traveling Salesman Problem with Drone (TSP-D) which allows cyclic drone flights. Starting from restricted problem assumptions, the heuristic uses two distinct mixed-integer programming formulations to progressively permit synchronous travel between the vehicle and drone as well as cyclic flights. By termination, the heuristic restores the full problem structure to yield high-quality feasible solutions. On a set of benchmark instances from the literature, the VNS improves upon the best-known results for 113/120 instances having up to 100 nodes, with comparable computational effort to existing approaches. We also propose pre-processing techniques to simplify the solution space associated with drone cycles, and investigate the conditions under which such cycles are optimal. Using the pre-processing procedures, the VNS yields solutions featuring drone cycles over roughly the same computational effort required for solving instances where cycles are disallowed. Furthermore, permitting drone cycles yields an average improvement of about 5% in both solution quality and the number of customers served via drone.

3.1 Introduction

To facilitate commercial drone deliveries, decision support systems are required. Specifically, decision-makers may benefit from tools that solve industrial-scale instances of the TSP-D quickly and efficaciously. The literature spans a variety of
approaches that have solved small instances with up to 20 nodes optimally [El-Adle et al., 2021, Schermer et al., 2020], with heuristic solutions for instances having up to 100 nodes [Agatz et al., 2018, Es Yurek and Ozmutlu, 2018, Schermer et al., 2019]. In contrast with classical routing problems, certain conditions in drone-and-vehicle routing problems may lead to a drone returning to its launching location after a delivery, thereby forming a cycle, before visiting the next customer location. This feature, a *drone cycle*, remains under-investigated [Agatz et al., 2018, Schermer et al., 2019].

To that end, this chapter makes the following contributions. From a modeling perspective, we introduce two mixed-integer programming (MIP) formulations for the TSP-D that respectively incorporate the drone cycles with or without limiting the vehicle’s intervening deliveries during the drone’s flight. Methodologically, we propose a VNS to exploit the complementary strengths of the MIPs: firstly, the heuristic employs a restricted formulation that limits the number of vehicle stops during a drone flight to make quick progress towards a good quality solution. The latter is then passed to the second MIP in order intensify the search for better local optima that restore this operational assumption. Computationally, the quality and speed of the VNS are demonstrated on benchmark TSP-D instances, in which the heuristic yields best-known solutions to 113/120 instances. Finally, the VNS provides a platform to assess the impact of drone cycles on solution quality, computational effort, and the sensitivity of those results to key input parameters, such as drone speed and flight range. This is timely, because there is no consensus in the literature as to whether allowing drone cycles bears significant impacts on the quality of solutions and the associated computational impact on solution techniques [Schermer et al., 2019].

The remainder of the chapter is organized as follows: §3.2 reviews related literature; §3.3 introduces the problem statement, the two MIP formulations underpinning the VNS, and a lemma on the conditions under which drone cycles may be featured in optimal solutions; §3.4 details the VNS approach; §3.5 presents benchmark compu-
tational results, including a sensitivity analysis for drone cycles; and §3.6 concludes this chapter with a summary of findings and related directions for future research.

3.2 Literature Review

Recent reviews [Macrina et al., 2020, Li et al., 2021] have highlighted the variety of novel features for the integration of drone technology. The vehicle routing problem with drones (VRP-D) features multiple capacitated vehicles (each of which may have one or more drones) and has been solved using exact [Wang and Sheu, 2019] and heuristic approaches [Schermer et al., 2019, Sacramento et al., 2019, Chen et al., 2021]. The TSP-D, the single-vehicle variant of the VRP-D, has also been investigated using both exact and heuristic optimization-based approaches, for which the interested reader may peruse the aforementioned reviews. This section focuses on heuristic approaches for the TSP-D with a single drone.

3.2.1 MIP-Based Heuristic Approaches

Murray and Chu [2015] introduced the Flying Sidekick Traveling Salesman Problem (matching the settings of the TSP-D), for which the authors developed an exact MIP formulation and a variety of heuristic strategies based on using vehicle-only tours to derive TSP-D solutions. The authors reported optimal solutions for instances with up to 9 nodes, and heuristic solutions for instances with up to 10 nodes.

Agatz et al. [2018] used carrier “operations” that combine complementary tours for the vehicle and drone, with at most one delivery by drone, and the remaining nodes served by the vehicle. An MIP formulation then sequences a combination of operations to serve all customers. In addition to optimal solutions for instances with up to 10 nodes, the authors also presented several route-first, cluster-second heuristic approaches. The most successful such strategy involved a dynamic programming (DP) algorithm that begins with an optimal vehicle-only tour, then assigns a subset
of customers to be visited by the drone using exact partitioning. The authors also analyzed heuristic solutions for instances having up to 100 nodes.

Boysen et al. [2018] present several MIPs in which the vehicle’s path is given, with one or more drones being launched to serve nearby customers at each stop. The authors also investigate problem settings in which the drone’s launching and landing locations are the same (i.e. a drone cycle), or when they may be limited (e.g. with only one, rather than an unrestricted, number of intervening vehicle visits). Notably, the authors do not combine both drone cycles and multiple intervening vehicle visits during a drone flight in the same problem variant. Solving instances with up to 100 nodes, the authors conclude that limiting the vehicle’s intervening visits during a drone flight is most effective when customers are clustered closely.

Es Yurek and Ozmutlu [2018] used a two-stage approach: first, TSP tours are generated by DP; then an MIP is used to generate complementary drone flights. For each partial vehicle-only tour generated in the first stage, a drone tour visiting the remaining nodes is generated in the second stage. The algorithm solves instances having up to 12 nodes optimally, and instances having up to 20 nodes heuristically.

Schermer et al. [2019] introduced both an exact MIP formulation and a heuristic approach for the VRP-D. After enhancing the formulation with a series of valid inequalities, the authors compare its performance with a matheuristic which partitions the VRP-D into sub-problems. The first set of sub-problems assign customers to a particular vehicle-drone tandem; the latter sub-problems, termed the drone assignment and scheduling problem (DASP) by the authors, find an optimal path for a single vehicle and its companion drone(s) through a set of customers. The DASP does not assume a given vehicle route, but instead requires the vehicle and drone(s) to be routed simultaneously. As such, a single-vehicle DASP with a single drone is equivalent to the TSP-D. For large instances having up to 100 nodes, for which solving the DASP was intractable, the authors proposed a scheme wherein a starting
feasible solution may be “partitioned” into several overlapping sections, each of which may be individually optimized.

3.2.2 Other Optimization-Based Approaches

Bouman et al. [2018] relied on the notion of “operations” for the drone and vehicle, whereby a node visited by both carriers bookends any intervening deliveries, to motivate a 3-pass DP approach. Solutions are built sequentially: first, the DP generates a TSP tour; then drone and vehicle operations are interwoven; and finally, those operations are optimally sequenced to span all nodes. The authors use a parameter, \( k \), to limit the number of nodes that may be visited by the vehicle while the drone is in flight. When \( k = \infty \), such that the vehicle’s tour is unrestricted, instances with up to 16 nodes were solved optimally. By scaling \( k \), the study explored the sensitivity of (heuristic) solutions.

Poikonen et al. [2019] developed a branch-and-bound (B&B) algorithm, with each B&B node being associated with a tour sequence. Starting at the root node, the farthest location from a customer already visited in the given sequence is chosen to be visited next. In the next level of the B&B tree, the children nodes represent a full enumeration of sequences that visit the new location. For each child node, the authors deployed an exact partitioning (EP) technique to assign locations to each carrier. The EP of a complete tour forms a valid upper bound, while the EP of a partial tour generally yields a lower bound. The study considered instances having up to 60 nodes.

The VNS proposed in this chapter extends the “partitioning” scheme of Schermer et al. [2019] to select variable neighborhoods within a given solution for re-optimization. The proposed approach departs from the existing literature in two ways: firstly, by using two distinct MIP formulations that begin by restricting key problem assumptions before progressively restoring them; and secondly through a
diversification scheme that iteratively visits distinct neighborhoods of different sizes to escape local optima.

3.3 Problem Statement & MIP Formulations

The TSP-D may be described as the problem of routing a single ground vehicle and its companion drone on a tour that serves each customer in a network exactly once, beginning and ending at a depot. Hereafter, the drone and vehicle are referred to generically as carriers unless an explicit reference is required. The objective of the TSP-D is to minimize the return time of both carriers to the depot. While the vehicle serves customers, the drone may be launched to serve customers simultaneously. Generally, the drone travels faster than the vehicle. The drone’s flight time and its carrying capacity are limited, such that after each delivery the drone must rendezvous with the vehicle. The drone is launched or collected at a demand node. The two carriers may meet and await one another at any node, in which case the node is called a rendezvous node. Should the drone land at the depot, it may not be launched again. After a rendezvous, the drone may be carried aboard the vehicle. All customers may be served by either carrier. Additionally, the drone may launch from a node $i$, serve a distinct node $j$, and then return to $i$, which is termed a drone cycle. In that drone cycle, the vehicle awaits the drone at $i$. Throughout, we make the following assumptions which are commonly adopted in the literature [Bouman et al., 2018, Schermer et al., 2019]:

1. The drone travels faster than the vehicle;

2. Flight times for the drone and vehicle travel times are symmetric, and follow the triangle inequality.
3.3.1 Restricted MIP Formulation

Before introducing the MIP formulation, this section outlines elements of the problem statement that are restricted. The restrictions are motivated by the idea that high-quality TSP-D solutions are likely maximize the parallelization of simultaneous vehicle and drone deliveries. Thus navigating shorter inter-node distances allows the two carriers to rendezvous more frequently, rather than requiring that one wait for the other.

1. **Network Sparsity**: for each node $i$ in a given network with $n$ nodes, let $\phi_i$ be a set comprising the nearest $\lceil \Phi n \rceil$ nodes to $i$, where $\Phi$ represents a specified ratio of the network’s edges to be retained. Setting $\Phi < 1$ and fixing to 0 the values of routing variables that connect $i$ to nodes outside of $\phi_i$ consequently reduces the solution space.

2. **Intervening Vehicle Visits**: During a drone flight, the problem statement allows the vehicle to serve any number of customers. As discussed in Section 2, both Bouman et al. [2018] and Boysen et al. [2018] limited intervening vehicle deliveries during drone flights to mitigate the solution effort and yield high-quality solutions. In our proposed restricted formulation, the vehicle may not deliver parcels during a drone flight. Specifically, the relevant constraints ensure that a drone flight launching from node $i$, serving node $k$, and landing at node $j$ is only feasible if the vehicle travels from node $i$ to $j$ directly.

3. **Disallowing Drone Multi-cycles**: Where feasible, multiple drone cycles may be launched from the same node, which are termed *multi-cycles* hereafter. The restricted formulation, however, permits only one drone cycle to launch from an eligible rendezvous node.

Combining the restrictions discussed above not only simplifies the vehicle routing decisions, but also by extension reduces the number of feasible drone flights. Note
that the restricted formulation’s solutions are always feasible to a TSP-D instance, even if they may not be optimal.

### 3.3.1.1 Formulation

The following notation is used to introduce the proposed restricted, starting MIP formulation for the TSP-D:

**Input Parameters**

- \( \hat{V} = \{0, 1, \ldots, n, n + 1\} \): Set of service nodes in the network.
- \( \hat{V}^* = \hat{V}\backslash\{0, n + 1\} \): Set of customer nodes, which excludes the central depot (represented by dummy nodes 0 and \( n + 1 \)).
- \( \tau_{ij} \): Vehicle travel time from \( i \) to \( j \), \( \forall \ i, j \in \hat{V}, i \neq j \).
- \( f_{ij} \): Drone travel time from \( i \) to \( j \), \( \forall \ i, j \in \hat{V}, i \neq j \).
- \( F \): Maximum flight duration for the drone to make a single delivery then return to the vehicle.
- \( \tilde{M}_{ij} \): A “large enough” scalar \( \forall \ i, j \in \hat{V}, i \neq j \).
- \( C_{\text{max}} \): An upper bound on the optimal objective value.
- \( \Phi \): A ratio of nearest neighbors considered for routing variables.
- \( \tau_{\text{max}}^i \): The value of \( \tau_{ij} \) such that \( j \) is the \( \lfloor \Phi n \rfloor \) nearest node to \( \forall \ i, j \in \hat{V}, i \neq j \).
- \( \phi_i = \{ j \in \hat{V} | \tau_{ij} \leq \tau_{\text{max}}^i \} \): A sparsified set of nodes neighboring \( i \in \hat{V} \).
- \( E_k = \{(i, j) \in \hat{V} | f_{ik} + f_{kj} \leq F, \forall i \in (\hat{V}\backslash\{n + 1\}), j \in \phi_i, k \in \hat{V}^* \} \): All pairs of rendezvous nodes permitting drone delivery to customer \( k \), with distinct \( i, j, k \).
- \( D = \{ j \in \hat{V}^* | |E_j| > 0 \} \): Set of nodes eligible for drone service.
• $\hat{C}_i = \{j \in \hat{V}^*: f_{ij} + f_{ji} \leq F\}$: Set of nodes which may be served via drone cycle commencing at node $i \forall j \in \hat{V}^*$.

• $\hat{f}_{ikj} = \max(0, f_{ij} + f_{jk} - \tau_{ij})$: The vehicle’s waiting time for a drone flight originating from $i$, delivering to $k$, and landing at $j$, $\forall i, j \in \hat{V}, k \in D$, with distinct $i, j, k$.

**Decision Variables**

• $\hat{x}_{ij} \in \{0, 1\}$: $\hat{x}_{ij} = 1$ $\iff$ the vehicle travels from $i$ to $j$, $\forall i \in \hat{V} \setminus \{n+1\}, j \in \phi_i$.

• $\hat{y}_{ikj} \in \{0, 1\}$: $\hat{y}_{ikj} = 1$ $\iff$ the drone launches from $i$, delivers to $k$, and lands at $j$, $\forall k \in D, (i, j) \in E_k$.

• $\hat{z}_{ij} \in \{0, 1\}$: $\hat{z}_{ij} = 1$ $\iff$ the drone commences a cycle from $i$, serves $j$, then flies back to $i$, $\forall i \in (\hat{V} \setminus \{n+1\}), j \in \hat{C}_i$.

• $\hat{\delta}_j \in [0, 1]$: $\hat{\delta}_j = 1$ $\iff$ the drone serves node $j$, $\forall j \in \hat{V}^*$.

• $b_j \in [0, C_{\text{max}}]$: The departure time of the vehicle at node $j$, $\forall j \in \hat{V}$.

A (restricted) formulation for the Traveling Salesman Problem with Drones is presented as follows, and denoted **Model TSPD2**:

**TSPD2:** Minimize $\sum_{i \in \hat{V} \setminus \{n+1\}} \sum_{j \in \phi_i} \tau_{ij} \hat{x}_{ij} + \sum_{i \in \hat{V} \setminus \{n+1\}} \sum_{k \in \hat{C}_i} (f_{ik} + f_{ki}) \hat{z}_{ik}$

$$+ \sum_{k \in D} \sum_{(i, j) \in E_k} \hat{f}_{ikj} \hat{y}_{ikj}$$

s.t. $\sum_{j \in \phi_0} \hat{x}_{0j} = 1,$ (3.1b)

$\sum_{j \in \hat{V}, j \in \phi_{n+1}} \hat{x}_{jn+1} = 1,$ (3.1c)

$\sum_{j \in \phi_k} \hat{x}_{kj} = 1 - \hat{\delta}_k, \forall k \in \hat{V}^*,$ (3.1d)
\[
\sum_{j \in \phi_i} \hat{x}_{ij} - \sum_{j \in \hat{V} \setminus \{n+1\} : i \in \phi_j} \hat{x}_{ji} = 0, \quad \forall i \in \hat{V}, \tag{3.1e}
\]

\[
\hat{y}_{ikj} \leq \hat{x}_{ij}, \quad \forall k \in D, (i, j) \in E_k, \tag{3.1f}
\]

\[
\sum_{k \in \hat{C}_i} \hat{z}_{ik} \leq 1 - \hat{\delta}_i, \quad \forall i \in \hat{V} : |\hat{C}_i| \geq 1, \tag{3.1g}
\]

\[
\sum_{k \in D : (i, j) \in E_k} \hat{y}_{ikj} \leq 1, \quad \forall i \in \hat{V} \setminus \{n+1\}, j \in \phi_i \tag{3.1h}
\]

\[
\sum_{(i, j) \in E_k} \hat{y}_{ikj} + \sum_{i \in \hat{V} \setminus \{n+1\}} \hat{x}_{ik} + \sum_{i \in \hat{C}_k} \hat{z}_{ik} = 1, \quad \forall k \in D : |\hat{C}_k| \geq 1, \tag{3.1i}
\]

\[
\sum_{i \in \hat{V} \setminus \{n+1\}} \hat{x}_{ik} + \sum_{i \in \hat{C}_k} \hat{z}_{ik} = 1, \quad \forall k \in \hat{V} \setminus D : |\hat{C}_k| \geq 1, \tag{3.1j}
\]

\[
\sum_{i \in \hat{V} \setminus \{n+1\}} \hat{x}_{ik} = 1, \quad \forall k \in \hat{V} \setminus D : |\hat{C}_k| = 0, \tag{3.1k}
\]

\[
b_j \geq b_i + \tau_{ij} + \sum_{k \in D : (i, j) \in E_k} \hat{f}_{ikj} \hat{y}_{ikj} + \sum_{k \in \hat{C}_i} (f_{ik} + f_{ki}) \hat{z}_{ik} - \hat{M}_{ij} (1 - \hat{x}_{ij}), \quad \forall i \in \hat{V} \setminus \{n+1\}, \forall j \in \phi_i, \tag{3.1l}
\]

\[
b_{n+1} \leq \sum_{i \in \hat{V} \setminus \{n+1\}} \sum_{j \in \phi_i} \tau_{ij} \hat{x}_{ij} + \sum_{i \in \hat{V} \setminus \{n+1\}} \sum_{k \in \hat{C}_i} (f_{ik} + f_{ki}) \hat{z}_{ik} + \sum_{k \in D : (i, j) \in E_k} \hat{f}_{ikj} \hat{y}_{ikj}, \tag{3.1m}
\]

\[b, \hat{\delta} \geq 0, \quad \hat{x}, \hat{y}, \hat{z} \text{ binary.}\]
The objective function (3.1a) minimizes the return time to the depot for all carriers across three relevant routing decisions: the summation of $\hat{x}$-variables minimizes the travel time of the vehicle; and the summation of $\hat{z}$-variables and $\hat{y}$-variables account for the vehicle’s idle time during a drone cycle and non-cyclic drone deliveries, respectively. Since all routing variables relating to $i \in \hat{V}$ are defined over sets $\phi_i$, setting $\Phi = 1$ exhaustively populates $\phi_i$ to ensure that all routing decisions relating to node $i$ remain feasible. But setting $\Phi < 1$ restricts the model and reduces the solution space, as discussed in §3.3.1. Constraints (3.1b) and (3.1c) ensure the vehicle’s path begins and ends at the depot (represented as nodes 0 and $n+1$, respectively), while Constraint (3.1e) ensures a Hamiltonian path through the intervening customer nodes. Constraint (3.1d) ensures that each customer must be served, while Constraints (3.1i) - (3.1k) ensure that each customer is served by either via vehicle delivery, a drone flight, or a drone cycle. Although every node may be served by the vehicle, the constraints limit the appropriate set of drone routing decisions: Constraint (3.1i) considers nodes that are eligible for both drone flights and drone cycles; Constraint (3.1j) considers only nodes that may be served by drone cycle; and Constraint (3.1k) considers nodes that may not be served by the drone at all. Constraint (3.1g) ensures the vehicle must visit a node from whence a cycle commences and Constraint (3.1h) allows at most one non-cyclic drone flight to launch from each rendezvous node. Constraint (3.1f) restricts the vehicle from intervening deliveries during a drone flight.

At each node, Constraint (3.1l) sets the departure time of the vehicle, with a secondary purpose of a subtour elimination constraint of the Miller-Tucker-Zemlin (MTZ) type. Since formulations with MTZ-type constraints are known to yield relatively weak continuous relaxations, the $\hat{M}$ values may be set as follows for tightness:
\[ M_{ij} = \begin{cases} 
0, & i = 0, j = n + 1, \\
2\tau_{n+1j}, & i = n + 1, \\
C_{\text{max}}, & j = 0, \\
C_{\text{max}} - \tau_{in+1} + \tau_{ij} - \tau_{0j}, & \text{otherwise.} 
\end{cases} \]

Constraint (3.1m) provides an upper bound for the value of the arrival time at the depot (which is represented by the departure time at the artificial node \(n + 1\)), since the latest possible departure time from any node must be less than the makespan of the entire tour.

### 3.3.2 Conditions for Drone Cycles

This section identifies conditions that must exist in an optimal solution featuring drone cycles:

**Lemma 1.** Given unlimited flight range, if there is a drone cycle from node \(i\) to node \(k\), then there must be one non-cyclic drone flight landing at \(i\) and one non-cyclic drone flight departing from \(i\).

**Proof:** See Appendix B.1.

**Remark.** In contrast, under limited flight range, an optimal solution may include a drone cycle from node \(i\) to node \(k\) absent a non-cyclic drone flight landing at \(i\). This occurs whenever a node \(k\) may not be served through a drone flight that departs from a predecessor of \(i\) and lands at \(i\). Similarly, an optimal solution may include a drone cycle from \(i\) to \(k\) absent a non-cyclic drone flight departing from \(i\). This latter case occurs whenever \(k\) may not be served via a drone flight departing \(i\) and landing at a successor of \(i\).

Based on Lemma 1, the following valid inequality may be appended to Model TSPD2 whenever the drone’s flight time is unlimited to ensure that a node \(k\) may
only be served via drone cycle from node $j$ if and only if $j$ has at least two incident non-cyclic drone flights:

$$2(\sum_{k \in \tilde{C}_j} \hat{z}_{jk}) \leq \sum_{k \in D} \sum_{(j,i) \in E_k} \hat{y}_{jki} + \sum_{k \in D} \sum_{(i,j) \in E_k} \hat{y}_{ikj}, \quad \forall j \in \tilde{V}^*$$

(3.2a)

### 3.3.3 Unrestricted MIP Formulation

This subsection proposes a second MIP, hereafter referenced as **Model TSPD3**, for the general TSP-D featuring drone cycles and no restrictions on the number of nodes that may be visited by the vehicle while the drone is in flight. Indeed, all of the restrictions discussed in §3.2 for Model TSPD2 are relaxed, thereby restoring the full problem setting for Model TSPD3. The latter formulation extends and generalizes one proposed in Chapter 2 to cater for the inclusion of drone cycles.

#### Input Parameters

- $V = \{0, 1, \ldots, n\}$: Set of service nodes in the network.
- $V^* = V \setminus \{0\}$: Set of customer nodes, which excludes the central depot (represented by node 0).
- $M_1, \ldots, M_6$: A series of “large enough” scalars.
- $C_i = \{j \in V^* | f_{ij} + f_{ji} \leq F\}$: Set of nodes that may be served via drone cycle commencing at node $i$, visiting $j$, then returning to $i$, $\forall i \in V^*$.

#### Decision Variables

- $x_{ij} \in \{0, 1\}$: $x_{ij} = 1 \iff$ the vehicle travels from $i$ to $j$, $\forall i, j \in V, i \neq j$.
- $y_{ij} \in \{0, 1\}$: $y_{ij} = 1 \iff$ the drone travels from $i$ to $j$, $\forall i, j \in V, i \neq j$. 
\( z_{ij} \in [0, 1] \): \( z_{ij} = 1 \iff \) the drone flies from \( i \) to \( j \), \( \forall \ i, j \in V, i \neq j \). Note that if the drone is carried aboard the vehicle, \( y_{ij} = 1 \) but \( z_{ij} = 0 \).

\( r_{ij} \in \{0, 1\} \): \( r_{ij} = 1 \iff \) the drone commences a cycle from \( i \), serves \( j \), then flies back to \( i \) \( \forall \ i \in V, j \in C_i \).

\( \delta_j \in [0, 1] \): \( \delta_j = 0 \iff \) the vehicle visits node \( j \), \( \forall j \in V \).

\( a_j \in [0, M] \): The arrival time of the delivery carrier at node \( j \), \( \forall j \in V \).

\( d_j \in [0, M] \): The departure time of the delivery carrier at node \( j \), \( \forall j \in V \).

\( e_j \in [0, f_{j}^{max}] \): The time spent by the vehicle waiting for the drone at node \( j \), \( \forall j \in V \), where \( f_{j}^{max} = \max\{\max_{i,k \in V|f_{ik}+f_{kj}\leq F}\{f_{ik} + f_{kj} - \tau_{ij}\}, 0\} \).

\( g_d \in [0, f_{0}^{max}] \): The drone’s arrival time at the depot.

\( g_v \in [0, f_{0}^{max}] \): The vehicle’s arrival time at the depot.

Model \textbf{TSPD3} is presented as follows:

\textbf{TSPD3:}\ Minimize \[ \sum_{i \in V} \sum_{j \in V} \tau_{ij} x_{ij} + \sum_{k \in V} e_k \] \hspace{1cm} (3.3a)

s.t. \[ \sum_{j \in V^*} x_{0j} = 1, \] \hspace{1cm} (3.3b)

\[ \sum_{j \in V} x_{ij} = 1 - \delta_i, \quad \forall i \in V, \] \hspace{1cm} (3.3c)

\[ \sum_{j \in V} x_{ij} - \sum_{j \in V} x_{ji} = 0, \quad \forall i \in V, \] \hspace{1cm} (3.3d)

\[ \sum_{j \in V^*} y_{0j} = 1, \] \hspace{1cm} (3.3e)

\[ \sum_{j \in V} y_{ij} - \sum_{j \in V} y_{ji} = 0, \quad \forall i \in V, \] \hspace{1cm} (3.3f)

\[ \delta_j \leq \sum_{i \in V} y_{ij} + \sum_{i \in V : j \in C_i} r_{ij} \leq 1, \quad \forall j \in V^*, \] \hspace{1cm} (3.3g)
\[2 - \delta_i - \delta_j \geq y_{ij}, \quad \forall i, j \in V | i \neq j, \] (3.3h)

\[y_{ij} + 2 - \delta_i - \delta_j \leq x_{ij} + 2, \quad \forall i, j \in V | i \neq j, \] (3.3i)

\[y_{ij} - x_{ij} \leq z_{ij} \leq 1 - x_{ij}, \quad \forall i, j \in V | i \neq j, \] (3.3j)

\[z_{ij} \leq y_{ij}, \quad \forall i, j \in V | i \neq j, \] (3.3k)

\[r_{ij} \leq 1 - \delta_i, \quad \forall i \in V, j \in C_i, \] (3.3l)

\[r_{ij} \leq \delta_j, \quad \forall i \in V, j \in C_i, \] (3.3m)

\[\sum_{j \in C_i} r_{ij} \leq 1, \quad \forall i \in V, \] (3.3n)

\[|C_i| \sum_{k \in V} y_{ki} \geq \sum_{j \in C_i} r_{ij}, \quad \forall i \in V^* : |C_i| > 0, \] (3.3o)

\[\sum_{i \in V} f_{ij} z_{ij} + \sum_{k \in V} f_{ijk} z_{jk} \leq F + M^1_j (1 - \delta_j), \quad \forall j \in V^*, \] (3.3p)

\[a_j \geq d_i + \tau_{ij} - M^2_{ij} (1 - x_{ij}), \quad \forall i, j \in V | i \neq j, \] (3.3q)

\[d_j \geq a_j + \sum_{k \in C_j} (f_{jk} + f_{kj}) r_{jk}, \quad \forall j \in V^*, \] (3.3r)

\[d_0 \geq \sum_{k \in C_0} (f_{0k} + f_{k0}) r_{0k}, \quad (3.3s)\]

\[a_j \geq d_i + f_{ij} - M^3_{ij} (1 - z_{ij}), \quad \forall i, j \in V | i \neq j, \] (3.3t)

\[d_j \geq d_i + f_{ij} - M^4_{ij} (1 - z_{ij}), \quad \forall i \in V, j \in V^*, \] (3.3u)

\[e_j \geq d_j - (d_i + \tau_{ij}) - M^5_{ij} (1 - x_{ij}), \quad \forall i \in V, j \in V^* | i \neq j \] (3.3v)

\[e_j \leq d_j - (d_i + \tau_{ij}) + M^5_{ij} (1 - x_{ij}), \quad \forall i \in V, j \in V^* | i \neq j \] (3.3v)

\[g_v \geq d_i + \tau_{i0} - M (1 - x_{i0}), \quad \forall i \in V^*, \] (3.3x)

\[g_d \geq d_i + f_{i0} - M (1 - z_{i0}), \quad \forall i \in V^*, \] (3.3y)

\[e_0 \geq g_d - g_v, \quad (3.3z)\]

\[a_0 \leq \sum_{i \in V} \sum_{j \in V} t_{ij} x_{ij} + \sum_{k \in V} e_k, \quad (3.3)\]
\[
\sum_{k \in C_i} r_{ik} \leq (\sum_{j \in V} z_{ij})|C_i|, \quad \forall i \in V : |C_i| > 0, \quad (3.3)
\]

\[
\sum_{k \in C_i} r_{ik} \leq (\sum_{j \in V} z_{ji})|C_i|, \quad \forall i \in V : |C_i| > 0, \quad (3.3)
\]

\[
a, d, \delta, z, e, g_d, g_v \geq 0, \quad \delta \leq 1,
\]

\[
a, d \leq M, \quad e, g_d, g_v \leq f^{\text{max}}, \quad x, y, r \text{ binary.}
\]

The objective function (3.3a) minimizes the duration of the carriers’ tour, accounting for the vehicle’s travel time (through the summation of \(x\)-variables) and its waiting time (through the summation of \(e\)-variables). Constraints (3.3b) and (3.3d) ensure that the vehicle departs the depot for exactly one destination, and thereafter travels from exactly one predecessor node to exactly one successor node. Constraints (3.3e) and (3.3f) serve the same purpose for the drone, whereas Constraint (3.3c) sets \(\delta_i = 0\) for any node \(i\) visited by the vehicle. Constraint (3.3g) enforces either a cyclic or non-cyclic flight by the drone in the absence of a vehicle visit. More generally, if the drone travels from \(i\) to \(j\), Constraint (3.3h) ensures that at least one of \(i\) or \(j\) must be visited by the vehicle. Constraint (3.3i) requires that if the drone travels from \(i\) to \(j\) where both \(i\) and \(j\) are served by the vehicle, then the vehicle must also travel from \(i\) to \(j\) (i.e. the drone is carried aboard the vehicle). Constraint (3.3j) requires that the drone flies from \(i\) to \(j\) if it was not transported aboard the vehicle. Constraint (3.3k) synchronizes the values of the \(z\)- and \(y\)-variables in case of a drone flight. In case of a drone cycle from \(i\) to \(j\) to \(i\), Constraint (3.3l) ensures the vehicle visits \(i\) (\(\delta_i = 0\)) and Constraint (3.3m) assigns the drone to \(j\) (\(\delta_j = 1\)). Constraint (3.3n) allows each node to serve as a rendezvous point for no more than one drone cycle. Constraint (3.3p) guarantees that for a given flight, the drone can travel for at most a duration of \(F\) before rendezvousing with the vehicle. Additionally, Constraints (3.3q) through (3.3u) determine arrival and departure times for the carriers in the spirit of MTZ-type
subtour elimination constraints. Depending on which carrier serves a node, several of (3.3q) through (3.3u) set the arrival and departure times at the node. Tightened $M$ values improving the continuous relaxation of this formulation are provided in B.2. Due to these constraints, the $\delta$- and $z$-variables are guaranteed to be binary, given the binariness of the $x$-, $y$, and $r$-variables.

Two conditions determine the vehicle’s idling time: a drone cycle, which is accounted for in Constraints (3.3r) and (3.3s); and in case of arriving at a node before the drone has arrived. In the latter case, Constraint (3.3v) sets the vehicle waiting time equal to the difference of the actual and expected departure times (i.e. the departure time with no waiting associated) from a node. To ensure that waiting times are assigned to the correct nodes, Constraint (3.3w) enforces an upper bound for each such waiting time. In the special case of waiting time at the depot, Constraints (3.3x) and (3.3y) determine the arrival time of the vehicle and drone at the depot, respectively. If the drone’s arrival time exceeds that of the vehicle, then Constraint (3.3z) determines the non-negative value of the vehicle’s waiting time. Constraints (3.3) and (3.3) aggregate the number of incident non-cyclic drone flights at a node from which drone cycle commence to ensure that Lemma 1 is enforced; these constraints are only valid when the drone’s flight time is unlimited. Finally, Constraint (3.3) bounds the arrival time at the depot from above by the makespan of the carriers’ tour.

### 3.4 Description of VNS Approach

The proposed VNS is hereafter referred to as the Directed Improvement Procedure (DIP). Beginning with a feasible solution, DIP iteratively re-optimizes sub-sequences of customers. The sub-sequences begin and end at nodes where both carriers are present (e.g. when the drone is carried aboard the vehicle, or when the drone launches/lands), and are hereafter referenced as rendezvous chains. Although both phases of DIP share a general structure, they rely upon the distinct MIP formulations
introduced in Section 3. The following subsections present DIP’s procedure, the focus of each phase, the perturbation/diversification schemes, and the termination criteria therein.

3.4.1 DIP Procedure

DIP proceeds with the following procedure repeatedly until convergence:

Phase 1

1. Rendezvous Chain Selection: Designate the rendezvous chain, a sub-sequence of nodes in the solution at hand.

   - The rendezvous chain begins with the last node explored in the preceding chain. If no prior chains have been explored, the chain commences at the depot.

   - DIP also designates a search direction: nodes are selected in a clockwise (used in the first iteration) or counterclockwise sequence mirroring the movement of the carriers.

   - Proceeding in the designated direction, rendezvous nodes (where both carriers are present), along with any nodes from/to which they are connected by a carrier’s path, are appended to the chain until it comprises at least $L_{\text{min}}$ nodes.

   - Every chain must begin and end with a node visited by both carriers. Bookending the chains with these rendezvous nodes ensures that intervening routing decisions may be re-optimized freely.

   - Since rendezvous nodes and their connections are added together to the chain, the chain length may exceed $L_{\text{min}}$.

   - If a chain has length greater than $L_{\text{max}}$, however, then the last rendezvous node appended to the chain (and any connected nodes) are removed. This
is repeated until the chain length is less than or equal to $L_{\text{min}}$. As such, $L_{\text{min}}$ and $L_{\text{max}}$ serve as a “soft” floor and “hard” ceiling for chain length, respectively, to vary the size of neighborhoods considered by DIP.

2. **Re-optimization:** Invoke Model TSPD2 to re-optimize the chain.
   - For all nodes outside the chain, fix the associated routing/assignment variables to their values in the solution at hand.
   - Note that the variables associated with carrier arrival/departure and idling time may not be fixed, since they may be updated in the course of exploring a new solution.
   - If the incumbent solution is improved, deactivate any perturbation/diversification schemes and reset the non-improving iterations ($NII$) to 0.

3. **Rendezvous Chain Splicing:** Given the re-optimized solution, select the final rendezvous node from the previous iteration as the starting node for the next chain.
   - In the clockwise direction, the final rendezvous node has the latest departure time in the chain; in the counterclockwise direction, it has the earliest arrival time.
   - In this way, DIP varies the composition of neighborhoods selected the procedure.

4. **Diversification/Perturbation Schemes:** Given a specified $NII$ value, activate schemes (detailed in §4.4) to vary neighborhood size, search direction, and computational effort per iteration. Then return to Step 1. If the schemes are already active, proceed to Step 5.

5. **Phase Termination:** Conclude Phase 1 and proceed to Step 6.
Phase 2

6. **Phase Transition:** Return to Step 1, replacing Model TSPD2 with Model TSPD3 and terminating the entire procedure at Step 5.

![Figure 3.1: A step-by-step illustration of DIP.](image)

To illustrate the approach, Figure 3.1 demonstrates chain selection on a toy-sized instance. For each iteration, the lower sequence of nodes shows the vehicle’s path, with drone flights shown above that. The departure time at each node is also shown, culminating with the arrival time at the depot (node 0), which is the objective value of the solution. Starting at the depot and proceeding in the clockwise direction in iteration 1, chain $C = \{0, 1, 6, 2, 5\}$ is chosen for re-optimization (the corresponding nodes are darkened). In iteration 2, node 5 is chosen to start, with the chain $C = \{5, 7, 4, 3, 1\}$ selected for re-optimization. After two iterations, DIP improves the objective value from 490.6 to 380.1.
3.4.2 Phase 1: Rapid Improvement

As highlighted in §3.2, Model TSPD2 restricts several key assumptions from the problem statement which affect the initial phase of DIP. Given these restrictions, Phase 1 solutions may be generated quickly at the potential expense of quality. Phase 2 thus counterbalances these restrictions.

3.4.3 Phase 2: Deeper Exploration

Phase 2 re-explores the Phase 1 solution, with the goal of rectifying elements of the problem that were previously over-constrained. To exhaustively consider all vehicle and drone routing decisions possible under the problem settings from §3.3.1, Model TSPD2 sets all edges connecting nodes as feasible routes for the vehicle. Secondly, the $y$- and $z$-variables determining drone flights are decoupled from the $x$-variables determining the vehicle’s path. This allows the vehicle to make unlimited intervening deliveries during a drone flight. Thusly Model TSPD3 fully explores the solution space.

A secondary difference between the phases of DIP is the length of the rendezvous chains described in Section 3.4.1. Longer rendezvous chains generally hold greater potential for an improved solution at increased computational expense. Shorter chains may yield only modest improvements to the solution, thereby requiring a greater number of algorithmic iterations until a termination criterion is met. Due to the relaxed settings deployed during Phase 1, DIP is capable of re-optimizing relatively large chains in that phase. In contrast, the deeper exploration required in Phase 2 generally requires consideration of shorter chains. The ideal chain length for each phase allows significant improvements within manageable computational efforts per iteration, and may only be ascertained empirically.
3.4.4 Diversification/Perturbation Schemes & Termination Criteria

Although DIP explores an evolution of rendezvous chains, it may be entrapped at a local optimum. To mitigate this risk, DIP is designed to diversify the search for solutions. In restoring problem assumptions and expanding the solution space, the transition between phases serves as a perturbation scheme. Following the VNS strategy, DIP deploys three additional tactics to vary neighborhoods given successive non-improving iterations (NII):

- **Directions:** Once a solution has been scanned completely (i.e. every node in the network has been part of at least one rendezvous chain that was re-optimized), DIP reverses the direction of the search from clockwise to counter-clockwise, or vice-versa.

- **Expanded Search:** Let $L_{\text{min}}^e$ be a larger number of nodes to be explored in rendezvous chains, such that $L_{\text{min}}^e > L_{\text{max}}$ in each phase. Under the expanded search, $L_{\text{min}}^e$ becomes the floor for rendezvous chain length. Moreover, the “hard” ceiling of $L_{\text{max}}$ for chain length is disabled, so that the minimum chain length is at least $L_{\text{min}}^e$.

- **Early Iterations:** During both phases, the earliest iterations of DIP are most likely to yield improvements, since the solution space has yet to be explored. Therefore DIP permits extended computational effort for re-optimization until the solution has been scanned once in both the clockwise and counter-clockwise directions.

While diversification schemes aid in exploring the solution space, they are also computationally demanding, and thus must be deployed parsimoniously. As described in §3.4.1, whenever DIP yields an improved solution, NII is reset and any diversification schemes are de-activated immediately until DIP reaches again the specified NII.
threshold. In our computational experience, DIP performed best with $\Phi = \frac{2}{3}$ for the network sparsity setting and $NII = 2$.

### 3.5 Computational Study

Computational results were obtained on a machine with an Intel i7-7700K processor and 32 GB of RAM. DIP was implemented using AMPL, with both formulations solved using GUROBI version 9.0.2. DIP is initialized with a *Double Greedy* solution introduced by El-Adle et al. [2021], which routes the vehicle (and where possible, the drone in parallel) to the nearest neighbor yet to be visited.

#### 3.5.1 Problem Instances

Agatz et al. [2018] published benchmark instances developed specifically for the TSP-D; this study focuses on instances having a Uniform topography, where nodes are distributed randomly about a central depot. Details of the generation scheme are available in Agatz et al. [2018], and the best-known results to date are provided by Schermer et al. [2019]. The benchmark instances have 20, 50, and 100 nodes (each size having 10 distinct instances) with the following assumptions:

- All nodes are connected with symmetrical Euclidean travel distances;

- The drone’s speed is defined as a multiple of the vehicle’s speed. Unless otherwise noted, let $\alpha = 0.5$ be the ratio of the vehicle speed to that of the drone, such that the drone moves twice as fast as the vehicle;

- Following Agatz et al. [2018] and Schermer et al. [2019], the drone’s flight range, $F$, changes for each instance, where $F = e_{max} \cdot e_{r}$; $e_{max}$ represents the longest vehicle arc in the network, and $e_r \in \{0.2, 0.4, 0.6, 1\}$. Setting $e_r = 1$ extends the flight range such that any node may be served via drone cycle or a non-cyclic drone flight from any other node in the network.
Solutions to each instance including objective values and the sequence of customers visited by both the vehicle and drone are published publicly\(^1\).

### 3.5.2 Performance on Benchmark Instances

The following notation is used to assess the results:

- \(\Delta\%\): The percentage deviation of \(Z_{DIP}^*\), the solution produced by DIP, over \(Z_S^*\), the best solution produced by the approach of Schermer et al. [2019] (i.e. \(100\left(\frac{Z_{DIP}^*-Z_S^*}{Z_S^*}\right)\)).

- \(\Delta_\mu\): The mean of \(\Delta\%\) over 10 instances of the same size.

- \(\Delta_\sigma\): The standard deviation of \(\Delta\%\) over 10 instances of the same size.

- \(\text{Imp.}\): The count of instances for which \(Z_{DIP}^*\) improved upon \(Z_S^*\).

- \(\text{CPU (s)}\): The mean computational effort in CPU seconds over 10 instances of the same size.

The results in Table 3.1 indicate DIP yielded consistent improvements over the approach of Schermer et al. [2019] across instance size and values of \(E_r\). Overall, DIP improved upon the results of Schermer et al. [2019] for 113/120 instances, by an average of 11.9%, 10.8%, and 6.7% for instances having 20, 50, and 100 nodes, respectively. For instances having 20 and 50 nodes, DIP yielded improvements on 80/80 instances in the test-bed.

Across instance size, larger values of \(E_r\) generally yielded smaller improvements in solution quality as shown in Figure 3.2, which plots the differential percentage (\(\Delta\%\)) for each instance in the test-bed. For example, with \(n = 50\), the average value of \(\Delta\) was 20.9% under \(E_r = 0.2\), while the remaining values of \(E_r\) yielded average differentials no higher than 8%. Larger \(E_r\) values complicate the combinatorics of

\(^{1}\text{https://github.com/amro3333/TSPD-BAS-Instance-Solutions}\)
Table 3.1: Summary of DIP’s Performance on Benchmark Instances

<table>
<thead>
<tr>
<th>N</th>
<th>$\xi_r$</th>
<th>Imp.</th>
<th>$\Delta_{\nu}$ (%)</th>
<th>$\Delta_{\sigma}$ (%)</th>
<th>CPU (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.2</td>
<td>10/10</td>
<td>-12.6</td>
<td>2.9</td>
<td>122.2</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td>10/10</td>
<td>-18.6</td>
<td>5.3</td>
<td>153.8</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>10/10</td>
<td>-11.0</td>
<td>5.6</td>
<td>166.9</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>10/10</td>
<td>-5.7</td>
<td>3.2</td>
<td>175.9</td>
</tr>
<tr>
<td>Summary</td>
<td>40/40</td>
<td>-11.9</td>
<td>6.3</td>
<td>155.8</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>0.2</td>
<td>10/10</td>
<td>-21.4</td>
<td>3.7</td>
<td>382.5</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td>10/10</td>
<td>-7.9</td>
<td>1.7</td>
<td>476.2</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>10/10</td>
<td>-7.3</td>
<td>1.3</td>
<td>465.4</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>10/10</td>
<td>-6.8</td>
<td>2.4</td>
<td>564.6</td>
</tr>
<tr>
<td>Summary</td>
<td>40/40</td>
<td>-10.8</td>
<td>6.6</td>
<td>473.5</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>0.2</td>
<td>10/10</td>
<td>-11.4</td>
<td>6.4</td>
<td>511.8</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td>8/10</td>
<td>-5.4</td>
<td>6.9</td>
<td>528.4</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>7/10</td>
<td>-4.9</td>
<td>6.9</td>
<td>577.6</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>8/10</td>
<td>-5.1</td>
<td>6.9</td>
<td>489.4</td>
</tr>
<tr>
<td>Summary</td>
<td>33/40</td>
<td>-6.7</td>
<td>7.1</td>
<td>526.8</td>
<td></td>
</tr>
</tbody>
</table>

the problem: many more customers are within flying range of the drone, which may adversely affect the efficacy of DIP. Note the valid inequalities in each formulation based on Lemma 1 had a favorable impact on DIP’s performance: DIP would have improved upon only 26/30 instances under $\xi_r = 1$ and absent the inequalities, as compared with 28/30 instances improved upon with the inequalities active.

Although Table 3.1 shows $\Delta_{\sigma}$ values up to nearly 7%, Figure 3.2 highlights the consistency and efficacy of DIP, including on the 7/120 instances with positive $\Delta$ values. In many cases DIP yielded solutions that improved upon those of Schermer et al. [2019] by more than 10%, and occasionally by more than 20%. Just as important, DIP is deterministic: repeated runs of DIP yield identical solutions whereas the approach of Schermer et al. [2019] depends on randomized procedures that yield variable results. Note that $\Delta$ compares a unique DIP solution against the best of five different solutions reported by Schermer et al. [2019]. Comparing against the average (115/120) or worst-case (118/120) solutions from all the solutions reported by Schermer et al. [2019] shows greater improvements by DIP.
Finally, the computational effort associated with DIP was less than 550 CPU seconds on average across the test-bed, which aligns closely with related approaches in the literature, including Schermer et al. [2019], who limited their approach to 600 seconds. As shown in Figure 3.3, there is a fixed computational cost for exploring solutions in both stages of DIP. For example, on instances having 20 nodes, Phase 1 improves the solution differential from roughly 30% to -10% on the Δ scale in about 135 seconds on average. Thanks to this high-quality solution, DIP then quickly terminates Phase 2 within 25 seconds on average. Thus the more extensively Phase 1 explores the solution space, the more quickly Phase 2 may converge. This also explains the average computational effort of instances having 50 and 100 nodes, the former of which are within 50 seconds of the latter.

Chain lengths in each phase also play a role in determining the pathway to a solution. As discussed in Section 4.1, relatively long chains hold greater potential for
improving the solution, while relatively short chains may require more iterations to unlock potential improvements. To balance this trade-off, chains in Phase 1 had up to 30 nodes, and chains in Phase 2 had up to 12 nodes. These parameters were chosen in accordance with the motivation for each phase: rapid improvement of restricted solutions in Phase 1 and expansion the feasible space in Phase 2.

### 3.5.3 Sensitivity Analysis of Drone Cycles

The results discussed thus far allow multiple drone cycles at each node, a setting referenced hereafter as **Multi-Cycle (MC)**. In contrast, by fixing to 0 all the $\hat{z}$-variables and the $r$-variables in Phases 1 and 2, respectively, drone cycles may be disallowed in DIP, referenced as **No Cycle (NC)** solutions hereafter. As described in §3.2, the MIP formulations and thus DIP may be adapted to allow at most one drone cycle launching at any node, referenced hereafter as **Single-Cycle (SC)** solutions. Given unlimited flight range for the drone (i.e. $\mathcal{E}_r = 1$), Table 3.2 compares optimal solutions for instances having 12 nodes across these settings. The solutions were obtained using Model TSPD3; after providing baseline statistics for the NC setting
in the top row, the remaining rows compare the relative change in objective values and computational effort.

Table 3.2: Exact Solutions for \( n = 12 \) Using Model TSPD3

<table>
<thead>
<tr>
<th></th>
<th>( \alpha = 0.75 )</th>
<th>( \alpha = 0.50 )</th>
<th>( \alpha = 0.25 )</th>
<th>Droned Customers (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( Z^* )</td>
<td>CPU (s)</td>
<td>( Z^* )</td>
<td>CPU (s)</td>
</tr>
<tr>
<td>NC</td>
<td>262.3</td>
<td>102.0</td>
<td>235.0</td>
<td>71.4</td>
</tr>
<tr>
<td>SC</td>
<td>-0.1%</td>
<td>29.3%</td>
<td>0.0%</td>
<td>64.7%</td>
</tr>
<tr>
<td>MC</td>
<td>-0.1%</td>
<td>37.4%</td>
<td>0.0%</td>
<td>79.6%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Two key inputs drive the impact of drone cycles: drone flight range and network density. Note the former is a function of both flight time (which is unlimited) and drone speed (which varies) across Table 3.2. Across smaller \( \alpha \) values, even as the drone’s speed relative to the vehicle increases, \( \text{MC} \) improves the objective by no more than 1.1%. Still, the inclusion of drone cycles does increase the number of customers served via drone, even on a toy-sized network having 12 nodes. On larger instances (having 100 nodes) with denser customer networks, incorporating \textbf{Multi-Cycles} improved solutions by nearly 5% when compared with the \textbf{NC} setting (with \( \alpha \) being constant). Further decreasing \( \alpha \) reduces the vehicle idle time associated with drone cycles, and increases the occurrence of these cycles. For example, with \( \alpha = 0.5 \), there was an average of 0.5 drone cycles per instance under the \textbf{MC} for \( n = 50 \); at \( \alpha = 0.25 \), there were 1.5 drone cycles on average.

Additionally, across instances having up to 100 nodes, DIP yielded solutions for the \textbf{NC}/\textbf{SC}/\textbf{MC} settings over roughly the same computational effort, indicating the robustness of the proposed approach to the problem setting. Finally, note the comparable and occasionally superior MIP performance using the \textbf{SC} rather than \textbf{MC} settings. In our computational experience using DIP, multi-cycles were extremely rare. Even in dense networks with low \( \alpha \) values, our results indicate that although the objective values associated with \textbf{SC} solutions may significantly improve upon
those of NC solutions, further allowing MC as compared with SC settings provides a marginal improvement that may be computationally burdensome and hamper the solution process.

3.6 Conclusion

This chapter introduces a variable neighborhood search heuristic denoted DIP, underpinned by two distinct MIP formulations designed to generate high-quality solutions to the TSP-D. In Phase 1 of DIP, a restricted formulation limits routing variables by precluding the visit of intermediate customer nodes by the vehicle between the nodes where the drone is launched and recollected. In Phase 2, an unrestricted formulation relaxes this constraint from Phase 1 to yield a deeper, yet computationally more onerous, exploration of the feasible space. To avoid entrapment at local optima, the proposed methodology is further enhanced through the use of diversification schemes that vary the size of neighborhoods and the re-optimization effort expended therein. On a set of benchmark TSP-D instances, DIP yields the best-known solutions to 113/120 instances having up to 100 nodes, with comparable computational performance to existing approaches.

In contrast with classical routing problems that allow at most one visit to each customer, the TSP-D is complicated by cyclic drone flights, in which the drone may launch, serve a customer, then land at the same node from which it launched. Using optimal solutions for small instances and DIP to solve larger instances, this work provides evidence of improvements up to 5% attained by permitting drone cycles versus disallowing them. We hope this insight advances the literature on the TSP-D since the need to incorporate (single or multiple) drone cycles was not clearly settled in this fast-growing stream of research. In this regard, we demonstrate how to incorporate drone cycles in the two proposed MIPs, provide an effective solution methodology, and numerically demonstrate their impact, under two governing parameters: the drone
speed (relative to that of the vehicle) and the customer density in the instances under investigation.

We recommend for future research investigating the growing potential for drone delivery technology, including longer flights, varying drone speed during a delivery, as well as coordination between vehicle and drone fleets.

Acknowledgments

We extend our gratitude to Bouman et al. [2018] for releasing TSP-D instances and solutions publicly, and to Schermer et al. [2019] for the objective values associated with their approach.
CHAPTER 4
LAST-MILE DELIVERY WITH DRONE ELIGIBILITY

We investigate a last-mile delivery problem with drone eligibility in which customers place recurring orders fulfilled weekly, across single or multiple deliveries. A retailer uses a vehicle and its companion drone in tandem to fulfill orders, seeking to assign customers durably for delivery by vehicle or by drone and thereby to optimize the synchronized operations of both carriers throughout the entire planning horizon. The problem is amenable to a multi-period Traveling Salesman Problem with Drone Eligibility considerations for which we develop a mixed-integer program (MIP) formulation that yields provably optimal solutions for instances having up to 40 customers over a 6-period horizon. For large-scale instances with up to 200 customers over the same horizon, we devise a matheuristic that deploys elements of data analysis based on customers’ geographical locations and frequency of demand. By orchestrating neighborhood searches along these two features, the proposed heuristic effectively identifies in manageable times customers who are best served by drone, consistently yielding solutions that exhibit optimality gaps within 0.5% on average.

4.1 Introduction

In 2019, there were more than 100 billion parcels delivered globally, with more than 200 billion expected to be delivered annually by 2025 [Spadafora, 2020]. While the growth in e-commerce sales has vastly expanded the volume of parcels being delivered, the nature of deliveries has also changed. According to a 2020 article about operations at UPS, the shipping giant experienced a 4.4% decrease in revenue per
parcel even as it saw a 21% increase in daily delivery volume “…as the carriers incur greater expenses from more miles driven and fewer packages delivered per stop as they deliver more to homes instead of businesses” [Ziobro, 2020b]. In a May 2020 survey of US adults, nearly half reported their shopping preferences had shifted permanently online [Ramachandran, 2020]. These consumers cited the “…peace-of-mind and convenience of knowing they’d receive their goods shipped directly to their door” [Ramachandran, 2020]. The shift has been especially dramatic due to challenges wrought by the the Covid-19 pandemic, which simultaneously upended physical interactions at many retail locations, spurred significant growth in e-commerce sales, and shrunk the pool of laborers available to logistics providers [Smith, 2020]. This reality has required new resources for accommodating volume, consistency, and physical distancing as last-mile logistics providers devise distribution plans. More than 17 million new American households are expected to rely on last-mile fulfillment as a result of the pandemic [Lewis, 2020]. The rise of these challenges has renewed interest in technologies that may improve the outlook of last-mile deliveries, including the integration of unmanned aerial systems (UAS) or drones.

In a collaboration between federal and state governments as well as private firms, the US in 2020 granted waivers for drone deliveries of library books and cookies in northern Virginia, prescription drugs to a senior community in Florida, and Covid-19 tests in Las Vegas, Nevada [Elassar, 2020, Black, 2020, Lindner, 2020]. In a more extensive trial, drone service firm DroneUp conducted 90 deliveries in an environment replicating real-world conditions, including small designated drop-sites for parcels, obstacles to deliveries such as trees and powerlines, and a payload of 1.275 lbs [Fuller, 2020]. In the trial, a team of operators working an eight-hour shift were able to conduct up to 49 drone deliveries, each one mile in total distance [Fuller, 2020]. The operational problem of synchronizing the routes of a vehicle and its companion drone for delivery through a customer network has been addressed in the literature as the
Traveling Salesman Problem with Drone (TSP-D, e.g. Murray and Chu, 2015, Agatz et al., 2018, El-Adle et al., 2021). While these approaches shed light on the logistical benefits of joint vehicle-drone delivery, broader tactical questions remain about how evolving customer demand may impact drone delivery decisions.

This chapter introduces the Traveling Salesman Problem with Drone Eligibility (TSP-DE), which investigates a new notion of *carrier consistency* whereby a customer is served by either vehicle or drone over a particular time horizon or season. The durable assignments of the TSP-DE, based on the geographic dispersion and demand patterns of customers, may then be enforced (during a particular season and revised periodically as needed) to significantly reduce the complexity of the residual operational problem of routing the carriers in each period while ensuring service consistency for customers. This chapter contributes a novel mixed-integer programming (MIP) formulation for the TSP-DE and a matheuristic approach for generating high-quality solutions. The matheuristic is underpinned by the MIP formulation, and initialized by a feasible solution generated from a multi-knapsack formulation derived for the TSP-DE. Over a variety of benchmark instances, the efficacy of the matheuristic compares favorably against that of the MIP formulation, yielding solutions that consistently deviate from their optimal counterparts by less than 0.5% on average with substantial computational savings. The matheuristic explores different neighborhoods of the feasible space using two underlying features of customer data: geographical locations and demand patterns. This study suggests that approaches focusing on one feature but not the other leads to sub-optimal solutions, whereas jointly exploiting spatial and temporal features can more reliably lead to near-optimality. While many studies in the literature introduce MIPs for routing problems only to state their computational inadequacy/failure due to the combinatorics of the problem, our study subscribes to a different research paradigm: the use of optimization analytics to address the computational challenge posed by the TSP-DE.
The remainder of this chapter is organized as follows: §4.2 presents a review of related literature; §4.3 introduces the formal problem statement along with the proposed MIP formulation; §4.4 discusses the matheuristic; §4.5 presents a computational study comparing exact and matheuristic results across a variety of instances; and §4.6 concludes the chapter with a summary of our findings.

4.2 Literature Review

This section introduces three streams of literature that relate to the notions of carrier synchronization, demand periodicity, and tactical planning in the TSP-DE.

4.2.1 Single-Period Variants

The single-period problem of routing a vehicle and its companion drone through a network of customer deliveries at minimal duration is known as the Traveling Salesman Problem with Drone (TSP-D). Like the TSP-DE, the TSP-D allows for collaborative deliveries between the carriers, and therefore requires that the vehicle and drone travel in sync so to accommodate en-route rendezvous between them. In this sense, each period of a TSP-DE instance, when separated from the remainder of the problem, is in fact an instance of the TSP-D. The TSP-DE also has additional requirements for periodicity and consistency. Firstly, each period of the TSP-DE features distinct expressions of demand, such that customers receiving a parcel in one period may not do so in the next. Thus solving one instance of the TSP-DE is akin to solving a series of instances of the TSP-D, each of which may have a distinct customer network. Moreover, the TSP-DE requires that the same carrier serve each customer across all time periods. As such, carrier assignments minimize the routing cost of serving a customer across all periods in the horizon. Should the carrier assignments in the TSP-DE be set, what remains is to solve in each time period a simplified TSP-D instance in which carrier assignments are fixed.
There is a growing literature for the TSP-D, spanning exact optimization [Murray and Chu, 2015, Carlsson and Song, 2017, Agatz et al., 2018, El-Adle et al., 2021, Poikonen et al., 2019] and heuristic approaches [Bouman et al., 2018, Schermer et al., 2019] that have solved networks having up to 32 nodes optimally and up to 200 nodes heuristically. Rapid developments in drone delivery technology, accelerated by the Covid-19 pandemic, have yielded a varying set of assumptions for vehicle-drone operations, including the flight time of the drone, and the rendezvous protocol [El-Adle et al., 2021]. For recent reviews, the interested reader is referred to Macrina et al. [2020] and Boysen et al. [2021].

4.2.2 Periodic & Consistent Routing Problems

The notions of carrier consistency and the periodicity of customer demand have been investigated separately in the literature. Whereas the classical TSP considers spatial relationships, problems like the Period Traveling Salesman Problem (PTSP) and the Periodic Vehicle Routing Problem (PVRP) “...take into account both spatial and temporal relationships” [Chao et al., 1995]. A variety of exact and heuristic solution approaches have been proposed for periodic routing problems, including: Christofides and Beasley [1984], Chao et al. [1995], Bertazzi et al. [2004], the latter of which investigated PTSP instances having up to 288 customers over a three-period horizon. For the PVRP, Mourgaya and Vanderbeck [2007] investigated instances having as many as 80 nodes over a five-period planning horizon. Consistent routing problems generally require a single vehicle from a fleet to serve the same customers over the planning horizon, with changes to the vehicle’s route allowed in each period [Mourgaya and Vanderbeck, 2007]. Subramanyam and Gounaris [2016] and Subramanyam and Gounaris [2018] solve instances of the Consistent Traveling Salesman Problem with up to 100 nodes being served over a five-period horizon. For a fuller review, the reader is referred to Francis et al. [2008] and Kovacs et al. [2014].
The combination of both consistency and periodicity features in the TSP-DE, as well as the presence of two carriers (the vehicle and drone), sets it apart from the extant literature. Periodic routing problems typically allow for flexible schedules: customers may be eligible to receive deliveries on one of several combinations of periods in a planning horizon. For example, in the PVRP, a customer scheduled to receive two deliveries in a seven-day horizon may receive them “...on Monday-Friday, Monday-Thursday, or Tuesday-Friday, but no other combinations are acceptable” [Christofides and Beasley, 1984]; the TSP-DE does not permit shifts of customer deliveries from one period to another. For consistent routing problems, temporal consistency is usually the key consideration: carriers must arrive within the same time window, or only on certain days in the planning horizon [Francis et al., 2008]. In contrast, consistency in the TSP-DE considers not only specific periods in which carriers must deliver, but also the challenge of ensuring that either the vehicle or the drone exclusively is available to deliver in all periods in which the customer expresses demand.

4.2.3 Tactical Planning

Several research streams focus on the interplay between inter-period tactical decisions that permit intra-period operational decisions. For example, Baloch and Gzara [2020] investigate a network design problem in which stations may be placed strategically to serve nearby customers over a planning horizon either by vehicle or drone, with the goal of maximizing profit based on customer service preferences. Janssens et al. [2015] consider tactical and routing assignments in which customers are assigned to microzones, with as single vehicle expected to serve all customers in the microzone within a certain time. In these problems and their solution approaches, however, tactical assignments are generally independent of operational assignments: as long as a feasible completion in operational assignments exists, any tactical as-
Assignment is feasible. For the TSP-DE, however, tactical and operational assignments are closely linked. Tactically assigning a customer to be served by drone necessitates the operational availability of nearby customers to serve as launching and landing sites. Thus the TSP-DE extends disparate elements of synchronization, periodicity/consistency, and tactical planning from a number of literature streams into a unified, computationally-challenging problem.

### 4.3 Problem Statement & MIP Formulation

This section firstly details the problem under examination alongside illustrative examples. Thereafter, the proposed MIP formulation is introduced.

#### 4.3.1 Problem Statement

Given a network of customer locations represented by \( n \) nodes, and edges representing vehicle routes and drone flight paths connecting those nodes, and a \( T \)-period horizon over which customers express demand, the objective of the TSP-DE is to assign either the vehicle or its companion drone to serve each customer node consistently across the horizon. For any period, the subset of customers requiring a delivery is known; likewise, for any customer, the subset of periods in which s/he must be visited is a given. In each period, the drone and vehicle traverse an optimized tour that begins and ends at a central depot while serving collaboratively all nodes expressing demand in that period. Every customer must be served by the same carrier in any period in which the node expresses demand, and demand must be met in the same period in which it is expressed.

In order to serve a customer \( k \) via drone, the vehicle may launch the drone from a distinct customer \( i \) (where \( i \) is served by the vehicle), and then the vehicle may retrieve the drone from another distinct customer \( j \) (where \( j \) is also served by the vehicle). Hereafter, \((i, j)\) are referred to as rendezvous nodes for \( k \). As discussed in
§4.2.1, drones are typically powered by rechargeable batteries and, therefore, have a limited capacity for flight. A drone serves exactly one customer per flight and its flight duration from \(i\) to \(k\) to \(j\) must not exceed \(F\), the drone’s flight range. When retrieved by the vehicle, the drone’s flight capacity is replenished, possibly by swapping batteries, such that it may immediately launch for a new delivery. When the drone is not flying, it is transported aboard the vehicle.

Although launching the drone requires both carriers to be present simultaneously at a node, the drone’s landing at a node and the vehicle’s arrival to the same may happen asynchronously. Should the vehicle await the drone at a landing site, that waiting time counts towards the duration of the tour. Any waiting time by the drone at the landing node is disregarded, since the drone may freely delay its flight speed to arrive at the same time as the vehicle. Additionally, the vehicle is not permitted to deliver to customers en route to the landing site. That is, given rendezvous nodes \((i, j)\) for \(k\), the vehicle must launch the drone at \(i\), then travel directly to \(j\) to retrieve the drone. While allowing more intervening deliveries by the vehicle is possible, a number of publications have shown that limiting intervening vehicle visits achieves comparable outcomes in tour duration at significantly reduced computational expense. The interested reader is referred to Boysen et al. [2018] and Bouman et al. [2018] for details.

A central challenge in the TSP-DE is that drone service necessitates the co-expression of customer demand. Even if node \(k\) may be served by rendezvous nodes \((i, j)\) in one period, the absence of demand by either \(i\) or \(j\) in a different period may jeopardize the ability of the drone to serve node \(k\). In what follows, this phenomenon is referenced as \textit{drone eligibility}: the availability of rendezvous nodes that co-express demand in the same periods as the node of interest. In the absence of rendezvous nodes in any single period in which a node expresses demand, that node is \textit{drone-ineligible}. Figure 4.1 illustrates a small example of multi-period demand. Node 11
expresses demand in period 1 (left-hand side), but not in period 2 (right-hand side). As the number of time periods increases, illustrating TSP-DE instances in this way becomes cumbersome. Alternatively, Figure 4.2 illustrates a visualization scheme that captures the phenomenon of drone eligibility more succinctly.

Figure 4.2 shows node locations using polar coordinates, with radii corresponding to half of the drone’s flight range. In order to be drone-eligible, a node must be no farther than one radial arc (and 60 degrees) from each node in a rendezvous pair that co-expresses demand. Thus node 8 is ineligible for drone service by virtue of its isolation from possible rendezvous nodes. Node 6 is also drone-ineligible due to a lack of rendezvous nodes (the pair (3,13) is available in period 1 but no such pair is available in period 2), as seen in the right-hand side of Figure 4.1). To highlight drone eligibility, Figure 4.2 designates the nodes served by drone, and amongst vehicle assignments, further distinguishes between drone-eligible nodes (e.g. node 10) and those that are drone-ineligible (e.g. node 9).

Finally, note that periods refer to user-defined windows of time in which nodes express demand. A period may represent a day, or a window of time that is longer or
shorter, as needed by the decision-maker. The computational study in §4.5 considers a 6-period horizon, in which a period represents a day and one day of the week is off.

4.3.2 MIP Formulation

We introduce parameters and variables for our formulation as follows:

**Input Parameters**

- $T = \{1, \ldots, T\}$: Set of service periods.
- $V = \{0, 1, \ldots, n, n + 1\}$: Set of service nodes in the network.
- $V^* = V \setminus \{0, n + 1\}$: Set of customer nodes, which excludes the central depot (represented by dummy nodes 0 and $n + 1$).
- $\alpha$: Ratio of drone to vehicle speed.
- $\tau_{ij}$: Vehicle travel time from $i$ to $j$, $\forall i, j \in V, i \neq j$. 
• \( f_{ij} \): Drone travel time from \( i \) to \( j \), \( \forall \ i, j \in V, i \neq j \).

• \( d^t_j \): \( d^t_j = 1 \iff \) customer \( j \) requires parcel delivery in period \( t \), \( \forall j \in V^*, t \in T \).

• \( F \): Maximum flight duration for the drone to make a single delivery then return to the vehicle.

• \( C^t \): Set of customers requiring parcel delivery in period \( t \), \( \forall t \in T \).

• \( R^t \): Set of nodes requiring carrier visits in period \( t \), \( \forall t \in T \).

• \( M^t_{ij} \): A “large enough” scalar \( \forall t \in T, i \in R^t, j \in R^t, i \neq j \).

• \( D^t_{max} \): The objective value of a feasible solution in period \( t \), which visits all nodes \( i \in R^t, \forall t \in T \).

• \( f_{ikj} = \max(0, f_{ij} + f_{jk} - \tau_{ij}) \): The vehicle’s waiting time for a drone flight originating from \( i \), delivering to \( k \), and landing at \( j \), \( \forall i, j \in V, k \in D \), where \( i, j, k \) distinct.

• \( E^t_k = \{(i, j) \in R^t | f_{ik} + f_{kj} \leq F, \forall t \in T, i, j \in R^t, k \in C^t \} \): All pairs of rendezvous nodes permitting a drone delivery to customer \( k \) in period \( t \), where \( i, j, k \) are distinct.

• \( D = \{j \in V^* | d^t_j > 0 \implies |E^t_j| > 0 \forall t \in T \} \): Set of drone-eligible nodes.

**Decision Variables**

• \( x^t_{ij} \in \{0, 1\} \): \( x^t_{ij} = 1 \iff \) the vehicle travels from \( i \) to \( j \) in period \( t \), \( \forall t \in T \), \( i, j \in R^t, i \neq j \).
\( y_{ikj}^t \in \{0, 1\} : \quad y_{ikj}^t = 1 \iff \) the drone launches from \( i \), delivers to \( k \), and lands at \( j \) in period \( t \), \( \forall \ t \in T, i,j \in R^t, k \in C^t \), \( (i,j) \in E^t_k \), where \( i,k,j \) distinct.

- \( \delta_j \in [0, 1] : \quad \delta_j = 1 \iff \) the drone delivers to customer \( j \) across all periods in which \( j \) expresses demand, \( \forall j \in V^* \).

- \( b_{jt}^t \in [0, D_{max}^t] : \) The departure time of the vehicle at node \( j \) in period \( t \), \( \forall t \in T, j \in R^t \).

The formulation for the tactical traveling salesman problem with drone eligibility, Model TSPDE, is presented as follows:

\[ \text{TSPDE: Minimize} \quad \sum_{t \in T} \sum_{i \in R^t} \sum_{j \in R^t} (\tau_{ij} x_{ij}^t) + \sum_{t \in T} \sum_{k \in (C^t \cap D^t)} \sum_{(i,j) \in E^t_k} \hat{f}_{ikj} y_{ikj}^t \quad (4.1a) \]

s.t. \[ \sum_{j \in C^t} x_{0j}^t = 1, \quad \forall t \in T, \quad (4.1b) \]

\[ \sum_{j \in C^t} x_{jn+1}^t = 1, \quad \forall t \in T, \quad (4.1c) \]

\[ \sum_{j \in R^t} x_{kj}^t = 1 - \delta_k, \quad \forall t \in T, k \in C^t, \quad (4.1d) \]

\[ \sum_{j \in R^t} x_{ij}^t - \sum_{j \in R^t} x_{ji}^t = 0, \quad \forall t \in T, i \in C^t, \quad (4.1e) \]

\[ y_{ikj}^t \leq x_{ij}^t, \quad \forall t \in T, k \in (C^t \cap D^t), \quad (4.1f) \]

\[ \sum_{k \in C^t \setminus (i,j) \in E^t_k} y_{ikj}^t \leq 1, \quad \forall t \in T, i,j \in R^t, \quad (4.1g) \]

\[ \sum_{(i,j) \in E^t_k} y_{ikj}^t + \sum_{i \in R^t} x_{ik}^t = 1, \quad \forall t \in T, k \in C^t, \quad (4.1h) \]
\[ b^t_j \geq b^t_i + \tau_{ij} + \sum_{k \in (C^t \cap D)} \hat{f}_{ikj}^t y_{ikj}^t - M^t_{ij}(1 - x_{ij}^t), \quad \forall t \in T, \forall i, j \in R_t | i \neq j, \quad (4.1i) \]

\[ b^t_{n+1} \leq \sum_{i \in R^t} \sum_{l \in R^t} \tau_{il} x_{il}^t + \sum_{k \in (C^t \cap D)} \sum_{(i,l) \in E_k^t} \hat{f}_{ikl}^t y_{ikl}^t, \quad \forall t \in T \quad (4.1j) \]

\[ b, \delta, \geq 0 \quad x, y \text{ binary}. \]

For each time period, the objective (4.1a) minimizes the return time to the depot for all carriers: the summation of \( x \)-variables minimizes the travel time of the vehicle, while the summation of \( y \)-variables accounts for the duration of the vehicle’s wait for the drone. Thus (4.1a) is the makespan of a tour in which both carriers return to the depot. Constraints (4.1b) and (4.1c) ensure the vehicle’s path begins and ends at the depot (represented as nodes 0 and \( n+1 \), respectively), while Constraint (4.1e) ensures a Hamiltonian path through the intervening customer nodes for each time period. Across time periods, Constraint (4.1d) ensures the consistent assignment of a single carrier to each customer, while Constraint (4.1h) ensures that each customer is delivered to either by the vehicle or by the drone. Constraint (4.1f) disallows vehicle deliveries between rendezvous nodes (if the drone is in flight), and Constraint (4.1g) allows at most one drone flight to launch from each rendezvous node.

Constraint (4.1i) determines the departure time of the vehicle from nodes, with the secondary purpose of a subtour elimination constraint of the Miller-Tucker-Zemlin-(MTZ-) type. Since formulations featuring MTZ-type constraints are known to yield relatively weak linear relaxations, Constraint (4.1i) has been enhanced by scaling \( M \), as suggested in the remark below and detailed in Appendix C.1. Constraint (4.1j) provides an upper bound for the value of the arrival time at the depot in each time period, since the latest possible departure time from any node must be less than the
makespan of the entire tour in the same period. Note that absent this upper bound, Model TSPDE still provides valid tactical and routing assignments; Constraint (4.1j) only affects the numerical values of \( b \), so that its inclusion in the model is left for the discretion of the decision-maker.

**Remark.** As shown in Appendix C.1, the \( M \) scalar in Constraints (4.1i) may be set to:

\[
M_{ij}^t = \begin{cases} 
0, & i = 0, j = n + 1, \\
2\tau_{n+1j}, & i = n + 1, \\
D_{max}^t, & j = 0, \\
D_{max}^t - \tau_{in+1} + \tau_{ij} - \tau_{0j}, & \text{otherwise}. 
\end{cases}
\]

### 4.4 Matheuristic for TSP-DE

This section introduces a matheuristic that tractably produces high-quality solutions for Model TSPDE for instances of industrial scale. The matheuristic generates an initial feasible solution for the problem by solving a customer-centric multiple knapsack problem, before improving it in two phases of neighborhood searches motivated by demand periodicity and geo-spatial analyses. Figure 4.3 presents the matheuristic procedure.

In what follows, *tactical assignments* refer to the values of the binary \( \delta \)-variables, indicating whether the vehicle (\( \delta = 0 \)) or the drone (\( \delta = 1 \)) serves a customer across all periods. In contrast, *operational assignments* refer to values of the routing binary \( x \)- and \( y \)-variables that determine the path of the vehicle and drone in each period, respectively. The incumbent solution refers to the best-known feasible MIP solution for Model TSPDE at that phase of the matheuristic.
4.4.1 Initialization

The matheuristic begins by generating feasible tactical assignments for Model TSPDE by solving a multiple knapsack problem (MKP). As discussed in §3.1, the drone’s requirement for rendezvous nodes sets an upper limit on the number of customers that may be served via drone in the network. In general, that upper bound is \( \lfloor \frac{(m-1)}{2} \rfloor \), where \( m \) is the number of nodes in the neighborhood of interest. In addition to this global upper bound on drone assignments, the TSP-DE is also characterized by upper bounds on drone assignments in local neighborhoods, based on the availability of rendezvous nodes therein. Combined, these local and global caps on drone assignments present a special structure in the TSP-DE that may be exploited. By analyzing the demand periodicity and geo-spatial dispersion around each drone-eligible
node, the MKP encapsulates this tactical trade-off as follows:

**Input Parameters**

- $s_k$: The minimal savings associated with serving a drone-eligible customer $k \in D$ via drone from the nearest rendezvous nodes, defined as $s_k = \sum_{t \in T} \min_{(i,j) \in E_t^k} (\tau_{ik} + \tau_{kj}) \alpha$.

- $c_{\text{max}}$: The maximum number of nodes that may be served by drone in the network, where $c_{\text{max}} = \lfloor \frac{(n+2)-1}{2} \rfloor = \lfloor \frac{(n+1)}{2} \rfloor$. Note that $n$ is the number of customer nodes in the entire network, and the depot is represented by 2 nodes.

- $\omega_{tk} = \bigcup_{(i,j) \in E_t^k} \{i, j\}$: The union of the neighborhoods of all customer nodes that may serve as rendezvous points for drone-eligible node $k \in D$ in period $t \in T$.

- $p_{1tk}$: The maximum number of neighboring nodes that are ineligible for drone service and may be used as rendezvous points for node $k \in D$ in period $t \in T$, where $p_{1tk} = \max(0, \lfloor \frac{|\omega_{tk} \setminus D| - 1}{2} \rfloor)$.

- $p_{2tk}$: The maximum number of neighboring nodes that are drone-eligible and may be used as rendezvous points for node $k \in D$ in period $t \in T$, where $p_{2tk} = \min(1, \max(0, \lfloor \frac{|\omega_{tk} \cap D| - 1}{2} \rfloor))$.

The parameters $p_{1tk}$ and $p_{2tk}$ serve to partition the local neighborhood of $k$ based on drone eligibility, which is required when a neighborhood contains more than one drone-eligible node. Figure 4.4 illustrates such an example, where nodes 6 and 16 are both drone-eligible, and node 6 may be served by the following rendezvous pairs in a particular period: (0, 16); (16, 3); (16, 4); (16, 11); (16, 13); as well as their reciprocals (e.g. (3, 16)). In this way, all the rendezvous pairs associated with node 6 are dependent on the availability of node 16 to as part of a rendezvous pair. If node 16 were assigned to the drone, then node 6 would no longer be eligible for drone service.

To enforce this mutually exclusive relationship, $p_{2tk}$ allows no more than one node from amongst the neighborhood of drone-eligible nodes to be served via drone using
the max operator. Considering node 6, for example, $\omega_{2,6} = \{0, 3, 4, 11, 13, 16, 21\}$, where nodes 0 and 21 are dummy nodes representing the depot. Of those nodes, only $(\omega_{2,6} \cap D) = \{3, 4, 11, 13, 16\}$ are drone-eligible, meaning that in general up to $\lfloor \frac{(5-1)}{2} \rfloor = 2$ nodes may be chosen for drone service. By construction, however, $p_{tk}^2 = \min(1, \max(0, 2)) = 1$. The inner max operator is required in the case $|\omega_{2,6} \cap D| = 0$, such that $p_{tk}^2$ is bounded below by 0.

Figure 4.4: Drone-Eligible Nodes in a Densely-Populated Neighborhood

Similarly, the max operator is required in order to enforce a lower bound of 0 for $p_{tk}^1$, which considers drone-ineligible customers. For example, for node 6 in Figure 4.4, only one such node (the depot) is available. Since drone-ineligible nodes must
always be served by the vehicle, the only upper bound on the number of such nodes that may be used as rendezvous points is \( \left\lfloor \frac{1-1}{2} \right\rfloor = 0 \). To complete the example cited heretofore for node 6 in period 2, \( p_{1,6}^2 = 0 \), \( p_{2,6}^2 = 1 \), and therefore no more than one node may be assigned to drone service in the neighborhood of node 6. To enforce such a constraint, Model MKP is presented as follows where the \( z \)-variables are binary such that \( z_k = 1 \iff \text{node } k \in D \) is served by the drone in all time periods:

\[
\text{MKP: Maximize } \sum_{k \in D} s_k z_k \tag{4.2a}
\]

s.t. \[
\sum_{k \in D} z_k \leq c_{\text{max}}, \tag{4.2b}
\]

\[
z_k + \sum_{j \in (\omega_k \cap D)} z_j \leq p_{1k}^1 + p_{2k}^2, \quad \forall k \in D, t \in T \mid k \in C^t, \tag{4.2c}
\]

\[z\text{ binary.}\]

The keystone of Model MKP is its sole consideration of drone-eligible tactical assignments, without regard for corresponding operational assignments. The result is a much smaller number of decision variables, on the order of the cardinality of \( D \), the set of drone-eligible nodes. Across all drone-eligible nodes, objective (4.2a) in Model MKP maximizes the savings associated with vehicle travel time that would be obtained by serving nodes via drone. Constraint (4.2b) sets \( c_{\text{max}} \) as the global maximum for the number of nodes that may be served via drone. At neighborhoods associated with each \( k \in D \), Constraint (4.2c) enforce localized knapsacks. The left-hand side of Constraints (4.2c) captures all drone-eligible nodes in the neighborhood of \( k \), including \( k \). The right-hand side of Constraints (4.2c) specify the maximum number of drone assignments according to a partition of the neighborhood into nodes that are drone-eligible (captured by \( p_{1k}^1 \)) and those that are not (captured by \( p_{2k}^2 \) ).
Once the tactical assignments in Model TSPDE are fixed on the basis of the MKP solution, the residual problem of solving for vehicle and drone routing decisions is separable. That is, Model TSPDE thus decomposes into $T$ independent instances of the TSP-D for each period (with fixed tactical assignments across the multi-period horizon) that may be solved by an exact or heuristic approach (e.g. see El-Adle et al. [2021]). In what follows, for a given set of tactical assignments, an operational completion is a conforming set of routing decisions assignments that are feasible.

While Model MKP generates feasible tactical assignments, it does so at the expense of compressing certain features of Model TSPDE. The loss of these features over-constrains Model MKP in comparison to Model TSPDE, and ultimately adversely affects the quality of tactical assignments in the former. Thus although solutions from Model MKP translate into feasible tactical assignments for Model TSPDE by setting $z_k = \delta_k \forall k \in D$ and $\delta_k = 0 \forall k \in (V^*\setminus D)$, tactical assignments from Model TSPDE are not necessarily feasible in Model MKP. Whereas Constraint (4.2c) permits only one drone assignment per neighborhood to preserve the feasibility of tactical assignments in some instances (e.g. Figure 4.4), Model TSPDE is not restricted as such. Moreover, the parameter $s_k$ estimates a lower bound on the savings accrued by serving node $k$ via drone from the nearest rendezvous pair; an alternate rendezvous pair assignment may increase those savings. These limitations are overcome in subsequent phases of the matheuristic.

### 4.4.2 Period-Specific Demand Improvement

Phase 1 of the matheuristic seeks improved tactical assignments by solving a series of sub-problems derived from Model TSPDE. Whereas the initialization phase seeks to quickly generate a set of feasible tactical assignments by ignoring routing considerations, Phase 1 establishes a series of sub-problems that connect those tactical assignments with their operational completion. By design, this connection exists in
the body of constraints of Model TSPDE. Thus this phase generates improvements upon the incumbent solution by fixing certain tactical and routing variables in Model TSPDE while a targeted group of the same variables are left free for re-optimization. The goal of this phase, however, is to target customers on the basis of their demand periodicity. Phase 1 proceeds as follows:

1. **Initialization:** Let \( P_t = \{ j \in V^* | \sum_{l \in T} d^l_j = t \} \), such that customers are partitioned based on the number of periods in which they express demand. The operational completion to the solution from Model MKP, denoted \( Z^*_{\text{initial}} \), is the starting incumbent.

2. **Main Loop:** Phase 1 aims to improve tactical assignments by solving a series of targeted sub-problems from Model TSPDE as follows:

   - Considering the total frequency of demand in descending order (i.e. \( \forall t \in \{T, \ldots, 1\} \)), fix the tactical assignment related to customers excluded from \( P_t \). This leaves free for re-optimization the tactical assignments relating only to customers included in \( P_t \). Moreover, the routing variables relating to all nodes are left free during the re-optimization procedure. Then solve Model TSPDE subject to the aforementioned partial variable fixings.

   - Whenever the re-optimized solution improves upon the incumbent, the latter is updated. Otherwise, the number of non-improving iterations, \( NII \), is incremented.

3. **Termination:** The procedure terminates after a successive number of \( NII \), or once all periods have been re-optimized.

In our experience, re-optimizing tactical assignments in descending order of demand frequency, \( P^T \) to \( P^1 \), improves the efficiency of the proposed methodology.
Customers having a more frequent demand are thus prioritized, because they correspondingly have a relatively large number of routing variables associated with them which directly factor into Objective (4.1a). Additionally, these nodes are likely to be staples of the network structure, such that their tactical assignments indirectly affect those of other customers with less frequent demand. For instance, erroneously assigning the drone to deliver to a node with high demand expression may preclude the neighboring nodes from being served via drone. Finally, although re-optimization of both tactical and operational assignments is possible through this procedure, the focus is on optimal tactical assignments. The next phase of the matheuristic provides the primary framework for the improvement of routing assignments.

4.4.3 Geo-spatial Improvement

Phase 2 of the matheuristic intensifies the search for both improved tactical and operational assignments using local neighborhoods. Re-optimizing a solution on the basis of demand periodicity alone disregards an essential component of the problem: the geo-spatial proximity necessary to find rendezvous pairs for drone-eligible nodes. Indeed, $F$, the flight range of the drone, establishes a natural local neighborhood wherein optimal tactical and operational assignments are closely linked. Without loss of generality, the following steps illustrate the Geo-spatial Improvement procedure for a specific node $k$:

1. **Initialization:** Since many local neighborhoods may overlap, Geo-spatial Improvement begins by identifying neighborhoods that are non-dominated. Recall that for drone-eligible node $k$, set $\omega_{tk}$ defines the set of all nodes that may serve as rendezvous nodes for $k$ in period $t$. Let $\Omega_k = \{t \in T|k \in C^t\}$ be the set of all periods in which $k$ expresses demand, and $\omega_k = \bigcup_{t \in \Omega_k} \omega_{tk}$ be the set of rendezvous nodes neighboring $k$ across all time periods. Note that for distinct drone-eligible nodes $j$ and $k$, if $\omega_j \subset \omega_k$, then re-optimizing the neighborhood
of $k$ ensures the re-optimization of the neighborhood of node $j$ as well.

Let $exploredNodes$ and $exploredNeighborhoods$ be sets that represent nodes that have been re-optimized and the corresponding neighborhoods of those nodes, respectively. During initialization, these sets are populated with dominated nodes and neighborhoods. Continuing the example above, since the neighborhood of Node $j$ was dominated, then $exploredNodes = \{l \in V^* | l \in \omega_j\}$ and $exploredNeighborhoods = \{j\}$.

2. **Local Search Diversification**: In order to re-optimize neighborhoods in a sequence most likely to yield improvements, Geo-spatial Improvement introduces the notion of diversification. Given $exploredNodes$, the set of nodes that have already been re-optimized, Geo-spatial Improvement seeks a drone-eligible node $k$ such that the neighborhood associated with $k$ includes the greatest number of nodes that have yet to be re-optimized. That is, Geo-Spatial Improvement selects the next local neighborhood to be optimized as that belonging to $k$, where $k$ maximizes the value of $|\omega_k \setminus exploredNodes|$.

3. **Local Search Expansion**: To ensure operational assignments may be altered effectively, Geo-spatial Improvement then augments $\omega_k$ with those nodes that are connected via routing variables. That is, given $l \in \omega_k$ and $q \notin \omega_k$, suppose that $x_{t_{lq}} = 1$ in time period $t$, such that the vehicle travels from node $l$ to node $q$. Re-optimizing a neighborhood that includes node $l$ but excludes node $q$ may forsake an opportunity to improve the incumbent solution. Thus this step updates a set called $expandedNeighborhood$ to include both $\omega_k$ as well as any nodes comprising an incoming or outgoing arc with a vertex in $\omega_k$ in the incumbent solution.

4. **Targeted Re-Optimization**: The $expandedNeighborhood$ set encapsulates all nodes in the network that are required for an optimal tactical and operational
assignments for the neighborhood of $k$. As such, Geo-spatial Improvement fixes
the value of all variables associated with nodes excluded from $\text{expandedNeigh-
borhood}$. For the nodes comprising $\text{expandedNeighborhood}$, Geo-spatial Improve-
ment further targets routing variables by time period: only variables associated
with a period $t \in \Omega_k$ are relevant to the neighborhood at hand, since node $k$
only expresses demand during those periods. Indeed, routing variables associ-
ated with periods excluded from $\Omega_k$ may be fixed. The only variables left free
for re-optimization are the tactical and routing variables that connect to the
neighborhood of node $k$, and only in those periods when $k$ expresses demand.
Model TSPDE is then re-optimized, with the variable values of the incumbent
solution being updated whenever the re-optimization yields an improved objec-
tive value.

5. **Termination Criteria:** After a certain number of successive non-improving
iterations, or once the neighborhoods of all drone-eligible nodes have been re-
optimized, Geo-Spatial Improvement terminates.

The unique advantage of separating the optimization of local neighborhoods from
the remainder of the problem is that due to the flight range of the drone, assign-
ments that are locally optimal in the neighborhood of a drone-eligible customer are
in fact globally optimal for the instance. Thus the assignments related to each local
neighborhood are independent from the remainder of the network. Indeed, exhaus-
tively improving tactical and operational assignments in the neighborhood of each
drone-eligible node would inexorably produce a globally optimal solution. This moti-
vates the diversification scheme embedded within Geo-spatial Improvement. Seeking
neighborhoods with maximum demand expression may yield early improvements, but
may also lead to repeated re-optimization of the same dominated neighborhood. To
mitigate the computational effort, Geo-spatial Improvement re-optimizes with greater
priority larger, more diverse neighborhoods and terminates when successive neighborhood re-optimizations yield no improvements.

Although the aforementioned phases of the matheuristic address the themes of periodicity and geo-spatial analyses that characterize solutions to the TSP-DE, the interplay between the two phases also serves a valuable role in diversifying the search for a solution. The discovery of improved solutions in each phase may cascade into the next, such that alternating between the two phases, rather than using either one in isolation, enhances the search and results in higher quality solutions. The impact of the matheuristic’s parameter settings and implementation is discussed further in §4.5.3.

4.5 Computational Study

This section introduces the instances on which the proposed exact optimization and matheuristic approaches were tested, followed by computational insights.

4.5.1 Description of Problem Instances

Instances are characterized by two main features: (i) a network configuration or the location of customers on a Cartesian plane; and (ii) a demand profile or the demand realization for customers over the entire multi-period planning horizon.

Network Configurations

Five network configurations were generated, each featuring the depot at the origin. Each configuration included 200 customer nodes distributed uniformly in a 360° pattern (spanning $2.5F$) around the origin. For each instance discussed below where $n \leq 200$, the first $n$ customer nodes from each configuration were chosen. The network is connected, with $\tau_{ij}$ (the vehicle’s travel time between nodes) equal to the Euclidean distance between $i$ and $j$, and $f_{ij}$ (the drone’s travel time) equal to $\alpha \times \tau_{ij}$,
where $\alpha = 0.5$ such that the drone travels twice as fast as the vehicle. The settings in this study match those used in benchmark instances for the TSP-D by Agatz et al. [2018], and extended by Bouman et al. [2018], Schermer et al. [2019] and El-Adle et al. [2021] amongst others.

**Demand Profiles**

The increasing volume and frequency of global parcel delivery discussed in §1 has motivated multiple customer categories in this study: $C_A, C_B, \text{ and } C_C$ which express demand with likelihood $\rho_A, \rho_B, \rho_C$, respectively. Each customer expresses demand in at least one time period and is assigned to exactly one of the disjoint customer categories. Without loss of generality, let $0 \leq \rho_C \leq \rho_B \leq \rho_A \leq 1$, such that customers in category $C_A$ have the highest probability of expressing demand.

<table>
<thead>
<tr>
<th>Country</th>
<th>Parcels (Billions)</th>
<th>Households (Millions)</th>
<th>Average Parcels Delivered Per Household (Annually)</th>
<th>Average Parcels Delivered Per Household (Daily)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Japan</td>
<td>9.00</td>
<td>53.33</td>
<td>168.8</td>
<td>0.46</td>
</tr>
<tr>
<td>United Kingdom</td>
<td>3.50</td>
<td>27.82</td>
<td>136.6</td>
<td>0.37</td>
</tr>
<tr>
<td>United States</td>
<td>13.00</td>
<td>119.73</td>
<td>122.8</td>
<td>0.34</td>
</tr>
<tr>
<td>Australia</td>
<td>0.87</td>
<td>7.76</td>
<td>120.4</td>
<td>0.33</td>
</tr>
<tr>
<td>Germany</td>
<td>3.50</td>
<td>41.54</td>
<td>89.1</td>
<td>0.24</td>
</tr>
</tbody>
</table>


Note that optimal solutions to the TSP-DE are shaped by the parameters defining each customer category. For example, if all nodes expressed demand with a probability of 1, then the TSP-DE would reduce to the TSP-D, where tactical assignments remain fixed across all periods. Similarly, if all nodes expressed demand infrequently, there may be insufficient rendezvous nodes available to allow for drone deliveries. The demand probability and formal construction of each customer category is set as follows:
• **Category A:** 10% of customers; let $\rho_A = \frac{2}{3}$, and define $C_A = \{1, \ldots, \lceil 0.1|V^*| \rceil \}$.

• **Category B:** 30% of customers; let $\rho_B = 0.35$, and define $C_B = \{\lceil 0.1|V^*| \rceil + 1, \ldots, \lceil 0.4|V^*| \rceil \}$.

• **Category C:** 60% of customers; let $\rho_C = 0.1$, and define $C_C = \{V^* \setminus (C_A \cup C_B)\}$.

Table 4.1 summarizes commercial parcel delivery trends for households around the world, some of which may be too heavy or bulky for drone delivery. For a variety of reasons, households may be more or less inclined to receive parcels than the average in their country; creating three separate customer categories helps account for this reality. The proposed construction of the demand profiles closely tracks to the average daily parcel delivery data from Table 4.1, since $\rho_A(0.1) \times \rho_B(0.3) \times \rho_C(0.6) = 0.23$.

**Instances**

We examine datasets S, M, and L, which involve small-, medium-, and large-sized instances, respectively. For each dataset, five distinct network configurations are examined and, for each of these, a number of distinct demand profiles (based on the demand parameters specified above) are generated. With planning horizons of $T = 3$ or $T = 6$ time-periods (typically days), the computational study includes:

- **Dataset S:** 150 instances; $(n = 20, 30, \text{ or } 40) \times (T = 3 \text{ or } 6) \times 5$ network configurations $\times 5$ demand profiles;

- **Dataset M:** 60 instances; $(n = 50, 75, \text{ or } 100) \times (T = 3 \text{ or } 6) \times 5$ network configurations $\times 2$ demand profiles;

---

2 All instances as well as best-known solutions available at: https://bit.ly/3l2phXV
• **Dataset L**: 20 instances; \((n = 150 \text{ or } 200) \times (T = 3 \text{ or } 6) \times 5\) network configurations \times 1 demand profile.

### Computational Settings
Models TSPDE and MKP as well as the matheuristic were implemented using AMPL and solved using GUROBI version 9.0.2, which was in our computational experience the most effective solver for the MIP formulation and problem instances presented. The runs were performed on a computer with an Intel i7-7700K processor and 32 GB of RAM.

#### 4.5.2 Base Results
The following statistics are used in the remainder of this section to assess the quality of the proposed matheuristic. In what follows, let \(Z^*_{\text{MIP}}\) be the best solution yielded by the exact optimization approach of Model TSPDE (within a time limit of 10,000 seconds), let \(Z^*_{\text{MH}}\) be the best solution yielded by the matheuristic, and let \(Z^*_{\text{MKP}}\) be the best operational completion (achieved within a time limit of 20 seconds for each period) of the tactical assignments generated from Model MKP.

### Summary Statistics
- \(\Delta\): The deviation between the best-known matheuristic solution and that yielded by the solver within 10,000 seconds using the proposed exact optimization approach, such that \(\Delta = \frac{(Z^*_{\text{MH}} - Z^*_{\text{MIP}})}{Z^*_{\text{MIP}}}\).
- \(\Delta_\mu\): The average of \(\Delta\) over a set of instances.
- \(\Delta_{\text{max}}\): The maximum of \(\Delta\) over a set of instances.
- \(\Delta_{\text{MKP}}\): The deviation between the best-known matheuristic solution and that yielded by a (time-limited) completion for Model MKP, such that \(\Delta_{\text{MKP}} = \ldots\)
\[
\frac{(Z_{\text{MH}}^* - Z_{\text{MKP}}^*)}{Z_{\text{MKP}}^*}.
\]

- \(\Delta_L\): The deviation between the best-known matheuristic solution and the upper bound yielded by the exact optimization approach within the same computational effort exerted by the matheuristic. For example, if the matheuristic solved a group instance within an average of 900 seconds, this was a time limit set for GUROBI for comparison.

- \(\text{LB}_\mu\): The deviation between the best-known matheuristic solution and the lower bound yielded by the solver at termination with a time limit of 10,000 seconds.

- \(\text{CPU}_{\text{MH}}\): The matheuristic’s average computational effort across a set of instances.

- \(\text{CPU}_{\text{MIP}}\): The average computational effort required by the exact optimization of Model TSPDE across a set of instances (and limited to 10,000 seconds).

Table 4.2 for Dataset S summarizes benchmark testing, where all instances were solved optimally using Model TSPDE. The matheuristic found optimal solutions for more than 73% of instances according to the \textbf{Opt.} column, with a deviation of at most 2.8% from the optimal solutions, as shown in the \(\Delta_{\text{max}}\) column in Table 4.2. Across Dataset S, the average deviation between matheuristic solutions and their optimal counterparts was 0.5% or less.

### Table 4.2: Dataset S

<table>
<thead>
<tr>
<th>n</th>
<th>T</th>
<th>(\Delta_\mu)(%)</th>
<th>(\Delta_{\text{max}})(%)</th>
<th>Opt.</th>
<th>(\text{CPU}_{\text{MH}}) (s)</th>
<th>(\text{CPU}_{\text{MIP}}) (s)</th>
<th>(\Delta_{\text{MKP}})(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>3</td>
<td>0.1</td>
<td>1.2</td>
<td>21/25</td>
<td>3.9</td>
<td>1.7</td>
<td>-19.4</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.2</td>
<td>1.4</td>
<td>19/25</td>
<td>4.6</td>
<td>1.5</td>
<td>-13.0</td>
</tr>
<tr>
<td>30</td>
<td>3</td>
<td>0.0</td>
<td>0.2</td>
<td>24/25</td>
<td>34.4</td>
<td>54.8</td>
<td>-29.0</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.1</td>
<td>1.1</td>
<td>20/25</td>
<td>36.4</td>
<td>30.3</td>
<td>-19.6</td>
</tr>
<tr>
<td>40</td>
<td>3</td>
<td>0.5</td>
<td>2.8</td>
<td>10/25</td>
<td>139.7</td>
<td>920.4</td>
<td>-33.0</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.2</td>
<td>0.9</td>
<td>16/25</td>
<td>158.2</td>
<td>1,759.2</td>
<td>-24.2</td>
</tr>
</tbody>
</table>
Table 4.2 also shows that as the size of the network and number of time periods approach practical relevance, the matheuristic’s computational performance scales far more advantageously than does that of the exact optimization approach. Although the exact optimization approach had a slight computational advantage for \( n = 20 \) and 30 under \( T = 3 \), this is directly attributable to a fixed computational expense associated with a progression through the phases of neighborhood searches, the iterative solution of smaller MIP problems therein, and the number of non-improving iterations the heuristic considered for convergence. Finally, the column entitled \( \Delta^{\text{MKP}} \) in Table 4.2 highlights the relative savings achieved by the matheuristic by comparing its final solution (after all phases) with its starting solution (the time-limited routing completion of tactical assignments from Model MKP). The magnitude of this metric is dependent on both the quality of the starting solution, and the ability of the matheuristic to improve upon it. For the purposes of benchmarking, it is included as part of the assessments in all datasets.

Table 4.3: Dataset M

<table>
<thead>
<tr>
<th>( n )</th>
<th>( T )</th>
<th>( \Delta_L(%) )</th>
<th>( \Delta_\mu(%) )</th>
<th>( \Delta_{\text{max}}(%) )</th>
<th>LB( _\mu(%) )</th>
<th>CPU_\text{MH} ( (s) )</th>
<th>CPU_\text{MIP} ( (s) )</th>
<th>( \Delta^{\text{MKP}}(%) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>3</td>
<td>-0.3</td>
<td>0.4(^3)</td>
<td>1.7</td>
<td>-0.6</td>
<td>333.6</td>
<td>3,922.8</td>
<td>-38.7</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>-3.9</td>
<td>0.6(^3)</td>
<td>2.0</td>
<td>-1.9</td>
<td>504.6</td>
<td>6,860.1</td>
<td>-29.0</td>
</tr>
<tr>
<td>75</td>
<td>3</td>
<td>-6.0</td>
<td>0.8</td>
<td>6.4</td>
<td>-5.7</td>
<td>692.8</td>
<td>10,000.0</td>
<td>-42.4</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>-0.5</td>
<td>-1.2</td>
<td>0.7</td>
<td>-6.5</td>
<td>1,074.9</td>
<td>10,000.0</td>
<td>-35.3</td>
</tr>
<tr>
<td>100</td>
<td>3</td>
<td>-4.1</td>
<td>0.2</td>
<td>2.7</td>
<td>-9.6</td>
<td>844.8</td>
<td>10,000.0</td>
<td>-46.0</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>-9.7</td>
<td>-0.6</td>
<td>4.0</td>
<td>-10.6</td>
<td>1,345.0</td>
<td>10,000.0</td>
<td>-38.8</td>
</tr>
</tbody>
</table>

\(^2\text{Opt.} = 2/8; \ ^3\text{Opt.} = 0/5\)

Tables 4.3 and 4.4 summarize results for Datasets M and L, respectively, where the time-limited exact optimization generally terminated before yielding a provably optimal solution (the footnotes for Table 4.3 highlight the exceptions, whereby the solver identified 8 and 5 optimal solutions for \( T = 3 \) and 6, respectively). Instead, the tables include columns for LB\( _\mu\), which compares the solutions yielded by the matheuristic against the best-known lower bounds derived at the termination of the
exact optimization approach, and $\Delta_L$, the deviation between the best-known upper
bounds produced the exact optimization approach over the same computational effort
consumed by the matheuristic, respectively.

In line with the values from Table 4.2, $\Delta_\mu$ remains consistently below 0.8% for
Dataset M. Consistently negative $\Delta_L$ values between -0.3% to nearly -10% indicate
the matheuristic outperformed the exact optimization approach when allowed a com-
parable computational time. Moreover, the difference between $\Delta_L$ and $\Delta_\mu$ shows
that even with the additional computational effort granted to the exact optimization
approach, the solver achieved only marginal improvements over the matheuristic so-
lutions. For example, for instances having 100 nodes under $T = 3$, the matheuristic
on average required about 850 seconds, yielding solutions that improved upon those
of the exact solution approach (limited to the same duration) by 4.1%, on average.
In the more than 9,000 seconds that followed, the best solutions yielded by the exact
optimization approach improved upon the matheuristic’s solutions by only 0.2%, on
average.

The difference between $\Delta_L$ and $\Delta_\mu$ also suggests that the computational effort
expended in the exact optimization approach is used primarily to improve the in-
cumbent lower bound, rather than to discover an improved feasible solution. To that
end, the $LB_\mu$ values between -0.6 to -10.6% indicate the magnitude of the deviation
between the matheuristic’s solution and the best-known lower bound for the instance.
Notably, the quality of both the upper and lower bounds deployed by the solver de-
teriorated significantly as the network size and the number of time periods increased.
In fact, for instances having more than 150 nodes, the exact optimization approach
yielded solutions that were non-competitive after 10,000 seconds, and thus were not
reported in Table 4.4. Instead, the reported $\Delta_{MKP}$ values conform to the same general
trend from Datasets S and M.
Table 4.4: Dataset L

<table>
<thead>
<tr>
<th>n</th>
<th>T</th>
<th>(\Delta_i)(%)</th>
<th>(\Delta_{\mu})(%)</th>
<th>(\Delta_{\text{max}})(%)</th>
<th>LB_{\mu}(%)</th>
<th>CPU_{\text{MH}} (s)</th>
<th>CPU_{\text{MIP}} (s)</th>
<th>(\Delta_{\text{MKP}})(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>150</td>
<td>3</td>
<td>-18.3</td>
<td>-4.3</td>
<td>-1.2</td>
<td>-16.9</td>
<td>1,620.5</td>
<td>10,000.0</td>
<td>-44.1</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>-21.2</td>
<td>-9.3</td>
<td>-4.3</td>
<td>-16.2</td>
<td>2,174.3</td>
<td>10,000.0</td>
<td>-41.6</td>
</tr>
<tr>
<td>200</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1,394.7</td>
<td>-</td>
<td>-34.1</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2,319.0</td>
<td>-</td>
<td>-36.6</td>
</tr>
</tbody>
</table>

As the \(\Delta_{\text{max}}\) column suggests, there was at least one instance where the matheuristic’s solution deviated from that of the exact optimization approach by 6.4%. More generally, of the 230 instances with solutions produced by both the matheuristic and (time-limited) exact optimization approach, the matheuristic matched or improved upon the exact solution in 64.5% of instances (\(\Delta \leq 0\%\)); 90.5% of instances exhibited \(\Delta \leq 1\%\); and all but 5 instances exhibited \(\Delta \leq 2.5\%\).

### 4.5.3 Matheuristic Progression & Parameter Tuning

To assess the matheuristic’s phases independently, Figure 4.5 summarizes results from instances having 40 nodes under \(T = 3\). Although the matheuristic deviated from the exact optimization approach by only 0.2% on average, the matheuristic yielded only 10/25 optimal solutions. This pattern highlights the presence of numerous near-optimal solutions, whereby a sub-optimal set of tactical assignments may be amenable to a routing completion that yields a near-optimal solution. Figure 4.5 tracks the quality of the incumbent solution produced by the matheuristic alongside the accuracy of customers assigned to the drone in that incumbent (as compared to the optimal solution). Even as solutions progressed from being 14.0% sub-optimal in Phase 0 to being 0.2% sub-optimal in the final phase of improvement in Figure 4.5, the accuracy of tactical assignments was below 90% on average.

An important limitation of the matheuristic’s performance are its search parameters, which modulate both solution quality and computational effort. The parameter settings chosen in this study (and outlined in Appendix C.2) resulted in one complete
main loop (Phase 1 and Phase 2), followed by the exploration of Phase 1 only, in a second, partial main loop iteration. Figure 4.5 shows a minimal gap between the (time-limited) actual and optimal routing completions for the tactical assignments from each phase, which attests to the adequacy of the selected solver settings. Although the solution quality improved only marginally during the second main loop iteration, that incremental effect was more significant for larger instances. Ultimately the matheuristic’s design serves as an optimization framework, leaving for users the freedom to parameterize the search according to their own needs for solution quality and computational efficiency.

4.6 Conclusion

This chapter investigates salient features of a last-mile delivery problem, whereby a decision-maker seeks to determine which customers to serve by drone throughout a planning horizon along with the accompanying vehicle-drone routing decisions. Our work contributes a novel MIP formulation that solves instances having up to 40 customers and 6 time-periods optimally, and a matheuristic that tractably and
consistently yields near-optimal solutions for instances having up to 200 nodes over the same horizon in our test-bed. The proposed matheuristic exploits two dimensions of the problem and iterates accordingly: the periodicity of customer demand and the geo-spatial dispersion of customer locations. From generating an initial solution to the neighborhood search procedures that follow, each phase of the proposed matheuristic is guided by a variety of analytical features that must be carefully balanced in a high-quality solution. The eligibility of rendezvous nodes for drone service (determined by the co-expression of demand amongst nodes), the savings accrued by assigning a customer to delivery service (determined by the frequency of demand and proximity to other nodes), as well as the subtle trade-offs involved in delivery to dense customer neighborhoods (which is determined by a number of geo-spatial features) are all intertwined features of the TSP-DE addressed by the matheuristic.

Whereas MIP formulations and commercial solvers generally resolve complex trade-offs for smaller-sized instances using a substantial computational effort, this study advocates for using optimization analytics for last-mile delivery problems to intuit algorithmic frameworks that drive towards near-optimal solutions for instances of practical size in more manageable times. In our computational study, the proposed matheuristic yielded optimal solutions for 110/150 instances with $n \leq 40$ customers and up to 6 time-periods; for instances with $50 \leq n \leq 100$ customers, it remarkably produced near-optimal solutions with an optimality gap of 0.5% on average; and for $n \geq 150$ customers, it produced similar computational patterns, suggesting high quality solutions in manageable computational efforts (within 1,800 seconds). For the latter instances, the solver’s solutions within a time limit of 10,000 seconds were non-competitive.

We advocate for the use of data and decision analytics jointly, in a manner that exploits the underlying data and problem features in order to intuit algorithmic frameworks that are both efficient and effective. In this context, our proposed MIP is first
used as a standalone methodology for small-sized instances and then as a backbone for our matheuristic where neighborhood searches are driven by customer demand frequency and location analytics. In light of the growing interest and need for unmanned vehicles such as aerial drones, this confluence of analytics and computational optimization is, in our opinion and experience, a promising research paradigm that can deliver both numerical performance, and insights into managerial policies, for challenging logistical problems.
CHAPTER 5
CONCLUSIONS AND FUTURE RESEARCH

This dissertation investigates the optimal deployment of joint vehicle-drone delivery systems for last-mile logistics using mathematical models enhanced by analytic insights. The first two essays explore operational decisions, via exact and heuristic optimization approaches, that route the carriers in tandem. The third essay investigates a variant of the traveling salesman problem with drones (TSP-D) over an extended planning horizon, the implications of which may aid decision-makers to choose which customers and network topographies may be most suited for delivery via drone. After summarizing the presented work, several managerial insights are highlighted below, followed by suggested future research topics.

Chapter 2 presents an MIP formulation for the TSP-D, enhanced by VIs, cut-set generation, and objective reformulation, that demonstrates a computational advantage over alternative formulations on benchmark instances in the literature. A significant breakthrough therein, at the time of publication, is the tractability of the proposed formulation in solving instances involving 24 customers in an optimal manner, with a breach into instances having as many as 32 nodes. Under less constrained operational problem settings, including flight ranges exceeding the current capability of drone delivery systems and in solving instances of typical size for commercial delivery networks, the performance of the exact formulation was more limited.

In this vein, Chapter 3 embeds two distinct formulations at the heart of a variable neighborhood search to generate heuristic solutions for instances of the TSP-D of practical size. In the first phase, the heuristic relies on a restricted formulation that
limits routing decisions relating to drone cycles and intervening deliveries by the vehicle during a drone flight. In the second phase, a different formulation relaxes the restrictions from Phase 1 to yield a deeper, yet computationally more onerous, exploration of the feasible space. Using diversification schemes that vary the size of neighborhoods and the re-optimization effort expended therein, the heuristic yields the best-known solutions to 113/120 benchmark instances of the TSP-D having up to 100 nodes, with comparable computational performance to existing approaches. The chapter also provides evidence of an improvement of up to 5% in objective values when drone cycles are permitted versus when they are disallowed. In contrast with Chapter 2, Chapter 3 also introduces a series of VIs that serve to simplify drone routing decisions under unlimited flight range settings, in which all customers are permitted to be served by the drone.

Chapter 4 introduces the Traveling Salesman Problem with Drone Eligibility (TSP-DE), which investigates a notion of carrier consistency whereby historical demand helps firms determine whether a customer might be better served by either vehicle or drone over a particular time horizon or season. The durable assignments of the TSP-DE, based on the geographic dispersion and demand patterns of customers, may then be enforced (during a particular season and revised periodically as needed) to significantly reduce the complexity of the residual operational problem of routing the carriers in each period, which remains challenging for instances of industrial scale. This chapter contributes both a novel mixed-integer programming (MIP) formulation for the TSP-DE as well as a heuristic approach for generating high-quality solutions. The heuristic is underpinned by the MIP formulation, and initialized by a feasible solution generated from a multi-knapsack formulation derived for the TSP-DE. The heuristic is driven by two underlying features of customer data to explore different neighborhoods of the feasible space: geographical locations and demand patterns. Over a set of benchmark instances, the proposed MIP formulation solves instances
having up to 40 customers and 6 time-periods optimally, while the heuristic tractably and consistently yields near-optimal solutions for instances having up to 200 nodes over the same horizon. The efficacy of the heuristic compares favorably against that of the MIP formulation, yielding 110/150 optimal solutions for instances having up to 40 nodes, with solutions deviating by less than 0.5% on average. This chapter suggests that the joint use of data and decision analytics, in a manner that exploits the underlying data and problem features, may yield algorithmic frameworks that are both efficient and effective.

5.1 Managerial Insights

To advance the adoption of drone delivery in general, and vehicle-drone tandem delivery systems in particular, this dissertation combines analytics and optimization to inform managerial decision-making. For practitioners in this domain, this section highlights several takeaways:

- **Analytically-informed Pre-processing:** Although the introduction of drone deliveries generally complicates routing decisions due to the flight range and autonomy of the drone, those same features may serve to simplify a variety of corresponding routing decisions for both the drone and the vehicle. Each chapter of this work presents a contribution in this vein:

  - Chapter 2 uses the flight range, speed of the drone and network density to pre-assign certain customers to vehicle deliveries *a priori* since they may not be feasibly served by the drone.

  - Chapter 3 introduces a lemma that specifies the conditions under which a drone cycle may exist in an optimal solution (under unlimited flight range for the drone), alongside a series of valid inequalities that enforce the lemma;
− Chapter 4 extends this notion of drone eligibility, to determine which customers may be served by the drone across a multi-period horizon based on the availability of nearby rendezvous nodes.

In all of these cases, data about customer demand and location are analyzed prior to optimization. When deployed, these pre-processing techniques improve the computational performance of both exact and heuristic optimization approaches. Moreover, these techniques signal the importance of having historical data about both customer demand and delivery locations when choosing deployment venues for drones.

• High-Resolution Data Analytics: According to Kopalle [2014], Amazon obtained several years ago a patent for so-called “anticipatory shipping,” in which the company proposed a system for boxing and shipping products to specific hubs or vehicles before a customer order is placed. Chapter 4 extends this concept to assign customers to a carrier based on elements of geo-spatial dispersion as well as the periodicity of demand. Firms like Amazon, which may have access to additional data about customer demand, inventory, as well as dynamic data such as real-time traffic (of ground or aerial vehicles) may further refine their decision-making to account for the impact of these factors. These trends also signal the importance of the integration of real-time data at scale. For example, a last-mile delivery provider may benefit from access to high-resolution data about traffic and weather patterns to predict not only optimized routes, but also the parking space to which a carrier may navigate [Winkenbach, 2018]. Reaping the maximal benefits from last-mile optimization requires integrating real-time demand, inventory and logistical data into analytical decision-making at every level of the organization [Winkenbach, 2018].
5.2 Future Research

With stakeholders in government, industry, and private citizenry, many questions about the utility of drone delivery systems and their applications remain open for investigation. In that vein, the FAA recently released a Concept of Operations (ConOps) report highlighting three major milestones for the integration of drones into civilian airspace: first, determining the safety standards by which drones will operate; second, management of unmanned aerial traffic; and third, management of drone-civilian interactions [Authority, 2020]. The following topics are suggested for future study:

- **Stochastic Variants:** While the deterministic settings investigated in this dissertation address some of the challenges associated with drone deliveries, there remains the challenge of accommodating the stochasticity of real-world data. Amongst others, factors such as customer demand, vehicular (and possibly aerial) traffic, as well as the weather may be challenging to predict in advance. As the use of drones scales for deliveries, these stochastic elements are likely to have a larger impact on the quality and possibly the feasibility of routing solutions. It is therefore essential to develop approaches that address inherent stochasticity in the problem setting.

- **Unmanned Aircraft Systems Traffic Management (UTM):** As highlighted in Figure 5.1, U.S. regulations limit drone flights to a maximum altitude of 400 feet above ground level. Within that airspace, flights may be further limited due to: the rapid growth in the number of commercial drones in the U.S.; the restriction of flights over secured airspace or geofencing [Zhang et al., 2020, 2021]. Furthermore, the FAA ConOps report further highlights civilian interactions as an area for future research [Authority, 2020].
Technological innovations relating to drones are likely to be double-sided in the coming years. In one direction, advancements in the speed, as well as flight and carrying capacities of drones, are bound to expand the range of logistical problems that may be addressed by UAVs as well as the operational assumptions inherent in modeling those problems. The latter may include networks with multiple depots, a fleet of unmanned vehicles, or the possibility for a vehicle to operate several drones, which we recommend for future research. It may be also interesting to analyze in the future the logistical impact of having ineligible customers for drone service, based on their geographical location, the weight-bulk ratio of the products ordered, or simply if they have not subscribed to this service.

Simultaneously, it is important for modeling and simulation-based studies to drive future drone designs. Beyond what is technologically feasible, many questions relating to the ideal functionality and flexibility of drone delivery systems – what the parcels,
networks, and depots of the future might require – remain unanswered. Furthermore, as drones become more closely integrated into civilian life, mitigating the risks associated with their failure in extreme conditions (e.g. inclement weather, engine failure) is critical to establishing them as reliable means of delivery in the perception of the public. In investigating these problems, we hope the optimization and analytic-based techniques at the heart of this dissertation will prove useful.
A.1 Cut-Set Generation Example

In what follows, we illustrate the cut-set constraint generation procedure described in §2.4.1.1 with a six-node example. The two cut-set constraints introduced therein are reproduced below for ease of reference:

\[
\begin{align*}
\sum_{(k,l) \in \delta^+(S)} x_{kl} & \geq w_i, & i \in V^*, S \subseteq V : 0 \in S, i \notin S, \\
\sum_{(k,l) \in \delta^+(S)} y_{kl} & \geq \sum_{j \in V} y_{ij}, & i \in V^*, S \subseteq V : 0 \in S, i \notin S.
\end{align*}
\]

<table>
<thead>
<tr>
<th>((x_{ij}, y_{ij}))</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>(0, 0)</td>
<td>(0, 0.31)</td>
<td>(0.28, 0)</td>
<td>(0, 0.58)</td>
<td>(0.72, 0.11)</td>
</tr>
<tr>
<td>1</td>
<td>(0, 0)</td>
<td>-</td>
<td>(0, 0.62)</td>
<td>(0.68, 0)</td>
<td>(0, 0.38)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>2</td>
<td>(0, 0)</td>
<td>(0, 0.72)</td>
<td>-</td>
<td>(0.04, 0.04)</td>
<td>(0, 0.04)</td>
<td>(0, 0.17)</td>
</tr>
<tr>
<td>3</td>
<td>(0.28, 0)</td>
<td>(0.68, 0)</td>
<td>(0.04, 0.04)</td>
<td>-</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>4</td>
<td>(0, 0.72)</td>
<td>(0, 0.28)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>-</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>5</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>-</td>
</tr>
</tbody>
</table>

Note that \(Z_{LPR}^* = 179.15\) at the outset of the procedure; Table A.1 shows the values of the \(x\)- and \(y\)-variables associated with this LP relaxation. For \(i \in V^*\), Table A.2 shows \(w_i\) and \(\sum_{j \in V | i \neq j} y_{ij}\), which form the right-hand side of Constraints (2.3a) and (2.3b), respectively. The former constraint is associated with the vehicle’s path...
(x-variables), and the latter is associated with the drone’s path (y-variables). In each case, we seek to construct $S \subset V : 0 \in S, j \notin S$ such that the violation between the left- and right-hand sides of the constraints are maximized. As noted in §2.4.1.1, the construction of the constraints as such makes them minimum cuts, which correspond to the solution of the maximum flow linear program.

Table A.2: Iteration 0, Minimum Cut Generation

| $i$ | $w_i$ | $\sum_{j\in V|i\neq j} y_{ij}$ | MF-X | MF-Y | $\text{Violation}_x$ | $\text{Violation}_y$ | $S_x$ | $S_y$ |
|-----|-------|---------------------------|------|------|----------------------|----------------------|------|------|
| 1   | 0.68  | 1.00                      | 0.28 | 0.59 | 0.40                 | 0.41                 | 0.5  | 0.4,5 |
| 2   | 0.04  | 0.96                      | 0.04 | 0.59 | -                    | 0.40                 | -    | 0,4,5 |
| 3   | 1     | 0.04                      | 0.28 | -    | 0.72                 | -                    | 0.5  | -    |
| 4   | 0     | 1.00                      | -    | 0.89 | -                    | -                    | -    | -    |
| 5   | 0.72  | 0.00                      | -    | 0.28 | -                    | 0.55                 | -    | 0,1,2,3,4 |

Table A.2 shows the solution of maximum flow problems for the x- (MF-X) and y-variables (MF-Y); a difference between the flow and the right-hand side of either constraint indicates that a violation of a cut-set $S$ (in x- or y-variables) exists. In other words, whenever MF-X (MF-Y) is less than $w_i \ (\sum_{j\in V|i\neq j} y_{ij})$, there exists $S$ such that Constraint (2.3a) (Constraint (2.3b)) constructed over $S$ is violated. Table A.2 lists these violations, and the associated construction of $S$ for each of the vehicle and drone paths. Note that for some nodes, there is no need to calculate the maximum flow since the value of the right-hand side of the cutset constraints precludes the existence of a valid inequality of that type (e.g. $w_4 = 0$). For other nodes, the maximum flow calculation may be skipped if it is clear that no cutset which produce a violation can exist (e.g. $x_{05} = 0.72 = w_5$).

We append a single deep cut, rather than all such violated constraints, to the formulation. Since node 3 provides the greatest violation in x-variables, the following constraint is appended to the formulation:
\[ x_{01} + x_{02} + x_{03} + x_{04} + x_{51} + x_{52} + x_{53} + x_{54} + \ge w_3. \]  

(A.2a)

Solving the model with Constraint (A.2a) appended yields \( Z_{LPR}^* = 187.34 \), an improvement of 4.6%. Table A.3 also shows that the routing variable values associated with the previous solution were cut off.

<table>
<thead>
<tr>
<th>((x_{ij}, y_{ij}))</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>(0, 0.24)</td>
<td>(0, 0.49)</td>
<td>(0.64, 0)</td>
<td>(0.09, 0.27)</td>
<td>(0.27, 0)</td>
</tr>
<tr>
<td>1</td>
<td>(0, 0.24)</td>
<td>-</td>
<td>(0, 0.09)</td>
<td>(0.08, 0.08)</td>
<td>(0, 0.17)</td>
<td>(0, 0.35)</td>
</tr>
<tr>
<td>2</td>
<td>(0, 0.49)</td>
<td>(0, 0.09)</td>
<td>-</td>
<td>(0.02, 0.02)</td>
<td>(0, 0.11)</td>
<td>(0, 0.29)</td>
</tr>
<tr>
<td>3</td>
<td>(0.91, 0)</td>
<td>(0.08, 0.08)</td>
<td>(0.02, 0.02)</td>
<td>-</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>4</td>
<td>(0.09, 0.27)</td>
<td>(0, 0.17)</td>
<td>(0, 0.11)</td>
<td>(0, 0)</td>
<td>-</td>
<td>(0, 0.36)</td>
</tr>
<tr>
<td>5</td>
<td>(0, 0)</td>
<td>(0, 0.35)</td>
<td>(0, 0.29)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>-</td>
</tr>
</tbody>
</table>

Table A.3: Iteration 1, Routing Variable Values

A.2 Derivation of Tightened Big “M” Values

In what follows, we derive the tightest possible \( M \) value for each of the constraints from §2.4.1.2. For ease of reference, each constraint has been reproduced below. We begin with several remarks that motivate later derivations. We assume \( \alpha \ge 1 \) (i.e. the drone travels at least as fast as the vehicle):

1. **Earliest Arrival**: Suppose \( i \in V^* \), and that \( i \) is the first node served by either carrier. Since \( \alpha \ge 1 \), the drone will never arrive at node \( i \) later than the vehicle. Therefore,

\[ a_i \ge f_{0i}, \tag{1} \]

Note that even if \( i \) cannot be serviced by the drone (e.g. \( f_{0i} > F \)), (1) is still valid, since \( \alpha \ge 1 \Rightarrow t_{0i} \ge f_{0i} \). Additionally, note \( a_i \le a_k \ \forall k \in V | i \neq k \).
2. **Earliest Departure:** Suppose again that \( i \) is the first node served by either carrier for \( i \in V^* \). Given (1) and Constraint (2.1n), the earliest departure from node \( i \) is

\[
d_i \geq f_{0i},
\]

(2) remains valid even if \( i \) may not be serviced by the drone (e.g. \( f_{0i} > F \)).

3. **Latest Departure:** Suppose \( j \in V^* \), and that \( j \) is the final node served by either carrier before returning to the depot. Let \( D_{\text{max}} \) be the objective value of a feasible solution. Since \( \alpha \geq 1 \), the latest time at which \( j \) may be serviced is

\[
d_j \leq D_{\text{max}} - f_{j0}.
\]

Note once again that even if \( j \) cannot be serviced by the drone (e.g. \( f_{j0} > F \)), (3) remains valid since \( \alpha \geq 1 \). Note also that \( d_j \geq d_k \forall k \in V \setminus j \neq k \).

\[
\sum_{i \in V} f_{ij}z_{ij} + \sum_{k \in V} f_{jk}z_{jk} \leq F + M_j^1w_j, \quad \forall j \in V^*, \tag{2.4a}
\]

**Constraint (2.4a):** As illustrated in Figure 1, Constraint (2.4a) is necessary when the drone delivers to node \( i \), meets with the vehicle at node \( j \), then delivers to node \( k \), such that \( z_{ij} = w_j = z_{jk} = 1 \). When \( w_j = 1 \), \( M_j^1 \) must be large enough to reset \( F \), the maximum flight duration of the drone. To scale \( M_j^1 \), we consider two cases:

1. **The drone may launch or land at node \( j \):** \( M_j^1 \) must be at least as large as the leg of a drone flight involving \( j \).
(a) If the drone lands at node \( j \), then the last leg of its flight from node \( i \in V \) has a duration no longer than
\[
\max_{i \in V} \frac{f_{ij}}{f_{ij} \leq F}
\]

(b) If the drone launches from node \( j \), then the first leg of its flight to node \( k \in V \) has a maximum duration of
\[
\max_{k \in V} \frac{f_{jk}}{f_{jk} \leq F}
\]

2. All flights to node \( j \) are infeasible: Under this case, Constraint (2.4a) is irrelevant; let \( M^1_j = 0 \). Furthermore, if no feasible flights exist to node \( j \), then the associated constraint relating to flights at \( j \) may be removed from the formulation.
\[
M^1_j = \max(0, \max_{i \in V} \frac{f_{ij}}{f_{ij} \leq F}, \max_{k \in V} \frac{f_{jk}}{f_{jk} \leq F}) \quad \Box
\]

\[
a_j \geq d_i + t_{ij} - M^2_{ij}(1 - x_{ij}), \quad \forall i, j \in V \mid i \neq j, \quad (2.4b)
\]

**Constraint (2.4b):** When \( x_{ij} = 0 \) for \( i, j \in V^* \), Constraint (2.4b) requires \( M^2_{ij} \geq d_i + t_{ij} \) such that \( a_j \geq f_{0j} \) in accordance with (1). We consider three cases for related \( M \) values:

1. \( i = 0 \): Under this setting, Constraint (2.4b) simplifies to \( a_j \geq d_0 + t_{0j} - M^2_{0j} \).
   By (1), \( a_j \geq f_{0j} \), which is achieved by letting
   \[
   M^2_{0j} = t_{0j} - f_{0j},
   \]
2. $j = 0$: Under this setting, Constraint (2.4b) simplifies to $a_0 \geq d_i + t_{i0} - M^2_{i0}$.

As Constraint (2.1s) shows, the arrival at the depot is determined by the latest departure time of a carrier from node $i$, plus the travel time from $i$ to the depot. Because the drone travels faster than the vehicle ($\alpha \geq 1$), the earliest arrival at the depot is $a_0 \geq d_i + f_{i0}$. To achieve this, let

$$M^2_{i0} = t_{i0} - f_{i0},$$

3. $i, j \in V^*$: Suppose that $i$ is the first node served by either carrier, and that $j$ is the last such node. Then Constraint (2.4b) simplifies to $a_j \geq d_i + t_{ij} - M^2_{ij}$. To ensure this holds, let

$$M^2_{ij} = \begin{cases} 
  t_{0j} - f_{0j}, & i = 0, \\
  t_{i0} - f_{i0}, & j = 0, \\
  (D_{\text{max}} - f_{i0} + t_{ij}) - f_{0j}, & \text{otherwise} 
\end{cases}$$

$$a_j \geq d_i + f_{ij} - M^3_{ij}(1 - z_{ij}) - M^4_{ij}w_j, \quad \forall i, j \in V | i \neq j, \quad (2.4c)$$

**Constraint (2.4c):** This constraint is necessary for nodes $i, j \in V$ when the drone lands at $j$ after having served $i$ ($z_{ij} = 1$), and the vehicle rendezvouses with the drone at $j$ ($w_j = 1$). Since Constraint (2.4c) associates a value of $M$ with two different sets of variables, we consider the following cases:
1. \( z_{ij} = 1, w_j = 0 \): This case applies when the drone serves node \( j \), in which case \( a_j \geq d_i + f_{ij} - M_{ij}^3 \). Since the arrival at \( j \) should be set by the drone’s path, let

\[ M_{ij}^3 \geq 0, \]

2. \( z_{ij} = 0, w_j = 1 \): This case applies when the vehicle serves node \( j \), in which case Constraint (2.4c) is irrelevant. Note that Constraint (2.4c) simplifies to \( a_j \geq d_i + f_{ij} - M_{ij}^3 - M_{ij}^4 \). Since \( w_j = 1 \) \( \implies \) \( a_j \geq t_{0j} \), we set the following \( M \) values

\[ M_{ij}^3 \geq 0, \]

\[ M_{ij}^4 \geq (D_{\text{max}} - f_{j0} + f_{ij}) - t_{0j}, \]

3. \( z_{ij} = w_j = 1 \): This case applies when the drone and vehicle rendezvous at node \( j \), in which case \( a_j \) marks the arrival time of the vehicle. The constraint simplifies to \( a_j \geq d_i + f_{ij} - M_{ij}^4 \). Consider the following sub-cases:

(a) \( i = 0 \): While mathematically valid, this setting is sub-optimal in the context of the TSP-D. If \( z_{0j} = 1 \), then the drone serves node \( j \in V^* \) directly from the depot. Therefore the drone should land at \( k \in V^* | k \neq j \) in order to rendezvous with the vehicle. If, however, \( w_j = 1 \), and the two carriers rendezvous, then it will always be sub-optimal to have served \( j \) via the drone. For completeness, \( 1 \) \( \implies \) \( a_j \geq t_{0j} \). Therefore,

\[ M_{0j}^4 = (f_{0j} - t_{0j}), \]

(b) \( j = 0 \): Unlike customer nodes, the arrival at the depot is set by the latest carrier. Thus \( z_{i0} = 1 \) \( \implies \) \( a_0 \geq d_i + f_{i0} - M_{i0}^4 \). Accordingly, let

\[ M_{i0}^4 \geq 0, \]
(c) \(i, j \in V^*\): Under this setting, Constraint (2.4b) simplifies to \(a_j \geq d_i + f_{ij} - M_{i0}^4\). Therefore,

\[M_{ij}^4 = (D_{\text{max}} - f_{j0} + f_{ij}) - t_{0j}\]

4. \(z_{ij} = w_j = 0\): Since \(w_j = 0\), the drone must deliver to node \(j\), and the constraint simplifies to \(a_j \geq d_i + f_{ij} - M_{ij}^3\). Consider the following cases to set the arrival time accordingly:

(a) \(i = 0\): (1) \(\implies a_j \geq f_{0j}\), and Constraint (2.4b) simplifies to \(a_j \geq d_0 + f_{0j} - M_{0j}^3\). Therefore let

\[M_{0j}^3 \geq 0,\]

(b) \(j = 0\): Since the vehicle originates at the depot, \(w_0 = 1\) by construction and this case is irrelevant. For completeness, let

\[M_{i0}^3 \geq 0,\]

(c) \(i, j \in V^*\): By (1), \(a_j \geq f_{0j}\), and Constraint (2.4b) simplifies to \(a_j \geq d_i + f_{ij} - M_{ij}^3\). Therefore,

\[M_{ij}^3 = (D_{\text{max}} - f_{j0} + f_{ij}) - f_{0j},\]

\[M_{ij}^3 = \begin{cases} 0, & i = 0 \text{ or } j = 0, \\ (D_{\text{max}} - f_{j0} + f_{ij}) - f_{0j}, & \text{otherwise,} \end{cases}\]

\[M_{ij}^4 = \begin{cases} (f_{0j} - t_{0j}), & i = 0, \\ 0, & j = 0, \\ (D_{\text{max}} - f_{j0} + f_{ij}) - t_{0j}, & \text{otherwise} \end{cases}\]
\[ d_j \geq d_i + f_{ij} - M_{ij}^5(1 - z_{ij}), \quad \forall i \in V, j \in V^*, \quad (2.4d) \]

**Constraint (2.4d):** When the drone serves node \( j \in V^* \), Constraint (2.4d) sets the departure time accordingly. We consider two cases:

1. \( i = 0 \): Note that \( d_0 = 0 \) by construction. Under this setting, Constraint (2.4d) simplifies to \( d_j \geq f_{0j} - M_{0j}^5 \). By (2), \( d_j \geq f_{0j} \). Accordingly, let

\[ M_{0j}^5 \geq 0, \]

2. \( i \in V^* \): Under this setting, Constraint (2.4d) similarly simplifies to \( d_j \geq d_i + f_{ij} - M_{ij}^5 \). By (2), \( d_j \geq f_{0j} \); let

\[ M_{ij}^5 = (D_{\text{max}} - f_{i0} + f_{ij}) - f_{0j}, \]

\[ M_{ij}^5 = \begin{cases} 
0, & i = 0, \\
(D_{\text{max}} - f_{i0} + f_{ij}) - f_{0j}, & \text{otherwise} \quad \Box 
\end{cases} \]

\[ e_j \geq d_j - (d_i + t_{ij}) - M_{ij}^6(1 - x_{ij}), \quad \forall i \in V, j \in V^* | i \neq j, \quad (2.4e) \]

**Constraint (2.4e):** When the drone and vehicle rendezvous at node \( j \), Constraint
(2.4e) sets the waiting time for the vehicle. If \( x_{ij} = 0 \), then \( M_6^0 \) must be large enough to set waiting time equal to 0. Consider the following cases:

1. \( i = 0 \): Under this setting, Constraint (2.4e) simplifies to \( e_j \geq d_j - (d_0 + t_{0j}) - M_0^6 \).

   By (3), \( d_j \leq D_{\text{max}} - f_{j0} \); thus let

   \[
   M_0^6 = (D_{\text{max}} - f_{j0}) - t_{0j},
   \]

2. \( i, j \in V^* \): In this setting, Constraint (2.4e) simplifies to \( e_j \geq d_j - (d_i + t_{ij}) - M_{ij}^6 \).

   By (3), \( d_j \leq D_{\text{max}} - f_{j0} \); thus let

   \[
   M_{ij}^6 = (D_{\text{max}} - f_{j0}),
   \]

   \[
   M_{ij}^6 = \begin{cases} 
   (D_{\text{max}} - f_{j0}) - t_{0j}, & i = 0, \\
   (D_{\text{max}} - f_{j0}), & \text{otherwise}
   \end{cases}
   \]

   \[
   e_0 \geq (d_j + f_{j0}) - (d_i + t_{i0}) - M_i^7 (1 - x_{i0}), \quad i, j \in V^* | i \neq j, \tag{2.4f}
   \]

**Constraint (2.4f):** Constraint (2.4f) is a special case of Constraint (2.4e) applied to the depot. When \( x_{i0} = 0 \), Constraint (2.4f) simplifies to \( e_0 \geq (d_j + f_{j0}) - (d_i + t_{i0}) - M_i^7 \).

Suppose that \( d_j \) represents the latest departure times for the carriers. By construction, \( d_j \geq d_k \forall k \in V \). We consider this case alone, since in any case when \( d_i \geq d_j \), Constraint (2.4e) becomes irrelevant. By (3), \( d_j \leq D_{\text{max}} - f_{j0} \); to ensure \( e_0 \geq 0 \), let

   \[
   M_i^7 = D_{\text{max}} - t_{i0}, \quad \forall i, j \in V^* | i \neq j \quad \Box
   \]
Constraint (2.4g): Finally, the value of all departure times may be bounded above. By (3), $d_j \leq D_{\text{max}} - f_{j0}$ for $j \in V^*$. As we have written, since $\alpha \geq 1$, this bound remains valid even if $f_{j0} > F$. Therefore, let

$$M = D_{\text{max}} - \min_{i,j \in V^*} f_{ij}, t_{j0} \quad \square$$

A.3 Detailed Results of Computational Study

Our data is organized by size and topography; for each instance, we show: $Z^*$, the best-known objective value; CPU (s), the total computation time; CPU4 (s), the computation time for Step 4 (which solves for the optimal solution) in our solution procedure; LP1, LP2, and RMIP Gap are the deviation percentages of each lower bound from $Z^*$; MIP Gap, the relative optimality gap at termination; and B&B, the number of nodes explored in the B&B/C search. Instances with optimal solutions have their CPU (s) time in bold.
Table A.4: Computational Results for $N = 16, 20,$ and $24$ for SC instances

<table>
<thead>
<tr>
<th>N=16</th>
<th>( Z^* )</th>
<th>CPU (s)</th>
<th>CPU4 (s)</th>
<th>LP1 Gap %</th>
<th>LP2 Gap %</th>
<th>RMIP Gap %</th>
<th>MIP Gap %</th>
<th>B&amp;B</th>
</tr>
</thead>
<tbody>
<tr>
<td>SingleCenter-61-n20</td>
<td>415.75</td>
<td>17.6</td>
<td>3.0</td>
<td>34.9</td>
<td>20.6</td>
<td>7.5</td>
<td>0.0</td>
<td>2,096</td>
</tr>
<tr>
<td>SingleCenter-62-n20</td>
<td>769.92</td>
<td>30.2</td>
<td>6.6</td>
<td>32.8</td>
<td>26.1</td>
<td>8.1</td>
<td>0.0</td>
<td>7,462</td>
</tr>
<tr>
<td>SingleCenter-63-n20</td>
<td>604.81</td>
<td>7.3</td>
<td>1.3</td>
<td>12.8</td>
<td>6.6</td>
<td>1.3</td>
<td>0.0</td>
<td>1,772</td>
</tr>
<tr>
<td>SingleCenter-64-n20</td>
<td>499.07</td>
<td>18.1</td>
<td>3.1</td>
<td>37.8</td>
<td>20.9</td>
<td>9.2</td>
<td>0.0</td>
<td>8,049</td>
</tr>
<tr>
<td>SingleCenter-65-n20</td>
<td>600.20</td>
<td>7.1</td>
<td>1.8</td>
<td>30.8</td>
<td>20.8</td>
<td>3.8</td>
<td>0.0</td>
<td>2,129</td>
</tr>
<tr>
<td>SingleCenter-66-n20</td>
<td>441.53</td>
<td>38.0</td>
<td>16.8</td>
<td>32.3</td>
<td>20.4</td>
<td>12.2</td>
<td>0.0</td>
<td>2,762</td>
</tr>
<tr>
<td>SingleCenter-67-n20</td>
<td>448.20</td>
<td>13.1</td>
<td>4.1</td>
<td>19.7</td>
<td>15.0</td>
<td>7.6</td>
<td>0.0</td>
<td>6,851</td>
</tr>
<tr>
<td>SingleCenter-68-n20</td>
<td>516.89</td>
<td>22.9</td>
<td>12.1</td>
<td>16.9</td>
<td>10.9</td>
<td>5.5</td>
<td>0.0</td>
<td>9,588</td>
</tr>
<tr>
<td>SingleCenter-69-n20</td>
<td>597.03</td>
<td>52.8</td>
<td>18.2</td>
<td>35.8</td>
<td>12.6</td>
<td>7.2</td>
<td>0.0</td>
<td>17,684</td>
</tr>
<tr>
<td>SingleCenter-70-n20</td>
<td>395.48</td>
<td>35.6</td>
<td>8.4</td>
<td>43.7</td>
<td>29.4</td>
<td>12.2</td>
<td>0.0</td>
<td>4,752</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>N=20</th>
<th>( Z^* )</th>
<th>CPU (s)</th>
<th>CPU4 (s)</th>
<th>LP1 Gap %</th>
<th>LP2 Gap %</th>
<th>RMIP Gap %</th>
<th>MIP Gap %</th>
<th>B&amp;B</th>
</tr>
</thead>
<tbody>
<tr>
<td>SingleCenter-61-n20</td>
<td>523.73</td>
<td>288.9</td>
<td>118.5</td>
<td>28.0</td>
<td>23.0</td>
<td>7.5</td>
<td>0.0</td>
<td>43,657</td>
</tr>
<tr>
<td>SingleCenter-62-n20</td>
<td>466.55</td>
<td>124.4</td>
<td>24.0</td>
<td>48.6</td>
<td>43.1</td>
<td>8.6</td>
<td>0.0</td>
<td>7,919</td>
</tr>
<tr>
<td>SingleCenter-63-n20</td>
<td>673.06</td>
<td>80.0</td>
<td>46.4</td>
<td>27.7</td>
<td>22.7</td>
<td>6.1</td>
<td>0.0</td>
<td>34,636</td>
</tr>
<tr>
<td>SingleCenter-64-n20</td>
<td>548.98</td>
<td>3,185.7</td>
<td>3,111.5</td>
<td>48.5</td>
<td>38.9</td>
<td>8.1</td>
<td>0.0</td>
<td>1,610,507</td>
</tr>
<tr>
<td>SingleCenter-65-n20</td>
<td>570.25</td>
<td>138.4</td>
<td>52.1</td>
<td>52.1</td>
<td>46.4</td>
<td>6.1</td>
<td>0.0</td>
<td>44,646</td>
</tr>
<tr>
<td>SingleCenter-66-n20</td>
<td>448.77</td>
<td>65.9</td>
<td>30.2</td>
<td>41.7</td>
<td>24.0</td>
<td>7.8</td>
<td>0.0</td>
<td>13,764</td>
</tr>
<tr>
<td>SingleCenter-67-n20</td>
<td>524.77</td>
<td>185.1</td>
<td>153.7</td>
<td>21.8</td>
<td>15.0</td>
<td>7.2</td>
<td>0.0</td>
<td>134,576</td>
</tr>
<tr>
<td>SingleCenter-68-n20</td>
<td>535.79</td>
<td>6,687.8</td>
<td>6,459.9</td>
<td>29.7</td>
<td>17.6</td>
<td>8.5</td>
<td>0.0</td>
<td>3,955,480</td>
</tr>
<tr>
<td>SingleCenter-69-n20</td>
<td>639.26</td>
<td>914.6</td>
<td>537.5</td>
<td>37.2</td>
<td>24.1</td>
<td>6.2</td>
<td>0.0</td>
<td>134,576</td>
</tr>
<tr>
<td>SingleCenter-70-n20</td>
<td>442.04</td>
<td>11,696.6</td>
<td>10,000.0</td>
<td>61.3</td>
<td>45.8</td>
<td>16.7</td>
<td>9.8</td>
<td>1,804,176</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>N=24</th>
<th>( Z^* )</th>
<th>CPU (s)</th>
<th>CPU4 (s)</th>
<th>LP1 Gap %</th>
<th>LP2 Gap %</th>
<th>RMIP Gap %</th>
<th>MIP Gap %</th>
<th>B&amp;B</th>
</tr>
</thead>
<tbody>
<tr>
<td>SingleCenter-71-n50</td>
<td>523.73</td>
<td>288.9</td>
<td>118.5</td>
<td>28.0</td>
<td>23.0</td>
<td>7.5</td>
<td>0.0</td>
<td>43,657</td>
</tr>
<tr>
<td>SingleCenter-72-n50</td>
<td>466.55</td>
<td>124.4</td>
<td>24.0</td>
<td>48.6</td>
<td>43.1</td>
<td>8.6</td>
<td>0.0</td>
<td>7,919</td>
</tr>
<tr>
<td>SingleCenter-73-n50</td>
<td>673.06</td>
<td>80.0</td>
<td>46.4</td>
<td>27.7</td>
<td>22.7</td>
<td>6.1</td>
<td>0.0</td>
<td>34,636</td>
</tr>
<tr>
<td>SingleCenter-74-n50</td>
<td>548.98</td>
<td>3,185.7</td>
<td>3,111.5</td>
<td>48.5</td>
<td>38.9</td>
<td>8.1</td>
<td>0.0</td>
<td>1,610,507</td>
</tr>
<tr>
<td>SingleCenter-75-n50</td>
<td>570.25</td>
<td>138.4</td>
<td>52.1</td>
<td>52.1</td>
<td>46.4</td>
<td>6.1</td>
<td>0.0</td>
<td>44,646</td>
</tr>
<tr>
<td>SingleCenter-76-n50</td>
<td>448.77</td>
<td>65.9</td>
<td>30.2</td>
<td>41.7</td>
<td>24.0</td>
<td>7.8</td>
<td>0.0</td>
<td>13,764</td>
</tr>
<tr>
<td>SingleCenter-77-n50</td>
<td>524.77</td>
<td>185.1</td>
<td>153.7</td>
<td>21.8</td>
<td>15.0</td>
<td>7.2</td>
<td>0.0</td>
<td>134,576</td>
</tr>
<tr>
<td>SingleCenter-78-n50</td>
<td>535.79</td>
<td>6,687.8</td>
<td>6,459.9</td>
<td>29.7</td>
<td>17.6</td>
<td>8.5</td>
<td>0.0</td>
<td>3,955,480</td>
</tr>
<tr>
<td>SingleCenter-79-n50</td>
<td>639.26</td>
<td>914.6</td>
<td>537.5</td>
<td>37.2</td>
<td>24.1</td>
<td>6.2</td>
<td>0.0</td>
<td>134,576</td>
</tr>
<tr>
<td>SingleCenter-80-n50</td>
<td>442.04</td>
<td>11,696.6</td>
<td>10,000.0</td>
<td>61.3</td>
<td>45.8</td>
<td>16.7</td>
<td>9.8</td>
<td>1,804,176</td>
</tr>
</tbody>
</table>

*This instance had an MIP Gap of 1.6% after 10,000 CPU seconds, before eventually proving an optimal solution with extended computational effort.
Table A.5: Computational Results for N = 28 and 32 for SC instances

<table>
<thead>
<tr>
<th>Instance</th>
<th>CPU (s)</th>
<th>CPU4 (s)</th>
<th>LP1 Gap %</th>
<th>LP2 Gap %</th>
<th>RMIP Gap %</th>
<th>MIP Gap %</th>
<th>B&amp;B</th>
</tr>
</thead>
<tbody>
<tr>
<td>N=28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SingleCenter-71-n50</td>
<td>12.089.2</td>
<td>10,000.0</td>
<td>63.2</td>
<td>30.9</td>
<td>28.4</td>
<td>9.5</td>
<td>1,705,730</td>
</tr>
<tr>
<td>SingleCenter-75-n50</td>
<td>10.261.3</td>
<td>10,000.0</td>
<td>44.7</td>
<td>38.8</td>
<td>7.9</td>
<td>2.2</td>
<td>2,134,455</td>
</tr>
<tr>
<td>SingleCenter-103-n50</td>
<td>11.228.0</td>
<td>10,000.0</td>
<td>70.1</td>
<td>70.1</td>
<td>40.2</td>
<td>13.5</td>
<td>806,345</td>
</tr>
<tr>
<td>SingleCenter-71-n70</td>
<td>6,479,1</td>
<td>6,221.7</td>
<td>49.4</td>
<td>34.6</td>
<td>11.5</td>
<td>0.0</td>
<td>2,086,405</td>
</tr>
<tr>
<td>SingleCenter-75-n70</td>
<td>10,471.7</td>
<td>10,000.0</td>
<td>38.8</td>
<td>33.1</td>
<td>10.9</td>
<td>1.3</td>
<td>4,131,449</td>
</tr>
<tr>
<td>SingleCenter-103-n70</td>
<td>11,581.0</td>
<td>10,000.0</td>
<td>54.6</td>
<td>45.9</td>
<td>14.6</td>
<td>4.2</td>
<td>1,915,610</td>
</tr>
<tr>
<td>SingleCenter-71-n90</td>
<td>500.55</td>
<td>10,000.0</td>
<td>35.1</td>
<td>29.5</td>
<td>16.0</td>
<td>6.2</td>
<td>1,256,445</td>
</tr>
<tr>
<td>SingleCenter-75-n90</td>
<td>11,008.7</td>
<td>10,000.0</td>
<td>36.7</td>
<td>31.8</td>
<td>15.7</td>
<td>6.9</td>
<td>1,435,444</td>
</tr>
<tr>
<td>SingleCenter-103-n90</td>
<td>11,685.4</td>
<td>10,000.0</td>
<td>58.6</td>
<td>46.0</td>
<td>16.3</td>
<td>8.2</td>
<td>1,874,655</td>
</tr>
<tr>
<td>SingleCenter-80-n90</td>
<td>914.01</td>
<td>10,806.0</td>
<td>30.6</td>
<td>31.1</td>
<td>9.3</td>
<td>4.6</td>
<td>1,395,724</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instance</th>
<th>CPU (s)</th>
<th>CPU4 (s)</th>
<th>LP1 Gap %</th>
<th>LP2 Gap %</th>
<th>RMIP Gap %</th>
<th>MIP Gap %</th>
<th>B&amp;B</th>
</tr>
</thead>
<tbody>
<tr>
<td>N=32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SingleCenter-71-n50</td>
<td>11,381.0</td>
<td>10,000.0</td>
<td>66.9</td>
<td>58.4</td>
<td>34.9</td>
<td>22.3</td>
<td>573,855</td>
</tr>
<tr>
<td>SingleCenter-73-n50</td>
<td>10,628.8</td>
<td>10,000.0</td>
<td>46.3</td>
<td>41.8</td>
<td>10.1</td>
<td>4.1</td>
<td>1,322,986</td>
</tr>
<tr>
<td>SingleCenter-75-n50</td>
<td>11,358.8</td>
<td>10,000.0</td>
<td>77.1</td>
<td>77.1</td>
<td>41.6</td>
<td>15.5</td>
<td>550,448</td>
</tr>
<tr>
<td>SingleCenter-77-n50</td>
<td>11,401.4</td>
<td>10,000.0</td>
<td>48.8</td>
<td>35.6</td>
<td>14.2</td>
<td>5.2</td>
<td>955,347</td>
</tr>
<tr>
<td>SingleCenter-79-n50</td>
<td>11,092.9</td>
<td>10,000.0</td>
<td>40.5</td>
<td>40.1</td>
<td>18.6</td>
<td>2.8</td>
<td>1,091,735</td>
</tr>
<tr>
<td>SingleCenter-81-n50</td>
<td>11,497.6</td>
<td>10,000.0</td>
<td>59.6</td>
<td>51.0</td>
<td>28.1</td>
<td>10.0</td>
<td>773,102</td>
</tr>
<tr>
<td>SingleCenter-77-n70</td>
<td>10,611.0</td>
<td>10,000.0</td>
<td>44.9</td>
<td>40.6</td>
<td>23.1</td>
<td>9.7</td>
<td>699,021</td>
</tr>
<tr>
<td>SingleCenter-79-n70</td>
<td>10,764.5</td>
<td>10,000.0</td>
<td>44.4</td>
<td>41.3</td>
<td>24.1</td>
<td>14.4</td>
<td>444,489</td>
</tr>
<tr>
<td>SingleCenter-81-n70</td>
<td>11,410.9</td>
<td>10,000.0</td>
<td>64.6</td>
<td>56.3</td>
<td>29.5</td>
<td>10.4</td>
<td>713,297</td>
</tr>
<tr>
<td>SingleCenter-79-n90</td>
<td>11,146.2</td>
<td>10,000.0</td>
<td>32.9</td>
<td>28.1</td>
<td>10.8</td>
<td>9.3</td>
<td>765,398</td>
</tr>
</tbody>
</table>

Table A.6: Computational Results for N = 16, 20, and 24 for DC instances

<table>
<thead>
<tr>
<th>Instance</th>
<th>CPU (s)</th>
<th>CPU4 (s)</th>
<th>LP1 Gap %</th>
<th>LP2 Gap %</th>
<th>RMIP Gap %</th>
<th>MIP Gap %</th>
<th>B&amp;B</th>
</tr>
</thead>
<tbody>
<tr>
<td>N=16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DoubleCenter-61-n20</td>
<td>848.53</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DoubleCenter-63-n20</td>
<td>785.61</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DoubleCenter-65-n20</td>
<td>911.14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DoubleCenter-67-n20</td>
<td>991.59</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DoubleCenter-69-n20</td>
<td>662.94</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DoubleCenter-71-n20</td>
<td>698.76</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DoubleCenter-73-n20</td>
<td>643.99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DoubleCenter-75-n20</td>
<td>608.55</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DoubleCenter-77-n20</td>
<td>30.6</td>
<td>14.2</td>
<td>35.4</td>
<td>22.6</td>
<td>7.2</td>
<td>0.0</td>
<td>401,174</td>
</tr>
<tr>
<td>DoubleCenter-79-n20</td>
<td>143.5</td>
<td>85.0</td>
<td>35.4</td>
<td>22.9</td>
<td>8.0</td>
<td>0.0</td>
<td>666,466</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instance</th>
<th>CPU (s)</th>
<th>CPU4 (s)</th>
<th>LP1 Gap %</th>
<th>LP2 Gap %</th>
<th>RMIP Gap %</th>
<th>MIP Gap %</th>
<th>B&amp;B</th>
</tr>
</thead>
<tbody>
<tr>
<td>N=20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DoubleCenter-61-n20</td>
<td>868.90</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DoubleCenter-63-n20</td>
<td>808.86</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DoubleCenter-65-n20</td>
<td>964.34</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DoubleCenter-67-n20</td>
<td>1,083.51</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DoubleCenter-69-n20</td>
<td>696.40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DoubleCenter-71-n20</td>
<td>126.6</td>
<td>15.9</td>
<td>51.6</td>
<td>18.5</td>
<td>24.1</td>
<td>8.7</td>
<td>48,446</td>
</tr>
<tr>
<td>DoubleCenter-73-n20</td>
<td>63.3</td>
<td>12.9</td>
<td>21.7</td>
<td>15.2</td>
<td>8.0</td>
<td>0.0</td>
<td>23,003</td>
</tr>
<tr>
<td>DoubleCenter-75-n20</td>
<td>45.8</td>
<td>11.1</td>
<td>47.4</td>
<td>19.4</td>
<td>8.7</td>
<td>0.0</td>
<td>5,825</td>
</tr>
<tr>
<td>DoubleCenter-77-n20</td>
<td>2,613.6</td>
<td>1,081.7</td>
<td>44.5</td>
<td>44.5</td>
<td>27.5</td>
<td>0.0</td>
<td>283,343</td>
</tr>
<tr>
<td>DoubleCenter-79-n20</td>
<td>794.15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DoubleCenter-81-n20</td>
<td>1,121.5</td>
<td>776.9</td>
<td>46.5</td>
<td>46.3</td>
<td>11.7</td>
<td>0.0</td>
<td>634,301</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instance</th>
<th>CPU (s)</th>
<th>CPU4 (s)</th>
<th>LP1 Gap %</th>
<th>LP2 Gap %</th>
<th>RMIP Gap %</th>
<th>MIP Gap %</th>
<th>B&amp;B</th>
</tr>
</thead>
<tbody>
<tr>
<td>N=24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DoubleCenter-71-n20</td>
<td>934.95</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DoubleCenter-73-n20</td>
<td>1,019.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DoubleCenter-75-n20</td>
<td>1,000.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DoubleCenter-77-n20</td>
<td>989.45</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DoubleCenter-79-n20</td>
<td>989.45</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DoubleCenter-81-n20</td>
<td>989.45</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DoubleCenter-83-n20</td>
<td>989.45</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

130
Table A.7: Computational Results for $N = 28$ and 32 for DC instances

<table>
<thead>
<tr>
<th>Instance</th>
<th>$Z^*$</th>
<th>$N=28$ CPU (s)</th>
<th>$N=28$ CPU4 (s)</th>
<th>LP1 Gap %</th>
<th>LP2 Gap %</th>
<th>RMIP Gap %</th>
<th>MIP Gap %</th>
<th>B&amp;B</th>
</tr>
</thead>
<tbody>
<tr>
<td>DoubleCenter-71-n50</td>
<td>941.01</td>
<td>10,390.6</td>
<td>10,000.0</td>
<td>34.8</td>
<td>32.9</td>
<td>21.4</td>
<td>9.8</td>
<td>1,804,176</td>
</tr>
<tr>
<td>DoubleCenter-72-n50</td>
<td>114.21</td>
<td>10,279.1</td>
<td>10,000.0</td>
<td>43.6</td>
<td>39.7</td>
<td>26.5</td>
<td>0.0</td>
<td>21,362,396</td>
</tr>
<tr>
<td>DoubleCenter-73-n50</td>
<td>823.80</td>
<td>10,471.4</td>
<td>10,000.0</td>
<td>49.3</td>
<td>45.9</td>
<td>29.0</td>
<td>0.0</td>
<td>5,416,906</td>
</tr>
<tr>
<td>DoubleCenter-74-n50</td>
<td>795.71</td>
<td>11,369.5</td>
<td>10,000.0</td>
<td>54.3</td>
<td>49.7</td>
<td>36.7</td>
<td>0.0</td>
<td>836,620</td>
</tr>
<tr>
<td>DoubleCenter-75-n50</td>
<td>994.42</td>
<td>10,945.8</td>
<td>10,000.0</td>
<td>42.7</td>
<td>37.4</td>
<td>25.6</td>
<td>0.0</td>
<td>360,984</td>
</tr>
<tr>
<td>DoubleCenter-76-n50</td>
<td>785.91</td>
<td>449.1</td>
<td>87.6</td>
<td>28.1</td>
<td>26.4</td>
<td>9.1</td>
<td>0.0</td>
<td>1,089,926</td>
</tr>
<tr>
<td>DoubleCenter-77-n50</td>
<td>889.97</td>
<td>11,100.5</td>
<td>10,000.0</td>
<td>43.1</td>
<td>21.3</td>
<td>10.1</td>
<td>0.0</td>
<td>2,212,864</td>
</tr>
<tr>
<td>DoubleCenter-78-n50</td>
<td>1,140.45</td>
<td>2,610.6</td>
<td>2,320.9</td>
<td>14.8</td>
<td>13.4</td>
<td>8.6</td>
<td>7.8</td>
<td>1,344,776</td>
</tr>
<tr>
<td>DoubleCenter-79-n50</td>
<td>892.82</td>
<td>805.3</td>
<td>188.7</td>
<td>33.0</td>
<td>33.0</td>
<td>9.6</td>
<td>0.0</td>
<td>41,242</td>
</tr>
<tr>
<td>DoubleCenter-80-n50</td>
<td>5,890.2</td>
<td>4,389.1</td>
<td>43.8</td>
<td>43.8</td>
<td>21.2</td>
<td>0.0</td>
<td>821,025</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instance</th>
<th>$Z^*$</th>
<th>$N=32$ CPU (s)</th>
<th>$N=32$ CPU4 (s)</th>
<th>LP1 Gap %</th>
<th>LP2 Gap %</th>
<th>RMIP Gap %</th>
<th>MIP Gap %</th>
<th>B&amp;B</th>
</tr>
</thead>
<tbody>
<tr>
<td>DoubleCenter-71-n50</td>
<td>1,117.80</td>
<td>10,572.7</td>
<td>10,000.0</td>
<td>28.0</td>
<td>26.6</td>
<td>17.3</td>
<td>14.0</td>
<td>2,536,894</td>
</tr>
<tr>
<td>DoubleCenter-72-n50</td>
<td>858.87</td>
<td>10,296.3</td>
<td>10,000.0</td>
<td>42.1</td>
<td>42.0</td>
<td>31.3</td>
<td>16.3</td>
<td>1,009,485</td>
</tr>
<tr>
<td>DoubleCenter-73-n50</td>
<td>875.38</td>
<td>11,058.2</td>
<td>10,000.0</td>
<td>46.9</td>
<td>41.7</td>
<td>21.7</td>
<td>21.7</td>
<td>1,544,964</td>
</tr>
<tr>
<td>DoubleCenter-74-n50</td>
<td>817.20</td>
<td>10,820.8</td>
<td>10,000.0</td>
<td>58.5</td>
<td>53.3</td>
<td>49.1</td>
<td>32.3</td>
<td>1,340,667</td>
</tr>
<tr>
<td>DoubleCenter-75-n50</td>
<td>1,098.95</td>
<td>10,153.1</td>
<td>10,000.0</td>
<td>54.5</td>
<td>52.4</td>
<td>35.9</td>
<td>14.9</td>
<td>449,438</td>
</tr>
<tr>
<td>DoubleCenter-76-n50</td>
<td>985.70</td>
<td>805.3</td>
<td>188.7</td>
<td>33.0</td>
<td>33.0</td>
<td>9.6</td>
<td>0.0</td>
<td>41,242</td>
</tr>
<tr>
<td>DoubleCenter-77-n50</td>
<td>1,085.04</td>
<td>11,147.8</td>
<td>10,000.0</td>
<td>39.1</td>
<td>36.2</td>
<td>23.4</td>
<td>19.5</td>
<td>1,265,497</td>
</tr>
<tr>
<td>DoubleCenter-78-n50</td>
<td>1,101.60</td>
<td>11,219.8</td>
<td>10,000.0</td>
<td>26.6</td>
<td>20.7</td>
<td>9.9</td>
<td>5.0</td>
<td>2,142,499</td>
</tr>
<tr>
<td>DoubleCenter-79-n50</td>
<td>905.13</td>
<td>11,983.3</td>
<td>10,000.0</td>
<td>45.0</td>
<td>40.2</td>
<td>17.9</td>
<td>3.5</td>
<td>2,289,784</td>
</tr>
<tr>
<td>DoubleCenter-80-n50</td>
<td>828.82</td>
<td>6,892.0</td>
<td>6,036.2</td>
<td>54.8</td>
<td>52.1</td>
<td>21.2</td>
<td>0.0</td>
<td>826,497</td>
</tr>
</tbody>
</table>
Table A.8: Computational Results for $N = 16$, 20, and 24 for U instances

### $N = 16$

<table>
<thead>
<tr>
<th>Instance</th>
<th>$Z^*$</th>
<th>CPU (s)</th>
<th>CPU4 (s)</th>
<th>LP1 Gap %</th>
<th>LP2 Gap %</th>
<th>RMIP Gap %</th>
<th>MIP Gap %</th>
<th>B&amp;B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform-61-n20</td>
<td>292.16</td>
<td>48.3</td>
<td>15.4</td>
<td>44.6</td>
<td>34.0</td>
<td>11.5</td>
<td>0.0</td>
<td>9,324</td>
</tr>
<tr>
<td>Uniform-62-n20</td>
<td>280.14</td>
<td>184.9</td>
<td>87.8</td>
<td>61.6</td>
<td>61.6</td>
<td>10.2</td>
<td>0.0</td>
<td>11,082</td>
</tr>
<tr>
<td>Uniform-63-n20</td>
<td>265.49</td>
<td>17.9</td>
<td>3.4</td>
<td>37.6</td>
<td>13.5</td>
<td>7.8</td>
<td>0.0</td>
<td>1,945</td>
</tr>
<tr>
<td>Uniform-64-n20</td>
<td>307.38</td>
<td>138.8</td>
<td>45.1</td>
<td>51.9</td>
<td>51.9</td>
<td>17.7</td>
<td>0.0</td>
<td>25,733</td>
</tr>
<tr>
<td>Uniform-65-n20</td>
<td>305.31</td>
<td>658.7</td>
<td>363.6</td>
<td>52.4</td>
<td>50.5</td>
<td>11.7</td>
<td>0.0</td>
<td>99,061</td>
</tr>
<tr>
<td>Uniform-66-n20</td>
<td>365.11</td>
<td>91.8</td>
<td>34.1</td>
<td>38.4</td>
<td>38.4</td>
<td>9.6</td>
<td>0.0</td>
<td>3,089</td>
</tr>
<tr>
<td>Uniform-67-n20</td>
<td>325.36</td>
<td>24.9</td>
<td>11.2</td>
<td>11.2</td>
<td>11.2</td>
<td>9.6</td>
<td>0.0</td>
<td>3,716</td>
</tr>
<tr>
<td>Uniform-68-n20</td>
<td>344.50</td>
<td>124.3</td>
<td>25.5</td>
<td>42.1</td>
<td>42.1</td>
<td>13.8</td>
<td>0.0</td>
<td>9,013</td>
</tr>
<tr>
<td>Uniform-69-n20</td>
<td>284.12</td>
<td>62.6</td>
<td>7.3</td>
<td>57.4</td>
<td>57.4</td>
<td>7.9</td>
<td>0.0</td>
<td>1,932</td>
</tr>
<tr>
<td>Uniform-70-n20</td>
<td>328.75</td>
<td>71.5</td>
<td>9.5</td>
<td>48.4</td>
<td>48.4</td>
<td>10.7</td>
<td>0.0</td>
<td>2,304</td>
</tr>
</tbody>
</table>

### $N = 20$

<table>
<thead>
<tr>
<th>Instance</th>
<th>$Z^*$</th>
<th>CPU (s)</th>
<th>CPU4 (s)</th>
<th>LP1 Gap %</th>
<th>LP2 Gap %</th>
<th>RMIP Gap %</th>
<th>MIP Gap %</th>
<th>B&amp;B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform-61-n20</td>
<td>296.19</td>
<td>1,861.0</td>
<td>739.4</td>
<td>64.8</td>
<td>52.5</td>
<td>26.5</td>
<td>0.0</td>
<td>224,662</td>
</tr>
<tr>
<td>Uniform-62-n20</td>
<td>293.84</td>
<td>12,004.0</td>
<td>10,000.0</td>
<td>63.4</td>
<td>63.4</td>
<td>37.5</td>
<td>4.5</td>
<td>1,177,205</td>
</tr>
<tr>
<td>Uniform-63-n20</td>
<td>274.05</td>
<td>2,215.6</td>
<td>982.1</td>
<td>59.7</td>
<td>59.3</td>
<td>13.1</td>
<td>0.0</td>
<td>469,977</td>
</tr>
<tr>
<td>Uniform-64-n20</td>
<td>307.31</td>
<td>3,828.8</td>
<td>1,926.6</td>
<td>63.8</td>
<td>63.8</td>
<td>22.2</td>
<td>0.0</td>
<td>1,222,058</td>
</tr>
<tr>
<td>Uniform-65-n20</td>
<td>305.11</td>
<td>1,580.6</td>
<td>367.7</td>
<td>63.8</td>
<td>63.8</td>
<td>22.2</td>
<td>0.0</td>
<td>469,977</td>
</tr>
<tr>
<td>Uniform-66-n20</td>
<td>320.35</td>
<td>2,608.6</td>
<td>752.0</td>
<td>63.9</td>
<td>63.9</td>
<td>30.2</td>
<td>0.0</td>
<td>1,633,893</td>
</tr>
<tr>
<td>Uniform-67-n20</td>
<td>344.51</td>
<td>503.5</td>
<td>261.9</td>
<td>48.4</td>
<td>48.2</td>
<td>17.4</td>
<td>0.0</td>
<td>73,576</td>
</tr>
</tbody>
</table>

### $N = 24$

<table>
<thead>
<tr>
<th>Instance</th>
<th>$Z^*$</th>
<th>CPU (s)</th>
<th>CPU4 (s)</th>
<th>LP1 Gap %</th>
<th>LP2 Gap %</th>
<th>RMIP Gap %</th>
<th>MIP Gap %</th>
<th>B&amp;B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform-71-n50</td>
<td>308.92</td>
<td>12,004.0</td>
<td>10,000.0</td>
<td>63.9</td>
<td>63.9</td>
<td>30.2</td>
<td>0.0</td>
<td>1,362,282</td>
</tr>
<tr>
<td>Uniform-72-n50</td>
<td>323.94</td>
<td>12,004.0</td>
<td>10,000.0</td>
<td>63.9</td>
<td>63.9</td>
<td>30.2</td>
<td>0.0</td>
<td>1,362,282</td>
</tr>
<tr>
<td>Uniform-73-n50</td>
<td>315.99</td>
<td>12,004.0</td>
<td>10,000.0</td>
<td>63.9</td>
<td>63.9</td>
<td>30.2</td>
<td>0.0</td>
<td>1,362,282</td>
</tr>
<tr>
<td>Uniform-74-n50</td>
<td>308.90</td>
<td>11,191.4</td>
<td>10,000.0</td>
<td>67.2</td>
<td>67.2</td>
<td>42.0</td>
<td>0.0</td>
<td>1,362,282</td>
</tr>
<tr>
<td>Uniform-75-n50</td>
<td>378.00</td>
<td>12,004.0</td>
<td>10,000.0</td>
<td>63.9</td>
<td>63.9</td>
<td>30.2</td>
<td>0.0</td>
<td>1,362,282</td>
</tr>
<tr>
<td>Uniform-76-n50</td>
<td>269.23</td>
<td>11,191.4</td>
<td>10,000.0</td>
<td>67.2</td>
<td>67.2</td>
<td>42.0</td>
<td>0.0</td>
<td>1,362,282</td>
</tr>
<tr>
<td>Uniform-77-n50</td>
<td>364.22</td>
<td>3,998.8</td>
<td>3,745.1</td>
<td>43.0</td>
<td>43.0</td>
<td>20.4</td>
<td>0.0</td>
<td>537,215</td>
</tr>
<tr>
<td>Uniform-78-n50</td>
<td>317.44</td>
<td>11,568.4</td>
<td>10,000.0</td>
<td>61.3</td>
<td>61.3</td>
<td>51.2</td>
<td>0.0</td>
<td>1,727,903</td>
</tr>
<tr>
<td>Uniform-79-n50</td>
<td>310.49</td>
<td>5,624.1</td>
<td>3,617.5</td>
<td>56.9</td>
<td>56.9</td>
<td>31.3</td>
<td>0.0</td>
<td>537,215</td>
</tr>
<tr>
<td>Uniform-80-n50</td>
<td>321.37</td>
<td>12,004.0</td>
<td>10,000.0</td>
<td>63.0</td>
<td>63.0</td>
<td>33.5</td>
<td>30.3</td>
<td>1,596,437</td>
</tr>
</tbody>
</table>
Table A.9: Computational Results for $N = 28$ and $32$ for $U$ instances

<table>
<thead>
<tr>
<th>Instance</th>
<th>$Z^*$</th>
<th>$N=28$</th>
<th>CPU (s)</th>
<th>CPU4 (s)</th>
<th>LP1 Gap %</th>
<th>LP2 Gap %</th>
<th>RMIP Gap %</th>
<th>MIP Gap %</th>
<th>B&amp;B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform-71-n50</td>
<td>315.06</td>
<td>12.014.0</td>
<td>10.000.0</td>
<td>59.4</td>
<td>59.4</td>
<td>57.2</td>
<td>46.2</td>
<td>649,590</td>
<td></td>
</tr>
<tr>
<td>Uniform-72-n50</td>
<td>327.61</td>
<td>11.404.6</td>
<td>10.000.0</td>
<td>62.0</td>
<td>62.0</td>
<td>59.7</td>
<td>37.3</td>
<td>889,075</td>
<td></td>
</tr>
<tr>
<td>Uniform-73-n50</td>
<td>318.01</td>
<td>12.012.1</td>
<td>10.000.0</td>
<td>58.7</td>
<td>58.7</td>
<td>55.7</td>
<td>19.2</td>
<td>743,877</td>
<td></td>
</tr>
<tr>
<td>Uniform-74-n50</td>
<td>363.77</td>
<td>12.013.1</td>
<td>10.000.0</td>
<td>66.7</td>
<td>66.7</td>
<td>65.4</td>
<td>34.2</td>
<td>526,135</td>
<td></td>
</tr>
<tr>
<td>Uniform-75-n50</td>
<td>355.00</td>
<td>11.124.4</td>
<td>10.000.0</td>
<td>42.5</td>
<td>42.5</td>
<td>36.4</td>
<td>24.3</td>
<td>789,046</td>
<td></td>
</tr>
<tr>
<td>Uniform-76-n50</td>
<td>380.66</td>
<td>11.148.1</td>
<td>10.000.0</td>
<td>64.3</td>
<td>64.3</td>
<td>62.2</td>
<td>37.5</td>
<td>175,539</td>
<td></td>
</tr>
<tr>
<td>Uniform-77-n50</td>
<td>347.78</td>
<td>11.675.7</td>
<td>10.000.0</td>
<td>61.5</td>
<td>61.5</td>
<td>51.7</td>
<td>8.6</td>
<td>326,448</td>
<td></td>
</tr>
<tr>
<td>Uniform-78-n50</td>
<td>361.71</td>
<td>12.013.9</td>
<td>10.000.0</td>
<td>66.1</td>
<td>66.1</td>
<td>45.2</td>
<td>40.5</td>
<td>710,703</td>
<td></td>
</tr>
<tr>
<td>Uniform-79-n50</td>
<td>320.50</td>
<td>11.388.7</td>
<td>10.000.0</td>
<td>58.3</td>
<td>58.3</td>
<td>40.9</td>
<td>27.0</td>
<td>693,824</td>
<td></td>
</tr>
<tr>
<td>Uniform-80-n50</td>
<td>359.64</td>
<td>12.012.0</td>
<td>10.000.0</td>
<td>52.4</td>
<td>52.4</td>
<td>33.5</td>
<td>28.0</td>
<td>672,905</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instance</th>
<th>$Z^*$</th>
<th>$N=32$</th>
<th>CPU (s)</th>
<th>CPU4 (s)</th>
<th>LP1 Gap %</th>
<th>LP2 Gap %</th>
<th>RMIP Gap %</th>
<th>MIP Gap %</th>
<th>B&amp;B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform-71-n50</td>
<td>352.68</td>
<td>11.126.1</td>
<td>10.000.0</td>
<td>63.8</td>
<td>63.8</td>
<td>62.9</td>
<td>42.3</td>
<td>145,024</td>
<td></td>
</tr>
<tr>
<td>Uniform-72-n50</td>
<td>357.14</td>
<td>11.587.5</td>
<td>10.000.0</td>
<td>64.8</td>
<td>64.8</td>
<td>64.1</td>
<td>59.0</td>
<td>475,876</td>
<td></td>
</tr>
<tr>
<td>Uniform-73-n50</td>
<td>348.10</td>
<td>12.012.3</td>
<td>10.000.0</td>
<td>61.3</td>
<td>61.3</td>
<td>58.5</td>
<td>41.3</td>
<td>387,262</td>
<td></td>
</tr>
<tr>
<td>Uniform-74-n50</td>
<td>378.89</td>
<td>11.660.0</td>
<td>10.000.0</td>
<td>67.4</td>
<td>67.4</td>
<td>65.7</td>
<td>41.5</td>
<td>195,091</td>
<td></td>
</tr>
<tr>
<td>Uniform-75-n50</td>
<td>364.21</td>
<td>11.765.0</td>
<td>10.000.0</td>
<td>44.0</td>
<td>44.0</td>
<td>39.2</td>
<td>36.9</td>
<td>288,715</td>
<td></td>
</tr>
<tr>
<td>Uniform-76-n50</td>
<td>363.39</td>
<td>12.033.5</td>
<td>10.000.0</td>
<td>64.5</td>
<td>64.5</td>
<td>42.6</td>
<td>41.3</td>
<td>372,910</td>
<td></td>
</tr>
<tr>
<td>Uniform-77-n50</td>
<td>401.17</td>
<td>12.430.1</td>
<td>10.000.0</td>
<td>60.9</td>
<td>60.9</td>
<td>62.5</td>
<td>48.5</td>
<td>320,678</td>
<td></td>
</tr>
<tr>
<td>Uniform-78-n50</td>
<td>403.04</td>
<td>12.012.2</td>
<td>10.000.0</td>
<td>60.6</td>
<td>60.6</td>
<td>64.8</td>
<td>51.2</td>
<td>365,120</td>
<td></td>
</tr>
<tr>
<td>Uniform-79-n50</td>
<td>499.11</td>
<td>12.012.8</td>
<td>10.000.0</td>
<td>73.2</td>
<td>73.2</td>
<td>71.4</td>
<td>61.9</td>
<td>539,985</td>
<td></td>
</tr>
<tr>
<td>Uniform-80-n50</td>
<td>359.80</td>
<td>12.010.3</td>
<td>10.000.0</td>
<td>66.9</td>
<td>66.9</td>
<td>63.6</td>
<td>38.0</td>
<td>142,879</td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX B

SUPPORTING MATERIAL FOR CHAPTER 3

B.1 Lemma 1 Proof

Lemma 1: Given unlimited flight range, if there is a drone cycle from node $i$ to node $k$, then there must be one non-cyclic drone flight landing at $i$ and one non-cyclic drone flight departing from $i$.

Proof: We proceed to prove Lemma 1 by contradiction. As per the settings in the computational study in §3.5, assume symmetrical travel times between nodes, and that the drone’s flight time may be expressed as a ratio of the vehicle’s travel time (i.e. $f_{ij} = \alpha \tau_{ij}$):

- Assume there exists an optimal solution, illustrated as Scenario 1, featuring a drone cycle from node $i$ to node $k$ absent a non-cyclic drone flight landing at node $i$. We derive from Scenario 1 a new solution, Scenario 2, that is similar
except that node \( k \) is now served by a non-cyclic drone flight that lands at \( i \) after launching from \( p \). By inspection, Scenario 2 is feasible because the drone has unlimited flight range.

Assuming Scenario 1 is optimal, then the vehicle’s departure time from node \( i \) must be earlier or equal to the departure time from the same node in Scenario 2. Comparing, we have

\[
\tau_{pi} + 2f_{ik} \leq \max(\tau_{pi}, f_{pk} + f_{ki}).
\]

If \( \tau_{pi} \geq f_{pk} + f_{ki} \), then by inspection Scenario 1 yields a longer completion time than Scenario 2. Alternatively, suppose \( \tau_{pi} < f_{pk} + f_{ki} \), in which case Scenario 1 is optimal if

\[
\tau_{pi} + 2f_{ik} \leq f_{pk} + f_{ki}.
\]

Since the drone flights are symmetric,

\[
\tau_{pi} + f_{ik} < f_{pk}.
\]

Furthermore, since \( f_{ij} = \alpha \tau_{ij} \),

\[
\tau_{pi} + \alpha \tau_{ik} \leq \alpha \tau_{pk}.
\]

By virtue of the triangle inequality,

\[
\tau_{pi} + \alpha \tau_{ik} \leq \alpha (\tau_{pi} + \tau_{ik}).
\]

But this is impossible, since by assumption \( \alpha < 1 \).
• Using a similar argument, the symmetric case in which a solution featuring a drone cycle from node $i$ to node $k$ absent a non-cyclic drone flight departing $i$ must be sub-optimal.

By contradiction, we have thus shown that in order for a drone cycle to launch from node $i$, there must be both at least one non-cyclic drone flight launching from $i$, and at least one non-cyclic drone flight landing at $i$. □

B.2 Tightened $M$ values for Model TSPD3

The scalars $M, M^1, M^2, \ldots, M^6$, introduced in §3.3.3 for Model TSPD3, may be validly set as follows (see Appendix A.2 for detailed derivations):

\[
M^1_j = \max(0, \max_{i \in V} \max_{f_{ij} \leq F} f_{ij}, \max_{k \in V} \max_{f_{jk} \leq F} f_{jk}),
\]

\[
M^2_{ij} = \begin{cases}
\tau_{0j} - f_{0j}, & i = 0, \\
\tau_{i0} - f_{i0}, & j = 0, \\
(C_{\max} - f_{i0} + t_{ij}) - f_{0j}, & \text{otherwise},
\end{cases}
\]

\[
M^3_{ij} = \begin{cases}
0, & i = 0 \text{ or } j = 0, \\
(C_{\max} - f_{j0} + f_{ij}) - f_{0j}, & \text{otherwise},
\end{cases}
\]

\[
M^4_{ij} = \begin{cases}
0, & i = 0, \\
(C_{\max} - f_{i0} + f_{ij}) - f_{0j}, & \text{otherwise},
\end{cases}
\]

\[
M^5_{ij} = \begin{cases}
(C_{\max} - f_{j0}) - \tau_{0j}, & i = 0, \\
(C_{\max} - f_{j0}) & \text{otherwise},
\end{cases}
\]

\[
M^6_i = C_{\max} - \tau_{i0}, \quad \forall i \in V^*,
\]
\[ M = C_{\text{max}} - \min_{i, j \in V^*} \{ f_{j0}, \tau_{j0} \}. \]

\[ f_{ij} + f_{j0} \leq F. \]
APPENDIX C
SUPPORTING MATERIAL FOR CHAPTER 4

C.1 Derivation of Tightened Big “M” Values

In what follows, valid values are derived for the $M$ parameter deployed in Constraint (4.1i) in §4.2:

$$b_j^t \geq b_i^t + \tau_{ij} + \sum_{k \in (C_t \cap D)} f_{ik}^t y_{ikj}^t - M_{ij}^t (1 - x_{ij}^t), \quad \forall t \in T, i, j \in R^t | i \neq j,$$  \hspace{1cm} (1i)

In each time period $t \in T$, Constraint (4.1i) sets the departure time, $b_j^t$, from a node $j \in R^t$ visited by the vehicle ($\sum_{i \in R^t} x_{ij}^t = 1$). If $j$ is not visited by the vehicle, $M_{ij}^t$ must be large enough to retain the validity of Constraint (4.1i). Two inequalities that determine the earliest and latest departure times for a node visited by the vehicle, respectively, follow:

$$b_i^t \geq \tau_{0i}, \quad \forall t \in T, i \in C^t,$$  \hspace{1cm} (1)

$$b_i^t \leq D_{\max}^t - \tau_{in+1}, \quad \forall t \in T, i \in C^t,$$  \hspace{1cm} (2)

Inequality (1) holds by construction, since the earliest arrival time for a node visited by the vehicle is the duration required to visit the node after departing from the depot ($\tau_{0j}$). In Inequality (2), $D_{\max}^t$ is the objective value associated with a feasible tour for a period $t \in T$. Thus the latest departure from a node visited by the vehicle
must be no more than the difference between $D_{\text{max}}^t$ and the vehicle’s travel time from that node to the depot, $\tau_{in+1}$. As specified in Section 3.1, nodes 0 and $n+1$ represent source and sink nodes, with shared coordinates at the central depot location. We also assume symmetrical inter-node distances.

To scale $M_{ij}^t$, suppose $x_{ij}^t = x_{ji}^t = 0$, which implies $\sum_{k \in (C \cap D)} \hat{f}_{ikj} y_{ikj}^t = 0$ by Constraint (4.1f). Thus Constraint (4.1i) simplifies to:

$$b_j^t \geq b_i^t + \tau_{ij} - M_{ij}^t \implies M_{ij}^t \geq b_i^t + \tau_{ij} - b_j^t,$$

Consider the following cases:

1. **Node $i$ is the depot:**

   (a) The departure time of the vehicle from the source node representing the depot is equal to 0 ($b_0^t = 0 \ \forall t \in T$), and thus Constraint (4.1i) simplifies to

   $$M_{0j}^t \geq \tau_{0j} - b_j^t$$

   By Inequality (1), the earliest departure time for a node $j$ is $b_j^t \geq \tau_{0j}$. Thus

   $$M_{0j}^t \geq 0$$

   (b) The departure time of the vehicle from the sink node representing the depot is equal to the length of a feasible tour ($b_{n+1}^t = D_{\text{max}}^t \ \forall t \in T$). Thus Constraint (4.1i) simplifies to

   $$M_{n+1j}^t \geq D_{\text{max}}^t + \tau_{n+1j} - b_j^t$$
By Inequality (2), the latest departure time for node \( j \) is \( b_j^i \leq D^t_{\text{max}} - \tau_{jn+1} \).

Thus

\[
M_{n+1,j}^t \geq D^t_{\text{max}} + \tau_{n+1,j} - (D^t_{\text{max}} - \tau_{jn+1}) \implies M_{n+1,j}^t \geq 2\tau_{n+1,j}
\]

2. Node \( j \) is the depot:

(a) The departure time of the vehicle from the source node representing the depot is equal to 0 \( (b_0^t = 0 \ \forall t \in T) \), and thus Constraint (4.1i) simplifies to

\[
M_{00}^t \geq b_i^t + \tau_{i0}
\]

By Inequality (2), the latest departure time for node \( i \) is \( b_i^t \leq D^t_{\text{max}} - \tau_{in+1} \).

Thus

\[
M_{i0}^t \geq D^t_{\text{max}} - \tau_{in+1} + \tau_{i0} \implies M_{i0}^t \geq D^t_{\text{max}}
\]

(b) The departure time of the vehicle from the sink node representing the depot is equal to the length of a feasible tour \( (b_{n+1}^t = D^t_{\text{max}} \ \forall t \in T) \). Thus Constraint (4.1i) simplifies to

\[
M_{in+1}^t \geq b_i^t + \tau_{in+1} - D^t_{\text{max}}
\]

By Inequality (2), the latest departure time for node \( i \) is \( b_i^t \leq D^t_{\text{max}} - \tau_{in+1} \).

Thus:

\[
M_{in+1}^t \geq D^t_{\text{max}} - \tau_{in+1} + \tau_{in+1} - D^t_{\text{max}} \implies M_{in+1}^t \geq 0
\]

(c) Nodes \( i, j \in V^* \): Assuming the worst-case bound, suppose the departure time at node \( i \) is as large as possible, and the departure time at node \( j \) is
as small as possible. By Inequalities (1) and (2), set \( b^t_i = D_{\text{max}}^t - \tau_{in+1} \) and \( b^t_j = \tau_{0j} \). Thus Constraint (4.1i) simplifies to:

\[
M^t_{ij} \geq D_{\text{max}}^t - \tau_{in+1} + \tau_{ij} - \tau_{0j},
\]

The valid values of \( M \) are summarized as follows:

\[
M^t_{ij} = \begin{cases} 
0, & i = 0, j = n + 1 \\
2\tau_{n+1j}, & i = n + 1, \\
D_{\text{max}}^t, & j = 0, \\
D_{\text{max}}^t - \tau_{in+1} + \tau_{ij} - \tau_{0j}, & \text{otherwise.}
\end{cases}
\]

### C.2 Parameter Tuning

As outlined in §4.4.3, the matheuristic’s parameters may be tuned to yield high-quality solutions with manageable computational effort. In our computational experience, the following settings provided the best such balance. Unless otherwise specified, all other solver settings were left in their default state.

- **Phase 0, Initialization:** The solver is limited to 300 CPU seconds.

- **Phase 1, Period-Specific Demand Improvement:** Set \( NII = 3 \), with a solver limit of 120 CPU seconds in each iteration.

- **Phase 2, Geo-spatial Improvement:** The solver settings are dependent on \( n \) as summarized below, with \( NII = 3 \) throughout.

  - \( n \leq 40 \): The solver is limited to 30 CPU seconds.
  - \( n \in [50, 100] \): The solver is limited to 40 CPU seconds.
  - \( n = 150 \): The solver is limited to 120 CPU seconds.
  - \( n = 200 \): The solver is limited to 180 CPU seconds, 10 MIP solutions, an MIP gap of 5%, with a focus on finding feasible solutions.


