

**MODELING AND CONTROL OF NETWORKED AUTONOMOUS MOBILITY
SYSTEMS**

A Dissertation
Presented to
The Academic Faculty

By

Qinshuang Wei

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering
Department of Electrical and Computer Engineering

Georgia Institute of Technology

May 2022

© Qinshuang Wei 2022

MODELING AND CONTROL OF NETWORKED AUTONOMOUS MOBILITY SYSTEMS

Thesis committee:

Dr. Samuel Coogan, Advisor
Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Matthieu Bloch
Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Fumin Zhang
Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Kyriakos Vamvoudakis
Aerospace Engineering
Georgia Institute of Technology

Dr. Yorai Wardi
Electrical and Computer Engineering
Georgia Institute of Technology

Date approved: April 8, 2022

ACKNOWLEDGMENTS

My sincere thanks go to my advisor, Dr. Samuel Coogan, who has been guiding me with invaluable suggestions and encouragements since Summer 2018. Sam has not only provided me with insightful thoughts and guidance, but also his kindness and patience. He nurtured a great lab environment in FACTS Lab. I am grateful that I was able to spend my 4-year PhD time with Sam's advisement.

I would also like to thank the rest of my thesis committee: Dr. Zhang, Dr. Wardi, Dr. Bloch, and Dr. Vamvoudakis for their patience and valuable comments.

I am fortunate to have been a part of the FACTS Lab. I would like to express my greatest appreciation to all former and current members in FACTS Lab for the support that I received. Everyone in the lab is encouraging and has inspired me greatly during my PhD program. I wish all of them the best for their future research.

Additionally, I want to express my thanks to Georgia Tech, where I has been spending my time since undergraduate program. I feel honored to be a student in Georgia Tech for 9 years.

Finally, I cannot finish this thesis without my parents. I would like to acknowledge their great understanding and tremendous support. I would also thank all my friends for their encouragement and company.

TABLE OF CONTENTS

Acknowledgments	iii
List of Tables	viii
List of Figures	ix
Summary	xii
Chapter 1: Introduction	1
1.1 Ride-Sharing Network	1
1.2 Urban Air Mobility Management	2
1.3 Preview of Thesis	2
Chapter 2: Literature Survey	9
2.1 Ride-Sharing with Autonomous Vehicles	9
2.2 UAM Network Modeling and Constraints	10
2.3 Routing and Scheduling of Vehicles	11
2.4 Disruptions to Transportation Systems	13
2.5 Resilience in Transportation Networks	14
Chapter 3: Mixed Autonomy in Ride-Sharing Networks	16
3.1 Model Definition	16

3.2	HV and AV Priority Assignments	18
3.3	Equilibrium Definition for HV Priority Assignment	19
3.4	Equilibrium Definition for AV Priority Assignment	21
Chapter 4: Profit-Maximization with Convexification For HV and AV Priority Assignments		23
4.1	Profit-Maximization for HV and AV Priority Assignment	23
4.2	Convexification of Profit Maximization	28
Chapter 5: The Relation Between HV Priority, AV Priority, and Weighted Priority Assignments and Special Networks		45
5.1	The Relation between HV Priority and AV Priority Assignments	45
5.2	Weighted Priority Assignment	56
5.2.1	Equilibrium Definition for Weighted Priority Assignment	56
5.2.2	Profit-Maximization Optimization Problem for Weighted Priority Assignment	57
5.3	Closed-Form Characterization for Star-to-Complete Networks	62
5.4	Concluding Remarks	65
Chapter 6: Capacity-Constrained Urban Air Mobility Scheduling		67
6.1	Problem Formulation	67
6.2	Theoretical Results for Scheduling	73
6.2.1	Necessary Condition for Dynamic Scheduling of Additional Demands	74
6.2.2	Fundamental Limits for Static Scheduling	78
6.2.3	Necessary and Sufficient Condition for a Schedule in a Local Network	82
6.2.4	Extension to General Star-Branch Network	86

6.3	Optimization Problem for Static Scheduling	87
6.4	Numerical Case Studies for Static Scheduling Algorithm	89
6.5	Concluding Remarks	92
Chapter 7: Efficient Algorithms For Dynamic Scheduling		93
7.1	A Detailed Explanation of the Dynamic Scheduling Algorithm	93
7.2	Numerical Study for Dynamic Scheduling Algorithm	102
7.2.1	Case Study 1: Dynamic Scheduling	103
7.2.2	Case Study 2: Atlanta Network	105
7.3	Concluding Remarks	107
Chapter 8: Safety Verification for UAM Scheduling: Simplest Case		108
8.1	Problem Formulation	109
8.1.1	Network Model and Nominal Scheduling	109
8.1.2	Disruption Model	110
8.2	Necessary and Sufficient Conditions for Safe Schedule	112
8.3	Case Study	118
8.4	Concluding Remarks	120
Chapter 9: Safety Verification for UAM Scheduling: General Case		122
9.1	Model Extension with Multiple Backup Nodes	122
9.2	Necessary and Sufficient Conditions for Safe Schedules	125
9.2.1	Necessary and Sufficient Condition for Worst-Case Safe Schedules	127
9.2.2	Necessary and Sufficient Condition for Best-Case Safe Schedules	130

9.3	Simplification for Verification	132
9.4	Case Study	138
9.5	Concluding Remarks	143
Chapter 10:Conclusion		145
Appendices		149
	Appendix A: Proof of Theorem 5	150
References		157

LIST OF TABLES

6.1	Time Needed for UAVs and Cars to Travel to Atlanta from Different Origins in Rush Hours	90
9.1	Backup nodes for each link in Example 3	124

LIST OF FIGURES

5.1	Optimal profits for a star-to-complete network with $n = 3$, $\xi = 0.2$ under a mixed autonomy deployment, a forced HV-only deployment, and a forced AV-only deployment. (Left) When $\beta = 0.8$, it is optimal for the platform to use only AVs when k , the ratio of the cost of AVs to HVs, satisfies $k \leq k_1 = 0.9053$, only HVs when $k \geq k_2 = 0.9181$, and a mix of AVs and HVs when $k_1 < k < k_2$. (Right) When $\beta = 0.95$, it is optimal for the platform to use only AVs when $k \leq k_1 = 0.9763$ and only HVs when $k > k_1$, and it is never optimal for the platform to use a mix of HVs and AVs.	65
6.1	A two-link network that is used to demonstrate the benefits of dynamic scheduling in Example 1.	72
6.2	A star-branch graph for a UAM network of Atlanta (ATL) with three ex-urbs: Alpharetta (ALP), Kennesaw (KEN) and Buford (BUF). We consider Atlanta as the central node, and there are 0, 1, 2 intermediate nodes between Atlanta node and the three leaf nodes “ALP”, “KEN” and “BUF”, respectively. The time interval need for traveling through each link is labeled beside the corresponding link.	78
6.3	The detailed optimal schedule of the second case study with $[h_1, h_2, h_3] = [4, 4, 19]$. The red, green and orange bars represent the reserved time slots for flights from v_1 , v_2 and v_3 , respectively. We label an “ID” above each bar to track the flights. The diamond represents the arrival deadline of a journey; the solid-color bar represents the time interval that the flight is scheduled to arrive. The flight will then stay at the node for time w_I (at intermediate node) or w (at v_0) after arrival (represented by the lightly shaded bar).	91
6.4	The detailed optimal schedule along branch 3. UAVs traveling along branch 3 stop at intermediate nodes b , c and v_0 . The dotted lines are showing that connected bars are the schedules of the same journey at different intermediate nodes and destination.	91

7.1	An 8-node, 8-link network used to illustrate the case study in 7.2.1.	103
7.2	Scheduled/realized occupation of the five parking spots at Node 8 for the case study in Section 7.2.1 in the time interval $[100, 160]$ determined at time 100 (left) and time 120 (right). The green, orange and blue bars represent the reserved time slots for flights traveling through R^2 , R^3 and R^4 , respectively. An ID number above each bar identifies the flights. The diamond represents the arrival deadline of a journey; the solid-color bar represents the time interval that the flight is scheduled to arrive. The flight stays at the node for time $w = 1$ after arrival (represented by the lightly shaded bar). Therefore, the entire bar represents the time interval the flight may or did appear at node 8. Comparing the top and the bottom figures, we can observe several representative changes: the journey 30 has not yet arrived at the destination in the top plot computed at time 100 but is completed in the bottom plot computed at time 120; the time slot reserved for journey 25 has shrunk because the flight arrived at an intermediate node in the time interval $[100, 120]$ so that the uncertainty of its arrival at node 8 reduced; journeys 37-43 are newly scheduled.	104
7.3	The computation time versus minimal SoD cost among all stored schedules obtained within the corresponding computation time for Case Study 2. The dashed line represents the optimal SoD computed from the optimal schedule obtained by the mixed-integer program.	106
8.1	(Bold partial graph) The sub-graph consisted of the bold lined 4 nodes and 3 links is used to illustrate the simple network in Example 2. (Entire graph) The entire graph is used to illustrate the network with 7 nodes and 7 links in the case study.	112
8.2	The number of flights rerouted to v_2 , $N_{v_2}(t_c, v_5)$ (simplified as N_{v_2}), represented by the upper blue rectangles and the maximum unaffected occupation at v_2 when the node v_5 is disabled at time t_c , $N_R(v_2, t_c, v_5)$ (simplified as N_R), represented by the lower white rectangles. The dotted line corresponds to the capacity at node v_2	120
8.3	The computation time for verifying the worst-case safety of schedules with different sizes. We test on 11 different sets of schedules with sizes from 20 to 1000. The data points constitute a parabola, which demonstrates the $O(n^2)$ computational complexity.	121
9.1	A network with 7 nodes and 7 links. This is the same network that appeared in Fig. 7.1.	123

9.2	The sub-graph consisted of the 4 nodes and 3 links is used to illustrate the simple network in Example 2	125
9.3	Observation of the network when node v_5 is disabled at any time $t_c > 0$. (Top) Expected landing-spot occupation at node v_3 . (Middle) Redistribution of flights on link e_5 to its backup nodes. (Bottom) Redistribution of flights on link e_6 to its backup nodes. We simplify the notation $N_R(\cdot, t_c, v_5)$ and $N_{\cdot,}(t_c, v_5)$ as $N_R(\cdot)$ and $N_{\cdot,}$ in the figure.	141
9.4	Observation of the network when node v_5 is disabled at any time $t_c > 0$ when we adjust the capacity of node v_4 to $C_{v_4} = 8$. (Top) Expected landing-spot occupation at node v_3 . (Middle) Redistribution of flights on link e_5 to its backup nodes. (Bottom) Redistribution of flights on link e_6 to its backup nodes. We simplify the notation $N_R(\cdot, t_c, v_5)$ and $N_{\cdot,}(t_c, v_5)$ as $N_R(\cdot)$ and $N_{\cdot,}$ in the figure.	142
9.5	The computation time for verifying the worst-case safety of schedules with different sizes. We test on 11 different sets of schedules with sizes from 20 to 1000. The data points demonstrates the $O(n^2)$ computational complexity.	143

SUMMARY

The objective of this thesis is to integrate autonomous transportation with the current transportation system, including introducing autonomous vehicles into ground transportation and utilizing urban airspace. We first study the ride-sharing networks with mixed autonomy where both autonomous vehicles (AVs) and human driven vehicles (HVs) are considered, and then focus on urban airspace mobility (UAM) management for urban air vehicles (UAVs).

Existing research in ride-sharing has largely focused on two ends of the autonomy. To fill this gap, this thesis explore the transition from traditional ride-sharing networks to totally automated mobility-on-demand systems, where the platform sets prices for riders, compensations for drivers of HVs, and operates AVs for a fixed price with the goal of maximizing profits.

In UAM networks, takeoff and landing sites, called vertiports, typically have limited landing capacity. For safety, it must be guaranteed that an air vehicle will be able to land before it can be allowed to take off. We present a model for the UAM network that accounts for uncertain travel times and limited landing capacity at vertiports. We explore the problem of scheduling UAM flights, where we established theoretical bounds on the achievable throughput of the network and developed a tractable algorithm for scheduling trips to satisfy safety constraints and arrival deadlines. The algorithm allows for dynamically updating the schedule to accommodate, e.g., new demands over time. Additionally, we investigate the safety verification problem for given UAM schedules in the network with disruption, where we consider the intermittent closures of the vertiports. We develop theoretical constraints and an efficient algorithm that, given a proposed UAM schedule, verifies whether all UAVs are able to safely reach a back-up landing site in the event of a vertiport closure without violating the limited landing capacity of each vertiport in the network.

CHAPTER 1

INTRODUCTION

There is growing attention towards ground transportation congestion and transition efficiency. Introducing autonomous vehicles into the ground transportation network and utilizing urban airspace are both reasonable solutions. Autonomous mobility systems has aroused the attention from researchers and companies, while the arrangement of both autonomous vehicles (AVs) and urban air vehicle (UAVs) are being studied. In our proposed research, we are exploring the urban air mobility (UAM) solutions and ride-sharing network with both AVs and human-driven vehicles (HVs).

1.1 Ride-Sharing Network

In ground transportation systems, ride-sharing network with mixed autonomy where both autonomous vehicles (AVs) and human-driven vehicles (HVs) are considered. Ride-sharing platforms, also known as transportation network companies, match passengers or riders with drivers using websites or mobile apps and have become ubiquitous in major cities across the world [1, 2]. The rise of ride-sharing platforms coincides with a decline in private car ownership due to high costs, lack of parking, and persistent traffic congestion [3, 1, 4, 5]. Traditionally, rides are provided by drivers who use their own personal vehicle to provide service.

However, ride-sharing platforms have indicated that they intend to incorporate autonomous vehicles (AVs) into their fleet in the near future [6]. AVs managed by the ride-sharing platform can be directed to specific locations as needed. However, owning and managing an AV fleet can be costly for the ride-sharing platform, and significant technological hurdles remain before these platforms could transition to 100% autonomous fleets. For these reasons, it is likely that ride-sharing platforms will, at least initially, adopt a

hybrid or *mixed* framework in which AVs operate along-side conventional, human-driven vehicles [7]. For example, the ride-sharing platform Lyft has partnered with Aptiv, a manufacturer of autonomous cars, to serve some of its ride requests in Las Vegas, Nevada with AVs [8]. In this setting, the AVs can be deployed to serve, for instance, locations with abnormally high demand.

1.2 Urban Air Mobility Management

Besides introducing autonomous vehicles into ground transportation systems, there is growing interest in utilizing urban airspace for transportation of people and goods. Both commercial mobility-on-demand operators [9] and government-sponsored research institutes such as NASA [10] are exploring such urban air mobility solutions in cities and surrounding regions. Studies such as [11, 12, 13, 14, 15, 16] propose various approaches to allow urban air vehicles (UAVs) to travel safely and efficiently through cities. These proposed ideas cover a wide range of possibilities such as allowing UAVs to land at *vertistops* or *vertiports* installed on roofs of existing buildings or within cloverleaf exchanges on freeways. Several simulation tools [17, 18, 19] have also been developed. At the same time, unforeseen disruptions such as intermittent closure of landing sites due to, e.g., extreme weather conditions must be considered for any UAM solution [11]. In particular, a key safety constraint is to ensure that a back-up landing spot is available for all in-flight UAVs.

1.3 Preview of Thesis

In this thesis, we first study the transition from traditional ride-sharing networks to totally automated mobility-on-demand systems in Chapter 3, where we introduce the problem formulation of ride-sharing networks with mixed autonomy. In Chapter 4, we pose the problems of profit maximization as non-convex optimization problems for AV and HV priority assignments and proposes an alternative convex optimization problem that provides the same optimal profits and from which a solution to the original problem can be recovered.

Due to its asymmetry to the AV and HV priority assignments, weighted priority assignment is introduced and studied separately in Chapter 5, where we also study the relation between the AV, HV and weighted priority assignments and show that they achieve the same optimal profits. We then turn our attention to the UAM management in Chapter 6, where we introduce our UAM network model with limited landing capacity and uncertain travel times and investigate the theoretical constraints on schedulability of given set of demands in both static and dynamic cases. In Chapter 7, we develop a tractable algorithm for dynamically scheduling trips to satisfy safety constraints and arrival deadlines. In Chapter 8, we study the safety verification problem in the event of a landing site closure in the UAM network with only one backup landing site assigned to each flight, and extend it into a more general setting where multiple backup landing sites can be assigned in Chapter 9.

Mixed Autonomy in Ride-Sharing Networks

In Chapters 3–5, we study the ride-sharing networks with mixed autonomy. In particular, in Chapter 3 we extend the model proposed in [20], which did not consider AVs, to the mixed autonomy setting under several assumptions on the vehicle-to-rider assignment possibilities, and we analyze the resulting models. We consider a network consisting of multiple locations, and potential riders arrive at these locations with desired destinations. The ride-sharing platform sets prices for riders and compensation to drivers of HVs. In addition, the platform has the option to deploy AVs for a fixed cost. Introducing AVs leads to an important assignment choice that must be made: if both an AV and an HV are available to serve a rider, which receives preference? We consider three possible assignment rules: AVs always receive priority (AV priority); HVs always receive priority (HV priority); and priority is determined in proportion to the number of available AVs and HVs at each location (weighted priority).

We focus on the equilibrium conditions that arise in the resulting mixed autonomy deployment when the platform seeks to maximize profits. In Chapter 4, we then study the profit-maximizing optimization problem under equilibrium for AV and HV priority

assignments, respectively. However, it is a non-convex optimization problem, which is difficult to analyze the performance. We therefore develop an alternative convex problem for both assignments, which we demonstrate to have the same maximum profits as the original non-convex problem, while an optimal solution to the original non-convex problem can be recovered from the convex alternative problem.

Due to its asymmetry to the AV and HV priority assignments, we also introduce and study weighted priority assignment in Chapter 5 under equilibrium. We also investigate the relation between AV, HV and weighted priority assignments, where we find them always achieve the same optimal profits. We then analyze the optimal solutions on a special class of networks.

We summarize our main findings in the problem of ride-sharing network with mixed autonomy as follows:

1. In all three priority assignments, the equilibrium conditions lead to a non-convex optimization problem. Nonetheless, we develop an alternative convex problem from which an optimal solution to the original non-convex problem can be recovered.
2. We find that, surprisingly, all three priority schemes result in the same maximum profits for the platform. This is because, at an optimal equilibrium, we show that all vehicles are assigned a ride and thus the priority assignment choice is immaterial at the optimal equilibrium.
3. Lastly, we consider the ratio of AVs to HVs that will be deployed by the platform in order to maximize profits for various operating costs of AVs. We show that, in some cases, there is a regime for which the platform will choose to mix HVs and AVs vehicles in order to maximize profits, while in other cases, the platform will use only HVs or only AVs, depending on the relative cost of AVs. For a specific family of networks, we fully characterize these thresholds analytically.

The main contributions are therefore two-fold. First, we develop a new model for study-

ing the emergence of AVs in ride-sharing networks. This model contributes substantial modifications to the foundational model developed in [20] in order to allow for the presence of AVs. Second, we conduct a detailed theoretical study of the resulting model focusing on the optimal profits obtainable by a ride-sharing platform that deploys AVs. While the AV priority assignment case is first investigated in [21], the main findings with all three assignments appear in [22].

Capacity-Constrained Urban Air Mobility Scheduling

We explore the scheduling problem of urban air mobility services with limited landing capacity and uncertain travel times in Chapters 6–7 and study a dynamic scheduling algorithm for UAM networks.

We model a UAM network as a graph with nodes that are finite capacity vertiports and edges that are transportation links between vertiports. A key feature of our model is the allowance of uncertain travel time between vertiports represented as an interval of possible travel times. These two aspects have not been considered together in any other UAM solutions. Flights depart from origin nodes at a scheduled departure time and visit one more more vertiports along a route through the UAM graph. When a vehicle arrives at a vertiport, it occupies one of a finite number of landing spots for a fixed ground service time t_0 , e.g, offload and load passengers. In this framework, the defining feature of safety is that a landing spot must always be available when the UAV arrives at the vertiport. The fact that travel times are uncertain adds to the complexity of the safety problem.

In Chapter 6 we present the above model for UAM networks that allows for a time-varying set of trip demands, limited landing capacity at destinations, and uncertainty in travel times that is modeled nondeterministically with lower and upper travel time bounds. Trip demands, *i.e.*, flights, must travel through designated routes and have arrival deadlines at their destinations. We consider the problem of scheduling flight departures to ensure that all flights arrive no later than their deadlines and that there is always an available landing spot at the destination and intermediate nodes upon arrival. Because demands are time-

varying, we refer to such networks as *dynamic*; in the case when all demands are available at once, we call the network *static* and refer to a *static* scheduling problem.

Next, we present necessary conditions for the existence of a schedule for the dynamic and static UAM network. In particular, for static scheduling problem, we focus on the practically relevant class of *star-branch* networks consisting of a main destination node, multiple leaf origin nodes, and possibly intermediate nodes between leaf nodes and the destination node. This model captures, for example, travel from exurbs to a main city with possible stops at additional suburbs along the way and is the configuration for all four UAM networks (metro areas of Austin, Atlanta, Boston, and San Francisco) considered the INRIX report [14]. We present necessary conditions for the existence of a feasible schedule for the static UAM network. When there are no intermediate nodes, we also show that this condition is sufficient for feasibility.

We propose to evaluate the cost of a schedule as the sum of the difference between arrival and departure time for all trips. We then present a mixed integer program to compute an optimal schedule for the static UAM network on star-branch network. We demonstrate our approach on a case study in Atlanta, Georgia based on the example considered in [14]. The static scheduling results can be found in [23].

The method for obtaining an optimal schedule in Chapter 6 is limited to static network for star-branch graph topology, and uses a mixed integer program. However, the mixed integer program quickly becomes intractable as the number of flights increases. To broaden the scope of UAM network under consideration for the scheduling problem, and enhance the scheduling efficiency, in Chapter 7 we present a computationally efficient dynamic scheduling algorithm to compute a schedule satisfying safety constraints and arrival deadlines, and we demonstrate our approach on several case studies.

The dynamic scheduling algorithm proposed in Chapter 7 uses a branch-and-bound heuristic that does not rely on a mixed integer formulation and therefore may only create a suboptimal schedule. However, we demonstrate through example that our algorithm is able

to quickly obtain feasible schedules with reasonable cost, i.e., in under 1 second for 200 trips in two case study networks. Moreover, the algorithm computation can be continued in search of a lower cost schedule and interrupted at any point. The dynamic network model and scheduling algorithm appear in [24].

Schedule Verification for Urban Air Mobility

In Chapters 6–7, the schedules are obtained so that the trip demands, i.e., flights, must travel through designated routes and meet their corresponding arrival deadlines at their destinations, while satisfying the landing-spot restrictions at the destination and intermediate nodes upon arrival. In Chapters 8–9, we further consider the problem of verifying the safety of a given schedule upon the closure of a node in the network.

We assume given a schedule that is *a priori* nominally safe obtained via, e.g., the methodology proposed in Chapters 7. Given such a schedule, the goal is to ensure that it remains safe even if a vertiport closes and in-flight UAVs must be rerouted. In particular, a key safety constraint is to ensure that a back-up landing spot is available for all in-flight UAVs. We use the same UAM network model as in Chapter 6 while focusing on the static case, as we are aiming to provide a safety verification for a given set of demands before their departure. But this can be easily extended to dynamic model.

In Chapter 8 we add the disruption model, which considers the closure of a landing site, to the existing UAM network model. For each flight under consideration, one link-related backup node will be assigned. We then develop necessary and sufficient conditions for a given UAM schedule to be guaranteed as safe in the disrupted scenario under worst-case travel time realization, which leads to our proposed safety verification algorithm. We also provide necessary conditions for guaranteed safety under best-case travel time realization. Lastly, we develop a safety verifying algorithm based on the worst-case safety constraints and demonstrate our verification algorithm and its computation efficiency on a UAM network with up to 1,000 UAVs. The results in the chapter are presented in [25].

In Chapter 9, we extend the safety verification problem in Chapter 8 to a more general

case, where we assume that each link in the UAM network poses a set of back-up nodes such that any flight on that link that is inbound for a closed vertiport must be safely rerouted to one of those nodes at the moment of closure such that landing capacity is not exceeded for any node within the network.

Our main contributions are as follows. First, we present necessary and sufficient conditions for ensuring safety in the event of a vertiport closure, i.e., for ensuring that all in-flight UAVs are able to land at a back-up vertiport without exceeding landing spot capacity constraints. These conditions ensure safety for any realization of the link travel times, which are uncertain and only assumed to lie between known lower and upper bounds. We therefore refer to these conditions as worst-case safety guarantees. Second, we present an efficient algorithm for checking whether a schedule satisfies the theoretical necessary and sufficient conditions for worst-case safety. This algorithm leverages the theory of totally unimodular matrices to losslessly convert a mixed integer program into a linear program, enabling scalability to schedules with large numbers of UAVs. In particular, the proposed algorithm scales quadratically with the number of scheduled flights. Third, we present necessary and sufficient conditions for safety under some realization of the travel times. We refer to this as best-case safety, in contrast to worst-case safety which must be safe for all travel time realizations. These conditions, for example, could help a UAM operator determine if a schedule could be rendered safe by reducing travel time uncertainty. We demonstrate our results on several examples. Extending to multiple backup nodes is a significant generalization requiring the theory of totally unimodular matrices for an efficient algorithm that allows for checking a much larger class of safe schedules.

CHAPTER 2

LITERATURE SURVEY

2.1 Ride-Sharing with Autonomous Vehicles

Ride-sharing platforms, also known as transportation network companies, have become commonplace due factors such as high costs of car ownership, lack of parking, and persistent traffic congestion [1, 2, 3, 4, 5]. Traditionally, rides are provided by drivers who use their personal vehicle to provide service. However, ride-sharing platforms are likely to incorporate autonomous vehicles (AVs) into their fleets in the near future [6].

AVs managed by the ride-sharing platform can be directed to specific locations as needed. However, owning and managing an AV fleet can be costly for the ride-sharing platform, and significant technological hurdles remain before these platforms could transition to 100% autonomous fleets. For these reasons, it is likely that ride-sharing platforms will, at least initially, adopt a hybrid or *mixed* framework in which AVs operate along-side conventional, human-driven vehicles [7]. For example, the ride-sharing platform Lyft has partnered with Aptiv, a manufacturer of autonomous cars, to serve some of its ride requests in Las Vegas, Nevada with AVs [8]. In this setting, the AVs can be deployed to serve, for instance, locations with abnormally high demand.

Existing research in ride-sharing has largely focused on two ends of the autonomy spectrum. On one end are futuristic *mobility-on-demand* systems consisting of only AVs [26, 27, 28, 29, 30]. These works focus on controlling the movement of AVs to achieve objectives such as maximum throughput. On the other end, models of rider and driver behavior in conventional ride-sharing markets with only HVs and no AVs have been considered in [20, 31, 32, 33]. A common approach in these works is to consider ride-sharing as a two-sided market with passengers willing to pay for rides and drivers willing to provide rides

for compensation.

2.2 UAM Network Modeling and Constraints

The modeling of UAM networks, including the new constraints and challenges that people need to take into account before UAVs can be widely used, is being studied. Commercial mobility-on-demand operator Uber is exploring the UAM solutions for the future of on-demand urban air transportation in [9] with the hope of radically improving the urban mobility. In [9], several critical challenges we need to resolve before bringing the on-demand urban air transportation to market are addressed, including battery technology, vehicle performance and reliability, air traffic control, cost, safety, vertiport/vertistop infrastructure in cities, etc. Among these challenges, the vertiport/vertistop infrastructure can be considered as the graph related to the network; the air traffic control involves scheduling the departures, assigning routes, and reserving landing spots for the UAVs; safety can include maintaining space separation for UAVs all the time, and reserving landing spots or backup landing spots for them upon arrival. NASA also provides high-level conceptual descriptions for the future UAM operations in [10]. The paper further discusses the scope that the UAM community may perform different missions and potential hazards. The paper suggests a few ideas of future work. In the proposed research, we are especially working on the scheduling problem, with disruption management (capacity drop) and contingency response management (backup parking-spot assignment).

Similarly, studies such as [11, 12, 13, 14, 15, 16] propose various approaches to allow urban air vehicles to travel safely and efficiently through cities. These proposed ideas cover a wide range of possibilities such as allowing UAVs to land at *vertistops* or *vertiports* installed on roofs of existing buildings or within cloverleaf exchanges on freeways. Implementing corridors can help to manage the traffic flow for airspace with high demands [11]. Therefore, the vertistops or vertiports can be naturally considered as the nodes of a directed graph, while the corridors can be regarded as the edges. However,

since the vertistops/vertiports suggested have limited space, and at the same time, the space separation of the UAVs needs to be maintained at all time to ensure safety, limited landing capacity at each vertistop/vertiport has to be considered and will be one of the most important constraints in the UAM network modeling.

A star graph with several leaf nodes, a central node and links directed from leaf nodes to the central node may be served as a simple baseline of the model. Then, some intermediate nodes can be added between leaf nodes and the destination node to serve as the intermediate stops for loading or unloading, and charging of the UAVs. This model captures, for example, travel from exurbs to a main city with possible stops at additional suburbs along the way and is the configuration for all four UAM networks (metro areas of Austin, Atlanta, Boston, and San Francisco) considered the INRIX report [14].

Several simulation tools [17, 18, 19] have also been developed, many of them based on how commercial airline traffic is managed today, while others focus on safe coordination of unmanned drones. The algorithm presented in [17] is an initial implementation of UAM network management based on AutoResolver algorithm developed for traditional aviation that schedules departure and arrival across the network, with continuous trajectory management to ensure safe separation between UAVs. The Flexible engine for Fast-time evaluation of Flight environments (Fe³) is another simulation tool introduced in [18]. It is able to analyze high-density traffic system that involve a large volume of UAVs statistically. In [19], a software framework for mission-level autonomy with unmanned vehicles, OpenUxAS, is presented.

2.3 Routing and Scheduling of Vehicles

In the transportation scheduling literature, prior works have considered uncertain travel times or limitations on parking capacity separately. Particularly, [34] investigates how the flow of UAVs depends on the congestion level and finds through simulations similarities with ground highway traffic with high traffic density. However, different from today's

mobility-on-demand services on the ground, the availability of landing spots at vertistops or vertiports will be limited for UAM networks. Since UAVs will have limited power storage, safety concerns will dictate that each UAV is guaranteed an available landing spot upon arrival. In addition, UAM travel is particularly vulnerable to uncertain travel times caused by the relatively short travel distances and high variability of factors such as weather. Moreover, for UAM solutions, it is likely that the UAV will only stay on the ground for a short amount of time (e.g., several minutes) to unload and load passengers, emphasizing the importance of timely operation of the whole system.

Uncertainty in routing problems has also been studied before for ground transportation. The paper [35] provides a literature review of stochastic vehicle routing problems. Examples of more recent work are presented in [36], which studies computation of minimum-cost paths through a time-varying network and considers several classes of waiting policies. In [37], a theoretical basis for optimal routing in transportation networks with highly varying traffic conditions is provided, where the goal is to maximize the probability of arriving on time at a destination given a departure time and a time budget. Besides the traditional shortest-path routing method, Software-defined Networking (SDN) and Data Center Network (DCN) are attracting increasing amount of researchers [38]. When traditional shortest path routing protocol among multiple data centers is used, the links in the shortest path can be congested easily, which may lead to low throughput and long delay, but using SDN can help with this problem.

As mentioned earlier, the availability of landing spots upon arrival is critical for safe operation of a UAM network. While the parking availability problem is not usual in ground transportation, it can be critical for truck scheduling, where the drivers are usually required by law to park and rest after a specified amount of driving time. This problem has been addressed in [39], where the authors consider deterministic travel time between different locations and the truck drivers only have access to parking spots during specific time windows, but space limitations at the parking spots are not considered. In [40] the authors

solve a similar problem with time-dependent travel times, but do not take the availability of parking spots into account.

Scheduling problems have also been well-studied in the real-time systems community, e.g., in [41, 42], where jobs often are assumed to arrive with a fixed periodicity and in some models have an uncertainty in their processing time. Both of them study the feasibility analysis that determines whether a specified collection of hard, real-time jobs executed on a processing platform can meet all designated deadlines prior to execution. [41] presents feasibility tests for uniprocessor real-time systems using preemptive earliest deadline first (EDF) scheduling, where EDF scheduling is considered optimal in the sense of feasibility. The paper [42] explores on the feasibility tests for multiprocessor.

Our proposed model is similar to non-preemptive scheduling with aperiodic tasks and hard deadlines when we consider a simple star graph, but with different goals. For example, for finite demands, we consider scheduling to achieve prescribed deadlines without excessively early departure times. Also, as the graph considered for the UAM network becoming more complicated, the studies of typical scheduling problems can not be directly applied to our proposed research, but they can provide an insight into the fundamental limits in our model. Moreover, as indicated in [43], when preemption is not allowed and tasks have arbitrary arrivals, even finding a feasible schedule will be NP-hard. Bratley's algorithm [44] is proposed to find a feasible schedule of a set of non-preemptive tasks with arbitrary arrival times. Although in certain cases, the algorithm may still fall into the dilemma of exhaustive search, it has a great chance of effectively reducing the search space with its pruning techniques.

2.4 Disruptions to Transportation Systems

Safety of UAV scheduling is one of the most important concerns for the research of UAM solutions, and has been explored in several papers. A risk assessment framework is developed in [16] that aims to provide real-time safety evaluation and tracking capability for the

UAM management. In [16], the risk of off-nominal conditions in a UAV is assessed by calculating the potential impact area and the effects of the impact to people on the ground. In [17], the simulation tool AutoResolver is proposed that has the ability of continuously ensuring safe separation between UAVs given both spatial and temporal constraints. However, AutoResolver assumes that UAVs are expected to arrive at the destination as early as possible; in contrast, under the setting considered in this paper, we relax this assumption and impose the separation constraint between UAVs from another aspect, i.e., the limited landing capacities at the vertistops or vertiports along the routes.

In ground transportation settings, most of the disruptions in the network can be modeled as capacity reductions, where totally disabled roads have zero capacity. The challenge is then to reroute the vehicle flows to ensure resilient operation of the network, where the flows are often assumed to be continuous quantities in the network [45, 46].

In this regard, our analysis is closer to classical airspace operation, where disruptions have previously been modeled and investigated to enable efficient recovery plans after the perturbations. Much of the existing literature focuses on generating a new recovery schedule [47, 48, 49, 50, 51, 52, 53], rerouting aircrafts [54, 55, 56, 57, 58, 59], or are integrated with recovering crew schedules [60, 61, 62, 63, 64, 65, 66] while minimizing a cost related to deviation to original schedules, available resources, and other system constraints. Other literature considers airport closures as disruptions [59, 53, 52]. However, these works do not consider the capacity constraints of the airports, as needed here for the vertiports. Moreover, the present paper views the scheduling problem as a hard safety constraint rather than from the perspective of efficient operation.

2.5 Resilience in Transportation Networks

Since the interruptions, e.g., adverse weather and emergency repair, can occur and affect the transportation systems, it is importance of introducing resilience into transportation networks as emphasized in [67]. This paper also observes that increasing the adaptability of

the transportation system while maintaining functions in the presence of a shock or disruption is one of the main themes within the concept of resilience. Analysis and evaluations for resilience of transportation systems have also been explored in [68, 69, 70, 71, 72]. A comprehensive review of the concepts and methodologies for resilience in transportation systems is provided in [68]. In [69], a quantifiable resilience evaluation approach is provided to analyze the ability to recover transportation function once partial network is shut down and the reduction in network resilience caused by the removal of nodes or edges of transportation networks. A model to optimize the structure of transportation networks is proposed according to the evaluation. In this paper, the transportation network is represented by an undirected graph with the nodes as cities and edges as traffic roads. The same model of transportation networks is also considered in [71]. This paper develops and calibrates a model to evaluate traffic delays using link loads, which helps to estimate the efficiency and resilience of the transportation network. A framework for evaluating transportation risk with some strategies for resilience enhancement is provided in [72]. [73] presents an emergency landing spot detection algorithm for UAVs. Unlike the model in our work where the landing spots are fixed and the emergency landing sites can only be chosen from the fixed spots, [73] focus on detection of any available landing spot in the surrounding area once a UAV fails to arrive at the landing spot. However none of these methods are directly applicable to the UAM network setting studied in this thesis.

CHAPTER 3

MIXED AUTONOMY IN RIDE-SHARING NETWORKS

For ride-sharing platforms, significant technological and regulatory hurdles remain before ride-sharing platforms could transition to 100% autonomous fleets [74, 75]. Therefore, it is likely that ride-sharing platforms will initially adopt a *mixed* framework in which autonomous vehicles (AVs) operate alongside conventional, human-driven vehicles (HVs) [7, 76, 8].

Existing research in ride-sharing has largely focused on two ends of the autonomy spectrum. On one end are futuristic *mobility-on-demand* systems consisting of only AVs. On the other end, models of rider and driver behavior in conventional ride-sharing markets with only HVs and no AVs have been considered.

In Chapter 3–5, we study the transition from traditional ride-sharing networks to totally automated mobility-on-demand systems. In particular, we extend the model proposed in [20], which did not consider AVs, to the mixed autonomy setting under several assumptions on the vehicle-to-rider assignment possibilities, and we analyze the resulting models. We consider a network with multiple equidistant locations, and potential riders arrive at these locations with desired destinations. The network operator or *platform* determines prices for rides and compensations to drivers within the network. The price of a ride may differ among locations, but does not depend on the desired destination of each rider.

In this chapter, we study the formal ride-sharing model definition and the equilibrium conditions.

3.1 Model Definition

We now formalize the mixed autonomous ride-sharing network described above.

Riders. Among a network of n equidistant locations, a mass of θ_i potential riders arrives

at location $i \in \{1, 2, \dots, n\}$ in each period of time. Throughout, when indices are omitted from a summation expression, it is assumed the summation is over all locations 1 to n . A fraction $\alpha_{ij} \in [0, 1]$ of riders at location i are traveling to location j so that $\sum_j \alpha_{ij} = 1$ for all i . We assume $\alpha_{ii} = 0$ for all i and construct the n -by- n adjacency matrix \mathbf{A} as $[\mathbf{A}]_{ij} = \alpha_{ij}$ where $[\mathbf{A}]_{ij}$ denotes the ij -th entry of \mathbf{A} .

Human-driven vehicles (HVs). After each time period, a driver exits the platform with probability $(1 - \beta)$ and serves another ride with probability β where $\beta \in (0, 1)$. Thus, a driver's expected lifetime in the network is $(1 - \beta)^{-1}$. Each driver has an outside option of earning ω over the same lifetime.

Autonomous vehicles (AVs). The platform can choose to operate an AV in the network for a fixed cost of s each time-step. Thus, $k = s(1 - \beta)^{-1}/\omega$ is the ratio of the cost of operating an AV for the equivalent time of a driver's expected lifetime to the outside option earnings. Unlike HVs, it is assumed that AVs are in continual use and do not leave the platform.

Platform. The platform sets a price p_i for a ride from location i and correspondingly compensates a driver with c_i for providing a ride at location i . The continuous cumulative distribution of the riders' willingness to pay is denoted by $F(\cdot)$ with support $[0, \bar{p}]$. That is, when confronted with a price p for a ride, a fraction $1 - F(p)$ of riders will accept this price, and the remaining $F(p)$ fraction will balk and leave the network without requesting a ride. Note that $\theta_i(1 - F(p_i))$ is then the effective demand for rides at location i .

The description of the riders, HVs, and the platform is the same as that presented in [20]. In this work, we also introduce AVs as described above. As developed below, this addition substantially alters how the model behaves and is analyzed as compared to [20]. In addition, we make the following assumption throughout.

Assumption 1. *The network's demand pattern is stationary, i.e., \mathbf{A} and θ_i are fixed for all i . Moreover, the directed graph defined by adjacency matrix \mathbf{A} is strongly connected and $\theta_i > 0$ for all $i \in \{1, \dots, n\}$, $n \geq 2$.*

In summary, the system consists of a *platform* that sets prices, *riders* that request rides among locations, *HVs* that seek to maximize their compensation, and *AVs* managed by the platform alongside the drivers.

3.2 HV and AV Priority Assignments

The number of riders willing to pay the platform's price may be less than, equal to, or greater than the total number of HVs and AVs available at that location. When it is greater than the total number of vehicles, some riders will not be served and will leave the network. When it is less than the total number of vehicles, the platform must decide how to assign riders to vehicles. Resolving this priority assignment problem is one of the main challenges presented by the model defined above as compared to the model with no AVs as proposed in [20]. When no AVs are present, it is assumed that riders are arbitrarily assigned to drivers and any remaining HVs choose to reroute to the location of highest expected earnings. In contrast, in this chapter, we consider several priority assignments.

The first priority assignment, called *HV priority*, assigns riders to HVs before AVs and is appropriate if, *e.g.*, the platform views HVs as customers that should be accommodated and given preference over AVs. We also consider an *AV priority* assignment in which AVs are assigned rides before HVs. This priority assignment is appropriate if, *e.g.*, the platform views HVs only as a supplement when insufficient AVs are available. In Section 5.2, we consider a third, intermediate *weighted* priority assignment that assigns rides in proportion to the availability of vehicles, but we defer its definition and analysis until later.

We sometimes refer to the above defined model under any of the three priority assignments as a *mixed autonomy deployment*. For comparison, the *HV-only deployment* is obtained by assuming no AVs at any location. An *HV-only deployment* may arise by the choice of a profit-maximizing platform if the platform decides not to use any AVs; alternatively, we may consider an HV-only deployment by enforcing the constraint of no AVs at any locations, in which case it is referred to as a *forced* HV-only deployment. Similarly,

the *AV-only deployment* is obtained from the mixed autonomy deployment when there are no HVs at any locations, and a *forced AV-only deployment* arises when this condition is enforced as a constraint on the system.

3.3 Equilibrium Definition for HV Priority Assignment

We now turn to the equilibrium conditions of the above model that are induced by the stationary demand as characterized in Assumption 1 and by fixed prices and compensations set by the platform. An equilibrium for the system is a time-invariant distribution of the mass of riders, HVs, and AVs at each location satisfying certain equilibrium constraints, as formalized next; all variables are understood to refer to an equilibrium and therefore no time index is included.

We consider first HV priority assignment. Let x_i denote the mass of HVs at location i . Recall $\theta_i(1 - F(p_i))$ the mass of riders willing to pay for a ride at location i . If there are fewer riders than HVs at a location, drivers can relocate to another location to provide service in the next time period. For each $i, j \in \{1, \dots, n\}$, let y_{ij} denote such drivers at location i who relocate to location j without providing a ride. It follows that

$$\sum_{j=1}^n y_{ij} = \max \{x_i - \theta_i(1 - F(p_i)), 0\}. \quad (3.1)$$

Further, let δ_i denote the mass of new drivers who choose to enter the platform and provide service at location i at each time step. At equilibrium, it must hold that

$$x_i = \beta \left[\sum_{j=1}^n \alpha_{ji} \min \{x_j, \theta_j(1 - F(p_j))\} + \sum_{j=1}^n y_{ji} \right] + \delta_i. \quad (3.2)$$

In (3.2), observe that $\min \{x_j, \theta_j(1 - F(p_j))\}$ is the total demand the platform serves with HVs at location j , and therefore $\sum_j \alpha_{ji} \min \{x_j, \theta_j(1 - F(p_j))\}$ is the mass of HVs that find themselves located at i after completing a ride.

When the demand $\theta_i(1 - F(p_i))$ at location i exceeds the mass of available HVs x_i , the platform can choose to use AVs to meet this extra demand. Let z_i denote the mass of AVs at location i , and for each $i, j \in \{1, \dots, n\}$, let r_{ij} denote the AVs which do not get a ride at i and are relocated to location j . Then

$$z_i = \sum_{j=1}^n \alpha_{ji} \min \{z_j, \max \{\theta_j(1 - F(p_j)) - x_j, 0\}\} + \sum_{j=1}^n r_{ji}. \quad (3.3)$$

In (3.3), observe that $\min \{z_j, \max \{\theta_j(1 - F(p_j)) - x_j, 0\}\}$ is the total demand that the platform serves with AVs at location j . Moreover, $\sum_j r_{ji}$ is the mass of AVs which do not get a ride to any other location and are relocated to location i . It follows that

$$\sum_{j=1}^n r_{ij} = \max \{z_i - \max \{\theta_i(1 - F(p_i)) - x_i, 0\}, 0\}. \quad (3.4)$$

Note that, under HV priority assignment, $\sum_j r_{ij}$ depends on x_i .

For each location i , define the *expected earnings* V_i to be the average total compensation earned by a driver arriving at location i . Recall that, for each ride served at location i , drivers are compensated c_i and travel to a new location according to the demand pattern \mathbf{A} . If a driver does not serve a ride due to insufficient demand, the driver earns no compensation but is free to reroute to the location with highest expected earnings. It thus follows that the expected earnings satisfy the relationship

$$V_i = \min \left\{ \frac{\theta_i(1 - F(p_i))}{x_i}, 1 \right\} \left(c_i + \sum_{k=1}^n \alpha_{ik} \beta V_k \right) + \left(1 - \min \left\{ \frac{\theta_i(1 - F(p_i))}{x_i}, 1 \right\} \right) \beta \max_j V_j \quad (3.5)$$

for all locations i where we observe $\theta_i(1 - F(p_i))/x_i$ is the fraction of drivers at location i that serve rides, provided $\theta_i(1 - F(p_i)) \leq x_i$.

Since drivers have an outside earnings option of ω , they will enter the network at lo-

cation i if and only if $V_i \geq \omega$. Moreover, the platform is able to independently adjust each compensation c_i , so a profit maximizing platform seeking to minimize V_i is able to achieve $V_i = \omega$ for all i , so called HVs' incentive-compatibility constraints, leading to the following definition.

Definition 1. For some prices and compensations $\{p_i, c_i\}_{i=1}^n$, the collection

$\{\delta_i, x_i, y_{ij}, z_i, r_{ij}\}_{i,j=1}^n$ is an equilibrium under $\{p_i, c_i\}_{i=1}^n$ for HV priority assignment if (3.1)–(3.4) is satisfied and V_i as defined in (3.5) satisfies $V_i = \omega$ for all $i = 1, \dots, n$ such that $\delta_i + \sum_{j=1}^n y_{ji} > 0$.

3.4 Equilibrium Definition for AV Priority Assignment

In this section, we parallel the development of the previous section for AV priority assignment. The analogous equilibrium conditions are

$$x_i = \beta \left[\sum_j \alpha_{ji} \min \{x_j, \max \{\theta_j(1 - F(p_j)) - z_j, 0\}\} + \sum_j y_{ji} \right] + \delta_i \quad (3.6)$$

$$\sum_{j=1}^n y_{ij} = \max \{x_i - \max \{\theta_i(1 - F(p_i)) - z_i, 0\}, 0\} \quad (3.7)$$

$$z_i = \sum_{j=1}^n \alpha_{ji} \min \{z_j, \theta_j(1 - F(p_j))\} + \sum_j r_{ji} \quad (3.8)$$

$$\sum_{j=1}^n r_{ij} = \max \{0, z_i - \theta_i(1 - F(p_i))\}. \quad (3.9)$$

In comparing (3.6)–(3.9) to (3.1)–(3.4), notice that AV priority assignment leads to $\sum_j y_{ij}$ dependent on z_i in (3.7) whereas $\sum_{j=1}^n r_{ij}$ does not depend on x_i in (3.9).

The expected earning V_i for a driver at location i now has the form

$$V_i = \min \left\{ \frac{M_i}{x_i}, 1 \right\} \left(c_i + \sum_{k=1}^n \alpha_{ik} \beta V_k \right) + \left(1 - \min \left\{ \frac{M_i}{x_i}, 1 \right\} \right) \beta \max_j V_j, \quad (3.10)$$

$$M_i = \max \{\theta_i(1 - F(p_i)) - z_i, 0\}. \quad (3.11)$$

Again, the platform chooses compensation such that $V_i = \omega$.

Definition 2. For some prices and compensations $\{p_i, c_i\}_{i=1}^n$, the collection $\{\delta_i, x_i, y_{ij}, z_i, r_{ij}\}_{i,j=1}^n$ is an equilibrium under $\{p_i, c_i\}_{i=1}^n$ for AV priority assignment if (3.6)–(3.9) is satisfied and V_i as defined in (3.10)–(3.11) satisfies $V_i = \omega$ for all $i = 1, \dots, n$ such that $\delta_i + \sum_{j=1}^n y_{ji} > 0$.

We discuss restrictions of the present model which posits several simplifying assumptions such as equidistant locations. These simplifying assumptions allow for fundamental insights such as in Theorem 3 and in Section 5.3 below that are not obscured or confounded by additional degrees of freedom. Moreover, such assumptions might be reasonable in certain settings. For example, about 75% of taxi rides in New York City are less than three miles¹, suggesting that distance may not be a major distinguishing attribute of most rides in that market, and, as a result, the equidistance assumption would be sufficient in many situations. Equidistance is required here because we adopt a discrete time model and all parameters are on a per-ride basis. Similar to [20], normalized distance between nodes can be introduced as a new coefficient to extend the model when the equidistance assumption is relaxed. This coefficient scales ride price and driver compensation, which are then interpreted on a per-distance-unit basis.

¹As determined from almost 7 million yellow taxi trips in June 2019 available at <https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page>

CHAPTER 4

PROFIT-MAXIMIZATION WITH CONVEXIFICATION FOR HV AND AV

PRIORITY ASSIGNMENTS

We have already discussed the model for the ride-sharing network with mixed autonomy, the three different priority assignments and the equilibrium. In this chapter, we mainly focus on the HV and AV priority assignments. We first formulate the non-convex profit-maximization problems for HV and AV priority assignments, followed by their convexification.

4.1 Profit-Maximization for HV and AV Priority Assignment

We now consider the problem of maximizing profits at equilibrium. We focus on the equilibrium under prices and compensations $\{p_i, c_i\}_{i=1}^n$. This analysis is reasonable when there are large populations of HVs, AVs and riders during periods of stationary rider demand. In this case, the equilibrium captures the flow constraints in (3.1)–(3.4) or (3.6)–(3.9) and the drivers’ earnings constraints in (3.5) or (3.10)–(3.11). We first consider profit maximization with HV priority assignment and then with AV priority assignment. Under HV priority assignment, maximizing the aggregate profit across the n locations subject to the system’s equilibrium constraints yields the following optimization problem:

$$\begin{aligned} & \max_{\{p_i, c_i\}_{i=1}^n} \sum_{i=1}^n [\min \{x_i + z_i, \theta_i(1 - F(p_i))\} \cdot p_i - \min \{x_i, \theta_i(1 - F(p_i))\} \cdot c_i - z_i \cdot s] \\ \text{s.t. } & \{\delta_i, x_i, y_{ij}, z_i, r_{ij}\}_{i,j=1}^n \text{ is an equilibrium under } \{p_i, c_i\}_{i=1}^n \text{ for HV priority assignment.} \end{aligned} \tag{4.1}$$

These equilibrium conditions capture the flow constraints of all vehicles while following HVs’ incentive-compatibility constraints. However, the optimization problem (4.1) is diffi-

cult to analyze directly. Instead, we propose an equivalent optimization problem, followed by a lemma establishing the equivalence. To this end, consider as an alternative

$$\begin{aligned}
& \max_{\{p_i, \delta_i, x_i, y_{ij}, z_i, r_{ij}\}} \sum_{i=1}^n p_i \theta_i (1 - F(p_i)) - \omega \sum_{i=1}^n \delta_i - s \sum_{i=1}^n z_i \\
& \text{s.t. } d_i = \theta_i (1 - F(p_i)) \\
& x_i = \beta \left[\sum_{j=1}^n \alpha_{ji} \min \{x_j, d_j\} + \sum_{j=1}^n y_{ji} \right] + \delta_i \\
& \sum_{j=1}^n y_{ij} = \max \{x_i - d_i, 0\} \\
& z_i = \sum_{j=1}^n \alpha_{ji} \max \{d_j - x_j, 0\} + \sum_{j=1}^n r_{ji} \\
& \sum_{j=1}^n r_{ij} = z_i - \max \{d_i - x_i, 0\} \\
& p_i, \delta_i, z_i, x_i, y_{ij}, r_{ij} \geq 0 \quad \forall i, j.
\end{aligned} \tag{4.2}$$

In a certain sense formalized in the next lemma, (4.2) is equivalent to (4.1).

Lemma 1. *Assume HV priority assignment and consider the optimization problems (4.1) and (4.2). Under Assumption 1, an optimal solution to (4.2) provides an optimal solution to (4.1). In particular, the following hold:*

1. *If $(1 - \beta)\omega < \bar{p}$ or $s < \bar{p}$, then any optimal solution $\{p_i^*, \delta_i^*, x_i^*, y_{ij}^*, z_i^*, r_{ij}^*\}_{i,j=1}^n$ for (4.2) is such that $d_i^* > 0$ for all i , i.e., some riders are served at all locations. In this case, there exist compensations $\{c_i^*\}_{i=1}^n$ such that $\{\delta_i^*, x_i^*, y_{ij}^*, z_i^*, r_{ij}^*\}_{i,j=1}^n$ constitutes an equilibrium under $\{p_i^*, c_i^*\}_{i=1}^n$ for HV priority assignment. Moreover, $\{p_i^*, c_i^*\}_{i=1}^n$ is optimal for (4.1).*
2. *Conversely, if $(1 - \beta)\omega \geq \bar{p}$ and $s \geq \bar{p}$, then any optimal solution for (4.2) and any optimal equilibrium from (4.1) is such that $\delta_i^* = d_i^* = x_i^* = z_i^* = 0$ for all i , i.e., no riders are served.*

Proof. The proof of the lemma closely follows that of [20, Lemma 1], where we adjust the claim and the proof so that it applies to the mixed autonomy setting here.

We first show that the optimal value of (4.2) upper bounds the optimal value of (4.1). For this, we need to show that any solution for (4.1) satisfies $d_i = \theta_i(1 - F(p_i)) \leq x_i + z_i$. By contradiction, suppose $d_i > x_i + z_i$, so that increasing the price p_i by a small amount (and thus decreasing $\theta_i(1 - F(p_i))$) will improve the value of the objective function. Therefore, $d_i \leq x_i + z_i$ at optimum. Hence we can write the first summation of (4.1) as

$$\sum_{i=1}^n \min \{x_i + z_i, \theta_i(1 - F(p_i))\} = \sum_{i=1}^n \theta_i(1 - F(p_i)). \quad (4.3)$$

The term $\omega \sum_i \delta_i$ is the cost rate for drivers of the platform, which is a lower bound for the platform's cost on human-driven vehicles at equilibrium. Moreover, the constraints in (4.2) correspond to the equilibrium constraints in (4.1). Therefore, the optimal value of (4.2) is an upper bound for that of (4.1).

Next, we'll see that the upper bound can be reached by the optimal solution supported by some compensations $\{c_i\}_{i=1}^n$ under equilibrium.

To prove the second part of the lemma, we construct a compensation $\{c_i\}_{i=1}^n$ so that $V_i = \omega$ for all i . To that end, let

$$c_i = \begin{cases} \frac{x_i}{d_i} \cdot \omega(1 - \beta) & \text{if } d_i < x_i \\ \omega(1 - \beta) & \text{if } d_i \geq x_i. \end{cases} \quad (4.4)$$

Since we assumed that $d_i > 0$ for all i , then $c_i < \infty$ for all i and thus the compensation is well-defined. Moreover, the probability that any driver at location i is assigned to a ride is $\frac{d_i}{x_i}$ when $d_i < x_i$ and is 1 when $d_i \geq x_i$ since the driver takes the priority when drivers and AVs both exist in the platform. Therefore, the expected earnings for a single time period for a driver at location i are equal to $\omega(1 - \beta)$. Thus, the expected lifetime earnings are $V_i = \sum_j \beta^j \omega(1 - \beta) = \omega$. Hence, the solution $\{p_i, \delta_i, x_i, y_{ij}, z_i, r_{ij}\}_{i,j=1}^n$ is supported as

an equilibrium using the compensations we constructed above.

Moreover, the cost incurred by the platform under these compensations per period is

$$\sum_{i=1}^n \min \{x_i, \theta_i(1 - F(p_i))\} \cdot c_i = \sum_{i=1}^n \min \{x_i, d_i\} \cdot c_i.$$

We construct a partition for the locations so that $I_1 = \{i : d_i < x_i\}$ and $I_2 = \{i : d_i \geq x_i\}$.

Therefore

$$\begin{aligned} \sum_i \min \{x_i, d_i\} \cdot c_i &= \sum_{i \in I_1} d_i c_i + \sum_{i \in I_2} x_i c_i \\ &= \sum_{i \in I_1} d_i \cdot \frac{x_i}{d_i} \omega (1 - \beta) + \sum_{i \in I_2} x_i \omega (1 - \beta) \\ &= \sum_i x_i \omega (1 - \beta) = \sum_{i=1}^n \delta_i \omega. \end{aligned}$$

The last equality follows from the fact that $\sum_{i=1}^n x_i (1 - \beta) = \sum_{i=1}^n \delta_i$ since, at equilibrium, the mass of drivers entering the platform is equal to the mass of drivers that are leaving.

The third part of the lemma follows directly from the second part of [20, Lemma 1] since $z_i > 0$ only if $d_i > 0$ in our scenario. \square

Turning now to the case of AV priority assignment, the analogous profit-maximization problem is given by (4.5) below and as in the case of HV priority assignment, we introduce (4.6) for AV priority assignment.

$$\begin{aligned} \max_{\{p_i, c_i\}_{i=1}^n} \sum_{i=1}^n & [\min \{x_i + z_i, \theta_i(1 - F(p_i))\} \cdot p_i \\ & - \min \{x_i, \max \{\theta_i(1 - F(p_i)) - z_i, 0\}\} \cdot c_i - z_i \cdot s] \end{aligned}$$

s.t. $\{\delta_i, x_i, y_{ij}, z_i, r_{ij}\}_{i,j=1}^n$ is an equilibrium under $\{p_i, c_i\}_{i=1}^n$ for AV priority assignment.

(4.5)

$$\begin{aligned}
& \max_{\{p_i, \delta_i, x_i, y_{ij}, z_i, r_{ij}\}} \sum_{i=1}^n p_i \theta_i (1 - F(p_i)) - \omega \sum_{i=1}^n \delta_i - s \sum_{i=1}^n z_i \\
& \text{s.t. } d_i = \theta_i (1 - F(p_i)) \\
& x_i = \beta \left[\sum_j \alpha_{ji} \max \{d_j - z_j, 0\} + \sum_j y_{ji} \right] + \delta_i \\
& \sum_{j=1}^n y_{ij} = x_i - \max \{d_i - z_i, 0\} \\
& z_i = \sum_{j=1}^n \alpha_{ji} \min \{d_j, z_j\} + \sum_{j=1}^n r_{ji} \\
& \sum_{j=1}^n r_{ij} = \max \{z_i - d_i, 0\} \\
& p_i, \delta_i, z_i, x_i, y_{ij}, r_{ij} \geq 0 \quad \forall i, j.
\end{aligned} \tag{4.6}$$

Mirroring Lemma 1, optimization problems (4.5) and (4.6) are equivalent in a certain sense.

Lemma 2. *Assume AV priority assignment and consider the optimization problems (4.5) and (4.6). Under Assumption 1, an optimal solution to (4.6) provides an optimal solution to (4.5). In particular, the following hold:*

1. *If $(1 - \beta)\omega < \bar{p}$ or $s < \bar{p}$, then any optimal solution $\{p_i^*, \delta_i^*, x_i^*, y_{ij}^*, z_i^*, r_{ij}^*\}_{i,j=1}^n$ for (4.6) is such that $d_i^* > 0$ for all i , i.e., some riders are served at all locations. In this case, there exist compensations $\{c_i^*\}_{i=1}^n$ such that $\{\delta_i^*, x_i^*, y_{ij}^*, z_i^*, r_{ij}^*\}_{i,j=1}^n$ constitutes an equilibrium under $\{p_i^*, c_i^*\}_{i=1}^n$ for AV priority assignment. Moreover, $\{p_i^*, c_i^*\}_{i=1}^n$ is optimal for (4.5).*
2. *Conversely, if $(1 - \beta)\omega \geq \bar{p}$ and $s \geq \bar{p}$, then any optimal solution for (4.6) and any optimal equilibrium from (4.5) is such that $\delta_i^* = d_i^* = x_i^* = z_i^* = 0$ for all i , i.e., no riders are served.*

Proof. The proof is similar to that of Lemma 1 by setting

$$c_i = \begin{cases} \frac{x_i}{d_i - z_i} \cdot \omega(1 - \beta) & \text{if } d_i > z_i \\ \omega(1 - \beta) & \text{if } d_i \leq z_i. \end{cases}$$

□

From Lemma 1 (resp., Lemma 2), we conclude that it is without loss of generality for us to focus on the optimization problem (4.2) (resp., (4.6)) for the rest of the chapter when considering HV (resp., AV) priority assignment.

Moreover, while the objective function of (4.2) (resp., (4.6)) is not concave in general, it is concave for distributions for which the term $p \cdot (1 - F(p))$ is concave in the fractional demand $d = 1 - F(p)$, which can be set by the platform by adjusting the price p (note that $p \cdot d = d \cdot F^{-1}(1 - d)$). For example, the uniform distribution, exponential distribution and Pareto distribution all satisfy this concavity requirement. Throughout the rest of the chapter, we focus on the case where the rider's willingness to pay is such that the revenue of the platform is concave in d .

Assumption 2. *The cumulative distribution $F(\cdot)$ of the riders' willingness to pay is such that $d \cdot F^{-1}(1 - d)$ is concave in d .*

Under HV (resp., AV) priority assignment, we have converted (4.1) (resp., (4.5)) to the alternative optimization problem (4.2) (resp., (4.6)). Next, we will further convert (4.2) (resp., (4.6), henceforth written as (4.2)/(4.6)) to an alternative optimization problem that is also convex, allowing for efficient—and in some cases, closed form—solution computation.

4.2 Convexification of Profit Maximization

Even when (4.2)/(4.6) possesses a concave objective function, the constraints are non-convex and cannot be simply convexified so that solving (4.2)/(4.6) remains computation-

ally difficult, *i.e.*, nonconvex. This section introduces alternative optimization problems of the mixed autonomy deployment for which the optimal profits will be the same as that of (4.2)/(4.6).

While the optimal profits are the same, the optimal solutions of the alternative optimization problems are not exactly the same as those calculated in the original problems (4.2)/(4.6). As a result, a main difference between the original problems and their alternatives is that, while the original problems and their optimal solutions can always be interpreted physically, the alternatives are purely mathematical problems. However, given the optimal solution of the alternative problems, we show that it is possible to compute an optimal solution for the original problems (4.2)/(4.6) with identical profit and vice versa. Moreover, by eliminating p_i using $d_i = \theta_i(1 - F(p_i))$ in substitution, the alternative optimization problems are seen to be convex optimization problems under Assumption 2. But, for clarity, we leave p_i in the alternative optimization problems to allow for comparison to the original problems. Furthermore, the alternative optimization problems become quadratic optimization problems with linear constraints when $F(\cdot)$ is a uniform distribution.

First, assume HV priority assignment, and consider the optimization problem given by

$$\begin{aligned}
& \max_{\{p_i, \delta_i, x_i, z_i, r_{ij}\}} \sum_{i=1}^n p_i \theta_i (1 - F(p_i)) - \omega \sum_{i=1}^n \delta_i - s \sum_{i=1}^n z_i \\
& \text{s.t.} \quad d_i = \theta_i (1 - F(p_i)) \\
& \quad \quad x_i = \beta \sum_{j=1}^n \alpha_{ji} x_j + \delta_i \\
& \quad \quad z_i = \sum_{j=1}^n \alpha_{ji} (d_j - x_j) + \sum_{j=1}^n r_{ji} \\
& \quad \quad \sum_{j=1}^n r_{ij} = z_i - (d_i - x_i) \\
& \quad \quad p_i, \delta_i, x_i, z_i, r_{ij} \geq 0 \quad \forall i, j.
\end{aligned} \tag{4.7}$$

In the following, we regard (4.2) as the *original* optimization problem and (4.7) as the *alternative* optimization problem for HV priority assignment.

Theorem 1 below states that (4.2) and (4.7) have the same optimal profits for any β , s , ω and adjacency matrix \mathbf{A} . Moreover, given one optimal solution for (4.2) or (4.7), it is possible to compute an optimal solution for the other.

Theorem 1. *Assume HV priority assignment, and consider the original optimization problem (4.2) and alternative optimization problem (4.7). Let*

$$\mathbf{u}^{ori*} = \{p_i^{ori*}, \delta_i^{ori*}, z_i^{ori*}, x_i^{ori*}, y_{ij}^{ori*}, r_{ij}^{ori*}\}_{i,j=1}^n \quad (4.8)$$

be an optimal solution for (4.2) and

$$\mathbf{u}^{alt*} = \{p_i^{alt*}, \delta_i^{alt*}, z_i^{alt*}, x_i^{alt*}, r_{ij}^{alt*}\}_{i,j=1}^n \quad (4.9)$$

be an optimal solution for (4.7). Then the following hold under Assumptions 1 and 2:

- *The original optimization problem and the alternative problem obtain the same optimal profits for all possible choices of β , s , ω and adjacency matrix \mathbf{A} .*
- *The optimal solutions satisfy $x^{ori*} = x^{alt*}$, $z^{ori*} = z^{alt*}$, $p^{ori*} = p^{alt*}$ and $\delta^{ori*} = \delta^{alt*}$.*
- *If $\theta_i(1 - F(p_i^{ori*})) \leq x_i^{ori*}$ for all i in the original optimization problem, then $z_i^{ori*} = 0$ for all i and setting $r_{ij}^{alt*} = y_{ij}^{ori*}$ for all i, j constitutes an optimal solution for the alternative problem.*
- *If $\theta_i(1 - F(p_i^{alt*})) \leq x_i^{alt*}$ for all i in the alternative optimization problem, then $z_i^{alt*} = 0$ for all i and setting $y_{ij}^{ori*} = r_{ij}^{alt*}$, $r_{ij}^{ori*} = 0$ constitutes an optimal solution for the original optimization problem.*

Proof. Let ϕ^{ori*} and ϕ^{alt*} be the optimal profits of the two problems (4.2) and (4.7), respectively, and let $d_i^{ori*} = \theta_i(1 - F(p_i^{ori*}))$ and $d_i^{alt*} = \theta_i(1 - F(p_i^{alt*}))$.

To prove that the optimal profits of the two problems are equal, we first show that $\phi^{ori*} \leq \phi^{alt*}$ and then $\phi^{ori*} \geq \phi^{alt*}$.

We first introduce Lagrange multipliers λ_i , μ_i , and γ_i and establish the following inequalities for all i, j derived from the KKT conditions that are necessary for any optimal solution of (4.7):

$$\text{(constraints on } \delta_i) \quad -\omega + \lambda_i \leq 0 \quad (4.10)$$

$$\text{(constraints on } x_i) \quad \sum_j \alpha_{ij}(\beta\lambda_j - \mu_j) - \lambda_i + \gamma_i \leq 0 \quad (4.11)$$

$$\text{(constraints on } z_i) \quad -s + \gamma_i - \mu_i \leq 0 \quad (4.12)$$

$$\text{(constraints on } r_{ij}) \quad \mu_j - \gamma_i \leq 0. \quad (4.13)$$

We now consider three cases to prove $\phi^{ori*} \leq \phi^{alt*}$.

Case 1: $d_i^{ori*} \geq x_i^{ori*}$ for all i . Then \mathbf{u}^{ori*} is feasible for the alternative problem because both problems are in fact the same optimization problem in this case. Therefore $\phi^{ori*} \leq \phi^{alt*}$.

Case 2: $d_i^{ori*} \leq x_i^{ori*}$ for all i . Then the AVs are not needed in any location and $z_i = 0, r_{ij} = 0 \quad \forall i, j$. Then the original optimization problem becomes

$$\begin{aligned} \max_{\{p_i, \delta_i, x_i, y_{ij}, z_i, r_{ij}\}} \quad & \sum_{i=1}^n p_i \theta_i (1 - F(p_i)) - \omega \sum_{i=1}^n \delta_i \\ \text{s.t.} \quad & d_i = \theta_i (1 - F(p_i)) \\ & x_i = \beta \left[\sum_{j=1}^n \alpha_{ji} d_j + \sum_{j=1}^n y_{ji} \right] + \delta_i \\ & \sum_{j=1}^n y_{ij} = x_i - d_i \\ & p_i, \delta_i, x_i, y_{ij} \geq 0 \quad \forall i, j. \end{aligned} \quad (4.14)$$

Let $z_i^{alt} = 0$ and $y_{ij}^{alt} = 0 \quad \forall i, j$. Then the alternative problem becomes exactly the same problem as (4.14) when we substitute r_{ij} with y_{ij} , which proves the claim.

Case 3: There exists some location i such that $x_i^{ori*} > d_i^{ori*}$ and some location j such that $x_j^{ori*} < d_j^{ori*}$. In this case, if there is no i such that $x_i^{ori*} = d_i^{ori*}$, then let $I_1 = \{i : x_i^{ori*} > d_i^{ori*}\}$ and let $I_2 = \{i : x_i^{ori*} < d_i^{ori*}\}$. We can then consider an aggregated network with locations 1 and 2 representing the combined locations in I_1 and I_2 , respectively.

Hence, in this aggregated network, $\alpha_{11} \geq 0$ and $\alpha_{22} \geq 0$; $\alpha_{12} > 0$ and $\alpha_{21} > 0$ by our assumption that the directed graph defined by adjacency matrix \mathbf{A} is strongly connected.

Since $d_1^{ori*} < x_1^{ori*}$, then $z_1^{ori*} = 0$. Meanwhile, $z_1^{ori*} = \max\{d_1^{ori*} - x_1^{ori*}, 0\}\alpha_{11} + \max\{d_2^{ori*} - x_2^{ori*}, 0\}\alpha_{21} + \sum_{j=1}^2 r_{j1}^{ori*} = (d_2^{ori*} - x_2^{ori*})\alpha_{21} + \sum_{j=1}^2 r_{j1}^{ori*}$ since $d_2^{ori*} - x_2^{ori*} > 0$ and $d_1^{ori*} - x_1^{ori*} < 0$. Hence $z_1^{ori*} > 0$ which leads to a contradiction. Therefore, if there is no i such that $x_i^{ori*} = d_i^{ori*}$, then either $x_i^{ori*} > d_i^{ori*}$ for all i or $x_i^{ori*} < d_i^{ori*}$ for all i .

If there exists i such that $x_i^{ori*} = d_i^{ori*}$, define I_1 and I_2 as above and introduce $I_3 = \{i : x_i^{ori*} = d_i^{ori*}\}$. Similar to the above argument, we can obtain that $z_1^{ori*} = z_3^{ori*} = 0$. Since $z_1^{ori*} = \sum_{j=1}^3 \alpha_{j1} \max\{d_j^{ori*} - x_j^{ori*}, 0\} + \sum_{j=1}^3 r_{j1}^{ori*}$ while $d_2^{ori*} - x_2^{ori*} > 0$, then $\alpha_{21} = 0$. Similarly, we must have $\alpha_{23} = 0$. Therefore, we have $\alpha_{22} = 1$ since $\sum_{j=1}^3 \alpha_{ij} = 1$. However, $\alpha_{22} = 1$ means that some components in the graph are not strongly connected with the others, which contradicts our assumption. Hence this mixed situation cannot be an optimal solution for the problem.

Thus, up to now, we have shown that $\phi^{ori*} \leq \phi^{alt*}$. Next we show that $\phi^{ori*} \geq \phi^{alt*}$.

Case 1: If $d_i^{alt*} \geq x_i^{alt*}$ for all i , then \mathbf{u}^{ori*} is feasible for the original problem because both problems are in fact the same optimization problem in this case. Therefore $\phi^{ori*} \geq \phi^{alt*}$.

Case 2: If $d_i^{alt*} \leq x_i^{alt*}$ for all i , we want to show that in this case, $z_i^{alt*} = 0$ for all i and then \mathbf{u}^{alt*} will be feasible for the original optimization by setting $y_{ij}^{ori} = r_{ij}^{alt}$ with $r_{ij}^{ori} = 0$ for all i, j .

Fix $d_i^{alt*} \leq x_i^{alt*}$ for all i , then if $z_i = 0$ is a feasible solution for (4.7), then it will be the optimal the solution since any increase in z_i will increase the cost and reduce the profit.

We'll show below that given $d_i^{alt*} \leq x_i^{alt*}$ and setting $z_i = 0$ for all i for (4.7), there exists r_{ij} that satisfies the constraints for (4.7) and thus constitutes a feasible solution for the alternative optimization problem.

$$\begin{aligned} \sum_{j=1}^n r_{ij} &= x_i - d_i \\ \sum_{j=1}^n r_{ji} &= \sum_{j=1}^n \alpha_{ji}(x_j - d_j) \\ r_{ij}, (x_i - d_i) &\geq 0 \quad \forall i, j. \end{aligned} \tag{4.15}$$

The new constraints can be described as in (4.15). We can reformulate (4.15) into (4.16) below where \mathbf{R} is an n by n matrix and $[\mathbf{R}]_{ij} = r_{ij}$; $\mathbf{\Delta}$ is an n by 1 vector and $[\mathbf{\Delta}]_i = x_i - d_i$; $\mathbf{1}$ is an n by 1 one's vector.

$$\begin{aligned} \mathbf{R}\mathbf{1} &= \mathbf{\Delta} \\ \mathbf{R}^T\mathbf{1} &= \mathbf{A}^T\mathbf{\Delta} \\ \mathbf{\Delta} &\geq 0 \\ \mathbf{R}_{ij} &\geq 0 \end{aligned} \tag{4.16}$$

We can then vectorize \mathbf{R} to $\hat{\mathbf{R}}$ (in row) so that (4.16) will transform into (4.17).

$$\begin{aligned} \mathbf{M}\hat{\mathbf{R}} &= \mathbf{b} \\ \hat{\mathbf{R}}_{ij} &\geq 0 \end{aligned} \tag{4.17}$$

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_1 \\ \mathbf{M}_2 \end{bmatrix} \text{ where } \mathbf{M}_1 \text{ and } \mathbf{M}_2 \text{ are both } n \text{ by } n^2 \text{ matrices:}$$

$$\mathbf{M}_1 = \mathbf{I}_{n \times n} \otimes \mathbf{1}^T = \begin{bmatrix} 1 & \dots & 1 & 0 & \dots & 0 & \dots & 0 & \dots & 0 \\ 0 & \dots & 0 & 1 & \dots & 1 & \dots & 0 & \dots & 0 \\ \vdots & & & \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \dots & 0 & 0 & \dots & 0 & \dots & 1 & \dots & 1 \end{bmatrix} \text{ and}$$

$$\mathbf{M}_2 = \mathbf{1}^T \otimes \mathbf{I}_{n \times n} = \begin{bmatrix} \mathbf{I}_{n \times n} & \mathbf{I}_{n \times n} & \dots & \mathbf{I}_{n \times n} \end{bmatrix}.$$

$$\hat{\mathbf{R}} = [\mathbf{R}_{11}, \mathbf{R}_{12}, \dots, \mathbf{R}_{1n}, \dots, \mathbf{R}_{n1}, \mathbf{R}_{n2}, \dots, \mathbf{R}_{nn}]^T \text{ is a } n^2 \text{ by } 1 \text{ vector.}$$

$$\mathbf{b} = \begin{bmatrix} \Delta \\ A^T \Delta \end{bmatrix} \text{ is a } 2n \text{ by } 1 \text{ vector.}$$

By Farka's Lemma, to prove that (4.17) has a feasible solution $\hat{\mathbf{R}}$: that is, $\exists \hat{\mathbf{R}}$ s.t. $\mathbf{M}\hat{\mathbf{R}} = \mathbf{b}$ and $\hat{\mathbf{R}} \geq 0$, we only need to disprove the claim that $\exists \mathbf{v} \in \mathbb{R}^{2n}$ s.t. $\mathbf{M}^T \mathbf{v} \geq 0$ and $\mathbf{b}^T \mathbf{v} < 0$. Denote v_i as the i th element of \mathbf{v} .

Let $\mathbf{v} \in \mathbb{R}^{2n}$ s.t. $\mathbf{M}^T \mathbf{v} \geq 0$,

$$\mathbf{M}^T \mathbf{v} = \begin{bmatrix} \mathbf{M}_1^T & \mathbf{M}_2^T \end{bmatrix} \mathbf{v} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{I}_{n \times n} \\ \mathbf{0} & \mathbf{1} & \dots & \mathbf{0} & \mathbf{I}_{n \times n} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{1} & \mathbf{I}_{n \times n} \end{bmatrix} \mathbf{v}.$$

Hence, $v_i + v_j \geq 0$ for all $i = 1, \dots, n$ and $j = n + 1, \dots, 2n$.

Now consider $\mathbf{b}^T \mathbf{v}$.

$$\begin{aligned} \mathbf{b}^T \mathbf{v} &= \begin{bmatrix} \Delta^T & \Delta^T A \end{bmatrix} \mathbf{v} = \Delta^T \begin{bmatrix} \mathbf{I}_{n \times n} & A \end{bmatrix} \mathbf{v} \\ &= \Delta^T \begin{bmatrix} \vdots \\ v_i + \sum_{j=1}^n \alpha_{ij} v_{j+n} \\ \vdots \end{bmatrix} = \Delta^T \begin{bmatrix} \vdots \\ \sum_{j=1}^n \alpha_{ij} (v_i + v_{j+n}) \\ \vdots \end{bmatrix} \end{aligned}$$

The last equality comes from the fact that $\sum_{j=1}^n \alpha_{ij} = 1$. Moreover, since $v_i + v_{j+n} \geq 0$

for all $i, j = 1, \dots, n$ as previously mentioned, and $\mathbf{\Delta} \geq 0$, then $\mathbf{b}^T \mathbf{v} \geq 0$. Hence we disproved the claim that $\exists \mathbf{v} \in \mathbb{R}^{2n}$ s.t. $\mathbf{M}^T \mathbf{v} \geq 0$ and $\mathbf{b}^T \mathbf{v} < 0$.

Therefore (4.17) has a feasible solution $\hat{\mathbf{R}}$ and thus (4.15) has feasible solution r_{ij} for all i, j . Hence $z_i^{alt*} = 0$ for all i and then \mathbf{u}^{alt*} will be feasible for the original optimization by setting $y_{ij}^{ori} = r_{ij}^{alt}$ with $r_{ij}^{ori} = 0$ for all i, j .

Case 3: There exist β and k such that the optimal solution \mathbf{u}^{alt*} does not satisfy the two situations above, which means there exist locations such that $d_i^{alt*} > x_i^{alt*}$ for some i and $d_j^{alt*} < x_j^{alt*}$ for some j . Let $I_1 = \{i : x_i^{alt*} < d_i^{alt*}\}$ and let $I_2 = \{i : x_i^{alt*} \geq d_i^{alt*}\}$ and we can consider an aggregated network with locations 1 and 2 representing the combined locations in I_1 and I_2 , respectively. Knowing $x_1^{alt*} < d_1^{alt*}$, suppose $x_2^{alt*} > d_2^{alt*}$ (since there exists at least an i such that $d_i^{alt*} < x_i^{alt*}$). Then we can rewrite the constraints of (4.7) as below:

$$\begin{aligned}
x_1 &= \beta(\alpha_{11}x_1 + \alpha_{21}x_2) + \delta_1 \\
x_2 &= \beta(\alpha_{12}x_1 + \alpha_{22}x_2) + \delta_2 \\
z_1 &= \alpha_{11}(d_1 - x_1) + \alpha_{21}(d_2 - x_2) + r_{11} + r_{21} \\
z_2 &= \alpha_{12}(d_1 - x_1) + \alpha_{22}(d_2 - x_2) + r_{12} + r_{22} \\
r_{11} + r_{12} &= z_1 - (d_1 - x_1) \\
r_{21} + r_{22} &= z_2 - (d_2 - x_2) \\
p_i, \delta_i, z_i, x_i, r_{ij} &\geq 0 \quad \forall i, j.
\end{aligned} \tag{4.18}$$

For convenience, denote $\Delta_1 = d_1^{alt*} - x_1^{alt*}$ and $\Delta_2 = d_2^{alt*} - x_2^{alt*}$. Obviously, $\Delta_1 > 0$ and $\Delta_2 < 0$.

Since $r_{11} + r_{12} \geq 0$, then $z_1^{alt*} > \Delta_1 > 0$ and this indicates that $\gamma_1 - \mu_1 = s$. Hence $\mu_1 - \gamma_1 = -s \neq 0$ and thus $r_{11}^{alt*} = 0$.

Since $x_2^{alt*} > d_2^{alt*} > 0$, then $x_1^{alt*} > 0$ since $\alpha_{21} > 0$ for strong connectivity of the network. Moreover, these indicates that $\delta_1^{alt*} + \delta_2^{alt*} = (1 - \beta)(x_1^{alt*} + x_2^{alt*}) > 0$

As $z_2 \geq 0$, then $r_{21}^{alt*} + r_{22}^{alt*} \geq x_2^{alt*} - d_2^{alt*} > 0$. Suppose $r_{21}^{alt*} = 0$, then $r_{22}^{alt*} > 0$, then $\mu_2 - \gamma_2 = 0$ and hence $z_2^{alt*} = 0$. Then $z_1^{alt*} = \alpha_{11}\Delta_1 + \alpha_{21}\Delta_2$. Knowing $z_1^{alt*} \geq \Delta_1$ requires $\alpha_{11} = 1, \alpha_{21} = 0$ (because $\Delta_2 < 0$) and this network is no longer strongly connected which contradicts the assumption. Therefore $r_{21}^{alt*} > 0$ and thus $\mu_1 - \gamma_2 = 0$. We can get $\mu_2 - \gamma_1 = (\mu_2 - \gamma_2) + (\gamma_2 - \mu_1) + (\mu_1 - \gamma_1) \leq 0 + 0 - s < 0$ so that $r_{12}^{alt*} = 0$. Therefore $z_1^{alt*} = \Delta_1; r_{21}^{alt*} = z_1^{alt*} - \alpha_{11}\Delta_1 - \alpha_{21}\Delta_2 = \alpha_{12}\Delta_1 - \alpha_{21}\Delta_2$.

With all the preliminary results above, we now divide the problem into two cases: $z_2^{alt*} = 0$ or $z_2^{alt*} > 0$.

Suppose $z_2^{alt*} = 0, r_{22}^{alt*} = \Delta_2 - r_{21}^{alt*} = -\alpha_{12}\Delta_1 - \alpha_{22}\Delta_2 > 0$, which implies that $\alpha_{12}\Delta_1 < -\alpha_{22}\Delta_2$ and $\mu_2 - \gamma_2 = 0$. Hence $\gamma_1 - \mu_2 = (\gamma_1 - \mu_1) + (\mu_1 - \gamma_2) + (\gamma_2 - \mu_2) = s + 0 + 0 = s$.

Then (4.11) yields that

$$\beta(\alpha_{11}\lambda_1 + \alpha_{12}\lambda_2) - \lambda_1 + \alpha_{11}s + \alpha_{12}s = 0 \quad (4.19)$$

$$\beta(\alpha_{21}\lambda_1 + \alpha_{22}\lambda_2) - \lambda_2 + \alpha_{21} \cdot 0 + \alpha_{22} \cdot 0 = 0 \quad (4.20)$$

If $\lambda_2 = \omega$, then (4.20) shows that $\beta\alpha_{21}\lambda_1 = (1 - \beta\alpha_{22})\lambda_2 > (\beta - \beta\alpha_{22})\lambda_2 = \beta\alpha_{21}\lambda_2$. Hence $\lambda_1 > \lambda_2 > \omega$ which contradicts to (4.10). Therefore, $\lambda_2 < \omega \Rightarrow \delta_2^{alt*} = 0$. Since $\delta_1^{alt*} + \delta_2^{alt*} > 0$, then $\delta_1^{alt*} > 0$ and $\lambda_1 = \omega, \lambda_2 = \frac{\beta\alpha_{21}}{1-\beta\alpha_{22}} \cdot \omega$. Applying this result to (4.19) gives $s = \frac{(1-\beta)(1+\beta\alpha_{12}-\beta\alpha_{22})}{1-\beta\alpha_{22}} \cdot \omega > (1 - \beta)\omega$.

Let $p_i^{ori} = p_i^{alt*}, r_{ij}^{ori} = 0, y_{ij}^{ori} = r_{ij}^{alt*}$ for all $i, j; \delta_1^{ori} = (1 - \beta)(d_1^{alt*} + x_2^{alt*}), \delta_2^{ori} = 0, z_1^{ori} = z_2^{ori} = 0, x_1^{ori} = d_1^{alt*}$ and $x_2^{ori} = x_2^{alt*}$. Then we can see that

$\mathbf{u}^{ori} = \{p_i^{ori}, \delta_i^{ori}, z_i^{ori}, x_i^{ori}, y_{ij}^{ori}, r_{ij}^{ori}\}_{i,j=1}^2$ would be a feasible solution for (4.2). This solution increases the cost by $\omega \cdot (\delta_1^{ori} - \delta_1^{alt*} + \delta_2^{ori} - \delta_2^{alt*}) = (1 - \beta)\omega\Delta_1$, decreases the cost by $s \cdot (z_1^{alt*} - z_1^{ori} + z_2^{alt*} - z_2^{ori}) = s \cdot \Delta_1 > (1 - \beta)\omega\Delta_1$. The net profit increases, hence there always exists a solution for the original optimization problem that has a higher profit and thus the solution is not optimal (since we've already proved that $\phi^{ori*} \leq \phi^{alt*}$).

Therefore $z_2^{alt*} > 0$, which indicates $r_{22}^{alt*} = 0$ and $\mu_2 - \gamma_2 = s$. Hence $\gamma_1 - \mu_2 = (\gamma_1 - \mu_1) + (\mu_1 - \gamma_2) + (\gamma_2 - \mu_2) = s + 0 + s = 2s$. Moreover, $z_2^{alt*} = \alpha_{12}\Delta_1 + \alpha_{22}\Delta_2 > 0$ implies $\alpha_{12}\Delta_1 > -\alpha_{22}\Delta_2$

Then (4.11) yields that

$$\beta(\alpha_{11}\lambda_1 + \alpha_{12}\lambda_2) - \lambda_1 + \alpha_{11}s + \alpha_{12} \cdot 2s = 0 \quad (4.21)$$

$$\beta(\alpha_{21}\lambda_1 + \alpha_{22}\lambda_2) - \lambda_2 + \alpha_{21} \cdot 0 + \alpha_{22} \cdot s = 0 \quad (4.22)$$

Suppose $\lambda_1 = \lambda_2 = \omega$, then $(1 + \alpha_{12})s = (1 - \beta)\omega = \alpha_{22}s$. But $s > 0$ and $1 + \alpha_{12} > 1 > \alpha_{22}$, thus $(1 + \alpha_{12})s < \alpha_{22}s$. Therefore we cannot have $\delta_1^{alt*} > 0$ and $\delta_2^{alt*} > 0$. Suppose $\lambda_2 = \omega$. Then solving the system of equations gives $\lambda_1 = \frac{1 + \alpha_{12} - \beta\alpha_{22}}{\beta(2\alpha_{21} + \alpha_{12} - 1) + \alpha_{22}} \cdot \omega > \omega$, which contradicts the KKT condition (4.10). Hence $\lambda_2 < \omega$ implies that $\delta_2^{alt*} = 0$ and thus $\delta_1^{alt*} > 0$. Therefore, $\lambda_1 = \omega$, $\lambda_2 = \frac{\beta(2\alpha_{21} + \alpha_{12} - 1) + \alpha_{22}}{1 + \alpha_{12} - \beta\alpha_{22}} \cdot \omega$ and $s = (1 - \beta) - \frac{(1 - \beta)^2\alpha_{12}}{1 + \alpha_{12} - \beta\alpha_{22}} \cdot \omega$.

Let $p_i^{ori} = p_i^{alt*}$, $y_{ij}^{ori} = 0$ for all i, j ; $x_1^{ori} = x_2^{ori} = \delta_1^{ori} = \delta_2^{ori} = 0$, $z_1^{ori} = d_1^{alt*}$ and $z_2^{ori} = \alpha_{12}d_1^{alt*} + \alpha_{22}d_2^{alt*}$; $r_{21}^{ori} = \alpha_{12}d_1^{alt*} - \alpha_{21}d_2^{alt*}$ and $r_{11}^{ori} = r_{12}^{ori} = r_{22}^{ori} = 0$ (notice that $r_{21}^{ori} > 0$ since $\alpha_{12}\Delta_1 > -\alpha_{22}\Delta_2$ implies that $\alpha_{12}d_1^{alt*} + \alpha_{22}d_2^{alt*} > \alpha_{12}x_1^{alt*} + \alpha_{22}x_2^{alt*} = \frac{x_2^{alt*}}{\beta} > x_2^{alt*} > d_2^{alt*}$ and thus $\alpha_{12}d_1^{alt*} - \alpha_{21}d_2^{alt*} > 0$). Then $\mathbf{u}^{ori} = \{p_i^{ori}, \delta_i^{ori}, z_i^{ori}, x_i^{ori}, y_{ij}^{ori}, r_{ij}^{ori}\}_{i,j=1}^2$ would be a feasible solution for (4.2). This solution decreases the cost by $\omega \cdot (\delta_1^{alt*} - \delta_1^{ori} + \delta_2^{alt*} - \delta_2^{ori}) = (1 - \beta)\omega(x_1^{alt*} + x_2^{alt*})$, increases the cost by $s \cdot (z_1^{ori} - z_1^{alt*} + z_2^{ori} - z_2^{alt*}) = s \cdot (x_1^{alt*} + \alpha_{12}x_1^{alt*} + \alpha_{22}x_2^{alt*}) = (1 - \beta)\omega(x_1^{alt*} + x_2^{alt*})$. The net profit is not changing, hence there always exists a solution for the original optimization problem that has the same profit which proves the claim. \square

Turning our attention to AV priority assignment case, consider the optimization problem

$$\begin{aligned} & \max_{\{p_i, \delta_i, x_i, y_{ij}, z_i, r_{ij}\}} \sum_{i=1}^n p_i \theta_i (1 - F(p_i)) - \omega \sum_{i=1}^n \delta_i - s \sum_{i=1}^n z_i \\ & \text{s.t. } d_i = \theta_i (1 - F(p_i)) \end{aligned}$$

$$\begin{aligned}
x_i &= \beta \left[\sum_j \alpha_{ji} (d_j - z_j) + \sum_j y_{ji} \right] + \delta_i \\
\sum_{j=1}^n y_{ij} &= x_i - (d_i - z_i) \\
z_i &= \sum_{j=1}^n \alpha_{ji} z_j \\
p_i, \delta_i, z_i, x_i, y_{ij} &\geq 0 \quad \forall i, j.
\end{aligned} \tag{4.23}$$

Similar to above, we regard (4.6) as the *original* optimization problem and (4.23) as the *alternative* optimization problem for AV priority assignment. The next theorem mirrors Theorem 1.

Theorem 2. *Consider the original optimization problem (4.6) and alternative optimization problem (4.23). Let*

$$\mathbf{u}^{ori*} = \{p_i^{ori*}, \delta_i^{ori*}, z_i^{ori*}, x_i^{ori*}, y_{ij}^{ori*}, r_{ij}^{ori*}\}_{i,j=1}^n \tag{4.24}$$

be an optimal solution for (4.6) and

$$\mathbf{u}^{alt*} = \{p_i^{alt*}, \delta_i^{alt*}, z_i^{alt*}, x_i^{alt*}, y_{ij}^{alt*}\}_{i,j=1}^n \tag{4.25}$$

be an optimal solution for (4.23). Then the following holds under Assumptions 1 and 2:

- *The original optimization problem and the alternative problem obtain the same optimal profits for all possible choices of β , s , ω and adjacency matrix \mathbf{A} .*
- *The optimal solutions satisfy $x_i^{ori*} = x_i^{alt*}$, $z_i^{ori*} = z_i^{alt*}$, $p_i^{ori*} = p_i^{alt*}$ and $\delta_i^{ori*} = \delta_i^{alt*}$.*
- *If $\theta_i(1 - F(p_i^{ori*})) \leq z_i^{ori*}$ for all i in the original optimization problem, then $x_i^{ori*} = 0$ for all i and setting $y_{ij}^{alt*} = r_{ij}^{ori*}$ for all i, j constitutes an optimal solution for the alternative problem.*

- If $\theta_i(1 - F(p_i^{alt*})) \leq z_i^{alt*}$ for all i in the alternative optimization problem, then $x_i^{alt*} = 0$ for all i and setting $r_{ij}^{ori*} = y_{ij}^{alt*}$, $y_{ij}^{ori*} = 0$ constitutes an optimal solution for the original optimization problem.

Proof. The proving strategy is the same as Theorem 1. Let ϕ^{ori*} and ϕ^{alt*} represent the optimal profits of the two problems (4.6) and (4.23), respectively, and let $d_i^{ori*} = \theta_i(1 - F(p_i^{ori*}))$ and $d_i^{alt*} = \theta_i(1 - F(p_i^{alt*}))$.

The KKT conditions related to all of the decision variables (except for the variable p_i since $F(p_i)$ can be some general function of p_i) are:

$$\text{(constraints on } \delta_i) : -\omega + \lambda_i \leq 0 \quad (4.26)$$

$$\text{(constraints on } x_i) : -\lambda_i + \gamma_i \leq 0 \quad (4.27)$$

$$\text{(constraints on } z_i) : -s - \sum_j \alpha_{ij}(\beta\lambda_j - \mu_j) + \gamma_i - \mu_i \leq 0 \quad (4.28)$$

$$\text{(constraints on } y_{ij}) : \beta\lambda_j - \gamma_i \leq 0. \quad (4.29)$$

Notice that for any of the inequalities, the equality holds if the corresponding variable is greater than zero.

To prove that the optimal profits of the two problems are equal, we first show that $\phi^{ori*} \leq \phi^{alt*}$ and then $\phi^{ori*} \geq \phi^{alt*}$. In both directions, the first two cases ($d_i \leq x_i$ for all i and $d_i \geq x_i$ for all i) use exactly the same method as the proof in Theorem 1, hence we omit those details, and only consider the third case to prove $\phi^{ori*} \leq \phi^{alt*}$.

Case 3: There exists some location i such that $z_i^{ori*} < d_i^{ori*}$ and some location j such that $z_j^{ori*} > d_j^{ori*}$. We will prove that the optimal solution for the original optimization problem (4.6) will not fall in this case.

Suppose there exist some location such that $z_i^{ori*} < d_i^{ori*}$, and let $I_1 = \{i : z_i^{ori*} < d_i^{ori*}\}$ and $I_2 = \{i : z_i^{ori*} \geq d_i^{ori*}\}$. We will show that for all $i \in I_2$, $z_i^{ori*} = d_i^{ori*}$. We can consider an aggregated network with locations 1 and 2 representing the combined locations

in I_1 and I_2 , respectively. Knowing $z_1 < d_1$ and $z_2 \geq d_2$, then for any d_2 , $z_2 = d_2$ will constitute a feasible solution for (4.6). Moreover, any z_2 such that $z_2 > d_2$ will increase the cost and thus decrease the profit for (4.6). Hence $z_2 = d_2$ is optimal. Therefore case 3 will not constitute an optimal solution for (4.6).

Next we consider the third case for proving $\phi^{ori*} \geq \phi^{alt*}$.

Case 3: There exists some location i such that $z_i^{alt*} < d_i^{alt*}$ and some location j such that $z_j^{alt*} > d_j^{alt*}$. We will prove that the optimal solution for the alternative optimization problem will not fall in this case.

Suppose there exists some location such that $z_i^{alt*} < d_i^{alt*}$, and let $I_1 = \{i : z_i^{alt*} < d_i^{alt*}\}$ and $I_2 = \{i : z_i^{alt*} \geq d_i^{alt*}\}$. We will show that for all $i \in I_2$, $z_i^{ori*} = d_i^{ori*}$.

As above, we can consider an aggregated network with locations 1 and 2 representing the combined locations in I_1 and I_2 , respectively. We know that $z_1^{alt*} < d_1^{alt*}$ and denote $\Delta_1 = d_1^{alt*} - z_1^{alt*} > 0$. Moreover, suppose that $z_2^{alt*} > d_2^{alt*}$ and $\Delta_2 = d_2^{alt*} - z_2^{alt*} < 0$. We then rewrite the constraints in (4.30) as below:

$$\begin{aligned}
x_1^{alt*} &= \beta[\alpha_{11}\Delta_1 + \alpha_{21}\Delta_2 + (y_{11}^{alt*} + y_{21}^{alt*})] + \delta_1^{alt*} \\
x_2^{alt*} &= \beta[\alpha_{12}\Delta_1 + \alpha_{22}\Delta_2 + (y_{12}^{alt*} + y_{22}^{alt*})] + \delta_2^{alt*} \\
y_{11}^{alt*} + y_{12}^{alt*} &= x_1^{alt*} - \Delta_1 \\
y_{21}^{alt*} + y_{22}^{alt*} &= x_2^{alt*} - \Delta_2 \\
z_1^{alt*} &= \alpha_{11}z_1^{alt*} + \alpha_{21}z_2^{alt*} \\
z_2^{alt*} &= \alpha_{12}z_1^{alt*} + \alpha_{22}z_2^{alt*} \\
p_i, \delta_i, z_i, x_i, y_{ij} &\geq 0 \quad \forall i, j.
\end{aligned} \tag{4.30}$$

First notice that $x_1 > 0$ and $y_{21} + y_{22} > 0$ since $\Delta_1 > 0$ and $\Delta_2 < 0$; then $x_1 + x_2 > 0$ and thus $\delta_1 + \delta_2 = (1 - \beta)(x_1 + x_2) > 0$. Moreover, we will show below that $\delta_1 + y_{21} > 0$.

Suppose that $\delta_1 = y_{21} = 0$. Since $y_{12} \geq 0$, then $y_{11} \leq x_1 - \Delta_1$. Then, from (4.30), $x_1 = \beta[\alpha_{11}\Delta_1 + \alpha_{21}\Delta_2 + y_{11}] \leq \beta[\alpha_{11}\Delta_1 + \alpha_{21}\Delta_2 + x_1 - \Delta_1] = \beta[-\alpha_{12}\Delta_1 + \alpha_{21}\Delta_2 + x_1] <$

$\beta x_1 < x_1$. This is a contradiction and thus $\delta_1 + y_{21} > 0$.

We next show that when $z_1^{alt*} < d_1^{alt*}$ and $z_2^{alt*} > d_2^{alt*}$, we are always able to obtain a solution in the original optimization problem that achieves greater profit. Since we have already proved that $\phi^{ori*} \leq \phi^{alt*}$, then the solution that falls in this case will not be an optimal solution for the alternative optimization problem.

Suppose $s > (1 - \beta)\omega$. We are able to obtain a higher profit by increasing the mass of HVs and decreasing the mass of AVs. In particular, this transformation to case 1 is accomplished by setting $d_i^{ori} = d_i^{alt}$, $r_{ij}^{ori} = 0$, and $y_{ij}^{ori} = y_{ij}^{alt}$ for all i, j ; $z_2^{ori} = d_2^{alt}$, $z_1^{ori} = \frac{\alpha_{21}}{\alpha_{12}} z_2^{ori}$, $x_1^{ori} = x_1^{alt} - \frac{\alpha_{21}}{\alpha_{12}} \Delta_2$, $x_2^{ori} = x_2^{alt} - \Delta_2$, $\delta_1^{ori} = \delta_1^{alt} - (1 - \beta) \frac{\alpha_{21}}{\alpha_{12}} \Delta_2$ and $\delta_2^{ori} = \delta_2^{alt} - (1 - \beta) \Delta_2$. Then, it is obvious that $\mathbf{u}^{ori} = \{p_i^{ori}, \delta_i^{ori}, z_i^{ori}, x_i^{ori}, y_{ij}^{ori}, r_{ij}^{ori}\}_{i,j=1}^2$ satisfies all the constraints of (4.6), and hence it is a feasible solution for (4.6).

This modified solution keeps the demand d_i and thus p_i unchanged, decreases the cost incurred by AVs by $s \cdot (z_1^{alt*} - z_1^{ori} + z_2^{alt*} - z_2^{ori}) = s \cdot (-\frac{\alpha_{21}}{\alpha_{12}} \Delta_2 - \Delta_2) = -s \cdot (1 + \frac{\alpha_{21}}{\alpha_{12}}) \Delta_2 < \omega(1 - \beta)(1 + \frac{\alpha_{21}}{\alpha_{12}}) \Delta_2$, and increases the cost incurred by HVs by $\omega \cdot (\delta_1^{ori} - \delta_1^{alt} + \delta_2^{ori} - \delta_2^{alt}) = \omega(1 - \beta)(1 + \frac{\alpha_{21}}{\alpha_{12}}) \Delta_2$. The net profit increases, hence there always exists a solution for the original optimization problem that achieves a higher profit. Thus, the original solution is not optimal.

Now consider when $s \leq (1 - \beta)\omega$. Suppose $x_2^{alt*} = 0$. Then $y_{21}^{alt*} + y_{22}^{alt*} = -\Delta_2$; since $x_1^{alt*} > 0$ (and thus $\lambda_1 = \gamma_1$), it must hold that $y_{11}^{alt*} = 0$ by KKT conditions. Moreover, we show that $y_{12}^{alt*} = 0$. Suppose $y_{12}^{alt*} > 0$ so that $\beta\lambda_2 - \gamma_1 = 0$. While $\gamma_1 = \lambda_1 \in [\beta\omega, \omega]$ (this is true if there exist $x_i^{alt*} > 0$ for any i), we must have $\lambda_2 = \omega$ and $\gamma_1 = \lambda_1 = \beta\omega$. If $\delta_1^{alt*} = 0$, then $y_{21}^{alt*} > 0$, and thus $\beta\lambda_1 - \gamma_2 = 0$. Hence $\gamma_2 = \beta^2\omega$. However, we require $\beta\lambda_2 - \gamma_2 \leq 0$ while $\beta\lambda_2 - \gamma_2 = \beta\omega - \beta^2\omega > 0$. Therefore $\delta_1^* > 0$. But then we obtain $\lambda_1 = \omega$ by KKT conditions, which contradicts with the fact that $\lambda_1 = \beta\omega$. Therefore, $y_{12}^{alt*} = 0$.

Since $y_{11}^{alt*} = y_{12}^{alt*} = 0$, it holds that $x_1^{alt*} = \Delta_1$. Since $x_2^{alt*} = 0$, we can thus compute $y_{22}^{alt*} = -\alpha_{12}\Delta_1 - \alpha_{22}\Delta_2 - \frac{\delta_2^{alt*}}{\beta} \geq 0$, $y_{21}^{alt*} = \alpha_{12}\Delta_1 - \alpha_{21}\Delta_2 + \frac{\delta_2^{alt*}}{\beta} > 0$. Notice that

$y_{21}^{alt*} > 0$ because $\Delta_1 > 0$ and $\Delta_2 < 0$. Also, $\delta_1^{alt*} + \delta_2^{alt*} = (1 - \beta)(x_1^{alt*} + x_2^{alt*}) = (1 - \beta)\Delta_1$.

Now consider the solution for the original optimization problem by setting $d_i^{ori} = d_i^{alt*}$, $x_i^{ori} = \delta_i^{ori} = y_{ij}^{ori} = 0$ for all i, j . Then a feasible solution of (4.6) is obtained according to $z_1^{ori} = d_1^{alt*}$, $z_2^{ori} = z_2^{alt*}$, $r_{11}^{ori} = r_{12}^{ori} = 0$, $r_{21}^{ori} = \alpha_{12}\Delta_1 - \alpha_{21}\Delta_2$ and $r_{22}^{ori} = -\alpha_{12}\Delta_1 - \alpha_{22}\Delta_2 = y_{22}^{alt*} + \frac{\delta_2^{alt*}}{\beta} > 0$. Then $\mathbf{u}^{ori} = \{p_i^{ori}, \delta_i^{ori}, z_i^{ori}, x_i^{ori}, y_{ij}^{ori}, r_{ij}^{ori}\}_{i,j=1}^2$.

Considering the cost of this modified solution compared to the original solution, the cost increases by $s \cdot (z_1^{ori} + z_2^{ori}) - s \cdot (z_1^{alt*} + z_2^{alt*}) = s \cdot \Delta_1$ and subsequently decreases by $\omega(\delta_1^{alt*} + \delta_2^{alt*}) - \omega(\delta_1^{ori} + \delta_2^{ori}) = (1 - \beta)\omega\Delta_1 > s \cdot \Delta_1$. Since we have already proved that $\phi^{ori*} \leq \phi^{alt*}$, this implies the original solution is not optimal, a contradiction.

Therefore, $x_2^{alt*} > 0$, and by KKT conditions, $y_{22}^{alt*} = y_{11}^{alt*} = y_{12}^{alt*} = \delta_2^{alt*} = 0$ and $y_{21}^{alt*} > 0$. Moreover, $\gamma_1 = \lambda_1 = \omega$ and $\gamma_2 = \lambda_2 = \beta\omega$. Hence $x_2^{alt*} = \beta(\alpha_{12}\Delta_1 + \alpha_{22}\Delta_2) > 0$ and $x_1^{alt*} = \Delta_1$.

By (4.28), we have

$$-s - \beta(\alpha_{11}\lambda_1 + \alpha_{12}\lambda_2) + \gamma_1 + (\alpha_{11}\mu_1 + \alpha_{12}\mu_2) - \mu_1 = 0 \quad (4.31)$$

$$-s - \beta(\alpha_{21}\lambda_1 + \alpha_{22}\lambda_2) + \gamma_2 + (\alpha_{21}\mu_1 + \alpha_{22}\mu_2) - \mu_2 = 0. \quad (4.32)$$

Hence $-s + (1 - \alpha_{11}\beta - \alpha_{12}\beta^2)\omega + \alpha_{12}(\mu_2 - \mu_1) = 0$ and $-s + \alpha_{22}(1 - \beta)\beta\omega + \alpha_{21}(\mu_1 - \mu_2) = 0$. By adding coefficients α_{21} and α_{12} , we obtain $-(\alpha_{12} + \alpha_{21})s + (1 - \alpha_{11}\beta - \alpha_{12}\beta^2)\alpha_{21}\omega + \alpha_{12}\alpha_{22}(1 - \beta)\beta\omega = 0$. By simplification, we then have $s = \frac{(1 + \beta)(\alpha_{21} - \alpha_{12}\beta)}{\alpha_{12} + \alpha_{21}} \cdot \omega$.

At the same time, the equation $x_i = \beta \left[\sum_j \alpha_{ji}(d_j - z_j) + \sum_j y_{ji} \right] + \delta_i$ can be reformulated into $x_i = \beta \left[\sum_j \alpha_{ji}d_j - z_i + \sum_j y_{ji} \right] + \delta_i$, and hence the KKT condition corresponding to the reformulated optimization problem becomes

$$(\text{constraints on } \delta_i) : -\omega + \lambda_i^1 \leq 0 \quad (4.33)$$

$$(\text{constraints on } x_i) : -\lambda_i^1 + \gamma_i^1 \leq 0 \quad (4.34)$$

$$(\text{constraints on } z_i) : -s + \sum_j \alpha_{ij} \mu_j^1 - \beta \lambda_i^1 + \gamma_i^1 - \mu_i^1 \leq 0 \quad (4.35)$$

$$(\text{constraints on } y_{ij}) : \beta \lambda_j^1 - \gamma_i^1 \leq 0. \quad (4.36)$$

By the same process as before, we obtain $\gamma_1^1 = \lambda_1^1 = \omega$, $\gamma_2^1 = \lambda_2^1 = \beta\omega$, and

$$-s + (1 - \beta)\lambda_1^1 + (\alpha_{11}\mu_1^1 + \alpha_{12}\mu_2^1) - \mu_1^1 = 0 \quad (4.37)$$

$$-s + (1 - \beta)\lambda_2^1 + (\alpha_{21}\mu_1^1 + \alpha_{22}\mu_2^1) - \mu_2^1 = 0. \quad (4.38)$$

Therefore, $s = \frac{(1-\beta)(\alpha_{21}+\alpha_{12}\beta)}{\alpha_{12}+\alpha_{21}} \cdot \omega$.

By establishing the equality $s = \frac{(1-\beta)(\alpha_{21}+\alpha_{12}\beta)}{\alpha_{12}+\alpha_{21}} \cdot \omega = \frac{(1+\beta)(\alpha_{21}-\alpha_{12}\beta)}{\alpha_{12}+\alpha_{21}} \cdot \omega$, we require $\alpha_{21} = \alpha_{12}$ and thus $s = \frac{\beta(1-\beta)\omega}{2}$.

Similar to the situation when $x_2^{alt*} = 0$, we obtain a feasible solution

$\mathbf{u}^{ori} = \{p_i^{ori}, \delta_i^{ori}, z_i^{ori}, x_i^{ori}, y_{ij}^{ori}, r_{ij}^{ori}\}_{i,j=1}^2$ for the original optimization problem by setting $d_i^{ori} = d_i^{alt*}$, $x_i^{ori} = \delta_i^{ori} = y_{ij}^{ori} = 0$ for all i, j ; $z_1^{ori} = d_1^{alt*}$, $z_2^{ori} = \alpha_{12}d_1^{alt*} + \alpha_{22}d_2^{alt*}$, $r_{11}^{ori} = r_{12}^{ori} = 0$, $r_{21}^{ori} = \alpha_{12}\Delta_1 - \alpha_{21}\Delta_2$ and $r_{22}^{ori} = 0$. All constraints of (4.6) are satisfied.

The cost incurred by HVs is decreased by $\omega(\delta_1^{alt*} + \delta_2^{alt*}) - \omega(\delta_1^{ori} + \delta_2^{ori}) = (1 - \beta)\omega(x_1^{alt*} + x_2^{alt*}) = \omega(1 - \beta)(\Delta_1 + \beta(\alpha_{12}\Delta_1 + \alpha_{22}\Delta_2))$ and the cost incurred by AVs is increased by $s \cdot (z_1^{ori} + z_2^{ori} - z_1^{alt*} - z_2^{alt*}) = s \cdot (\Delta_1 + \alpha_{12}\Delta_1 + \alpha_{22}\Delta_2) = \frac{\beta(1-\beta)\omega}{2}(\Delta_1 + \alpha_{12}\Delta_1 + \alpha_{22}\Delta_2) = \frac{\omega(1-\beta)}{2}(\beta\Delta_1 + \beta(\alpha_{12}\Delta_1 + \alpha_{22}\Delta_2)) < \omega(1 - \beta)(\Delta_1 + \beta(\alpha_{12}\Delta_1 + \alpha_{22}\Delta_2))$. Hence the cost decreases and the profit is not optimal for the original solution, a contradiction.

Therefore the optimal solution does not fall in case 3. \square

Corollary 1 follows from Theorems 1 and 2.

Corollary 1. *Under Assumptions 1 and 2, the optimal profit for the mixed autonomy deployment under HV (resp., AV) priority assignment is no less than the optimal profit computed from (4.2)/(4.6) with the additional forced HV-only deployment constraint, i.e., the constraint $z_i = 0$ for all i .*

Proof. The mixed autonomy optimization problem can be transformed into (4.14) by setting $\mathbf{z} = \mathbf{0}$ and $\mathbf{r} = \mathbf{0}$. Furthermore, (4.14) is exactly the optimization problem for the system without any AVs. Therefore, by letting $\mathbf{z} = \mathbf{0}$ and $\mathbf{r} = \mathbf{0}$ and the other variables equal to the optimal solution for the optimization problem for the system without AV, we obtain a feasible solution for the mixed autonomy system. Therefore the optimal profit for the mixed autonomy system will be no less than that of the system without autonomous system. \square

Corollary 1 emphasizes that in our model, the AVs will be introduced into the platform only if they increase the optimal profit for the platform.

CHAPTER 5

THE RELATION BETWEEN HV PRIORITY, AV PRIORITY, AND WEIGHTED PRIORITY ASSIGNMENTS AND SPECIAL NETWORKS

Now that we have introduced the alternative optimization problems for maximizing the profits in both HV and AV priority assignments, we next compare the optimal profits for these two priority assignments in this chapter. We are also exploring the weighted priority assignment in detail in Section 5.2 and compare all three priority assignments. We then study the problem on a special class of networks through the case study.

5.1 The Relation between HV Priority and AV Priority Assignments

The main result of this section is Theorem 3 which shows that the two priority assignments actually lead to the same optimal profits.

Before presenting the main theorem, we first introduce some preliminary lemmas that are interesting in their own right. In the remainder of the chapter, we denote an optimal solution with superscript $*$, e.g., x_i^* .

The next lemma establishes that under HV priority assignment, if some location has departing AVs without passengers, then that location also does not have incoming AVs without passengers.

Lemma 3. *Consider the alternative optimization problem (4.7) for HV priority assignment under Assumptions 1 and 2. Suppose there exist some location i such that both $x_i^* > 0$ and $z_i^* > 0$. Then $d_i^* \geq x_i^*$ for all i . Moreover, for any i_0 , if there exists some location j such that $r_{i_0,j}^* > 0$, then $r_{j,i_0}^* = 0$ for all j .*

Proof. Step 1: We first show that $d_i^* \geq x_i^*$ for all i . This part follows similar to the corresponding part in Lemma 5 which will be proved later.

Step 2: We complete the proof by contradiction. Assume i_0, j_0 are locations that $r_{i_0 j_0}^* > 0$. By (4.13) we'll have $\mu_{j_0} - \gamma_{i_0} = 0$.

Since $\sum_{j=1}^n r_{ij} = z_i - (d_i - x_i)$ by (4.7), then $z_{i_0} = d_{i_0} - x_{i_0} + \sum_{j=1}^n r_{i_0 j} = d_{i_0} - x_{i_0} + \sum_{j=1, j \neq j_0}^n r_{i_0 j} + r_{i_0 j_0}$. Since $r_{i_0 j} \geq 0$ for all j , $r_{i_0 j_0}^* > 0$ and from step 1 we have $d_{i_0}^* \geq x_{i_0}^*$, then $z_{i_0}^* > 0$. And (4.12) gives that $\gamma_{i_0} - \mu_{i_0} = s$. Combining the two results yields that $\mu_{j_0} - \mu_{i_0} = s$.

Suppose there exists a location j that $r_{j i_0}^* > 0$, then $\mu_{i_0} - \gamma_j = 0$. Hence $\mu_{i_0} = \gamma_j$ and $\mu_{j_0} - \gamma_j = \mu_{j_0} - \mu_{i_0} = s > 0$ which contradicts (4.13). Therefore, for any location j , $r_{j i_0}^* = 0$. \square

Next, we show that if it is optimal for the platform to use both HVs and AVs at some location, then every vehicle in the network will be assigned to a ride.

Lemma 4. *For optimization problem (4.7) under Assumption 1 and 2, if there exists a location i_0 such that $x_{i_0}^* > 0$ and $z_{i_0}^* > 0$, then $r_{ij}^* = 0$ for all i, j .*

Proof. Since there exists a location i such that $x_i^* > 0$ and $z_i^* > 0$, from Lemma 3 we know that $d_i^* \geq x_i^*$ for all i .

Suppose there exist a location i_0 such that $r_{i_0 j}^* > 0$. First, we partition the n locations into two groups: $I_1 = \{i : i \neq i_0\}$, $I_2 = \{i_0\}$. Then we aggregate those into a 2-location system with locations 1 and 2 such that $\alpha_{22} = 0$, $\alpha_{21} = 1$.

Step 1: We show $r_{11}^* = r_{12}^* = r_{22}^* = 0$, $r_{21}^* > 0$.

Notice that since $r_{i_0 j}^* > 0$, then $r_{21}^* > 0$. Hence by Lemma 3, $r_{12}^* = r_{22}^* = 0$. Moreover, since $z_1 = \alpha_{11}(d_1 - x_1) + \alpha_{21}(d_2 - x_2) + r_{11} + r_{21}$ and $d_i^* \geq x_i^*$ for $i = 1, 2$, then $z_1^* > 0$ and $\gamma_1 - \mu_1 = s$ by (4.12). From (4.13), $\mu_1 - \gamma_1 = -s < 0$ implies that $r_{11}^* = 0$.

Step 2: We show that $\delta_2^* = 0$ and $\delta_1^* > 0$ using KKT conditions.

To reason about the 2-group problem, first rewrite the optimization constraints below by combining with the conditions $\alpha_{22} = 0$, $\alpha_{21} = 1$.

$$\begin{aligned}
x_1 &= \beta(\alpha_{11}x_1 + x_2) + \delta_1 \\
x_2 &= \beta\alpha_{12}x_1 + \delta_2 \\
z_1 &= \alpha_{11}(d_1 - x_1) + (d_2 - x_2) + r_{11} + r_{21} \\
z_2 &= \alpha_{12}(d_1 - x_1) + r_{12} + r_{22} \\
r_{11} + r_{12} &= z_1 + x_1 - d_1 \\
r_{21} + r_{22} &= z_2 + x_2 - d_2 \\
\delta_i, x_i, z_i, r_{ij} &\geq 0 \quad \forall i, j.
\end{aligned} \tag{5.1}$$

Clearly, as $x_i^* > 0$ for $i = 1$ or 2 , then $x_1^* > 0$ and $x_2^* > 0$ since $\alpha_{12} > 0$ when the actual ride-sharing network has no less than two locations and is strongly connected. Similarly, since there exists a location i such that $x_i^* > 0$ and $z_i^* > 0$, then $d_i^* - x_i^* > 0$ and $d_1^* - x_1^* > 0$ or $d_2^* - x_2^* > 0$. Hence $z_1^* > 0$ and $z_2 = r_{21} + r_{22} + d_2 - x_2$ implies that $z_2^* > 0$.

We can therefore conclude the corresponding KKT conditions:

$$\begin{aligned}
r_{21} > 0 &\Rightarrow \mu_1 - \gamma_2 = 0 \\
z_1 > 0 &\Rightarrow \gamma_1 - \mu_1 = s \\
z_2 > 0 &\Rightarrow \gamma_2 - \mu_2 = s \\
x_1 > 0 &\Rightarrow \alpha_{11}(\beta\lambda_1 - \mu_1) + \alpha_{12}(\beta\lambda_2 - \mu_2) - \lambda_1 + \gamma_1 = 0 \\
x_2 > 0 &\Rightarrow (\beta\lambda_1 - \mu_1) - \lambda_2 + \gamma_2 = 0.
\end{aligned}$$

Notice that the first 3 equations above imply that $\gamma_1 - \mu_2 = 2s + \mu_1 - \gamma_2 = 2s$. By recombination of the equations, we derive

$$\begin{aligned}
\beta(\alpha_{11}\lambda_1 + \alpha_{12}\lambda_2) - \lambda_1 + \alpha_{11}(\gamma_1 - \mu_1) + \alpha_{12}(\gamma_1 - \mu_2) &= 0 \\
\beta(\alpha_{11}\lambda_1 + \alpha_{12}\lambda_2) - \lambda_1 + \alpha_{11} \cdot s + \alpha_{12} \cdot 2s &= 0
\end{aligned}$$

$$\beta(\alpha_{11}\lambda_1 + \alpha_{12}\lambda_2) - \lambda_1 + (1 + \alpha_{12})s = 0 \quad (5.2)$$

and

$$\begin{aligned} \beta\lambda_1 - \lambda_2 - (\mu_1 - \gamma_2) &= 0 \\ \beta\lambda_1 - \lambda_2 &= 0. \end{aligned} \quad (5.3)$$

Since $\delta_1 + \delta_2 = (1 - \beta)(x_1 + x_2)$ and now $x_1^* + x_2^* > 0$, then $\delta_1^* + \delta_2^* > 0$. Suppose $\delta_2^* > 0$, then by (4.10), $\lambda_2 = \omega$, and hence $\lambda_1 = \frac{\lambda_2}{\beta} = \frac{\omega}{\beta} > \omega$, which contradicts the KKT condition. Hence $\delta_2^* = 0$ and thus $\delta_1^* > 0$.

Step 3: Determine the range of s that satisfies the given conditions.

Since $\delta_1^* > 0$ then $\lambda_1 = \omega$ and thus $\lambda_2 = \beta\lambda_1 = \beta\omega$. Substituting those into (5.2) yields that

$$s = -\frac{\alpha_{11}\beta\omega + \alpha_{12}\beta^2\omega - \omega}{1 + \alpha_{12}} \quad (5.4)$$

$$= \frac{(1 - \beta)(1 + \alpha_{12}\beta)}{1 + \alpha_{12}} \cdot \omega. \quad (5.5)$$

Therefore, $s = \frac{(1 - \beta)(1 + \alpha_{12}\beta)}{1 + \alpha_{12}} \cdot \omega$ is the only value that is feasible.

Step 4: We show that it is possible for the platform to realize the same profit using only AVs ($x_i = 0, z_i > 0$ for all i). Now that

$$x_1^* = \beta(\alpha_{11}x_1 + x_2) + \delta_1^*$$

$$x_2^* = \beta\alpha_{12}x_1^*$$

$$z_1^* = \alpha_{11}(d_1^* - x_1^*) + (d_2^* - x_2^*) + r_{21}^*$$

$$z_2^* = \alpha_{12}(d_1^* - x_1^*)$$

$$0 = z_1^* + x_1^* - d_1^*$$

$$r_{21}^* = z_2^* + x_2^* - d_2^*,$$

suppose $d_1^* \leq d_2^*$. Since $x_2^* = \beta\alpha_{12}x_1^* < x_1^*$ and $z_1^* = d_1^* - x_1^* = \alpha_{11}(d_1^* - x_1^*) + (d_2^* - x_2^*) + r_{21}^*$, then $d_1^* - x_1^* \geq d_2^* - x_2^* = d_2^* - \beta\alpha_{12}x_1^* > d_2^* - x_1^*$. This implies that $d_1^* > d_2^*$, which contradicts the assumption. Therefore $d_1^* > d_2^*$.

Moreover, since $z_2^* = r_{21}^* + d_2^* - x_2^* > d_2^* - x_2^*$, then $z_2^* = \alpha_{12}(d_1^* - x_1^*) > d_2^* - x_2^* \Rightarrow \alpha_{12}d_1^* > d_2^* - x_2^* + \alpha_{12}x_1^* = d_2^* + (1 - \beta)\alpha_{12}x_1^*$. We can also reformulate that $\delta_1^* = (1 - \beta)(x_1^* + x_2^*) = (1 - \beta)(1 + \beta\alpha_{12})x_1^*$ and $z_1^* + z_2^* = (1 + \alpha_{12})(d_1^* - x_1^*)$.

It is straightforward to verify that

$$\{z_1 = d_1^*, z_2 = \alpha_{12}d_1^*, x_1 = x_2 = \delta_1 = \delta_2 = 0, r_{ij} = r_{ij}^*\}$$

is also a feasible solution for the problem.

We now consider the modified costs under this alternative feasible solution. The increase of the cost is

$$\begin{aligned} & (z_1 + z_2) \cdot s - (z_1^* + z_2^*) \cdot s \\ &= [d_1^* + \alpha_{12}d_1^* - (1 + \alpha_{12})(d_1^* - x_1^*)] \cdot s \\ &= (1 + \alpha_{12})x_1^* \cdot \frac{(1 - \beta)(1 + \alpha_{12}\beta)}{1 + \alpha_{12}} \cdot \omega \\ &= (1 - \beta)(1 + \alpha_{12}\beta)x_1^* \cdot \omega, \end{aligned} \tag{5.6}$$

and the cost is subsequently decreased by $(\delta_1^* + \delta_2^*) \cdot \omega - (\delta_1 + \delta_2) \cdot \omega = \delta_1^*\omega = (1 - \beta)(1 + \beta\alpha_{12})x_1^*\omega$. Thus the total cost does not change while the prices and demands are also unchanged. Therefore the profit is not changed.

Hence, it is possible to achieve the same profit using only AVs.

Step 5: We next complete the proof by contradiction. Denote the solutions above as $\mathbf{u}_{x>0, z>0}^{d*}$ for the mixed case of both HVs and AVs and by $\mathbf{u}_{x=0, z>0}^{d*}$ for the case with only AVs. Denote the optimal profit obtained in these two scenarios as π_m and π_{AV} , respectively, and from Step 4 we know $\pi_m = \pi_{AV}$. Consider the alternative form of AV priority

assignment optimization problem (4.23).

Suppose with the same ω, s, β and α_{ij} for all i, j , the optimal solution for AV priority assignment falls into the mixed autonomy case with $\mathbf{u}_{x>0, z>0}^{a*}$. Notice that since $y_{ij, x>0, z>0}^{a*} = 0$ for all i, j , then the solution $\mathbf{u}_{x>0, z>0}^{a*}$ is feasible for HV priority assignment by substituting r_{ij}^d with $y_{ij, x>0, z>0}^{a*}$, and moreover the profit will be exactly the same. Additionally, the solution $\mathbf{u}_{x>0, z>0}^{d*}$ is also feasible for AV priority assignment by substituting y_{ij}^a with $r_{ij, x>0, z>0}^{d*}$ with the profit $\hat{\pi}_m$. However, since there exist i, j such that $r_{ij, x>0, z>0}^{d*} > 0$, and from Lemma 6 (as we will prove later) we know that this is not optimal for AV priority assignment, it follows that $\pi_m < \hat{\pi}_m$. Hence π_m is not an optimal profit for HV priority assignment, which gives the contradiction.

Suppose the optimal solution $\mathbf{u}_{x=0, z>0}^{a*}$ for AV priority assignment falls into the pure-AV case, *i.e.*, $x_i = 0$ for all i . Again, $\mathbf{u}_{x>0, z>0}^{d*}$ is feasible for AV priority assignment. Moreover, under the case with only AVs, the two optimization problems are exactly the same by substituting r_{ij}^d with y_{ij}^a . Therefore $\mathbf{u}_{x=0, z>0}^{d*}$ and $\mathbf{u}_{x=0, z>0}^{a*}$ yield the same profit, denoted as π_{AV} . However, since $\mathbf{u}_{x>0, z>0}^{d*}$ cannot be optimal for AV priority assignment as shown above, $\pi_m < \pi_{AV}$ which contradicts the above result that $\pi_m = \pi_{AV}$.

Finally, if the optimal solution $\mathbf{u}_{x>0, z=0}^{a*}$ for AV priority assignment falls into the pure-HV case, *i.e.*, $z_i = 0$ for all i , then the optimal profit gained from this solution, denoted as π_{HV} , will be greater than π_m (since π_m is not the optimal profit). Moreover, since the solution will also be feasible for HV priority optimization problem, then π_{HV} is also attainable for HV priority assignment. This contradicts the result that π_m is the optimal profit for HV priority assignment.

Therefore, $\mathbf{u}_{x>0, z>0}^{d*}$ cannot be the optimal solution for (4.7) and our assumption that there exist i, j such that $r_{ij}^* > 0$ is false. Hence, in the situation under consideration, $r_{ij}^* = 0$ for all i, j . □

Similar properties exist under AV priority assignment, as summarized in the following lemmas.

Lemma 5. Consider the alternative optimization problem (4.23) for AV priority assignment under Assumptions 1 and 2. Suppose there exist some location i such that both $x_i^* > 0$ and $z_i^* > 0$. Then $d_i^* \geq z_i^*$ for all i . Moreover, for any i_0 , if there exist some location j such that $y_{i_0j} > 0$, then $y_{ji_0} = 0$ for all j .

Proof. Step 1: We show that $d_i^* \geq z_i^*$ for all i . Assume location $i_{>0} \in \{1, \dots, n\}$ is such that $x_{i_{>0}}^* > 0$ and $z_{i_{>0}}^* > 0$. From the construction of the model, we know that the platform uses AVs only to meet the excess demand, hence $d_{i_{>0}}^* > z_{i_{>0}}^*$. Therefore, from Theorem 2, we know that for the optimal problem (4.23), $d_{i_{>0}}^* > z_{i_{>0}}^*$. Moreover, in the proof of the theorem, we have also shown that the mixed case where there exist some locations such that $d_i > z_i$ and some locations such that $d_i < z_i$ will not be the optimal solution. Thus, it follows that $d_i^* \geq z_i^*$ for all i under this circumstance.

Step 2: We complete the proof by contradiction. Assume i_0, j_0 are locations such that $y_{i_0j_0}^* > 0$. By (4.29), we have $\beta\lambda_{j_0} - \gamma_{i_0} = 0$. Since $\sum_{j=1}^n y_{ij} = x_i - (d_i - z_i)$ by (4.23), then $x_{i_0} = d_{i_0} - z_{i_0} + \sum_{j=1}^n y_{i_0j} = d_{i_0} - z_{i_0} + \sum_{j=1, j \neq j_0}^n y_{i_0j} + y_{i_0j_0}$. Since $y_{i_0j} \geq 0$ for all j , $y_{i_0j_0}^* > 0$, and from Step 1 above we have $d_{i_0}^* \geq z_{i_0}^*$, then $x_{i_0}^* > 0$. Therefore, (4.27) gives that $\gamma_{i_0} = \lambda_{i_0}$.

Notice also (4.26), (4.27) and (4.29) together indicate that $\lambda_i \in [\beta\omega, \omega]$ and $\gamma_i \in [\beta\omega, \omega]$ for all i when there exists at least one location i' such that $\delta_{i'} > 0$ (or $x_{i'} > 0$). Therefore, $\lambda_{j_0} = \omega, \gamma_{i_0} = \beta\omega$ is the only possible choice. Thus $\gamma_{i_0} = \lambda_{i_0} = \beta\omega$.

Suppose there exists a location j such that $y_{ji_0}^* > 0$. Then $\beta\lambda_{i_0} - \gamma_j = 0$. This indicates that $\lambda_{i_0} = \omega$ and $\gamma_j = \beta\omega$, which contradicts the result $\lambda_{i_0} = \beta\omega$ obtained above. Therefore, for any location j , $y_{ji_0}^* = 0$. □

Lemma 6. For optimization problem (4.23) under Assumptions 1 and 2, if there exists a location i_0 such that $x_{i_0}^* > 0$ and $z_{i_0}^* > 0$, then $y_{ij}^* = 0$ for all i, j .

Proof. We partition the locations into two groups: $I_1 = \{i : y_{ij}^* = 0 \quad \forall j\}$ and $I_2 = \{i : \exists j \quad y_{ij}^* > 0\}$. By aggregating these groups into two locations, we henceforth

regard this as a two-location problem indexed by 1 and 2. By Lemma 5, we know that $y_{22}^* = 0$, $y_{21}^* > 0$ and $d_i^* \geq z_i^*$ for $i = 1, 2$.

As $z_1^* > 0$ or $z_2^* > 0$ and $z_i = \sum_{j=1}^2 \alpha_{ji} z_j$, since the network is strongly connected, then $z_1^* > 0$ and $z_2^* > 0$. Knowing $d_2^* \geq z_2^*$, $x_2 = (d_2 - z_2) + y_{21} + y_{22}$ and $y_{21}^* > 0$ implies that $x_2^* > 0$; similarly, $x_1^* = \beta[\alpha_{11}(d_1^* - z_1^*) + \alpha_{21}(d_2^* - z_2^*) + y_{11}^* + y_{21}^*] > 0$. Moreover, (4.27) implies that $\gamma_1 = \lambda_1$ and $\gamma_2 = \lambda_2$.

Also, since $\delta_1 + \delta_2 = (1 - \beta)(x_1 + x_2)$, then there exists $i \in \{1, 2\}$ such that $\delta_i^* > 0$ and hence $\lambda_i = \omega$ by (4.26). Combining with (4.27) and (4.29), we know that $\lambda_i \in [\beta\omega, \omega]$ and $\gamma_i \in [\beta\omega, \omega]$. Since $y_{21}^* > 0$, then $\beta\lambda_1 - \gamma_2 = 0$ which indicates that $\lambda_1 = \omega = \gamma_1$ and $\gamma_2 = \beta\omega = \lambda_2$. We further conclude that $\delta_2^* = 0$ and thus $\delta_1^* > 0$, $y_{11}^* = y_{12}^* = 0$. The KKT variables are the same as the proof of Theorem 2 in the situation where $s \leq (1 - \beta)\omega$ and $x_2^{alt*} > 0$. Without loss of generality, we therefore conclude that

$$s = \frac{(1 + \beta)(\alpha_{21} - \beta\alpha_{12})}{\alpha_{21} + \alpha_{12}} \cdot \omega \quad (5.7)$$

$$= \frac{(1 - \beta)(\alpha_{21} + \beta\alpha_{12})}{\alpha_{21} + \alpha_{12}} \cdot \omega \quad (5.8)$$

$$= \frac{1}{2}(1 - \beta)\beta\omega \quad (5.9)$$

and $\alpha_{11} = \alpha_{22}$.

Now consider the possible optimal solutions

$$x_1^* = \beta[\alpha_{11}(d_1^* - z_1^*) + \alpha_{21}(d_2^* - z_2^*) + y_{21}^*] + \delta_1^*$$

$$x_2^* = \beta[\alpha_{12}(d_1^* - z_1^*) + \alpha_{22}(d_2^* - z_2^*)]$$

$$x_1^* = d_1^* - z_1^*$$

$$y_{21}^* = z_2^* + x_2^* - d_2^*$$

$$z_1^* = z_2^* .$$

Suppose $d_1^* \geq d_2^*$. Then $d_1^* - z_1^* \geq d_2^* - z_2^*$ and $x_2^* = \beta[\alpha_{12}(d_1^* - z_1^*) + \alpha_{22}(d_2^* - z_2^*)] =$

$\beta[\alpha_{21}(d_1^* - z_1^*) + \alpha_{22}(d_2^* - z_2^*)] \geq \beta(d_2^* - z_2^*)$. Now let $z_1 = z_2 = d_2^*$ (increase both by $d_2^* - z_2^*$). Then decrease x_2 by $\beta(d_2^* - z_2^*)$ and x_1 by $(d_2^* - z_2^*)$, thus we decrease δ_1 by $(1 - \beta)(1 + \beta)(d_2^* - z_2^*) < (1 - \beta)(x_1^* + x_2^*)$. Hence we increase the cost by $2(d_2^* - z_2^*) \cdot s = 2(d_2^* - z_2^*) \cdot \frac{1}{2}(1 - \beta)\beta\omega = (1 - \beta)\beta(d_2^* - z_2^*) \cdot \omega$ and subsequently decrease the cost by $(1 - \beta)(1 + \beta)(d_2^* - z_2^*) \cdot \omega > (1 - \beta)\beta(d_2^* - z_2^*) \cdot \omega$ (by 4.6, $x_2 > 0$ indicates that $d_2^* - z_2^* > 0$). Hence the total profit increases, which contradicts the fact that this is a profit-maximizing optimum.

Suppose $d_1^* < d_2^*$. With the same process as before, we increase z_1 and z_2 by $(d_1^* - z_1^*)$, decrease x_2 by $\beta(d_1^* - z_1^*)$ and x_1 by $d_1^* - z_1^*$, that is, we decrease δ_1 by $(1 - \beta)(1 + \beta)(d_1^* - z_1^*) < (1 - \beta)(x_1^* + x_2^*)$. Hence we increase the cost by $2(d_1^* - z_1^*) \cdot s = (1 - \beta)\beta(d_1^* - z_1^*) \cdot \omega$ and subsequently decrease the cost by $(1 - \beta)(1 + \beta)(d_1^* - z_1^*) \cdot \omega > (1 - \beta)\beta(d_1^* - z_1^*) \cdot \omega$, with the net effect of increasing the profit, which again is a contradiction.

Therefore $y_{ij} > 0$ is not an optimal solution in this situation. \square

The main result of this section below uses the above lemmas to establish that a profit-maximizing platform is able to realize the same optimal profits under either the HV priority or AV priority assignments.

Theorem 3. *Under Assumptions 1 and 2, for any choice of ω, s, β and \mathbf{A} ,*

$\mathbf{u}^* = \{p_i^*, \delta_i^*, z_i^*, x_i^*, y_{ij}^*, r_{ij}^*\}_{i,j=1}^n$ *is an optimal solution of the optimization problem for HV priority assignment (4.2) if and only if it is an optimal solution of the optimization problem for AV priority assignment (4.6), and therefore the optimal profits of the two optimization problems are the same.*

Proof. First notice that in each priority assignment, an optimal solution falls into one of three cases: HV-only (i.e., $z_i = 0$ for all i), mixed autonomy (i.e., there exists some i, j such that $x_i > 0$ and $z_j > 0$), and AV-only (i.e., $x_i = 0$ for all i). In the case of HV-only or AV-only, it is straightforward to observe that when a solution is feasible for either HV priority assignment or AV priority assignment, it will also be feasible for the other

AV assignment (consider the original optimization problems here). This is also true for the mixed case, since from Lemmas 4 and 6, we know that $r_{ij} = y_{ij} = 0$ in both priority assignments. Therefore, the solutions for the two optimization problems are convertible: given β, ω, s and A , if a solution is optimal for one priority assignment, it is also optimal for the other priority assignment.

Since the objective functions of the two optimization problems (4.2) and (4.6) are the same, then the result above implies that they have the same optimal profits. \square

We can then derive a threshold on the cost of AVs above which the platform does not find it optimal to deploy any AVs.

Proposition 1. *Under Assumptions 1 and 2, if $k > 1$, then, under any priority assignment, it is optimal for the platform to use an HV-only deployment, i.e., there is no benefit to introducing AVs into the ride-sharing network.*

Proof. Firstly we will develop another necessary condition.

Since we have proved that the two priority assignments achieve the same optimal solutions, then the following are equivalent:

- the inequality/equality in (4.10)/(4.11)/(4.12)/(4.13) holds
- the inequality/equality in (4.26)/(4.27)/(4.28)/(4.29) holds
- the inequality/equality in (4.33)/(4.34)/(4.35)/(4.36) holds.

Moreover, consider the corresponding KKT condition for prices p_i , and denote the variables in (4.10)–(4.13) using superscript d . The KKT conditions require

$$\begin{aligned} \frac{\partial(p_i d_i)}{\partial p_i}(p_i^*) + \frac{\partial d_i}{\partial p_i}(p_i^*) \left(\sum_j \alpha_{ij} \mu_j^d - \gamma_i^d \right) &= \frac{\partial(p_i d_i)}{\partial p_i}(p_i^*) + \frac{\partial d_i}{\partial p_i}(p_i^*) \left(\sum_j \alpha_{ij} \beta \lambda_j - \gamma_i \right) \\ &= \frac{\partial(p_i d_i)}{\partial p_i}(p_i^*) + \frac{\partial d_i}{\partial p_i}(p_i^*) \left(\sum_j \alpha_{ij} \beta \lambda_j^1 - \gamma_i^1 \right) = 0. \end{aligned}$$

The last equality holds because $p_i^* > 0$ for all i obviously. Hence $\sum_j \alpha_{ij} \mu_j^d - \gamma_i^d = \sum_j \alpha_{ij} \beta \lambda_j - \gamma_i = \sum_j \alpha_{ij} \beta \lambda_j^1 - \gamma_i^1$.

Therefore, satisfying the relation of (4.10)–(4.13) with (4.33)–(4.36) requires $-\omega + \lambda_i^d = -\omega + \lambda_i^1$, $\sum_j \alpha_{ij} (\beta \lambda_j^d - \mu_j^d) - \lambda_i^d + \gamma_i^d = -\lambda_i^1 + \gamma_i^1$, $-s + \gamma_i^d - \mu_i^d = -s - \beta \lambda_i^1 + \sum_j \alpha_{ij} \mu_j^1 + \gamma_i^1 - \mu_i^1$ and $\mu_j^d - \gamma_i^d = \beta \lambda_j^1 - \gamma_i^1$ for all i, j .

These requirements yield that $\lambda_i^d = \lambda_i^1$ and $\gamma_i^1 = \gamma_i^d + c$ where $c = \beta \lambda_j^d - \mu_j^d$ for any j . In addition,

$$\begin{aligned} -s + \gamma_i^d - \mu_i^d &= -s - \beta \lambda_i^1 + \sum_j \alpha_{ij} \mu_j^1 + \gamma_i^1 - \mu_i^1 \\ \gamma_i^d - \mu_i^d &= -\beta \lambda_i^1 + \sum_j \alpha_{ij} \mu_j^1 + \gamma_i^1 - \mu_i^1 \\ \gamma_i^d - \mu_i^d &= -\beta \lambda_i^1 + \sum_j \alpha_{ij} \mu_j^1 + (\gamma_i^d + \beta \lambda_i^d - \mu_i^d) - \mu_i^1 \\ 0 &= \sum_j \alpha_{ij} \mu_j^1 - \mu_i^1 \end{aligned}$$

and applying this to (4.35) gives a new necessary condition that must be satisfied for any optimal solution for the optimization problem (4.23):

$$-s - \beta \lambda_i^1 + \gamma_i^1 \leq 0 \quad (5.10)$$

where the equality holds when $z_i > 0$.

With the condition described in (5.10) held, we can construct the threshold for the cost of AV above which the mixed-autonomy won't be beneficial for the platform.

Assume the optimal profit of the mixed autonomy deployment is strictly greater than that of the HV-only deployment. Then there exists a location i such that $z_i > 0$. Hence by (5.15), $-s - \beta \lambda_i^1 + \gamma_i^1 = 0$. Moreover, from (4.29), (4.33) and (4.36), $\beta \lambda_j^1 \leq \gamma_i^1 \leq \lambda_i^1 \leq \omega$ for any j . Therefore, $s = \gamma_i^1 - \beta \lambda_i^1 \leq \lambda_i^1 - \beta \lambda_i^1 \leq (1 - \beta)\omega$. Hence $k = \frac{s}{\omega} \leq \frac{(1-\beta)\omega}{\omega} = 1 - \beta$. \square

5.2 Weighted Priority Assignment

Besides assigning the rides to one type of vehicle—HVs or AVs—first, and then using the other type to satisfy any remaining demand, it is also reasonable to consider that any vehicle in the platform can be chosen randomly with equal probability. Therefore, in this section, we introduce the *weighted priority assignment* in which the platform assigns the rides at each location to HVs and AVs at that location with the same probability, *i.e.*, in proportion to the relative fraction of HVs and AVs to the total number of vehicles.

5.2.1 Equilibrium Definition for Weighted Priority Assignment

As described above, in weighted priority assignment, HVs and AVs are assigned to riders with equal possibility: $Prob\{\text{rider assigned to HV}\} = Prob\{\text{rider assigned to AV}\} = \min\left\{\frac{\theta_i(1-F(p_i))}{x_i+z_i}, 1\right\}$ for all i . The resulting equilibrium constraints for the model are:

$$x_i = \beta \left[\sum_j \alpha_{ji} \min \left\{ 1, \frac{\theta_j(1-F(p_j))}{x_j+z_j} \right\} \cdot x_j + \sum_j y_{ji} \right] + \delta_i \quad (5.11)$$

$$\sum_j y_{ij} = \max \left\{ 1 - \frac{\theta_i(1-F(p_i))}{x_i+z_i}, 0 \right\} \cdot x_i \quad (5.12)$$

$$z_i = \sum_j \alpha_{ji} \min \left\{ 1, \frac{\theta_j(1-F(p_j))}{x_j+z_j} \right\} \cdot z_j + \sum_j r_{ji} \quad (5.13)$$

$$\sum_j r_{ij} = \max \left\{ 0, 1 - \frac{\theta_i(1-F(p_i))}{x_i+z_i} \right\} \cdot z_i. \quad (5.14)$$

The expected lifetime earnings V_i for a driver at location i takes the form

$$\begin{aligned} V_i = & \min \left\{ \frac{\theta_i(1-F(p_i))}{x_i+z_i}, 1 \right\} \left(c_i + \sum_{k=1}^n \alpha_{ik} \beta V_k \right) \\ & + \left(1 - \min \left\{ \frac{\theta_i(1-F(p_i))}{x_i+z_i}, 1 \right\} \right) \beta \max_j V_j. \end{aligned} \quad (5.15)$$

As before, the platform chooses compensation such that $V_i = \omega$.

Definition 3. For some prices and compensations $\{p_i, c_i\}_{i=1}^n$, the collection

$\{\delta_i, x_i, y_{ij}, z_i, r_{ij}\}_{i,j=1}^n$ is an equilibrium under $\{p_i, c_i\}_{i=1}^n$ for weighted priority assignment if (5.11)–(5.14) is satisfied and V_i as defined in (5.15) satisfies $V_i = \omega$ for all $i = 1, \dots, n$ such that $\delta_i + \sum_{j=1}^n y_{ji} > 0$.

To further study weighted priority assignment, we now introduce the following assumption which ensures that the platform can make some profit by offering rides at an appropriate price.

Assumption 3. *The parameters β, ω and s are such that $(1 - \beta)\omega < \bar{p}$ or $s < \bar{p}$.*

5.2.2 Profit-Maximization Optimization Problem for Weighted Priority Assignment

We now establish the following profit-maximization problem for weighted priority assignment:

$$\begin{aligned}
& \max_{\{p_i, c_i\}_{i=1}^n} \sum_{i=1}^n \left[\min \{x_i + z_i, \theta_i(1 - F(p_i))\} \cdot p_i \right. \\
& \quad \left. - \min \left\{ x_i, \theta_i(1 - F(p_i)) \frac{x_i}{x_i + z_i} \right\} \cdot c_i - z_i \cdot s \right] \\
& \text{s.t. } \{\delta_i, x_i, y_{ij}, z_i, r_{ij}\}_{i,j=1}^n \text{ is an equilibrium under} \\
& \quad \{p_i, c_i\}_{i=1}^n \text{ for weighted priority assignment.} \tag{5.16}
\end{aligned}$$

As in Section 4.1, we establish an equivalent optimization problem

$$\begin{aligned}
& \max_{\{p_i, \delta_i, x_i, y_{ij}, z_i, r_{ij}\}} \sum_{i=1}^n p_i \theta_i (1 - F(p_i)) - \omega \sum_{i=1}^n \delta_i - s \sum_{i=1}^n z_i \\
& \text{s.t. } d_i = \theta_i (1 - F(p_i)) \\
& \quad x_i = \beta \left[\sum_j \alpha_{ji} d_j \frac{x_j}{x_j + z_j} + \sum_j y_{ji} \right] + \delta_i \\
& \quad \sum_{j=1}^n y_{ij} = x_i - d_i \frac{x_i}{x_i + z_i} \\
& \quad z_i = \sum_{j=1}^n \alpha_{ji} d_j \frac{z_j}{x_j + z_j} + \sum_{j=1}^n r_{ji}
\end{aligned}$$

$$\sum_{j=1}^n r_{ij} = z_i - d_i \frac{z_i}{x_i + z_i}$$

$$p_i, \delta_i, z_i, x_i, y_{ij}, r_{ij} \geq 0 \quad \forall i, j, \quad (5.17)$$

followed by a lemma showing the equivalence.

Lemma 7. *Assume weighted priority assignment and consider the optimization problems (5.16) and (5.17). Under Assumptions 1, 2 and 3, an optimal solution to (5.17) provides an optimal solution to (5.16). In particular, any optimal solution $\{p_i^*, \delta_i^*, x_i^*, y_{ij}^*, z_i^*, r_{ij}^*\}$ for (5.17) is such that $d_i^* > 0$ for all i , i.e., some riders are served at all locations, and there exist compensations $\{c_i^*\}_{i=1}^n$ such that $\{\delta_i^*, x_i^*, y_{ij}^*, z_i^*, r_{ij}^*\}_{i,j=1}^n$ constitutes an equilibrium under $\{p_i^*, c_i^*\}_{i=1}^n$ for weighted priority assignment. Moreover, $\{p_i^*, c_i^*\}_{i=1}^n$ is optimal for (5.16).*

Proof. The proof for the first two points are similar to that of the HV and AV priority assignments. Obviously, $d_i^* \leq x_i^* + z_i^*$ for (5.16). Hence we can turn the equilibrium constraints into the constraints in (5.17). By setting the compensation $c_i = \omega(1 - \beta) \cdot \frac{x_i + z_i}{d_i}$ for all i , we obtain the equivalent optimization (5.17).

Consider the optimization problem (5.17) of weighted priority assignment and compare it with that of HV priority assignment (4.2). By observation, if for any optimal solution of HV priority assignment, we can obtain that $\min\{x_i^*, d_i^*\} = d_i^* \cdot \frac{x_i^*}{x_i^* + z_i^*}$ and $\max\{d_i^* - x_i^*, 0\} = d_i^* \cdot \frac{z_i^*}{x_i^* + z_i^*}$ (notice that $\max\{x_i^* - d_i^*, 0\} = x_i^* - \min\{x_i^*, d_i^*\}$), then it follows that any optimal solution for HV priority assignment will be feasible for weighted priority assignment.

By Assumption 3, we have that $(1 - \beta)\omega < \bar{p}$ or $s < \bar{p}$. Hence Lemma 1 establishes that $x_i^* + z_i^* \geq d_i^* > 0$ for all i . We then consider the optimal solution in the three cases.

If it falls in the HV-only case, i.e., $x_i^* > 0, z_i^* = 0$ for all i , then this implies $d_i^* \leq x_i^*$

for all i . Therefore, we have

$$\begin{cases} d_i^* \cdot \frac{x_i^*}{x_i^* + z_i^*} = d_i^* & = \min \{x_i^*, d_i^*\} \\ d_i^* \cdot \frac{z_i^*}{x_i^* + z_i^*} = 0 & = \max \{d_i^* - x_i^*, 0\}. \end{cases}$$

Similarly, if the optimal solution is in the AV-only case, i.e., $x_i^* = 0, z_i^* > 0$, then $d_i^* \geq x_i^*$ for all i . Hence

$$\begin{cases} d_i^* \cdot \frac{x_i^*}{x_i^* + z_i^*} = 0 & = \min \{x_i^*, d_i^*\} \\ d_i^* \cdot \frac{z_i^*}{x_i^* + z_i^*} = d_i^* & = \max \{d_i^* - x_i^*, 0\}. \end{cases}$$

Lastly, when the optimal solution is in the mixed autonomy case, i.e., $x_i^* > 0, z_i^* > 0$ for some i , then $d_i^* \geq x_i^*$ for all i . Also, Proposition 4 implies that $y_{ij}^* = r_{ij}^* = 0$ here for all i, j , and then $d_i^* = x_i^* + z_i^*$ for all i . Therefore, we observe that

$$\begin{cases} d_i^* \cdot \frac{x_i^*}{x_i^* + z_i^*} = x_i^* & = \min \{x_i^*, d_i^*\} \\ d_i^* \cdot \frac{z_i^*}{x_i^* + z_i^*} = d_i^* - x_i^* & = \max \{d_i^* - x_i^*, 0\}. \end{cases}$$

Thus, the optimal solutions for the HV and AV priority assignments are always feasible for weighted priority assignment. Hence, under Assumption 3, any optimal solution $\{p_i^*, \delta_i^*, x_i^*, y_{ij}^*, z_i^*, r_{ij}^*\}$ for (5.17) is such that $d_i^* > 0$ for all i . \square

The following theorem establishes that weighted priority assignment obtains the same optimal profits as the HV and AV priority assignments, which were already shown to obtain the same optimal profits in Theorem 3.

Theorem 4. *Under Assumptions 1, 2 and 3, for any choice of ω, s, β and \mathbf{A} , a feasible solution \mathbf{u} for (4.2) or (4.6) is optimal for (4.2) or (4.6) if and only if \mathbf{u} is an optimal solution for (5.17).*

Proof. By recombining the constraints in (5.17), we can obtain another optimization problem given by

$$\begin{aligned}
& \max_{\{p_i, \delta_i, x_i, y_{ij}, z_i, r_{ij}\}} \sum_{i=1}^n p_i \theta_i (1 - F(p_i)) - \omega \sum_{i=1}^n \delta_i - s \sum_{i=1}^n z_i \\
& \text{s.t. } d_i = \theta_i (1 - F(p_i)) \\
& x_i + \beta z_i = \beta \left[\sum_j \alpha_{ji} d_j + \sum_j y_{ji} + \sum_j r_{ji} \right] + \delta_i \\
& \sum_{j=1}^n y_{ij} + \sum_{j=1}^n r_{ij} = x_i + z_i - d_i \\
& z_i - \sum_{j=1}^n \alpha_{ji} z_j = \sum_{j=1}^n r_{ji} - \sum_{j=1}^n \alpha_{ji} \sum_{k=1}^n r_{jk} \\
& \delta_i, z_i, x_i, y_{ij}, r_{ij} \geq 0 \quad \forall i, j.
\end{aligned} \tag{5.18}$$

By construction, any optimal solution for (5.17) will be feasible for (5.18) and thus the optimal profit of (5.18) will be no less than that of (5.17).

As we have already proved in Lemma 7, the optimal solution of the optimization problem in priority assignment is always a feasible solution for (5.17).

Consider the optimization problem (5.18), and rewrite it by considering d_i as the variable instead of p_i . Notice that since $d_i = \theta_i (1 - F(p_i))$ is monotonically decreasing, we are able to write p_i as a function d_i because the inverse mapping exists. Moreover, we can relax the constraint $p_i \geq 0$ for all i since d_i is always positive and thus a negative price cannot be optimal.

Below lists the KKT conditions related to (5.18) while regarding d_i as a variable instead of p_i :

$$(\text{constraints on } d_i) : \frac{\partial p_i d_i}{\partial d_i} + \beta \sum_j \alpha_{ij} \lambda_j - \gamma_i = 0 \tag{5.19}$$

$$\text{(constraints on } \delta_i) : -\omega + \lambda_i \leq 0 \quad (5.20)$$

$$\text{(constraints on } x_i) : -\lambda_i + \gamma_i \leq 0 \quad (5.21)$$

$$\text{(constraints on } z_i) : -s - \sum_j \alpha_{ij}(\beta\lambda_j - \mu_j) + \gamma_i - \mu_i \leq 0 \quad (5.22)$$

$$\text{(constraints on } y_{ij}) : \beta\lambda_j - \gamma_i \leq 0 \quad (5.23)$$

$$\text{(constraints on } r_{ij}) : \beta\lambda_j - \gamma_i - \mu_j + \sum_j \alpha_{ij}\mu_j \leq 0. \quad (5.24)$$

By Assumption 2 and 3, (5.18) is a convex optimization problem with affine constraints, and thus the KKT conditions are not only necessary, but also sufficient for optimality. Hence in order to show a solution to be optimal for (5.18), it is enough to show that it satisfies all the KKT conditions (5.19)–(5.24):

Given the optimal solution and the KKT variables λ_i^1 and γ_i^1 resolved from the optimal solution of AV priority assignment with the conditions (4.33)–(4.36), let $\mu_i = \mu_j$ for all i, j . Then the conditions (5.19)–(5.24) and the constraints for weighted priority assignment can all be satisfied. Therefore, any optimal solution for AV priority assignment is also optimal (and feasible) for (5.18).

At the same time, since the optimal profits for (5.18) are higher than or equal to that of (5.17), and since any optimal solution for AV priority assignment is feasible for (5.17), then we can conclude that any optimal solution for AV priority assignment is also optimal (and feasible) for (5.17). \square

Theorems 3 and 4 show that, even though the three priority assignments prescribe different models for incorporating AVs into a ride-sharing platform, the resulting profits at an optimal equilibrium are the same in all three cases under Assumptions 1, 2 and 3. This is because no location will have both AVs and HVs present at an optimal equilibrium. Intuitively, on the one hand, the platform is able set compensation for drivers and to deploy AVs as desired, so that there is considerable freedom in dictating system operation. On the other hand, locations are coupled through the rider demand pattern and cannot be managed

independently by the platform, highlighting the surprising nature of this result.

5.3 Closed-Form Characterization for Star-to-Complete Networks

In this section, we consider the family of *star-to-complete networks* introduced in [20]. For this large class of networks, we derive closed form expressions for the thresholds of relative cost between HVs and AVs for which the platform finds it optimal to use an HV-only deployment, AV-only deployment, or a mixed autonomy deployment.

Definition 4. *The class of demand patterns $(\mathbf{A}^\xi, \mathbf{1})$ with $n \geq 3$, $\xi \in [0, 1]$, and*

$$\mathbf{A}^\xi = \begin{bmatrix} 0 & \frac{1}{n-1} & \frac{1}{n-1} & \cdots & \frac{1}{n-1} \\ c_1 & 0 & c_2 & \cdots & c_2 \\ c_1 & c_2 & 0 & \cdots & c_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_1 & c_2 & \cdots & c_2 & 0 \end{bmatrix}, \quad (5.25)$$

$$c_1 = \frac{\xi}{n-1} + (1-\xi), \quad c_2 = \frac{\xi}{n-1} \quad (5.26)$$

is the family of *star-to-complete networks*. It is a *star network* when $\xi = 0$ for which we write $\mathbf{A}^S = \mathbf{A}^0$ and a *complete network* when $\xi = 1$ for which we write $\mathbf{A}^C = \mathbf{A}^1$. Therefore the general adjacency matrix of a *star-to-complete network* can be written as $\mathbf{A}^\xi = \xi \mathbf{A}^C + (1-\xi) \mathbf{A}^S$.

In addition, we make the following assumption throughout this section.

Assumption 4. *All locations have the same mass of potential riders, which we normalize to one, i.e., $\theta = 1$. Also, the riders' willingness to pay is uniformly distributed in $[0, 1]$ so that $F(p) = p$ for $p \in [0, 1]$.*

Consider fixed outside option earnings ω , and recall the parameter k determining the cost of operating AVs for the same lifetime of an HV relative to ω . In this section, we

confirm the intuition that, for large k , i.e. high relative cost of AVs, the profit maximizing strategy for the platform is an HV-only deployment, and for small k , i.e. low relative cost of AVs, the profit maximizing strategy for the platform is an AV-only deployment. We also show that in some cases, but not all, for some values of k , the platform finds it optimal to use both HVs and AVs at equilibrium, *i.e.*, a true mixed autonomy deployment.

Recall that Proposition 1 provides a sufficient condition for when a platform will not find it optimal to use AVs. In the next Theorem, we sharpen this result for the class of star-to-complete networks and fully characterize the regions in which the profit-maximizing platform will deploy an HV-only deployment, an AV-only deployment, and a truly mixed autonomous network.

Theorem 5. *Consider a star-to-complete network under Assumption 4. Define*

$$\begin{aligned}
k_1 &= \frac{1 + \beta c_1}{c_1 + 1}, \\
k_2 &= \begin{cases} 1 & \text{if } \xi \in \left[\frac{\beta(n-1)-1}{\beta(n-2)}, 1\right] \\ \frac{c_1(1+\beta)+(n-1)\beta^2 c_1^3+1}{(c_1+1)((n-1)\beta^2 c_1^2+1)} & \text{if } \xi \in \left[\beta_{lim}, \frac{\beta(n-1)-1}{\beta(n-2)}\right) \\ \frac{1+\beta c_1}{c_1+1} & \text{if } \xi \in [0, \beta_{lim}), \end{cases} \\
k_3 &= \frac{(n-1)c_1 - 1}{(1-\beta)(n-1)(1+c_1)c_1}, \\
k_4 &= \frac{(1+\beta)c_1 + (n-1)\beta c_1^3 + 1}{(c_1+1)(\beta(n-1)c_1^2+1)},
\end{aligned}$$

where

$$\beta_{lim} = \max \left\{ \frac{n-1}{2(1-\beta)\beta(n-2)} \left[\beta(1-2\beta) + \sqrt{\frac{\beta^2(n-1) + 4\beta - 4}{n-1}} \right], 0 \right\}.$$

Suppose $k_3 \geq k_1$, equivalently, $\beta c_1(n-1)(1-c_1+\beta c_1) \geq 1$. When $k \in [0, k_1]$, it is always optimal for the platform to deploy an AV-only deployment, *i.e.*, optimal profits are obtained with $x_i = 0$ for all i . If $k_1 < k_2$, then: when $k \in (k_1, k_2)$, it is optimal for

the platform to deploy a mixed autonomous network, i.e., optimal profits are obtained with $x_i > 0$ and $z_j > 0$ for some i, j ; when $k \geq k_2$, it is optimal for the platform to deploy an HV-only deployment, i.e., optimal profits are obtained with $z_i = 0$ for all i . If $k_1 \geq k_2$, then: when $k > k_1$, it is optimal for the platform to deploy an HV-only deployment.

Now suppose $k_3 < k_1$, equivalently, $\beta c_1(n-1)(1-c_1+\beta c_1) \geq 1$. When $k \in [0, k_4]$, it is optimal for the platform to deploy an AV-only deployment; when $k \in (k_4, k_2)$, it is optimal to deploy a mixed autonomy deployment; when $k \geq k_2$, it is optimal to deploy an HV-only deployment.

The proof of Theorem 5 can be found in Appendix A.

To demonstrate Theorem 5 and provide intuition for the fundamental theoretical results of this chapter, we study a star-to-complete network with $n = 3$, $\xi = 0.2$. We consider two cases: $\beta = 0.8$ and $\beta = 0.95$, and we compute optimal equilibria and profits using the optimization problems formulated above. In both cases, applying Theorem 5, we can verify that $k_3 \geq k_1$. For the first case with $\beta = 0.8$, we obtain $k_1 = 0.9053$ and $k_2 = 0.9181$ so that $k_1 < k_2$. Figure 5.1 (Left) confirms that for $k \leq k_1$, it is optimal for the platform to deploy only AVs, for $k_1 < k < k_2$, it is optimal for the platform to use both AVs and HVs, and for $k \geq k_2$, it is optimal for the platform to use only HVs. In contrast, when $\beta = 0.95$ so that the expected lifetime of HVs in the network is longer, then $k_1 = 0.9763$ and $k_1 \geq k_2$. Figure 5.1 (Right) confirms that for $k \leq k_1$, the platform finds it optimal to deploy only AVs, and for $k > k_1$, the platform finds it optimal to use only HVs; there is no regime in which the platform finds it optimal to use both AVs and HVs. The plots in Figure 5.1 are generated by solving the optimization problem (4.7) in MATLAB using CVX, a package for specifying and solving convex programs[77, 78]. Theorem 5 guarantees that the basic qualitative results demonstrated here apply to arbitrarily large star-to-complete networks.

It is interesting to note from the above thresholds that even if AVs are cheaper than HVs, when the price difference is small, the platform may still choose to deploy only HVs or to

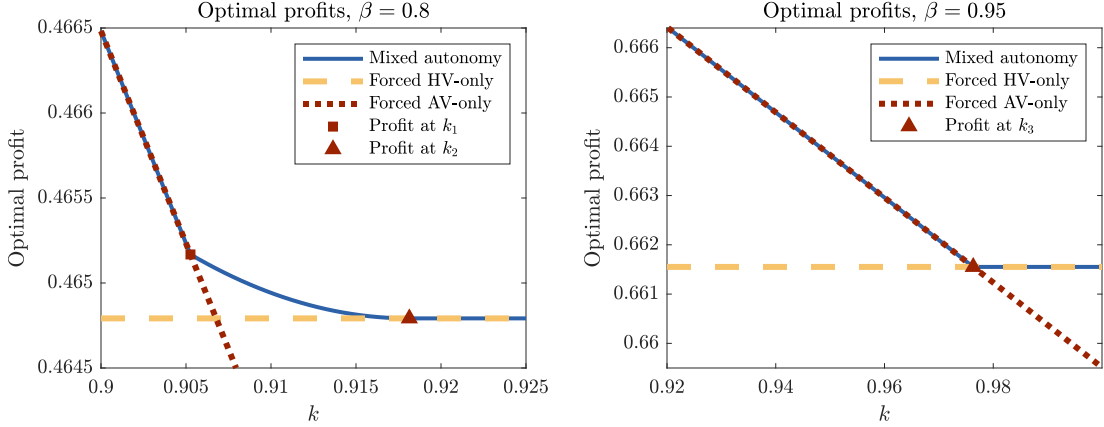


Figure 5.1: Optimal profits for a star-to-complete network with $n = 3$, $\xi = 0.2$ under a mixed autonomy deployment, a forced HV-only deployment, and a forced AV-only deployment. (Left) When $\beta = 0.8$, it is optimal for the platform to use only AVs when k , the ratio of the cost of AVs to HVs, satisfies $k \leq k_1 = 0.9053$, only HVs when $k \geq k_2 = 0.9181$, and a mix of AVs and HVs when $k_1 < k < k_2$. (Right) When $\beta = 0.95$, it is optimal for the platform to use only AVs when $k \leq k_1 = 0.9763$ and only HVs when $k > k_1$, and it is never optimal for the platform to use a mix of HVs and AVs.

deploy a mix of AVs and HVs. An explanation for this observation is as follows. Recall that with probability $1 - \beta$, a driver leaves the network and does not seek to be matched to a new rider after finishing a ride and thus essentially provides one-way service. In contrast, AVs are assumed to remain in the network and must be recirculated to a new location. When the demand is uneven so that some destinations are more popular than others, the platform can exploit this one-way service to obtain a higher profit with HVs, even if AVs are less expensive on a per ride basis.

5.4 Concluding Remarks

We proposed three models for ride-sharing systems with mixed autonomy under different ride-assigning schemes and showed that under equilibrium conditions, the optimal profits can be computed efficiently by converting the original problems into alternative convex programs. In addition, we proved that the optimal profits of the three models are the same.

We found that the optimal profits for the ride-sharing platform with AVs in the fleet will be the same as that of the human-only network when k is large, *i.e.*, the cost for operating

an AV is relatively high compared to the outside option earnings for drivers' lifetime. In particular, in Proposition 1, we showed that if the cost of operating an AV exceeds the expected compensation to a driver in the system, the platform will find it optimal to not use AVs, an intuitive result.

The case study illustrates that the platform may not necessarily find it optimal to use AVs even when the cost of operating an AV is less than the expected compensation to a driver in the system. Moreover, there are some situations when it is optimal to have both drivers and AVs in the platform. For star-to-complete networks, we quantified the conditions for which the mixed autonomy deployment allows for higher profits than a forced AV-only or forced HV-only deployment.

The model proposed and studied here includes a several simplifying assumptions that can be relaxed in future work. For example, destinations are often not equidistant and ride costs might then depend on destination. Nonetheless, these simplifying assumptions are important for illuminating fundamental properties of ride-sharing in a mixed autonomy setting.

CHAPTER 6

CAPACITY-CONSTRAINED URBAN AIR MOBILITY SCHEDULING

We study a UAM network model that accounts for uncertainty in travel time and limited landing capacity in this chapter. We consider the problem of scheduling flight departures to ensure that all flights arrive no later than designated deadlines and that there is always an available landing spot at the destination and intermediate nodes upon arrival, which can be regarded as safety constraints. We are taking both static and dynamic scheduling into account in this chapter. The model we are exploring can be suitable for both dynamic and static scheduling, and we gain some theoretical insights in both cases in Section 6.2. We also develop an algorithm for static scheduling that minimizes the summation of time difference between the arrival of the flights and their deadlines, while each flight has to arrive on time at the destination with no two flight can take the same parking space at the same time in Section 6.3, followed by its illustration through a case study.

We let \mathbb{N} denote the natural numbers without zero while \mathbb{N}_0 the natural numbers with zero, and \mathbb{R} the reals while \mathbb{R}_+ the positive reals. For a finite set \mathcal{A} , we let $\mathbb{R}^{\mathcal{A}}$, denote the set of vectors indexed by the elements in \mathcal{A} .

6.1 Problem Formulation

We model an urban air mobility (UAM) network with an acyclic¹ directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where \mathcal{V} is the set of nodes and \mathcal{E} is the set of links for the network. Nodes are physical landing sites for the UAVs, sometimes called *vertistops* or *vertiports*. Links are corridors of airspace connecting nodes. Each node $v \in \mathcal{V}$ has capacity $C_v \in \mathbb{N}_0$, that is, there are C_v *parking spots* at node v where each parking spot allows at most one UAV to

¹In practice, a directed graph can often be naturally decomposed into two or more acyclic graphs, e.g., flights inbound versus outbound of a city center.

stay at any time. We denote the vector of capacities $C = \{C_v\}_{v \in \mathcal{V}}$.

We define $\tau : \mathcal{E} \rightarrow \mathcal{V}$ and $\sigma : \mathcal{E} \rightarrow \mathcal{V}$ so that for all $e = (v_1, v_2) \in \mathcal{E}$ where $v_1, v_2 \in \mathcal{V}$, $\tau(e) = v_1$ is the tail of edge e and $\sigma(e) = v_2$ is the head of edge e . Let $S \subseteq \mathcal{V}$ (resp., $T \subseteq \mathcal{V}$) be the set of nodes that are not the head (resp., tail) of any edge, $S = \{v \in \mathcal{V} \mid \sigma(e) \neq v \forall e \in \mathcal{E}\}$ and $T = \{v \in \mathcal{V} \mid \tau(e) \neq v \forall e \in \mathcal{E}\}$. We assume $S \cap T = \emptyset$.

A *route* R is a sequence of connected links. Denote the number of links in route R by k_R and enumerate the links in the route $1^R, 2^R, \dots, k_R^R$ and the nodes in the route $0^R, 1^R, \dots, k_R^R$. To avoid cumbersome notation, we use ℓ^R to denote both a link and its head node along a route, i.e., $\ell^R = \sigma(\ell^R)$ for all $\ell \in \{1, \dots, k_R\}$; the intended meaning will always be clear from context. Thus the route links and nodes are enumerated so that $0^R = \tau(1^R)$ is the origin node, k_R^R is the destination node, and $\sigma(\ell^R) = \tau((\ell + 1)^R)$ for all $\ell \in \{1, \dots, k_R\}$ ensures the sequence is connected. Further, when the route R is clear from context, we drop the superscript- R notation. We denote the set of nodes that R travels through as $V(R)$. We assume that, due to operational reasons, the UAVs are only allowed to travel along a set of routes \mathcal{R} .

Since, in reality, the travel time depends on external factors such as weather conditions, we assume that the travel time for each link is not exact, but rather bounded by a time interval. For each link $i \in \mathcal{E}$, let \bar{x}_i and \underline{x}_i with $\bar{x}_i \geq \underline{x}_i > 0$ denote the maximum travel time and minimum travel time, respectively, for the link, and let $\underline{x} \in \mathbb{R}_+^{\mathcal{E}}$ and $\bar{x} \in \mathbb{R}_+^{\mathcal{E}}$ be the corresponding aggregated vectors. Once a UAV has landed at any node, it is assumed to block a landing spot for a fixed ground service time $w \in \mathbb{R}_+$. For ease of notation, we assume the ground service time is uniform at all nodes, but this assumption is straightforward to relax.

Definition 5 (UAM Network). *A UAM network \mathcal{N} is a tuple $\mathcal{N} = (\mathcal{G}, C, \mathcal{R}, \underline{x}, \bar{x}, w)$ where $\mathcal{G}, C, \mathcal{R}, \underline{x}, \bar{x}, w$ are the network graph, node capacities, routes, and minimum and maximum link travel times as defined above.*

To model the demand of UAV flights in a UAM network $\mathcal{N} = (\mathcal{G}, C, \mathcal{R}, \underline{x}, \bar{x}, w)$, we

assume that every flight is associated to a route $R \in \mathcal{R}$ and stops at intermediate nodes along the route. Therefore, a *demand* is a pair (R, f) where $R \in \mathcal{R}$ and $f \in \mathbb{R}_+$ is the latest time the UAV must arrive (i.e., land) at the destination $k_R^R \in \mathcal{V}$, i.e., its deadline.

A demand profile for a UAM network is a time-varying set $\mathcal{D}(t) = \{(R_j, f_j)\}_{j \in \mathcal{J}(t)}$ where $\mathcal{J}(t)$ is a time-varying index set. Then $\mathcal{J}(t)$, and, hence, $\mathcal{D}(t)$, is monotonically increasing and is assumed finite for all finite t , but may become infinite in the limit. To coordinate the operation of the UAVs, a centralized scheduler aims to schedule all available demands $\mathcal{D}(t)$ at each time t . For example, new demands may become available in batches at certain times, requiring action from the scheduler, and other demands will reach their final destination, requiring no further consideration from the scheduler, although without loss of generality completed demands remain within the set $\mathcal{D}(t)$. Let $\mathcal{D} := \cup_t \mathcal{D}(t) = \lim_{t \rightarrow \infty} \mathcal{D}(t)$ denote all demands that will ever be considered by the scheduler, and likewise let $\mathcal{J} := \cup_t \mathcal{J}(t) = \lim_{t \rightarrow \infty} \mathcal{J}(t)$.

The scheduler associates to each demand a journey consisting of an assigned departure time and the set of realized arrival times along edges in the route. Therefore, a journey is updated over time as the UAV arrives at intermediate nodes. Formally, for each $j \in \mathcal{J}(t)$, a *journey* for demand $(R_j, f_j) \in \mathcal{D}(t)$ is a tuple $(A_j(t), \delta_j)$, where $\delta_j \in \mathbb{R}$ is the departure time and is a decision variable of the scheduler, and $A_j(t)$ is the realized set of arrival times of the UAV at nodes along R_j that have occurred by time t . Here, δ_j is a decision variable of the scheduler and A_j is a result of realized travel times. In particular, when $R_j = \{\ell\}_{\ell=1}^{k_{R_j}}$, $A_j(t) = \{A_{j,\ell}(t)\}_{\ell=1}^{k_{R_j}}$ where $A_{j,\ell}(t)$ is the arrival time at node ℓ if the arrival has occurred by time t , and $A_{j,\ell}(t) = \infty$ if demand j has not arrived at node ℓ by time t . Therefore, the departure time of the UAV from node ℓ is $\delta_{j,\ell} = A_{j,\ell} + w$ where we drop the explicit dependence on time since the equation is understood to hold only for times t such that $A_{j,\ell}(t) < \infty$ and we let $\delta_{j,0} = \delta_j$.

For safety reasons, it is assumed that a UAV must be able to land immediately upon arrival at any node along its route. Whenever a UAV arrives at a node along the route of

a journey, the journey will be updated accordingly, so that the uncertainty of the rest of the journey decreases. In particular, suppose the UAV departs some node $\ell_1 - 1$ along its route. The latest arrival time at some other node $\ell_2 \geq \ell_1$ along the route is denoted a_{ℓ_1, ℓ_2}^j and given by

$$a_{\ell_1, \ell_2}^j = \delta_{j, \ell_1 - 1} + \sum_{\ell=\ell_1}^{\ell_2} \bar{x}_\ell + (\ell_2 - \ell_1)w, \quad (6.1)$$

i.e., a_{ℓ_1, ℓ_2}^j is the departure time from node $\ell_1 - 1$ plus the upper bound of the time interval it takes to travel through the links $\{\ell\}_{\ell=\ell_1}^{\ell_2}$ with the time spent at each node. Note that it is time-dependent because $\delta_{j, \ell_1 - 1}$ is updated over time. Further, the time interval that the UAV will potentially block a landing spot at node ℓ_2 when departing from ℓ_1 is given by

$$\mathcal{M}_{\ell_1, \ell_2}^j = \left[\delta_{j, \ell_1 - 1} + \sum_{\ell=\ell_1}^{\ell_2} \underline{x}_\ell + (\ell_2 - \ell_1)w, a_{\ell_1, \ell_2}^j + w \right]. \quad (6.2)$$

We also define $m_{\ell_1, \ell_2}^{R_j}$ as the length of the time interval above, so that

$$m_{\ell_1, \ell_2}^{R_j} = \sum_{\ell=\ell_1}^{\ell_2} \bar{x}_\ell - \sum_{\ell=\ell_1}^{\ell_2} \underline{x}_\ell + w. \quad (6.3)$$

Note that the superscript of m is R_j because it depends only on the route of demand j and not its particular arrival and departure times. We let $\mathcal{M}_{v_1, v_2}^j = \mathcal{M}_{\ell_1, \ell_2}^j$ if v_1, v_2 are two nodes along the route $R_j \in \mathcal{R}$, $v_1 = \ell_1^{R_j}$, $v_2 = \ell_2^{R_j}$ and $\ell_1 \leq \ell_2$. For all $\ell \in \{1, \dots, k_R\}$, we let $a_\ell^j = a_{1, \ell}^j$, $\mathcal{M}_\ell^j = \mathcal{M}_{1, \ell}^j$ and $m_\ell^{R_j} = m_{1, \ell}^{R_j}$. In the same manner, we let $a_v^j = a_\ell^j$, $\mathcal{M}_v^j = \mathcal{M}_\ell^j$ and $m_v^{R_j} = m_\ell^{R_j}$ if $v = \ell^{R_j}$.

The task, then, is to assign to each demand a journey such that capacity constraints and arrival times are satisfied.

Definition 6 (Schedules and Journeys). *Given a set of demands $\mathcal{D}(t) = \{(R_j, f_j)\}_{j \in \mathcal{J}(t)}$ at time t , a corresponding set of departure times $\mathcal{S}(t) = \{\delta_j\}_{j \in \hat{\mathcal{J}}(t)}$ with $\hat{\mathcal{J}}(t) \subseteq \mathcal{J}(t)$ and each $\delta_j \in \mathbb{R}$ is a **schedule** for $\mathcal{D}(t)$ if:*

1. The latest arrival time is not later than the deadline for all demands $j \in \hat{\mathcal{J}}(t)$, that is, $\alpha_{k_{R_j}}^j \leq f_j$ where we recall that k_{R_j} is the final destination node for demand j along route R_j and $\alpha_{k_{R_j}}^j$ is computed as in (6.1) which depends on any realized arrival/departure times at intermediate nodes;

2. the number of vehicles at a node never exceeds capacity, i.e., for all $v \in \mathcal{V}$ and all $t' \geq 0$,

$$\sum_{j:v \in V(R_j)} \mathbf{1}(t'; \mathcal{M}_v^j(t)) \leq C_v$$

where the notation $\mathbf{1}(\cdot; \cdot)$ is an indicator such that $\mathbf{1}(t; [a, b]) = 1$ if $t \in [a, b]$ and $\mathbf{1}(t; [a, b]) = 0$ otherwise; and

3. In any finite time interval, only a finite number of UAVs depart.

A schedule is a **complete schedule** if $\hat{\mathcal{J}}(t) = \mathcal{J}(t)$; otherwise it is a **partial schedule**. For any scheduled demand $j \in \hat{\mathcal{J}}(t_0)$, the pair $(A_j(t), \delta_j)$ is the **journey** of demand j where $A_j(t)$ is the set of realized arrival times as defined above.

In the above definition, items 1 and 2 are the main features of a schedule, while item 3 is a mild technical requirement that is always satisfied in practice. In this paper, we consider the following problem.

Scheduling Problem. Given a set of demands $\mathcal{D}(t)$ and a sequence of scheduling times $\mathcal{T} = \{t_0, t_1, t_2, \dots\}$ that may be finite or infinite satisfying $t_i < t_{i+1}$ for all i , at each scheduling time t_i , for the set of available demands $\mathcal{D}(t_i)$, determine $\hat{\mathcal{J}}(t_i) \subseteq \mathcal{J}(t_i)$ the subset of flights to be scheduled and compute a schedule $\mathcal{S}(t_i) = \{\delta_j\}_{j \in \hat{\mathcal{J}}(t_i)}$ satisfying the properties

1. $\hat{\mathcal{J}}(t_i) \supseteq \hat{\mathcal{J}}(t_{i-1})$ and $\mathcal{S}(t_i) = \mathcal{S}(t_{i-1}) \cup \{\delta_j\}_{j \in \hat{\mathcal{J}}(t_i) \setminus \hat{\mathcal{J}}(t_{i-1})}$, and
2. For all $j \in \hat{\mathcal{J}}(t_i) \setminus \hat{\mathcal{J}}(t_{i-1})$, $\delta_j \geq t_i$

with $\mathcal{J}(t_{-1}) := \emptyset$ and $\mathcal{S}(t_{-1}) := \emptyset$.

Property 1 of the above problem statement implies that once a demand is scheduled for departure, its departure time does not change at future scheduling times. Property 2 captures a causality requirement and implies that newly scheduled demands cannot have departure times in the past.

If \mathcal{D} is time-invariant and only a single scheduling time $t_0 = 0$ is considered, the scheduling problem is called *static*. Otherwise, it is called a *dynamic* scheduling problem.

Dynamic scheduling allows the schedule to be updated dynamically at certain scheduling times. Schedule updates may be needed to accommodate new demands or because it may not be possible to schedule all known demands at some time, i.e., only a partial schedule can be obtained. Then, at future scheduling times after some flights arrive at intermediate nodes and reduce uncertainty in the remaining travel time, these previously unscheduled demands can be scheduled. Note that scheduling times need not be fixed in advance and can, for example, be triggered when a flight lands at an intermediate node.

We demonstrate how, even with a fixed and known set of demands, dynamic scheduling may be beneficial.

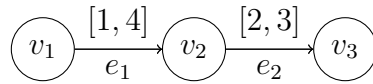


Figure 6.1: A two-link network that is used to demonstrate the benefits of dynamic scheduling in Example 1.

Example 1. Consider a network with 3 nodes and 2 links as shown in Fig. 6.1. This network has a single route $R = \{e_1, e_2\}$. Suppose the travel time interval of link e_1 is $[1, 4]$ and of link e_2 is $[2, 3]$, the wait time at nodes v_2 and v_3 is $w = 1$, and the capacity of nodes v_2 and v_3 are $C_2 = C_3 = 1$. Consider the time-invariant set of demands $\mathcal{D} = \{(R, 8), (R, 11)\}$, i.e., there are two demands $\mathcal{J} = \{1, 2\}$ with deadlines $f_1 = 8$ and $f_2 = 11$ that each take the only route R . A partial schedule at time $t = 0$ is $\mathcal{S}(0) = \{\delta_1\}$ with $\delta_1 = 0$, the departure time for demand $j = 1$. It is not possible to include a departure time for demand $j = 2$ in schedule $\mathcal{S}(0)$ because of the uncertain travel time. Suppose, however, that demand $j = 1$

arrives at node v_2 at time $t = 2$. Then a new complete schedule at time $t = 2$ is given by $\mathcal{S}(2) = \{\delta_1, \delta_2\}$ with $\delta_1 = 0$ and now $\delta_2 = 3$.

Even when the set of demands is time-invariant and finite, the static scheduling problem is computationally challenging because of the combinatorial decision in determining arrival order at nodes. In static scheduling, we propose an integer program which exactly solves this static scheduling problem, but this approach is not suitable for large networks or when schedules are updated dynamically.

In the dynamic scheduling case, we propose an algorithm based on branch-and-bound heuristics that considers at any time only the $K_0 \in \mathbb{N}$ most urgent demands that must be scheduled, where K_0 is a design parameter. Journeys are scheduled for these demands which determine the time intervals that the UAVs are potentially blocking any landing spot. To avoid conflict, the blocking intervals cannot overlap, and hence these intervals are used for scheduling remaining demands. As the indeterminacy of traveling aggregates along the route and reduces whenever an intermediate node is reached, the schedule is dynamically updated over time to improve the efficiency of the UAM network. The detailed scheduling and updating method will be explained in Chapter 7. In the next section, we first derive fundamental theoretical limits for the scheduling problem.

6.2 Theoretical Results for Scheduling

In this section, we present fundamental theoretical results on when complete schedules are available for a set of demands. We first consider the dynamic scheduling case and obtain necessary conditions for feasibility in the case of dynamic scheduling with finite number of demands at each scheduling time. Then, we turn to the static scheduling problem and focus on the class of star networks representative of the case when several links lead to a central, main destination. In this case, we obtain sufficient and necessary conditions for scheduling an infinite set of demands representing, e.g., regular periodic flights. For more general network topologies, by considering a star sub-network consisting of a node and its

neighbors, this result immediately leads to necessary conditions for feasibility.

6.2.1 Necessary Condition for Dynamic Scheduling of Additional Demands

Consider the general network setting defined in Section 6.1. Given a set of demands $\mathcal{D}(t)$ and a sequence of increasing scheduling times $t_0 < t_1 < t_2 < \dots$, we consider the set of available demands $\mathcal{D}(t_i) = \{(R_j, f_j)\}_{j \in \mathcal{J}(t_i)}$ and a schedule $\mathcal{S}(t_i) = \{\delta_j\}_{j \in \hat{\mathcal{J}}(t_i)}$, for any $i = 0, 1, \dots$, where $\hat{\mathcal{J}}(t_i) \subseteq \mathcal{J}(t_i)$ and let $\mathcal{J}(t_{-1}) = \emptyset$.

We introduce a cumulative departure function in time interval $[t_1, t_2]$ as $\Delta_v^R(t_1, t_2, \mathcal{J}) : \mathbb{R}^2 \rightarrow \mathbb{N}$ for all $v \in \mathcal{V} \setminus T$, where we recall the set of tail nodes T , so that $\Delta_v^R(t_1, t_2, \mathcal{J})$ is the number of UAVs departing from node v in the time interval $[t_1, t_2]$, while traveling through route $R \in \mathcal{R}$, i.e.,

$$\Delta_v^R(t_1, t_2, \mathcal{J}) = |\{j \in \mathcal{J} \mid R_j = R \text{ and } \delta_{j,v} \in [t_1, t_2]\}|. \quad (6.4)$$

We then introduce a cumulative arrival function, $\Gamma_v^R(t_1, t_2, \mathcal{J}) : \mathbb{R}^2 \rightarrow \mathbb{N}$ as the cumulative number of UAVs that travel through route $R \in \mathcal{R}$ and must arrive at the node $v \in V(R) \setminus \{0^R\}$ in the time interval $[t_1, t_2]$, i.e.,

$$\Gamma_v^R(t_1, t_2, \mathcal{J}) = |\{j \in \mathcal{J} \mid R_j = R \text{ and } \mathcal{M}_v^j \subseteq [t_1, t_2]\}|. \quad (6.5)$$

We then define the flow rate at node v through route R in a finite interval $[t_1, t_2]$ as

$$r_v^R(t_1, t_2, \mathcal{J}) = \frac{\Delta_v^R(t_1, t_2, \mathcal{J})}{t_2 - t_1}, \quad (6.6)$$

where $v \in \mathcal{V} \setminus T$ and $R \in \mathcal{R}$.

Before exploring the necessary conditions of the schedules and demands, we need to define the bottleneck of a network, which is a basis of the necessary conditions and the algorithms.

Definition 7 (Flow and Maximizing Flow). A set $\{\tilde{r}_v^R\}_{v \in \mathcal{V}, R \in \mathcal{R}}$ with each $\tilde{r}_v^R \in \mathbb{R}_+$ is a **flow** if the following are satisfied:

$$\tilde{r}_v^R \geq 0 \quad \forall R \in \mathcal{R}, v \in \mathcal{V} \quad (6.7)$$

$$\tilde{r}_v^R = 0 \quad \forall R \in \mathcal{R}, v \in \mathcal{V} \setminus V(R) \quad (6.8)$$

$$\tilde{r}_v^R \leq r_v^R \quad \forall R \in \mathcal{R}, v \in \mathcal{V} \quad (6.9)$$

$$\tilde{r}_v^R = \tilde{r}_{v+1}^R \quad \forall R \in \mathcal{R}, v \in \mathcal{V} \setminus T \quad (6.10)$$

$$\sum_{R \in \mathcal{R}} \tilde{r}_{v-1}^R m_{v-1}^R \leq C_v \quad \forall v \in \mathcal{V} \setminus S \quad (6.11)$$

where $r_{\ell_1, \ell_2}^R = C_{\ell_2} / m_{\ell_1, \ell_2}^R$ with m_{ℓ_1, ℓ_2}^R as defined in (6.3) and we let $r_{\ell_2}^R = r_{1, \ell_2}^R$. Then r_{ℓ_1, ℓ_2}^R is the maximum flow rate through the node v along the route R .

A flow is a **maximizing flow** if it maximizes $\sum_{R \in \mathcal{R}} \tilde{r}_{k_R}^R$ over all flows.

Definition 8 (Bottleneck). The **bottleneck** of a UAM network $\mathcal{N} = (\mathcal{G}, C, \mathcal{R}, \underline{x}, \bar{x}, w)$ is a set of nodes \bar{V} such that for all $R \in \mathcal{R}$, there exists a node $v \in \bar{V}$ and a maximizing flow $\{\tilde{r}_v^R\}_{v \in \mathcal{V}, R \in \mathcal{R}}$ such that $\tilde{r}_v^R = r_v^R$ and that in the reduced graph $\mathcal{G}_{\text{reduced}} = (\mathcal{V}, \mathcal{E} \setminus (\mathcal{E}_1 \cup \mathcal{E}_2))$, for any $v_1 \in S$ and $v_2 \in T$, v_1 and v_2 are not connected, where $\mathcal{E}_1 = \{e \in \mathcal{E} \mid \tau(e) \in \bar{V}\}$ and $\mathcal{E}_2 = \{e \in \mathcal{E} \mid \sigma(e) \in \bar{V}\}$.

The lemma below provides an upper bound on the number of UAVs traveling through a node $v \in \mathcal{V} \setminus S$ that may be newly assigned in a schedule at any scheduling time.

Lemma 8. Consider a network $\mathcal{N} = (\mathcal{G}, C, \mathcal{R}, \underline{x}, \bar{x}, w)$ where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a sequence of increasing scheduling times $t_0 < t_1 < t_2 < \dots$. Given the set of demands $\mathcal{D}(t)$ and $R \in \mathcal{R}$, let $\mathcal{S}(t_i) = \{\delta_j\}_{j \in \hat{\mathcal{J}}(t_i)}$ be a schedule for the set of available demands $\mathcal{D}(t_i)$ at the scheduling time t_i for all $i > 0$ where $\hat{\mathcal{J}}(t_i) \subseteq \mathcal{J}(t_i)$ and let $\mathcal{J}(t_{-1}) = \emptyset$. Then $C_v(t_b - t_a) \geq \sum_{R \in \mathcal{R}_v} \Gamma_v^R(t_a, t_b, \hat{\mathcal{J}}_i^\Delta) m_v^R$ for any time interval $[t_a, t_b]$ and for all $v \in \mathcal{V} \setminus S$, where $\mathcal{R}_v = \{R \in \mathcal{R} \mid v \in V(R)\}$ for all $v \in \mathcal{V}$, $\hat{\mathcal{J}}_i^\Delta = \hat{\mathcal{J}}(t_i) \setminus \hat{\mathcal{J}}(t_{i-1})$, $t_0 \leq t_a < t_b$ and $t_a, t_b \in \mathbb{R}$.

The proof of Lemma 8 follows from the Definition 6 and is omitted. We then deduce a necessary condition on nodes for given set of demands to have a complete schedule. Lemma 8 implies that, given a set of demands $\mathcal{D}(t_i)$ at the scheduling time t_i , $i > 0$, if there exists a node $v \in \mathcal{V} \setminus S$ such that

$$C_v \cdot \left(\max_{j \in \mathcal{J}_v} f_j - t_i \right) < \sum_{j \in \mathcal{J}_v} m_v^{R_j}, \quad (6.12)$$

where $\mathcal{J}_v = \{j \in \mathcal{J}_i^\Delta \mid v \in V(R_j)\}$ and $\mathcal{J}_i^\Delta = \mathcal{J}(t_i) \setminus \hat{\mathcal{J}}(t_{i-1})$, then there does not exist a complete a schedule for $D(t_i)$. We can further narrow this necessary condition by denoting $\underline{\mathcal{M}}_{\ell_1, \ell_2}^j$ (resp., $\overline{\mathcal{M}}_{\ell_1, \ell_2}^j$) as the shortest time (resp., longest time) it takes to travel from node $\ell_1 - 1$ to ℓ_2 , so that $\underline{\mathcal{M}}_{\ell_1, \ell_2}^j = \sum_{\ell=\ell_1}^{\ell_2} \underline{x}_\ell + (\ell_2 - \ell_1)w$ and $\overline{\mathcal{M}}_{\ell_1, \ell_2}^j = \sum_{\ell=\ell_1}^{\ell_2} \overline{x}_\ell + (\ell_2 - \ell_1)w$. We denote $\underline{\mathcal{M}}_{\ell_2}^j = \underline{\mathcal{M}}_{1, \ell_2}^j$ and $\overline{\mathcal{M}}_{\ell_2}^j = \overline{\mathcal{M}}_{1, \ell_2}^j$. We let

$$f_{j,v} = f_j - \overline{\mathcal{M}}_{v, k_{R_j}}^j \quad (6.13)$$

for all $j \in \mathcal{J}$ and $v \in V(R_j)$ so that $f_{j,v}$ is the latest arrival (resp., departure) time for demand j at node v when $v \neq 0$ (resp., $v = 0$) such that the deadline at the destination k_{R_j} is not violated. Therefore, for all $v \in \mathcal{V} \setminus S$,

$$C_v \cdot \left(\max_{j \in \mathcal{J}_v} f_{j,v} - \min_{j \in \mathcal{J}_v} \underline{\mathcal{M}}_v^j - t_i \right) \geq \sum_{j \in \mathcal{J}_v} m_v^{R_j} \quad (6.14)$$

is a necessary condition for the set of demands $D(t_i)$ to have a complete schedule.

Theorem 6 below can help to come up with another necessary condition for the existence of a complete schedule for a set of demands at any scheduling time.

Theorem 6. *Consider a network $\mathcal{N} = (\mathcal{G}, C, \mathcal{R}, \underline{x}, \overline{x}, w)$ where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a sequence of increasing scheduling times $t_0 < t_1 < t_2 < \dots$. Given the set of demands $\mathcal{D}(t)$ and $R \in \mathcal{R}$, let $\mathcal{S}(t_i) = \{\delta_j\}_{j \in \hat{\mathcal{J}}(t_i)}$ be a schedule for the set of available demands $\mathcal{D}(t_i)$ at the scheduling time t_i , for any $i > 0$ and let $\mathcal{J}(t_{-1}) = \emptyset$. If there exists a schedule*

$\mathcal{S}(t_i) = \{\delta_j\}_{j \in \hat{\mathcal{J}}(t_i)}$, then it must satisfies:

$$\sum_{R \in \mathcal{R}} \Delta_0^R(t_a, t_b, \hat{\mathcal{J}}_i^\Delta) \leq (t_b - t_a) \sum_{v \in \bar{V}} \sum_{R \in \mathcal{R}_v} r_v^R \quad (6.15)$$

for any t_a, t_b such that $t_0 \leq t_a < t_b$, where \bar{V} is the bottleneck, $\hat{\mathcal{J}}_i^\Delta = \hat{\mathcal{J}}(t_i) \setminus \hat{\mathcal{J}}(t_{i-1})$, and $\mathcal{R}_v = \{R \in \mathcal{R} \mid v \in V(R)\}$ for all $v \in \mathcal{V}$, i.e., the number of new departures assigned in the interval $[t_a, t_b]$ must be less than the product of the length of the time interval and the sum of maximum possible flow through the bottleneck of the network.

Proof. We first rewrite (6.15) as $\sum_{R \in \mathcal{R}} r_0^R(t_1, t_2, \hat{\mathcal{J}}_i^\Delta) \leq \sum_{v \in \bar{V}} \sum_{R \in \mathcal{R}_v} r_v^R$. To complete the proof, we only need to show that flow rate $r_v^R(t_1, t_2, \hat{\mathcal{J}}_i^\Delta)$ at node v through route R is actually limited by the constraints (6.7)–(6.11). Noticeably, we can consider it as a static scheduling because the previous schedule will only reduce the number of UAVs be scheduled to depart in a time interval. Therefore, we assume there are no other assigned demands.

First of all, consider $R \in \mathcal{R}$, then by definition it holds that $r_v^R(t_1, t_2, \hat{\mathcal{J}}_i^\Delta) \geq 0$ for all $v \in \mathcal{V}$, and $r_v^R(t_1, t_2, \hat{\mathcal{J}}_i^\Delta) = 0$ for all $v \notin V(R)$. By utilizing Lemma 8 it follows that $r_v^R(t_1, t_2, \hat{\mathcal{J}}_i^\Delta) \leq r_v^R$ for all $R \in \mathcal{R}$ and $v \in \mathcal{V}$, and $\sum_{R \in \mathcal{R}} r_{v-1}^R(t_1, t_2, \hat{\mathcal{J}}_i^\Delta) m_{v-1}^R \leq C_v$ for all $v \in \mathcal{V} \setminus T$. In the static scheduling scenario we can consider $r_v^R(t_1, t_2, \hat{\mathcal{J}}_i^\Delta)$ to be the static along the route, as a result, (6.10) becomes trivial. Therefore, the flow rate $r_v^R(t_1, t_2)$ is actually limited by the constraints (6.7)–(6.11) and by Definition 8 of the bottleneck of a network, it follows that $\sum_{R \in \mathcal{R}} r_0^R(t_1, t_2, \hat{\mathcal{J}}_i^\Delta) \leq \sum_{v \in \bar{V}} \sum_{R \in \mathcal{R}_v} r_v^R$. \square

Similar to the necessary condition of each node, we can conclude from Theorem 6 that, for any scheduling time t_i , where $i > 0$, there exists a complete schedule for the set of demands $\mathcal{D}(t_i)$ only if

$$|\mathcal{J}_i^\Delta| \leq \left(\max_{j \in \mathcal{J}_i^\Delta} (f_{j,0}) - t_i \right) \cdot \sum_{v \in \bar{V}} \sum_{R \in \mathcal{R}_v} r_v^R. \quad (6.16)$$

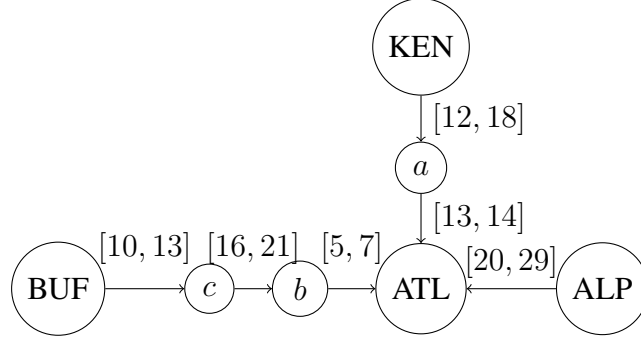


Figure 6.2: A star-branch graph for a UAM network of Atlanta (ATL) with three exurbs: Alpharetta (ALP), Kennesaw (KEN) and Buford (BUF). We consider Atlanta as the central node, and there are 0, 1, 2 intermediate nodes between Atlanta node and the three leaf nodes “ALP”, “KEN” and “BUF”, respectively. The time interval need for traveling through each link is labeled beside the corresponding link.

6.2.2 Fundamental Limits for Static Scheduling

We investigate fundamental limits for static scheduling on a special class of directed graph $\mathcal{G}_* = (\mathcal{V}_*, \mathcal{E}_*)$, where the set of nodes \mathcal{V}_* consists of: a central node v_0 ; L leaf nodes v_l for $l = 1, 2, \dots, L$; and, for each leaf node v_l , a set of intermediate nodes between v_l and the central node v_0 denoted $v_{l,k}$ for $k = 1, 2, \dots, K_l$ for $K_l \in \mathbb{N}_0$. If $K_l = 0$, then there are no intermediate nodes between v_l and v_0 . Then $\mathcal{E}_* = \{(v_{l,k}, v_{l,k-1}) \mid 1 \leq l \leq L \text{ and } 1 \leq k \leq K_l + 1\}$ is the set of links for the network, where we let $v_{l,0} = v_0$ and $v_{l,K_l+1} = v_l$ for all l . We define any graph that satisfies the conditions above as a *star-branch graph*. An example star-branch graph is shown in Fig. 6.2 and serves as the case study below, where the central node is labeled ATL, the leaf nodes are labeled BUF, KEN, and ALP, and the intermediate nodes are labeled a , b , and c .

In a star-branch graph, for each leaf node v_l , there exists a unique set of connected links $\{e_{l,k}\}_{k=1}^{K_l+1}$ along the path to the central node v_0 , where we let $e_{l,k} = (v_{l,k}, v_{l,k-1})$. We define *branch* l as the pair $(\{e_{l,k}\}_{k=1}^{K_l+1}, \{v_{l,k}\}_{k=0}^{K_l+1})$. We consider there to be L routes in \mathcal{R} and each route $R_l \in \mathcal{R}$ consists of the edges of a branch, i.e., $R_l = \{e_{l,k}\}_{k=1}^{K_l+1}$.

In this section, we first consider a star-branch network with no intermediate nodes, i.e., a star network. In this case, the set of nodes \mathcal{V}_* consists of a central node v_0 and L leaf

nodes v_l for $l = 1, 2, \dots, L$. Then $\mathcal{E}_* = \{(v_l, v_0) \mid 1 \leq l \leq L\}$ is the set of links for the network. We label edge (v_i, v_0) simply as edge i . We call such a network a *local network* because it can be interpreted as a local portion of a larger network where we study only incoming flights to a particular node. Consider the static set of demands $\mathcal{D} = (R_j, f_j)_{j \in \mathcal{J}}$. We then define the collection of demands \mathcal{D} as *feasible* if there exists a schedule for \mathcal{D} . If \mathcal{D} is a finite collection of demands, then there always exists a schedule since departure times may be scheduled as early as needed to satisfy capacity constraints and desired arrival times. When \mathcal{D} is an infinite collection of demands, then \mathcal{D} may or may not be feasible. We assume that the number of deadlines within the time interval $[\tau, \tau + T]$ is finite for all $T < \infty$ and the limiting average number of deadlines in $[\tau, \tau + T]$ as $T \rightarrow \infty$ exists and is constant for varying $\tau \in \mathbb{R}$, e.g., \mathcal{D} is a periodic set of demands. We next explore the fundamental limitations on the feasibility of the infinite set of demands \mathcal{D} in the remainder of this section.

We introduce a cumulative departure function in time interval $[t_1, t_2]$ as $\Delta_v(t_1, t_2) : \mathbb{R}^2 \rightarrow \mathbb{N}$ for all $v \in \mathcal{V} \setminus \{v_0\}$ so that $\Delta_v(t_1, t_2)$ is the number of UAVs departing from node v in the time interval $[t_1, t_2]$, i.e.,

$$\Delta_v(t_1, t_2) = |\{j \in \mathcal{J} \mid o_j = v \text{ and } \delta_j \in [t_1, t_2]\}|.$$

Given the cumulative departure function, we define the long-term average departure rate r_v at node $v \in \mathcal{V} \setminus \{v_0\}$ as

$$r_v := \lim_{T \rightarrow +\infty} \frac{1}{T} \Delta_v(\tau, \tau + T), \quad \forall \tau \in \mathbb{R}. \quad (6.17)$$

In the same manner, we introduce a cumulative arrival function $\Gamma_v(t_1, t_2) : \mathbb{R}^2 \rightarrow \mathbb{N}$ as the cumulative number of UAVs that depart from origin v and arrive at the destination in the time interval $[t_1, t_2]$, i.e., $\Gamma_v(t_1, t_2) = |\{j \in \mathcal{J} \mid o_j = v \text{ and vehicle arrives in } [t_1, t_2]\}|$.

For any schedule, the average departure and the arrival rates must be equal, as stated

next.

Lemma 9. *Consider a local UAM network \mathcal{N} with a set of demands \mathcal{D} . For any schedule \mathcal{S} and for all nodes $v \in \mathcal{V} \setminus \{v_0\}$, it holds that the average arrival rate from the node equals the average departure rate from the node, i.e.,*

$$r_v = \lim_{T \rightarrow +\infty} \frac{1}{T} \Gamma_v(\tau, \tau + T), \quad \forall v \in \mathcal{V} \setminus \{v_0\}, \forall \tau \in \mathbb{R}. \quad (6.18)$$

The proof of Lemma 9 is straightforward follows directly from the conservation of the total number of UAVs in the network.

Proof. Let t_0 be any time in \mathbb{R} and $T > 0$ be a duration of time. Any UAV departed from origin $v_i \in \mathcal{V} \setminus \{v_0\}$ will arrive at the destination in the interval $[t_0, t_0 + T]$ only if it departs in the interval $[t_0 - \bar{x}_i, t_0 + T - \underline{x}_i]$. Hence it must hold that $\Gamma_{v_i}(t_0, t_0 + T) \leq \Delta_{v_i}(t_0 - \bar{x}_i, t_0 + T - \underline{x}_i)$. In a similar manner, if a UAV departs from the origin v_i in the interval $[t_0 - \underline{x}_i, t_0 + T - \bar{x}_i]$, then it must arrive at the destination within the interval $[t_0, t_0 + T]$ and hence $\Gamma_{v_i}(t_0, t_0 + T) \geq \Delta_{v_i}(t_0 - \underline{x}_i, t_0 + T - \bar{x}_i)$. Let $t_1 = t_0 - \underline{x}_i$, we have

$$\begin{aligned} \Delta_{v_i}(t_0 - \underline{x}_i, t_0 + T - \bar{x}_i) &\leq \Gamma_{v_i}(t_0, t_0 + T) \\ \Delta_{v_i}(t_1, t_1 + T - (\bar{x}_i - \underline{x}_i)) &\leq \Gamma_{v_i}(t_0, t_0 + T). \end{aligned}$$

By writing $[t_1, t_1 + T - (\bar{x}_i - \underline{x}_i)]$ as $[t_1, t_1 + T] \setminus [t_1 + T - (\bar{x}_i - \underline{x}_i), t_1 + T]$, while $[t_1 + T - (\bar{x}_i - \underline{x}_i), t_1 + T] \subseteq [t_1, t_1 + T]$, we can then derive from above that

$$\Delta_{v_i}(t_1, t_1 + T) - \Delta_{v_i}(t_1 + T - (\bar{x}_i - \underline{x}_i), t_1 + T) \leq \Gamma_{v_i}(t_0, t_0 + T). \quad (6.19)$$

Since $\lim_{T \rightarrow +\infty} \frac{1}{T} \Delta_{v_i}(\tau, \tau + T) = r_{v_i}$ for any τ , then there exist $T_0 > 0$ such that for any $T \geq T_0$, $(r_{v_i} - r_{v_i}) \leq \frac{1}{T} \Delta_{v_i}(\tau, \tau + T) \leq (r_{v_i} + r_{v_i})$. It follows that, $0 \leq \Delta_{v_i}(\tau, \tau + T) \leq 2r_{v_i}T$ for any $T \geq T_0$, hence we can conclude $\Delta_{v_i}(\tau, \tau + T) \leq 2r_{v_i}T_0$ for any $T < T_0$

because $\Delta_{v_i}(\tau, \tau + t)$ is monotonically increasing with t . Without loss of generality, we can assume that $T_0 > \bar{x}_i - \underline{x}_i$. Thus we have $\Delta_{v_i}(t_1 + T - (\bar{x}_i - \underline{x}_i), t_1 + T) \leq 2r_{v_i}T_0$. Then we can obtain

$$\Delta_{v_i}(t_1, t_1 + T) - \Delta_{v_i}(t_1 + T - (\bar{x}_i - \underline{x}_i), t_1 + T) \geq \Delta_{v_i}(t_1, t_1 + T) - 2r_{v_i}T_0. \quad (6.20)$$

By combining (6.19) and (6.20), we get

$$\Delta_{v_i}(t_1, t_1 + T) - 2r_{v_i}T_0 \leq \Gamma_{v_i}(t_0, t_0 + T).$$

Dividing both sides with T , and taking the limit yields

$$r_{v_i} \leq \lim_{T \rightarrow +\infty} \frac{1}{T} \Gamma_{v_i}(t_0, t_0 + T). \quad (6.21)$$

Without loss of generality, the fact that $\Gamma_{v_i}(t_0, t_0 + T) \leq \Delta_{v_i}(t_0 - \bar{x}_i, t_0 + T - \underline{x}_i)$ gives that

$$\lim_{T \rightarrow +\infty} \frac{1}{T} \Gamma_{v_i}(t_0, t_0 + T) \leq r_{v_i}. \quad (6.22)$$

By (6.21) and (6.22) while substituting v_i by v , we can therefore conclude that

$$\lim_{T \rightarrow +\infty} \frac{1}{T} \Gamma_v(t_0, t_0 + T) = r_v$$

for all $t_0 \in \mathbb{R}$ and $v \in \mathcal{V} \setminus \{v_0\}$. □

In the same manner, it follows that for any schedule, the arrival rate must satisfy

$$r_v = \lim_{T \rightarrow +\infty} \frac{1}{T} \sum_{j \in \mathcal{J} | o_j = v} \mathbf{1}(f_j; [\tau, \tau + T]) \forall v \in \mathcal{V} \setminus \{v_0\}, \forall \tau \in \mathbb{R}. \quad (6.23)$$

6.2.3 Necessary and Sufficient Condition for a Schedule in a Local Network

In the following theorem, we obtain a necessary and sufficient condition for the existence of a schedule for a local network when the set of demands \mathcal{D} is infinitely large, so that we can say immediately if there exists any schedule. This will hence provide fundamental limits for how large demands a local network can handle.

Theorem 7. *Consider a local UAM network \mathcal{N} . An infinite set of demands \mathcal{D} is feasible if and only if*

$$\sum_{1 \leq i \leq L} r_{v_i} \cdot (\bar{x}_i - \underline{x}_i + w) \leq C_{v_0}, \quad (6.24)$$

where r_{v_i} is as given in (6.17) for all $v_i \in \mathcal{V} \setminus \{v_0\}$ and we recall C_{v_0} is the capacity of the destination node v_0 .

The proof of Theorem 7 is based on the two lemmas below, which focus on the special case when $C_{v_0} = 1$. We can then extend this special case to prove Theorem 7.

Lemma 10. *Consider a local UAM network \mathcal{N} with $C_{v_0} = 1$. If an infinite set of demands \mathcal{D} is feasible, then*

$$\sum_{1 \leq i \leq L} r_{v_i} \cdot (\bar{x}_i - \underline{x}_i + w) \leq 1, \quad (6.25)$$

where r_{v_i} is as given in (6.17) for all $v_i \in \mathcal{V} \setminus \{v_0\}$.

Proof. Recall that, in a local network, a UAV departing from node v_i travels along link i for a non-deterministic amount of time within the interval $[\underline{x}_i, \bar{x}_i]$ and occupies a landing spot at the central node v_0 for time w . No other UAVs are allowed to depart if it is possible that it will arrive at the central node v_0 at a time when there is no landing capacity. We first define

$$g(t_0, t_1) = \sum_{1 \leq i \leq L} \Gamma_{v_i}(t_0, t_1)(\bar{x}_i - \underline{x}_i + w). \quad (6.26)$$

We claim that $g(t_0, t_1)$ is the total time needed to be reserved at the destination node v_0 for all journeys arriving in the time interval $[t_0, t_1]$.

Consider three consecutive scheduled arrival times at the destination, t_{n-1} , t_n and t_{n+1} , where $t_{n-1} \leq t_n \leq t_{n+1}$. We know that the corresponding UAVs travel through links i_1 , i_2 and i_3 where $i_1, i_2, i_3 \in \mathcal{E}$. Therefore the departure time of the three UAVs should be $t_{n-1} - \bar{x}_{i_1}$, $t_n - \bar{x}_{i_2}$ and $t_{n+1} - \bar{x}_{i_3}$. Since the latest time of the UAV staying at the destination has to be earlier than the earliest time that the next UAV arrives, we conclude $t_{n-1} + w \leq t_n - \bar{x}_{i_2} + \underline{x}_{i_2}$, and $t_n + w \leq t_{n+1} - \bar{x}_{i_3} + \underline{x}_{i_3}$. We can then obtain $(t_{n+1} - \bar{x}_{i_3} + \underline{x}_{i_3}) - (t_{n-1} + w) \geq \bar{x}_{i_2} - \underline{x}_{i_2} + w$. That is, upon scheduling, the earliest time that the third UAV may arrive at the destination needs to be at least $(\bar{x}_{i_2} - \underline{x}_{i_2} + w)$ later than the first one's latest leaving time in order to fit the schedule of the second UAV at destination. Since i_2 can be any link, we substitute it with i . Hence the time reserved at destination for each UAV traveling through i is at least $(\bar{x}_i - \underline{x}_i + w)$. Then $g(t_0, t_1)$ defined in (6.26) sums the product of time reserved for a UAV traveling through link i and the corresponding number of arrivals through that link, which is the total time to be reserved at the destination node v_0 for all journeys arriving in the time interval $[t_0, t_1]$.

Assume the infinite set of demands \mathcal{D} is feasible. Then for any $T > 0$, we must have $g(t_0, t_0 + T) - (\bar{x}_{i_T} - \underline{x}_{i_T} + w) \leq T$, where i_T is the link of the last journey scheduled to arrive at the destination in the interval $[t_0, t_0 + T]$. We denote $m_i = \bar{x}_i - \underline{x}_i + w$, since $\bar{x}_{i_T} - \underline{x}_{i_T} + w \leq \max_{\ell} \{\bar{x}_{\ell} - \underline{x}_{\ell} + w\} = \bar{m}$, and then $T \geq g(t_0, t_0 + T) - (\bar{x}_{i_T} - \underline{x}_{i_T} + w) \geq g(t_0, t_0 + T) - \bar{m}$. We thus obtain

$$\begin{aligned}
T &\geq g(t_0, t_0 + T) - \bar{m} = \sum_{1 \leq i \leq L} \Gamma_{v_i}(t_0, t_0 + T) \cdot m_i - \bar{m} \\
\frac{T}{T} &\geq \frac{1}{T} \left[\sum_{1 \leq i \leq L} \Gamma_{v_i}(t_0, t_0 + T) \cdot m_i - \bar{m} \right] \\
1 &\geq \lim_{T \rightarrow +\infty} \frac{1}{T} \left[\sum_{1 \leq i \leq L} \Gamma_{v_i}(t_0, t_0 + T) \cdot m_i - \bar{m} \right] \\
1 &\geq \sum_{1 \leq i \leq L} \left[\lim_{T \rightarrow +\infty} \frac{1}{T} \Gamma_{v_i}(t_0, t_0 + T) \cdot m_i - 0 \right] \\
1 &\geq \sum_{1 \leq i \leq L} r_{v_i} \cdot m_i.
\end{aligned}$$

Therefore, we conclude that a feasible solution is possible only if (6.25) is satisfied. \square

Lemma 11. *Consider a local UAM network \mathcal{N} with $C_{v_0} = 1$ and a countably infinite set of demands $\mathcal{D} = \{(o_j, f_j)\}_{j \in \mathcal{J}}$ with $f_j > t_0$ for all $j \in \mathcal{J} = \mathbb{N}$. If $\sum_{1 \leq i \leq L} r_{v_i} \cdot (\bar{x}_i - \underline{x}_i + w) \leq 1$, then there always exists $\bar{M} \in \mathbb{R}$ such that*

$$\sum_{1 \leq i \leq L} \Gamma_{v_i}(t_0, t_0 + T) \cdot (\bar{x}_i - \underline{x}_i + w) - f_T \leq \bar{M} \quad (6.27)$$

for any $T > 0$, where $f_T = \max_{\{j \in \mathcal{J} | f_j \leq T\}} \{f_j\}$ is the last deadline that needs to be achieved before T . In particular, this implies that the set of demands \mathcal{D} is feasible.

Proof. We will prove the lemma by contradiction. We let $m_i = \bar{x}_i - \underline{x}_i + w$. Suppose no \bar{M} exists such that (6.27) holds. That is, for any D , there exists T such that $\sum_{1 \leq i \leq L} \Gamma_{v_i}(t_0, t_0 + T) \cdot m_i - f_T > D$. Let $\hat{r} = \sum_{1 \leq i \leq L} r_{v_i}$. Since $\lim_{T \rightarrow +\infty} \frac{1}{T} \Gamma_{v_i}(t_0, t_0 + T) = \sum_{1 \leq i \leq L} r_{v_i} = \hat{r} > 0$, for any $\epsilon > 0$, there exists $T_0 > 0$ such that for any $T \geq T_0$, $(\hat{r} - \epsilon)T < \sum_{1 \leq i \leq L} \Gamma_{v_i}(t_0, t_0 + T) < (\hat{r} + \epsilon)T$. With the choice $\epsilon = \hat{r}$, we conclude there exists $T_1 > 0$ such that

$$T \geq T_1 \quad \text{implies} \quad 0 < \sum_{1 \leq i \leq L} \Gamma_{v_i}(t_0, t_0 + T) < 2\hat{r}T. \quad (6.28)$$

Without loss of generality, assume demands are numbered in increasing order according to deadline, i.e., such that $t_0 < f_k \leq f_{k+1}$ for all $k \in \mathbb{N}$. Let $\bar{m} = \max_i m_i$. Choose $D_0 > 2\hat{r}T\bar{m}$. We claim that if $\sum_{1 \leq i \leq L} [\Gamma_{v_i}(t_0, f_k) \cdot m_i] \geq D_0$ for some $k \in \mathbb{N}$, then $f_k \geq T_1 + t_0$. To prove the claim by contraposition, suppose k is such that $f_k < T_1 + t_0$. Then, using (6.28), $\sum_{1 \leq i \leq L} [\Gamma_{v_i}(t_0, f_k) \cdot m_i] \leq \sum_{1 \leq i \leq L} \Gamma_{v_i}(t_0, T_1 + t_0) \cdot m_i \leq \sum_{1 \leq i \leq L} \Gamma_{v_i}(t_0, T_1 + t_0) \cdot \bar{m} < 2\hat{r}T_1 \cdot \bar{m} < D_0$, and the claim is proved.

Let $\Lambda_k = \max\{\sum_{1 \leq i \leq L} \Gamma_{v_i}(t_0, f_k) \cdot m_i - f_k, 0\}$ for all $k = 1, 2, \dots$. Since $0 \leq f_{k-1} \leq f_k$, then $\Lambda_k \leq \Lambda_{k-1} + \bar{m}$.

Let $D_n = nD_0$ for $n = 1, 2, \dots$. We then construct a sub-sequence $\{f_{k_n}\}_{n=1}^{\infty}$ such that $k_n = \arg \min_{k_n > k_{(n-1)}} \{\sum_{1 \leq i \leq L} \Gamma_{v_i}(t_0, f_{k_n}) \cdot m_i - f_{k_n} > D_n\} = \arg \min_{k_n > k_{(n-1)}} \{\Lambda_{k_n} >$

$D_n\}$. We are able to construct such a sequence since we assumed that for any D , exist T such that $\sum_{1 \leq i \leq L} \Gamma_{v_i}(t_0, T + t_0) \cdot m_i - f_T > D$. Notice that this is equivalent to that for any D , there exists n such that $\sum_{1 \leq i \leq L} \Gamma_{v_i}(t_0, f_{k_n}) \cdot m_i - f_{k_n} > D$.

Therefore, suppose there exists k_{n_0} such that we cannot find the next element, that is, $\Lambda_{k_n} < D_{(n_0+1)}$ for all $k_n > k_{n_0}$. Let $\bar{D} = \max\{\max_{k=1,2,\dots,k_{n_0}} \Lambda_k, D_{(n_0+1)}\}$, then $\Lambda_k \leq \bar{D}$ for all k which is a contradiction.

We claim that $\{f_{k_n}\}_{n=1}^{\infty}$ is an unbounded, monotonically increasing sequence. By construction, $k_n > k_{(n-1)}$, and since the sequence $\{f_k\}_{k=1}^{\infty}$ is monotonically increasing, then $f_{k_n} > f_{k_{(n-1)}}$ for all n . Therefore, $\{f_{k_n}\}_{n=1}^{\infty}$ is monotonically increasing. Moreover, suppose it is bounded by \hat{F} (i.e., $f_{k_n} \leq \hat{F}$ for all n). There are at most $\lceil \frac{\hat{F}-t_0}{T_1} \rceil$ number of time periods with length T_1 , each can have less than $2\hat{r}T_1$ incidences. Therefore, $\sum_{1 \leq i \leq L} \Gamma_{v_i}(t_0, f_{k_n}) \cdot m_i \leq \sum_{1 \leq i \leq L} \Gamma_{v_i}(t_0, f_{k_n}) \cdot \bar{m} \leq \sum_{1 \leq i \leq L} \Gamma_{v_i}(t_0, \hat{F}) \cdot \bar{m} < \lceil \frac{\hat{F}-t_0}{T_1} \rceil \cdot 2\hat{r}T_1 \cdot \bar{m} < \lceil \frac{\hat{F}-t_0}{T_1} \rceil \cdot D_0$ for all v_i . However, since $\sum_{1 \leq i \leq L} \Gamma_{v_i}(t_0, f_{k_n}) \cdot m_i - f_{k_n} > D_n$ for all n , then $n \cdot D_0 = D_n < f_{k_n} + D_n < \sum_{1 \leq i \leq L} \Gamma_{v_i}(t_0, f_{k_n}) \cdot m_i < \lceil \frac{\hat{F}-t_0}{T_1} \rceil \cdot D_0$, which contradicts with the fact that n can become infinitely large. Therefore $\{f_{k_n}\}_{n=1}^{\infty}$ is an unbounded, monotonically increasing sequence and thus $\lim_{n \rightarrow \infty} f_{k_n} \rightarrow \infty$.

Then we have

$$\begin{aligned}
& \sum_{1 \leq i \leq L} \Gamma_{v_i}(t_0, f_{k_n}) \cdot m_i - f_{k_n} > D_n \\
& \sum_{1 \leq i \leq L} \Gamma_{v_i}(t_0, f_{k_n}) \cdot m_i > D_n + f_{k_n} \\
& \frac{1}{f_{k_n}} \sum_{1 \leq i \leq L} \Gamma_{v_i}(t_0, f_{k_n}) \cdot m_i > \frac{D_n + f_{k_n}}{f_{k_n}} \\
& \sum_{1 \leq i \leq L} \left[\lim_{n \rightarrow \infty} \frac{1}{f_{k_n}} \Gamma_{v_i}(t_0, f_{k_n}) \cdot m_i \right] > \lim_{n \rightarrow \infty} \frac{D_n + f_{k_n}}{f_{k_n}} \\
& \sum_{1 \leq i \leq L} \left[\lim_{f_{k_n} \rightarrow \infty} \frac{1}{f_{k_n}} \Gamma_{v_i}(t_0, f_{k_n}) \cdot m_i \right] > \lim_{f_{k_n} \rightarrow \infty} \frac{D_n + f_{k_n}}{f_{k_n}} \\
& 1 \geq \sum_{1 \leq i \leq L} r_{v_i} \cdot m_i > \lim_{f_{k_n} \rightarrow \infty} \frac{D_n}{f_{k_n}} + 1 \geq 1
\end{aligned}$$

$$1 > 1, \quad (6.29)$$

a contradiction.

Therefore, there exist a positive real number \bar{M} , so that $\sum_{1 \leq i \leq L} \Gamma_{\tau(i)}(t_0, T + t_0) \cdot m_i - f_T \leq \bar{M}$ for any $T > 0$. As a result, the departure time of the journeys is bounded below by $\min_j f_j - \bar{M} - \max_{1 \leq i \leq L} \{\bar{x}_i\}$ and thus the set of demands \mathcal{D} is feasible. \square

6.2.4 Extension to General Star-Branch Network

We now show that the necessary condition of Theorem 7 extends to general star-branch networks with intermediate nodes between origins and the destination in the following corollary.

Corollary 2. *Consider a UAM network $\mathcal{N} = (\mathcal{G}_*, C, \underline{x}, \bar{x})$ where \mathcal{G}_* is a star-branch graph with L leaf nodes and thus L branches, $(\{e_{l,k}\}_{k=1}^{K_l+1}, \{v_{l,k}\}_{k=0}^{K_l+1})$, for all $l = 1, \dots, L$. If a countably infinite set of demands \mathcal{D} is feasible, then*

$$r_{v_l} \leq \hat{r}_{v_l} = \min_{1 \leq k_I \leq K_l+1} \hat{r}_{l,k_I}, \quad \forall 1 \leq l \leq L \quad (6.30)$$

where \hat{r}_{l,k_I} is obtained by the equation below:

$$C_{v_l, k_I} = \left(\sum_{k=1}^{k_I} \bar{x}_{e_{l,k}} - \sum_{k=1}^{k_I} \underline{x}_{e_{l,k}} + w_{l,k_I} \right) \cdot \hat{r}_{l,k_I}. \quad (6.31)$$

Proof. Consider branch l of the graph, $(\{e_{l,k}\}_{k=1}^{K_l+1}, \{v_{l,k}\}_{k=0}^{K_l+1})$. The leaf node on the branch is $v_l = v_{l, K_l+1}$. The time interval to reach any other node v_{l, k_I} on the branch from the leaf node will be

$$\left[\sum_{k=1}^{k_I} \underline{x}_{e_{l,k}} + (k_I - 1)w_I, \sum_{k=1}^{k_I} \bar{x}_{e_{l,k}} + (k_I - 1)w_I + w_{l,k_I} \right]. \quad (6.32)$$

We can then consider the pair of nodes v_l and v_{l, k_I} as a simple star graph with a link

(v_l, v_{l,k_I}) with travel time through the link falling in the interval computed in (6.32). Moreover, the UAV will stay at the node v_{l,k_I} for time w_{l,k_I} , hence we can apply Theorem 7 to this two-node network and conclude that the long-term average departure rate r_{v_l} from v_l needs to satisfy

$$r_{v_l} \cdot \left(\sum_{k=1}^{k_I} \bar{x}_{e_{l,k_I}} - \sum_{k=1}^{k_I} \underline{x}_{e_{l,k_I}} + w_{l,k_I} \right) \leq C_{v_{l,k_I}}. \quad (6.33)$$

As a result, the maximum value of r_{v_l} can be obtained in this two-node local network is $\hat{r}_{v_{l,k_I}}$ and thus $r_{v_l} \leq \hat{r}_{v_{l,k_I}}$ where $\hat{r}_{v_{l,k_I}}$ is defined in (6.31). Meanwhile, (6.33) needs to be satisfied for all k_I , hence $r_{v_l} \leq \hat{r}_{v_{l,k_I}}$ for all $k_I = 1, \dots, K_l + 1$, which is equivalent to (6.30). \square

The observation that the above theoretical guarantees extend to the class of star-branch networks significantly expands the practical usefulness of these results. This is because, in many practical scenarios, a UAM network is reasonably decomposed as several independent star-branch networks. For example, during the morning commute, flights will travel to a central city from regional exurbs with possible stops at intermediate suburbs. This is the situation studied in the case study below. During the evening commute, reverse travel from the city to the exurbs is modeled as independent star-branch networks, each with a single branch.

6.3 Optimization Problem for Static Scheduling

When the set of demands \mathcal{D} is finite, it is always possible to find a schedule as we can set the flights to depart early. However, some schedules are less desirable if, e.g., they require the UAVs to depart too early so that the UAVs arrive too far ahead of the corresponding deadlines. Therefore, we propose an optimization program that seeks to minimize the gaps between the arrival times and the corresponding deadlines while remaining feasible in this section. We denote the order that journeys land at the destination v_0 as z , ordered according to arrival a_j , using the index $\gamma \in \{1, \dots, |\mathcal{J}|\}$. For each $j \in \mathcal{J}$, z_γ^j is an

indicator for whether the journey j is scheduled as the γ -th arrival or not: $z_\gamma^j = 1$ means that the assigned UAV is the γ -th flight entering v_0 while $z_\gamma^j = 0$ if not. In particular, consider a star-branch network $\mathcal{N}_* = (\mathcal{G}_*, C, \underline{x}, \bar{x})$ and the optimization program

$$\min_{\delta_j, a_j, z_\gamma^j} \sum_{j \in \mathcal{J}} (f_j - \delta_j)$$

$$a_j \leq f_j \quad \forall j \in \mathcal{J} \quad (6.34)$$

$$a_j = \delta_j + \sum_{k=1|v_l=o_j}^{K_l+1} \bar{x}_{e_l,k} + K_l w_I \quad \forall j \in \mathcal{J} \quad (6.35)$$

$$\sum_{\gamma=1}^{|\mathcal{J}|} z_\gamma^j = 1 \quad \forall j \in \mathcal{J} \quad (6.36)$$

$$\sum_{j \in \mathcal{J}} z_\gamma^j = 1 \quad \forall 1 \leq \gamma \leq |\mathcal{J}| \quad (6.37)$$

$$\sum_{j \in \mathcal{J}} z_\gamma^j \cdot a_j + w \leq \sum_{j \in \mathcal{J}} z_{\gamma+C_{v_0}}^j \cdot \left[\delta_j + \sum_{k=1|v_l=o_j}^{K_l+1} \bar{x}_{e_l,k} + K_l w_I \right] \quad \forall 1 \leq \gamma \leq |\mathcal{J}| \quad (6.38)$$

$$\sum_{j|o_j=v_l} z_\gamma^j \cdot a_j \leq \sum_{j|o_j=v_l} z_{\gamma+1}^j \cdot \left(a_j - \frac{1}{\hat{r}_{v_l}} \right) \quad \forall 1 \leq \gamma \leq |\mathcal{J}|, \forall 1 \leq l \leq L \quad (6.39)$$

$$\delta_j, a_j \in \mathbb{R}, z_\gamma^j \in \{0, 1\}, \forall j \in \mathcal{J}, \forall 1 \leq \gamma \leq |\mathcal{J}|. \quad (6.40)$$

In the optimization problem above, the objective function minimizes the sum of the difference between the deadline and the departure time of each trip. Constraint (6.34) requires each UAV to arrive by the deadline of the corresponding trip. The constraint (6.35) defines the latest arrival time for the journey j as the departure time δ_j plus the longest time it may need to travel along the journey. As depicted in (6.36), each journey is assigned exactly one order index to enter the destination and (6.37) shows that each order index must be occupied by one journey. Since all journeys stay at destination v_0 for a fixed time w , we implicitly assign the parking slots to the journeys by their order index γ cyclically. The constraint (6.38) says for each parking slot, the earliest time that the flight $(\gamma + C_{v_0})$ can enter the parking slot must be later than the latest time that flight γ leaves the slot (the latest

time that a flight leaves the slot is the latest time that the flight will arrive at the slot plus the time it will stay at the node). Similarly, the constraint (6.39) guarantees the rate that the flights entering each branch l will not exceed its maximal allowed average departure rate \hat{r}_{v_l} . Lastly, (6.40) provides the range for all the variables. Notice that time 0 is arbitrarily fixed so that, in particular, $\delta_j < 0$ and $a_j < 0$ are possible.

Even though the constraints (6.38) and (6.39) contain the multiplication of two decision variables, we can reformulate them into a set of affine constraints because z_γ^j is binary and a_j is bounded for any $1 \leq \gamma \leq \mathcal{J}$ and $j \in \mathcal{J}$. Hence, the problem can be transformed into a mixed-integer linear program (MILP). Although the optimization problem after transformation is still non-convex, it can be efficiently solved using solvers such as Gurobi[79] or Cplex[80].

6.4 Numerical Case Studies for Static Scheduling Algorithm

A recent report by INRIX suggests that a UAM local network traveling to the city of Atlanta from three exurbs, Alpharetta, Kennesaw and Buford, has the potential to offer significant time savings compared to ground transportation during peak travel times [14], and we construct our case study from data reported in [14]. Table 6.1 reports plausible travel time intervals for flights to travel from the three exurbs to Atlanta and the corresponding time savings versus peak hours. The fourth column of Table 6.1 is taken from [14], and the third column is inferred from the fourth column and free flow travel times obtained via Google Maps. We further assume there is an intermediate vertistop a at a suburb between Kennesaw and Atlanta, and intermediate vertistops b and c at suburbs between Buford and Atlanta, as shown in Fig. 6.2. The corresponding travel time intervals are labeled beside the links. Conforming to the local network model in Section 6.2.2, we take Atlanta as node number 0 and the exurbs as numbered in Table 6.1. This is therefore a star-branch network with 3 branches, where $v_{2,1} = a$, $v_{3,1} = b$ and $v_{3,2} = c$.

We now present the case study with demand sets $\mathcal{D} = \{(o_j, f_j)\}_{j \in \mathcal{J}}$ defined subse-

Table 6.1: Time Needed for UAVs and Cars to Travel to Atlanta from Different Origins in Rush Hours

Node	Origin	UAV Travel Time (min)	UAV Savings Versus Car Travel Time (min)
1	Alpharetta	$[\underline{x}_1, \bar{x}_1] = [20, 29]$	-41
2	Kennesaw	$[\underline{x}_2, \bar{x}_2] = [25, 32]$	-19
3	Buford	$[\underline{x}_3, \bar{x}_3] = [31, 41]$	-32

quently. Note that the destination for all demands is Atlanta, node v_0 . We consider a time horizon of three hours ($T = 180$ minutes) and assume there are two landing spots in Atlanta, but only one landing spot at vertistops a, b and c , i.e., $C_{v_0} = 2, C_a = C_b = C_c = 1$. Each UAV stays at the intermediate vertistops along its path for $w_I = 1$ minute and at the Atlanta vertiport for $w = 5$ minutes after landing. The optimization problem (6.40) is solved in MATLAB using Gurobi with YALMIP toolbox to obtain a schedule $\mathcal{S} = \{(\{e_{l,k}\}_{k=1}^{K_l+1}, \delta_j)\}_{j \in \mathcal{J}}$ where $l \in \{1, 2, 3\}$ and $K_l = l - 1$. By solving the optimization problem (6.40), we minimize $\sum_{j \in \mathcal{J}} (f_j - \delta_j)$, the sum of the difference between the deadline and the departure time of each trip.

In the case study, we assume the number of UAVs that need to arrive at Atlanta during the three hour horizon varies across the three origins and is denoted h_1, h_2, h_3 . Deadlines are set at regularly intervals, i.e., if origin v_i is tasked with sending h_i flights to Atlanta, the deadlines are $\left\lfloor \frac{180}{1+h_i} \cdot k + 0.5 \right\rfloor, k = 1, 2, \dots, h_i$. We set the deadlines as integers for a shorter convergence time when running the algorithm. We limit the total number of flights under consideration, since the complexity of the non-convex optimization problem in Section 6.3 will lead to a long computational time when there are too many flights. The example we present below takes around two hours for scheduling. In addition to considering a fixed three hour time window, we also consider the case when the demands are repeated indefinitely, providing a countably infinite set of demands.

Fig. 6.3 shows the optimal schedule computed as in Section 6.3 for $[h_1, h_2, h_3] = [4, 4, 19]$. The ‘‘ATL 1’’ and ‘‘ATL 2’’ nodes represents the two parking slots at Atlanta

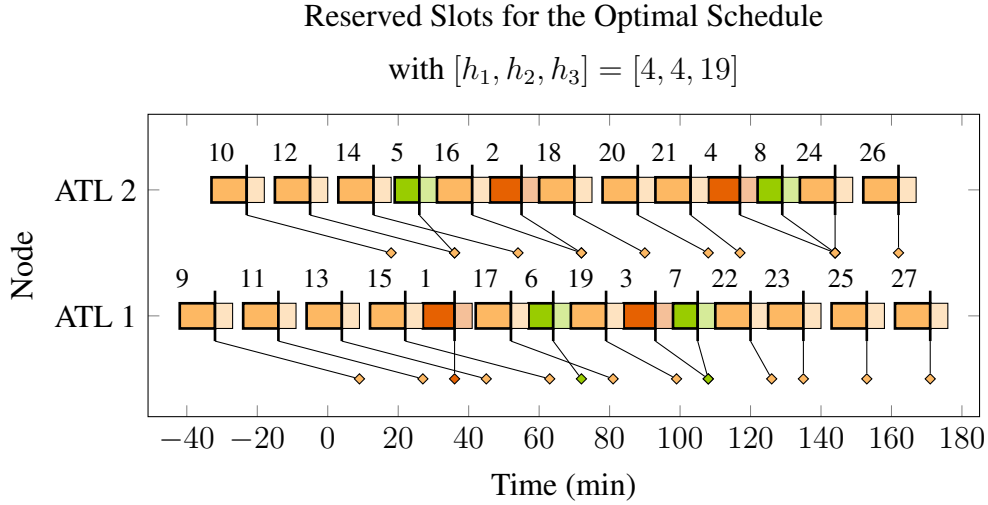


Figure 6.3: The detailed optimal schedule of the second case study with $[h_1, h_2, h_3] = [4, 4, 19]$. The red, green and orange bars represent the reserved time slots for flights from v_1 , v_2 and v_3 , respectively. We label an “ID” above each bar to track the flights. The diamond represents the arrival deadline of a journey; the solid-color bar represents the time interval that the flight is scheduled to arrive. The flight will then stay at the node for time w_I (at intermediate node) or w (at v_0) after arrival (represented by the lightly shaded bar).

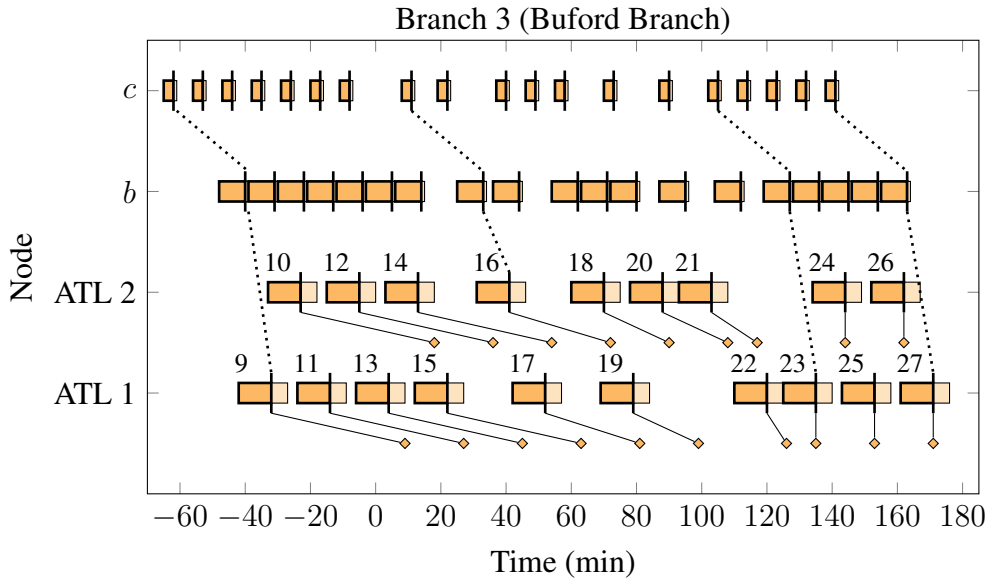


Figure 6.4: The detailed optimal schedule along branch 3. UAVs traveling along branch 3 stop at intermediate nodes b , c and v_0 . The dotted lines are showing that connected bars are the schedules of the same journey at different intermediate nodes and destination.

node (v_0). The red bars indicate UAVs from v_1 , green for v_2 and orange for v_3 . For illustrative purposes, we denote each UAV with a unique ID index, which is labeled above the

bar of the corresponding schedule. In this case, $\sum_{l=1}^3 \frac{h_l}{180} \cdot (\bar{x}_{br_l} - \underline{x}_{br_l} + w) > 2$ so that, if we consider an infinitely repeating demand, Theorem 7 implies that this infinite set of demands is not feasible. As we can verify from the figure, no matter how we adjust the schedule while satisfying all the constraints (6.34)–(6.40), we require more parking slots or more than 180 minutes to fit in all schedules for all the journeys. As a result, when we repeat the demand in $[0, T]$ periodically, there does not exist any schedule for the infinitely repeating demand. Fig. 6.4 shows the schedules of the UAVs along branch 3.

On the other hand, with the same total number of UAVs, if the number of departing UAVs is instead $[h_1, h_2, h_3] = [4, 19, 4]$, then $\sum_{l=1}^3 \frac{h_l}{180} \cdot (\bar{x}_{br_l} - \underline{x}_{br_l} + w) < 2$, satisfying the necessary condition for feasibility, and it can be verified that a schedule (not shown) is obtained from our proposed scheduling algorithm in this case.

6.5 Concluding Remarks

In this chapter, we studied the problem of scheduling in UAM networks with uncertain travel time. One main challenge is that nodes in a UAM network, unlike in a ground transportation network, have limited parking spaces. As a result, a schedule for each UAV in the network has to be made before it takes off to ensure that a parking space is available upon arrival. We incorporated these challenges as constraints in a UAM network scheduling model and presented theoretical results establishing necessary conditions for a schedule to exist. We further showed that these conditions become also sufficient conditions in certain cases. Further, for static scheduling with general star-branch networks, we presented a mixed integer program to obtain an optimal schedule. We demonstrate these theoretical and computational results for static scheduling on a numerical case study for a UAM network in the city of Atlanta.

CHAPTER 7

EFFICIENT ALGORITHMS FOR DYNAMIC SCHEDULING

In Chapter 6, we showed how to obtain the optimal schedule for a finite set of static demands through a mixed integer program, which can be time-consuming, and thus not suitable for general dynamic scheduling problems. In this chapter, we are investigating how dynamic scheduling can be incorporated in the model. We are first explaining the dynamic scheduling algorithm in detail in Section 7.1. We then demonstrate the performance of the dynamic scheduling algorithm in the numerical study in Section 7.2. In the same section, we also compare the performance between the dynamic scheduling method with the static scheduling method in last chapter over the same network and demands.

7.1 A Detailed Explanation of the Dynamic Scheduling Algorithm

In this section, we present an algorithm for dynamic scheduling and, as a special case, static scheduling with finite demands. At its core, the algorithm creates a schedule at each scheduling time using branch-and-bound heuristics to efficiently determine candidate orderings for departure times and then a linear program to determine optimal departure times from the set of fixed orderings. We divide the algorithm into four algorithm blocks. The outermost block, Algorithm 1, creates schedules $\mathcal{S}(t_i)$ at scheduling times $\mathcal{T} = \{t_0, t_1, t_2, \dots\}$ given a UAM network \mathcal{N} , time-varying demands $\mathcal{D}(t)$, and flight arrival times $A(t)$ that are updated according to the description in Section 6.1. We introduce $\mathcal{C}_j = \{c_\ell^j\}_{\ell=1}^{k_{R_j}}$ as the set of parking spots occupied by the flight of the j 'th journey along its route, where $c_\ell^j \in \{1, \dots, C_{\ell R_j}\}$ is the parking spot at node ℓ that the flight occupies upon its arrival. Therefore $\mathcal{C}(t) = \{\mathcal{C}_j\}_{j \in \mathcal{J}(t)}$ is updated while scheduling. The scheduling times \mathcal{T} correspond to when new demands become available to the scheduler or when a UAV lands at a node along its journey, and these times are generally not known in advance. In

Algorithm 1 Event-triggered Scheduling Algorithm

Require: UAM network $\mathcal{N} = (\mathcal{G}, C, \mathcal{R}, \underline{x}, \bar{x}, w)$, maximum schedule size K_0

- 1: $\hat{\mathcal{J}}(t_{-1}) := \emptyset, \mathcal{S}(t_{-1}) := \emptyset, i := 0, t_0 = \text{NOW}$
- 2: $\mathcal{D}(t_0) := \{(R_j, f_j)\}_{j \in \mathcal{J}(t_0)}, A(t_0) := \emptyset, \mathcal{C}(t_0) := \emptyset$ \triangleright Initialize demands, flight arrival times and flight parking spots
- 3: **for** scheduling time t_i **do**
- 4: $\mathcal{J}_i^\Delta := \mathcal{J}(t_i) \setminus \hat{\mathcal{J}}(t_{i-1})$ \triangleright indices of demands eligible to be scheduled
- 5: $K := \max\{K_0, |\mathcal{J}_i^\Delta|\}$
- 6: **while** $K > 0$ **do**
- 7: $(\mathcal{D}^\dagger, \mathcal{J}^\dagger) :=$ subset of K demands and indices with earliest deadlines from set $\{(R_j, f_j)\}_{j \in \mathcal{J}_i^\Delta}$
- 8: $A := A(t_i), \mathcal{C} := \mathcal{C}(t_i)$
- 9: $OptSche := \text{INSERTION}(\mathcal{D}^\dagger, \mathcal{S}(t_{i-1}), t_i, A, \mathcal{C}, \mathcal{N})$
- 10: **if** $OptSche \neq \mathcal{S}(t_{i-1})$ **then**
- 11: $\mathcal{S}(t_i) := OptSche$ \triangleright feasible schedule found
- 12: $\hat{\mathcal{J}}(t_i) := \hat{\mathcal{J}}(t_{i-1}) \cup \mathcal{J}^\dagger$
- 13: **BREAK**
- 14: **else**
- 15: $K := K - 1$
- 16: **end if**
- 17: **end while**
- 18: Update $\mathcal{C}(t_i)$
- 19: **WAITFOREVENT** \triangleright wait for next event
- 20: $i := i + 1$
- 21: $t_i = \text{NOW}$
- 22: Update $D(t_i)$ and $A(t_i)$
- 23: **end for**

this way, Algorithm 1 acts as an event-triggered algorithm that creates a new schedule each time a vehicle lands or a new demand is introduced. For computational considerations, Algorithm 1 also allows for considering only the first K_0 UAV flights with the earliest deadline, and other available demands are scheduled at later scheduling times. In practice, we found that including such a ceiling significantly increases computational speed with negligible effect on the final schedules for all demands.

It is possible that a complete schedule accommodating worst case travel times may not be possible at each scheduling event. In this case, Algorithm 1 creates a partial schedule. The unassigned demands are then considered at the next scheduling time, and any demand that cannot be fulfilled anymore is ultimately dropped.

Algorithm 2 Schedule-Computing Algorithm

```

1: function INSERTION( $\mathcal{D}, \mathcal{S}_{old}, t, A, \mathcal{C}, \mathcal{N}$ )
2:   Inputs: demands  $\mathcal{D} = \{D_j\}_{j \in \mathcal{J}}$ , existing schedule  $\{\delta_j\}_{j \in \mathcal{J}_{old}} := \mathcal{S}_{old}$ , time  $t$ ,
   arrival times  $A$ , parking-spot assignments  $\{\{c_v^j\}_{v \in V(R_j)}\}_{j \in \mathcal{J}(t)} = \mathcal{C}$ , UAM network
    $((\mathcal{V}, \mathcal{E}), C, \mathcal{R}, \underline{x}, \bar{x}, w) := \mathcal{N}$ 
3:    $\ell^j := \arg \min_{\ell \in V(R_j)} a_{\ell, k_{R_j}}^j(t) \quad \forall j \in \mathcal{J}_{old}$ 
4:    $\tilde{M}_{v,c} := \cup_{j \in \mathcal{J}_{old} | v = \ell^{R_j}, \ell \geq \ell^j, c_v^j = c} \mathcal{M}_{\ell^j, v}^j \quad \forall v \in \mathcal{V}, c = 1, \dots, C_v$ 
5:    $\tilde{M} := \{\tilde{M}_{v,c}\}_{v \in \mathcal{V}, c=1, \dots, C_v}$ 
6:    $\tilde{M}_{max} := \max_{v \in \mathcal{V}, c=1, \dots, C_v} (\sup \tilde{M}_{v,c})$ 
7:    $\mathcal{D}_1 := \{(R_j, f_j) \in \mathcal{D} \mid f_j - m_{k_{R_j}}^{R_j} + w \leq \tilde{M}_{max}\}$ , sort descending wrt deadline  $f_j$ 
8:    $\mathcal{D}_2 := \mathcal{D} \setminus \mathcal{D}_1$ , sort ascending wrt  $f_j$ 
9:   while  $\mathcal{D}_2 \neq \emptyset$  do
10:     $\mathcal{S} := \mathcal{S}_{old}$  ▷ initialization
11:    for  $D_j \in \mathcal{D}_1$  do
12:      go through  $\tilde{M}$  and find the latest feasible departure time for  $D_j$  which satisfies
      scheduling constraints at each node along  $R_j$  and assign the available spot as  $c_v^j$  for all
       $v \in V(R_j)$ 
13:      if a feasible departure time  $\delta_j$  is found then
14:         $\mathcal{S} = \mathcal{S}_{old} \cup \{\delta_j\}$ ; update  $\tilde{M}$ 
15:      else
16:        return  $\mathcal{S}_{old}$  ▷ No feasible schedule found
17:      end if
18:    end for
19:     $\mathcal{S}_2 := \text{PREPARE\_SCHEDULE}(\mathcal{D}_2)$ 
20:    if  $\mathcal{S}_2 \neq \emptyset$  then
21:       $\mathcal{J}_a :=$  index set of  $\mathcal{D}_2$ 
22:       $f_{v,c}^{node} := \max_{j \in \mathcal{J}} f_j + w, \forall v \in \mathcal{V}, 1 \leq c \leq C_v$ 
23:      for  $\mathcal{J}_a \neq \emptyset$  do
24:         $j := \arg \max_{j' \in \mathcal{J}_a} \sup \mathcal{M}_{k_{R_{j'}}, v}^{j'}$ 
25:         $c_v^j := \arg \max_{c=1, \dots, C_v} f_{v,c}^{node} \quad \forall v \in V(R_j)$ 
26:         $f_{v,c_v^j}^{node} := \inf \mathcal{M}_v^j \quad \forall v \in V(R_j)$ 
27:         $\mathcal{J}_a := \mathcal{J}_a \setminus \{j\}$ 
28:      end for
29:       $\mathcal{S} = \mathcal{S} \cup \mathcal{S}_2$ 
30:      return  $\mathcal{S}$ 
31:    else
32:      remove the first journey  $D_j \in \mathcal{D}_2$  and append to the head of  $\mathcal{D}_1$ 
33:    end if
34:  end while
35:  return  $\mathcal{S}_{old}$  ▷ no feasible schedule found
36: end function

```

Algorithm 1 seeks a feasible partial or complete schedule minimizing the *Sum of Difference (SoD)* cost

$$\text{SoD} = \sum_{j \in \hat{\mathcal{J}}} (f_j - \delta_j) \quad (7.1)$$

where $\hat{\mathcal{J}}$ is the index set for the schedule. We refer to this cost as the SoD cost since it is the sum of the difference between arrival time and departure time for the flights. This cost corresponds to the typical preferences of customers to leave no earlier than necessary while still guaranteeing arrival by a desired deadline. An immediate lower bound for the SoD cost is obtained by considering worst case travel times along the routes for all demands $\hat{\mathcal{J}}$, ignoring capacity constraints at nodes.

With relatively few demands and a simple network topology, e.g., a star network as considered in Section 6.2.2, the exact optimal schedule minimizing (7.1) can be obtained from a mixed-integer program. However, for general acyclic directed graphs and/or a large set of demands, obtaining a schedule from a mixed-integer program is not computationally tractable. We next propose an efficient but possibly suboptimal approach for these cases in Algorithm 2 – 4.

Algorithm 2 considers a set of demands that need to be scheduled and partitions them into two sets according to their deadlines and existing schedule. To achieve this, the algorithm first computes the time intervals that flights will potentially occupy each parking spot of each node according to (6.2) and the parking spot assignment $\mathcal{C}(t)$. This is time-varying since it depends on time-varying arrivals $A_j(t)$, and we let $\tilde{M}(t)$ collect all of the blocked time intervals computed by (6.2) for all journeys at all parking spots of all nodes at time t . When a flight lands at a node along its route, the time interval that the flight will potentially occupy any node along the rest of its route reduces since the uncertainty of traveling is reduced, so that we can update $A(t)$ and $\tilde{M}(t)$ accordingly. When journeys are assigned to new demands at time t , $\tilde{M}(t)$ is used to avoid parking spot conflicts. Using $\tilde{M}(t)$, Algorithm 2 divides demands into those with earlier deadlines, \mathcal{D}_1 , which are inserted into

the current schedule if possible while the demands with later deadlines, \mathcal{D}_2 , are scheduled using Algorithm 3 beyond the latest time of the existing schedule. If a feasible complete schedule for \mathcal{D}_2 cannot be found, the demand with the earliest deadline is moved from \mathcal{D}_2 to \mathcal{D}_1 and the process repeats.

Algorithm 3 prepares the variables for Algorithm 4, picks the best solution returned from Algorithm 4, and transforms that into a schedule. Algorithm 4 is a branch-and-bound algorithm that finds a list of possible schedules for the given demands. While the algorithm is inspired by the classical Bratley's algorithm for task scheduling [44], several adaptations have been made for the problem we are addressing. For instance, Bratley's algorithm generally seeks a single schedule with the earliest possible departure times, while Algorithm 4 returns a set of possible departure times aiming to minimize the SoD cost (7.1).

Algorithm 4 also employs a pruning technique different than Bratley's algorithm. Consider the set of candidate schedules as a rooted tree. The algorithm starts from scheduling the last journey for the unassigned demands by visiting the demands in descending order according to their deadlines, picking a demand, recording the latest possible departure time and removing the demand from the unassigned list. The process is repeated until all demands are assigned or the branch is stopped by the pruning mechanism. If all demands are assigned, the current branch will be stored as a possible schedule and the algorithm will continue with another branch until all branches are considered (visited or pruned). The algorithm then returns all stored schedules.

We suggest several pruning mechanisms to avoid an exhaustive search, with the first two inspired by Bratley's Algorithm:

1. If an unassigned demand cannot be scheduled according to the assigned schedules without considering other unassigned demands, the current branch will be discarded.
2. If any node cannot afford the unassigned demands according to Lemma 8 with the assigned schedules, the current branch will be pruned.

3. If the demand with latest deadline can be scheduled without interfering with any other demand, then it is scheduled as the last journey to arrive at the destination.
4. If two demands choose the same route, their order follows the order of their deadline, i.e., if $D_{j_1}, D_{j_2} \in \mathcal{D}_{in}$ and $R_{j_1} = R_{j_2}$, while $f_{j_1} < f_{j_2}$, then only the branches with D_{j_1} arriving earlier than D_{j_2} will be preserved.
5. When searching along a branch, the current scheduled demands are compared with stored schedules. If the SoD cost along the current branch must exceed that of any of the stored schedules, the current branch will be abandoned.

Even with the above pruning techniques, Algorithm 4 may still produce a large set of feasible schedules. Although there can be several branches to explore, it is possible to stop the search at any point, given that at least one branch has been found, to get a timely but sub-optimal solution.

We then explain Algorithm 3 and 4 in details. While Algorithm 4 is the main schedule-finding algorithm that provides a set of possible schedules, Algorithm 3 prepares the variables for Algorithm 4, picks the best solution returned from Algorithm 4, and transforms that into a schedule.

Algorithm 3 first computes $f_{j,v}$ by (6.13) for all $j \in \mathcal{J}$ and $v \in V(R_j)$, based on the set of unassigned demands $\mathcal{D}_0 = \{D_j\}_{j \in \mathcal{J}}$. Recall that $f_{j,v}$ is the latest time for the flight to arrive at node v if $v \neq 0^{R_j}$ and is the latest departure time if $v = 0^{R_j}$ without passing its deadline f_j . We initialize DDL , PT and RT as zero matrices of dimension $|\mathcal{J}| \times |\mathcal{V}|$. We let $DDL(j, v) = f_{j,v}$, $PT(j, v) = m_v^{R_j}$ and $RT(j, v) = \underline{\mathcal{M}}_v^j$ for all j and $v \in V(R_j) \setminus \{0^{R_j}\}$, and let $DDL(j, 0^{R_j}) = f_{j,0^{R_j}}$, so that DDL , PT , and RT represent the deadline for arrival/departure, possible time for occupying the nodes, and the shortest time for the flight to travel from origin to node v along its route, respectively. It then calls the function *schedule* in Algorithm 4 and picks the schedule that achieves the smallest SoD cost.

Algorithm 3 Prepare Schedule

```

1: function PREPARE_SCHEDULE( $\mathcal{D}_0 = \{D_j\}_{j \in \mathcal{J}}$ )
2:   Let  $DDL, PT, RT$  be three  $N_0$  by  $|\mathcal{V}|$  zero matrices.
3:   for  $j \in \mathcal{J}$  do
4:     for  $v \in R_j$  do
5:        $DDL(j, v) := f_{j,v}$   $\triangleright f_{j,v}$  from (6.13)
6:        $PT(j, v) := m_v^{R_j}$   $\triangleright m_v^{R_j}$  from (6.3)
7:        $RT(j, v) := \underline{\mathcal{M}}_v^j$   $\triangleright \underline{\mathcal{M}}_v^j$  from (6.2)
8:     end for
9:   end for
10:   $RID := \text{zeros}(N_0, 1), RdpT := \text{zeros}(N_0, 1), SID := \emptyset, SdpT := \emptyset$   $\triangleright$  Initialize
11:   $(SID, SdpT) := \text{SCHEDULE}(\mathcal{D}_0, SID, SdpT, RID, RdpT, |\mathcal{D}_0|, DDL, PT, RT)$ 
12:   $\mathcal{S}_{new} := \emptyset$ 
13:  if  $SdpT \neq \emptyset$  then
14:     $i^* := \arg \max_i \sum_j SdpT(j, i)$   $\triangleright i^*$  index of the optimal schedule
15:    Let  $\mathcal{S}_{new}$  be the new schedule, with departure times  $SdpT(:, i^*)$  ordered as  $SID(:, i^*)$ 
16:  end if
17:  return  $\mathcal{S}_{new}$ 
18: end function

```

Algorithm 4 is a branch-and-bound algorithm that considers the set of candidate schedules as a rooted tree. Given the unassigned demands with ascending deadlines $\mathcal{D}_0 = \{D_j = (R_j, f_j)\}_{j \in \mathcal{J}_0}$, the algorithm visits the unassigned demands in descending order according to f_j to pick the last journey in the schedule. $RdpT$ records the departure time while RID records the index of the picked demand as in line 18.

We then remove the demand from the unassigned list and delete the row of the corresponding demand from DDL, PT and RT before continuing with the current branch. If the set of demands $\mathcal{D}_1 = \{D_j\}_{j \in \mathcal{J}_1}$, where $\mathcal{J}_1 \subseteq \mathcal{J}_0$, is already scheduled along the current branch, then for each node v , we can then compute $f_{v,c}^{node}$ as in Algorithm 2 line 21–28 for all $v \in \mathcal{V}$ and $c = 1, \dots, C_v$ by substituting $\mathcal{J}_a = \mathcal{J}_1$ with line 21. We implicitly assume that we will assign the journey to a parking spot where the earliest arrival time of the next journey at the same spot will be the latest among the C_v parking spots for all assigned demands. We let $f_v^{node} = \max_{1 \leq c \leq C_v} f_{v,c}^{node}$ for all $v \in \mathcal{V} \setminus S$ and let $f'_{j,v} = \min(DDL(j, v), f_v^{node})$ for

all $j \in \mathcal{J}_0 \setminus \mathcal{J}_1$ and all $v \in V(R_j)$. We then update DDL so that

$$DDL(j, 0^{R_j}) = \min_{v \in V(R_j) \setminus \{0\}} (f'_{j,v} - \overline{\mathcal{M}}_v^j) \quad (7.2)$$

and

$$DDL(j, v) = DDL(j, 0^{R_j}) + \overline{\mathcal{M}}_v^j. \quad (7.3)$$

The search-and-assign process is repeated until all demands are assigned or the branch is stopped by the pruning mechanism, which will be described later. If all demands are assigned, then the current branch, RID and $RdpT$, will be stored into SID and $SdpT$ as in line 13. The algorithm will then continue with another branch until all branches are considered (visited or pruned). The algorithm then returns SID and $SdpT$. As mentioned previously, we have five main pruning conditions, while the first, second and the fourth conditions are easy to be realized by line 6, 8 and 31, respectively. We next explain the third and the fifth conditions in detail.

The third condition indicates that if the demand with latest deadline can be scheduled without interfering with any other demand, then we schedule it as the last journey to arrive at the destination. The following explains line 16 in Algorithm 4. Let $j_0 = \arg \max_{j \in \mathcal{J}_0 \setminus \mathcal{J}_1} f_j$. For each $v \in V(R_{j_0})$, the number of unassigned demands that need to be considered is

$$num = \min \left(C_v, |\{D_j\}_{j \in \mathcal{J}_0 \setminus \mathcal{J}_1 | v \in V(R_{j_0})}| - 1 \right). \quad (7.4)$$

By our parking-spot assigning assumption, if assigning D_{j_0} as the last journey among all the unassigned demands will not interfere with any other demand, then

$$\max^k([f_{v,1}^{node}, \dots, f_{v,C_v}^{node}]) \geq \max^{k+1}(DDL(:, v)) \quad (7.5)$$

for all $v \in V(R_{j_0})$ and $k = 1, \dots, num$, where we let $\max^k(\cdot)$ represents the k 'th largest

Algorithm 4 Schedule

```
1: function SCHEDULE( $\{D_{in,j}\}_{j \in \mathcal{J}}$ ,  $SID$ ,  $SdpT$ ,  $RID$ ,  $RdpT$ ,  $N_0$ ,  $DDL$ ,  $PT$ ,  $RT$ )
2:    $\mathcal{D}_{in} = \{D_{in,j} = (R_{in,j}, f_{in,j})\}_{j \in \mathcal{J}}$ ,  $N := |\mathcal{D}_{in}|$ 
3:    $\mathcal{D} = \{D_j = (R_j, f_j)\}_{j=1}^N :=$  the sorted sequence of  $\mathcal{D}_{in}$ , so that  $\mathcal{D} = \mathcal{D}_{in}$  and
    $f_{j_1} \leq f_{j_2}$  if  $j_1 \leq j_2$ 
4:    $K := \text{zeros}(N, 1)$ 
5:    $K(j) := j'$  if and only if  $D_j = D_{in,j'}$ 
6:   if  $\exists j, v$  that  $DDL(j, v) < 0$  then
7:     return  $SID, SdpT$ 
8:   else if  $\exists v$  that  $\sum_j PT(j, v) < (\max_j DDL(j, v) - \min_j RT(j, v))$  then
9:     return  $SID, SdpT$ 
10:  else if  $N == 1$  then
11:     $dpT := DDL(N, 0^{R_N})$ 
12:     $RID(1) := K(1)$ ,  $RdpT(1) := dpT$ 
13:     $SID := [SID, RID]$ ,  $SdpT := [SdpT, RdpT]$ 
14:    return  $SID, SdpT$ 
15:  else
16:    if  $D_N$  can arrive at the each node at last without requiring any other flight to depart
    earlier then ▷ (7.5)
17:       $i := N$ 
18:       $RID(N) := K(i)$ ,  $RdpT(N) := DDL(i, 0^{R_i})$ 
19:       $\mathcal{D}_2 := \{D_j\}_{j \neq i}$ ,  $DDL_2 := DDL$ ,  $DDL_2(i, :) := []$ 
20:       $PT_2 := PT$ ,  $PT_2(i, :) := []$ ,  $RT_2 := RT$ ,  $RT_2(i, :) := []$ 
21:      update  $DDL_2$  according to (7.2) and (7.3)
22:      compare  $SID, SdpT$  with  $\mathcal{D}_2$ 
23:      if condition in (7.6) is satisfied then
24:         $RID(1 : N - 1) := SID_2$ ,  $RdpT(1 : N - 1) := SdpT_2$ 
25:         $SID := [SID, RID]$ ,  $SdpT := [SdpT, RdpT]$ 
26:      else
27:         $[SID, SdpT] :=$  SCHEDULE( $\mathcal{D}_2, SID,$ 
 $SdpT, RID, RdpT, N_0, DDL_2, PT_2, RT_2)$ 
28:      end if
29:    else
30:      for  $i \in \{N, \dots, 1\}$  do
31:        if  $\forall j$  such that  $R_j = R_i, f_j \leq f_i$  then
32:          same as line 18 – 28
33:        end if
34:      end for
35:    end if
36:  end if
37:  return  $SID, SdpT$ 
38: end function
```

number in the vector (\cdot) .

We now demonstrate how we can realize the last pruning condition with lines 22 and 23. When searching along a branch, we will compare the current scheduled demands with stored assignments in SID and $SdpT$. If the expected SoD cost along the current branch already exceeds any of the stored branches, the current branch will be abandoned. Consider the current unassigned demands $\{D_j\}_{j \in \mathcal{J}_0 \setminus \mathcal{J}_1}$. If $SID(1 : |\mathcal{J}_0 \setminus \mathcal{J}_1|, col) = \mathcal{J}_0 \setminus \mathcal{J}_1$ for some $col \in \mathbb{N}$, we then compute $f_{v,c}^{node}$ for all $v \in V(R_j)$ and $c = 1, \dots, C_v$. Similarly, we can use $SID(:, col)$ and $SdpT(:, col)$ to obtain $\{\delta_j\}_{j \in \mathcal{J}_1}$ and compute $f_{v,c}^{node,S}$ for all $v \in V(R_j)$ and $c = 1, \dots, C_v$. If

$$\max^k([f_{v,1}^{node}, \dots, f_{v,C_v}^{node}]) = \max^k([f_{v,1}^{node,S}, \dots, f_{v,C_v}^{node,S}]) \quad (7.6)$$

for all $v \in V(R_j)$ and $k = 1, \dots, num$, where num can be computed as in (7.4), is satisfied, then we can stop searching the current branch and instead use the stored schedule, as initiated on line 23.

If more than one col schedule satisfies (7.6), we pick the one with the least SoD cost, col_{min} . We let $SID_2 = SID(1 : |\mathcal{J}_0 \setminus \mathcal{J}_1|, col_{min})$ and $SdpT_2 = SdpT(1 : |\mathcal{J}_0 \setminus \mathcal{J}_1|, col_{min})$. We then merge $SID_2/SdpT_2$ with $RID/RdpT$ as in line 24–25 and stop searching this branch.

7.2 Numerical Study for Dynamic Scheduling Algorithm

In this section, we demonstrate our algorithm on two case studies with up to 200 flights each. We first demonstrate the algorithm on an eight-node network with dynamic scheduling updates. In the second case study, we consider a static scheduling example for a network of the Atlanta region in Chapter 6. For this case, we show that the proposed algorithm generates a schedule with a nearly optimal cost in considerably less computational time than the exact integer program proposed in Section 6.3. In both case studies, the proposed algorithm obtained reasonable schedules within 1 second.

7.2.1 Case Study 1: Dynamic Scheduling

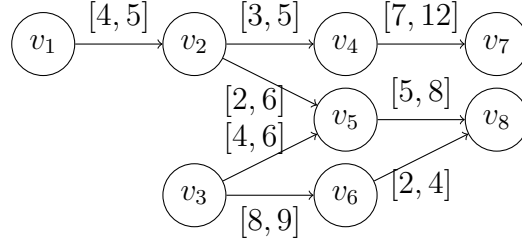


Figure 7.1: An 8-node, 8-link network used to illustrate the case study in 7.2.1.

The first case study considers the network in Fig. 7.1. The set of all possible origins (resp., destinations) is $S = \{v_1, v_3\}$ (resp., $T = \{v_7, v_8\}$). We assume the origins v_1, v_3 do not have capacity constraints, while $C_{v_2} = 1$, $C_{v_4} = 2$, $C_{v_5} = 3$, $C_{v_6} = 1$, $C_{v_7} = 3$ and $C_{v_8} = 5$. The links are indicated in the figure and the corresponding travel time intervals are labeled beside the links. We consider four routes $\mathcal{R} = \{R^1, R^2, R^3, R^4\}$ with $R^1 = \{(v_1, v_2), (v_2, v_4), (v_4, v_7)\}$, $R^2 = \{(v_1, v_2), (v_2, v_5), (v_5, v_8)\}$, $R^3 = \{(v_3, v_5), (v_5, v_8)\}$ and $R^4 = \{(v_3, v_6), (v_6, v_8)\}$. Each flight remains at the vertistops along its path for $w = 1$ minute after landing (all times are given in minutes). Algorithms 1 – 4 are implemented in MATLAB to obtain a schedule $\mathcal{S} = \{(\delta_j)\}_{j \in \mathcal{J}}$.

In simulation, realized travel times are uniformly randomly drawn from the travel time intervals. We first consider 43 randomly generated demands. Subsets of demands become available for scheduling across scheduling times $\mathcal{T} = \{0, 15, 30, 45, 50, 60, 80, 85, 100, 120\}$.

Fig. 7.2 demonstrates the resulting schedule. The figure compares the scheduled and actual arrivals at node 8 in the time interval $[100, 160]$ minutes as observed at time $t = 100$ minutes (left) and $t = 120$ minutes (right). The green, orange and blue bars represent the reserved time slots for flights traveling through R^2 , R^3 and R^4 , respectively. Flights on route R^1 are not shown because R^1 does not go through node 8. We label an ID number above each bar to identify the flights. The diamond represents the arrival deadline of a demand; the solid-color bar represents the time interval that the flight could possibly arrive given the schedule and uncertain travel times. The flight will then stay at the node for time

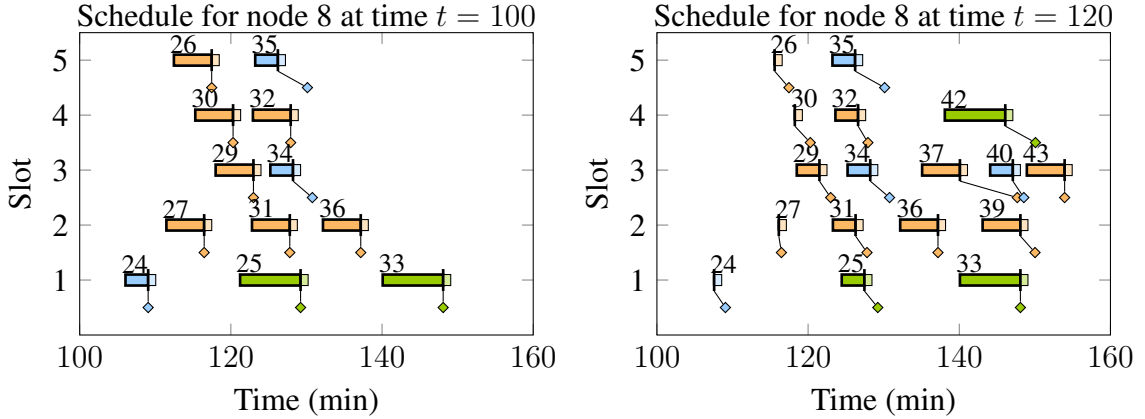


Figure 7.2: Scheduled/realized occupation of the five parking spots at Node 8 for the case study in Section 7.2.1 in the time interval $[100, 160]$ determined at time 100 (left) and time 120 (right). The green, orange and blue bars represent the reserved time slots for flights traveling through R^2 , R^3 and R^4 , respectively. An ID number above each bar identifies the flights. The diamond represents the arrival deadline of a journey; the solid-color bar represents the time interval that the flight is scheduled to arrive. The flight stays at the node for time $w = 1$ after arrival (represented by the lightly shaded bar). Therefore, the entire bar represents the time interval the flight may or did appear at node 8. Comparing the top and the bottom figures, we can observe several representative changes: the journey 30 has not yet arrived at the destination in the top plot computed at time 100 but is completed in the bottom plot computed at time 120; the time slot reserved for journey 25 has shrunk because the flight arrived at an intermediate node in the time interval $[100, 120]$ so that the uncertainty of its arrival at node 8 reduced; journeys 37-43 are newly scheduled.

w after arrival, which is represented by the lightly shaded bars in the figure. Therefore, the entire bar represents the time interval the flight may appear at node 8 under best and worst case travel times. Comparing the left and the right figures, we observe several representative changes: the journey 30 has not yet arrived at destination by time 100 in the top plot but is completed by time 120 in the right plot; the time slot reserved for journey 25 has shrunk because the flight arrived at an intermediate node along its route in the time interval $[100, 120]$ minute so that the uncertainty of its arrival at node 8 reduces; and journeys 37–43 are newly scheduled in the right plot.

A schedule for the 43 flights described above is obtained in 0.8 seconds with a SoD cost of 759.3 minutes. By considering worst case travel times for all demands but not capacity constraints, the lower bound for the SoD cost is 746 minutes. Thus, the algorithm produces

a schedule that is at least within 2% of optimal, and since an exact optimal schedule is not computationally tractable, it is unknown exactly how far from optimal the obtained schedule is, or even if it is, in fact, optimal itself.

Next, we consider a larger set of demands with size 200. The routes are chosen randomly among the four routes and the deadlines are picked randomly in the interval [40, 1540] minutes. The algorithm finds its first feasible schedule at 0.2801 seconds with SoD cost equal to 3759.9 minutes, which is again within 2% of the lower bound of 3693 minutes.

7.2.2 Case Study 2: Atlanta Network

We next recall the Atlanta star-branch network shown in Fig. 6.2 that was presented in Section 6.2.4. As a reminder, we take Atlanta as node number 0 and the exurbs Alpharetta, Kennesaw and Buford 1, 2 and 3, respectively. We let $\mathcal{R} = \{R^1, R^2, R^3\}$, where $R^1 = \{(v_1, v_0)\}$, $R^2 = \{(v_2, a), (a, v_0)\}$, and $R^3 = \{(v_3, c), (c, b), (b, v_0)\}$. We consider demands $\mathcal{D} = \{(R_j, f_j)\}_{j \in \mathcal{J}}$ defined subsequently, where $R_j \in \{R^1, R^2, R^3\}$. Note that the destination for all demands is Atlanta, node v_0 . We consider a time horizon of three hours ($T = 180$ minutes) and assume there are two landing spots in Atlanta, but only one landing spot at vertistops a, b and c , i.e., $C_{v_0} = 2, C_a = C_b = C_c = 1$. Each flight stays at the intermediate vertistops along its path for $w_I = 1$ minute and at the Atlanta vertiport for $w = 5$ minutes after landing.

We use the same set of demands where the number of UAM demands across the three origins is $[h_1, h_2, h_3] = [4, 4, 19]$ and that arrival deadlines are set at regular intervals, i.e., if origin v_i is tasked with sending h_i flights to Atlanta, the deadlines are $\left\lfloor \frac{180}{1+h_i} \cdot k + 0.5 \right\rfloor$, $k = 1, 2, \dots, h_i$.

For comparison, the mixed-integer optimization problem in Section 6.3 is solved in MATLAB using Gurobi with YALMIP toolbox to obtain a schedule that exactly minimizes the SoD cost (7.1). Using the exact mixed-integer program from Section 6.3 takes around

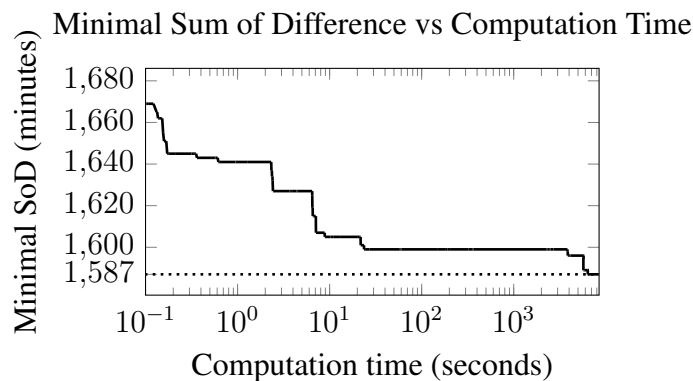


Figure 7.3: The computation time versus minimal SoD cost among all stored schedules obtained within the corresponding computation time for Case Study 2. The dashed line represents the optimal SoD computed from the optimal schedule obtained by the mixed-integer program.

150 minutes of computation time on a standard laptop, and the optimal SoD cost as in (7.1) is 1587 minutes. Note that the lower bound computed using worst cast travel time but ignoring capacity constraints is 1173 minutes. Thus, the capacity constraints at the nodes play a significant role in overall network SoD cost.

In contrast, the algorithm proposed in this chapter obtains a feasible schedule in 0.05 seconds. The SoD cost of this sub-optimal schedule is 1669 minutes. The SoD cost reduces to 1605 minutes when the algorithm finds an alternative schedule after 5.8 seconds of computation time. Fig. 7.3 demonstrates how the minimal SoD cost of all stored schedules decreases with computation time.

The exact mixed-integer program cannot practically compute a solution if the number of demands exceeds about 50. On the other hand, the scheduling algorithm in this paper provides a sub-optimal schedule quickly for even a large demand set. To demonstrate this, we now consider 200 demands assigned randomly to the three routes with random deadlines in the interval $[40, 1540]$ minutes. The algorithm finds the first feasible schedule at 0.2 seconds with Sum of Difference equal to 12662 minutes and, after 10.1 seconds, obtains a schedule with Sum of Difference equal to 12637 minutes. The lower bound for this set of demands is 7786 minutes. We note, however, that this is only a lower bound and

it is likely that the optimal schedule has SoD cost significantly greater than this bound as was the case in the prior example with 27 flights.

7.3 Concluding Remarks

In this chapter, we studied a dynamic scheduling algorithm. We've come up with a mixed integer program for static scheduling in Section 6.3, which is too time-expensive when the complexity of network and the size of demands increase. An exact schedule can sometimes be obtained from a mixed integer program, but this is not computationally tractable for larger networks and/or large sets of demands. Instead, we present a dynamic scheduling algorithm that is able to consider new demands over time and uses branch-and-bound heuristics to identify feasible but possibly sub-optimal schedules in this chapter.

CHAPTER 8

SAFETY VERIFICATION FOR UAM SCHEDULING: SIMPLEST CASE

For any UAM solution, unforeseen disruptions such as intermittent closure of landing sites needs to be considered for the sake of safety. It must be ensured that, once the network is disrupted, there will be enough landing spots available for all UAVs conforming to the emergency rerouting rules. In this chapter, we consider schedule disruptions within the model proposed in Chapter 6, which accounts for uncertain travel time and limited landing capacity, and we develop a safety verification algorithm for a given UAM schedule. In Chapter 6, the schedules are obtained so that the flights must travel through designated routes and meet their corresponding arrival deadlines at their destinations, while satisfying the landing-spot restrictions at the destination and intermediate nodes upon arrival. In this chapter, we consider the static UAM model, where the schedules are not updated over time, and we further consider the problem of verifying the safety of a given schedule upon the closure of a node in the network while only one backup landing site will be assigned to each flight in Section 8.1. We provide necessary and sufficient conditions for a given UAM schedule to be guaranteed as safe in the disrupted scenario, which leads to our proposed safety verification algorithm. We also provide necessary conditions for guaranteed safety under best-case travel time realization and demonstrate our verification algorithm and its computation efficiency on a UAM network with up to 1,000 UAVs.

The remainder of this chapter is organized as follows: In Section 8.1, we first recall the UAM network model in Chapter 6 with static settings, and then introduce the disruption model that reduces capacity of the network. In Section 8.2, we establish safety criteria and develop necessary and sufficient conditions for the schedule to be safe under disruptions. We then demonstrate our safety verification algorithm on a UAM network in Section 8.3 and compare the verification time for different sizes of schedules.

8.1 Problem Formulation

8.1.1 Network Model and Nominal Scheduling

In Definition 5, the UAM network are defined with network graph, limited node capacities, routes, and minimum and maximum link travel times. We again assume that every flight is associated to a route $R \in \mathcal{R}$ and stops at intermediate nodes along the route. In the context of safety verification problem, we assume that each schedule assigned to a given demand satisfying the constraint of the deadline. Therefore, for each flight, we will only concern about the route of the corresponding demand, and the departure time. Therefore, we define *assured schedule* as a pair (R, δ) where $R \in \mathcal{R}$ and $\delta \in \mathbb{R}_+$ is the appointed departure time from the first node along the route. A *schedule profile* for a UAM network is a set $\mathcal{S}^p = \{(R_j, \delta_j)\}_{j \in \mathcal{J}}$. In this chapter, we assume that \mathcal{J} is a finite index set of *flights*, and hence, \mathcal{S}^p , is assumed finite. To be succinct, when we mention “schedule” in the context of verification for schedules in the rest of the thesis, it represents the assured schedule.

We recall that the latest arrival time at the node $\sigma(\ell)$ along the route is a_ℓ^j , and the time interval that the flight will potentially block a landing spot at node ℓ is \mathcal{M}_ℓ^j , where a_ℓ^j , \mathcal{M}_ℓ^j can be computed through (6.1) and (6.2) in the static case starting from origin. Therefore,

$$a_\ell^j = \delta_j + \sum_{\ell_1=1}^{\ell} \bar{x}_{\ell_1} + (\ell - 1)w, \quad (8.1)$$

and

$$\mathcal{M}_\ell^j = \left[\delta_j + \sum_{\ell_1=1}^{\ell} x_{\ell_1} + (\ell - 1)w, a_\ell^j + w \right]. \quad (8.2)$$

We let $\mathcal{M}_v^j = \mathcal{M}_\ell^j$ and $a_v^j = a_\ell^j$ if $v \in V(R_j)$ and $v = \ell^{R_j}$.

Definition 9 (Realization). A **realization** of an assured schedule (R, δ) refers to a set of realizations for the travel times on each link along route R that obeys the minimum and maximum travel time limits. We do not assume any probability distribution on realizations, but we will consider certain conditions that hold in the worst-case, i.e., for all realizations,

or in the best-case, i.e., there exists some realization satisfying the condition.

The feasible schedule profile we are defining below is mirroring the schedule in Definition 6.

Definition 10 (Feasible Schedule Profile). *A schedule profile $\mathcal{S}^p = \{(R_j, \delta_j)\}_{j \in \mathcal{J}}$ where $\delta_j \in \mathbb{R}_+$ for all $j \in \mathcal{J}$ is a **feasible schedule profile** if, for all realizations (that is, in worst-case), the number of vehicles at a node never exceeds capacity, i.e., for all $v \in \mathcal{V}$ and all $t \geq 0$,*

$$\sum_{j:v \in V(R_j)} \mathbf{1}(t; \mathcal{M}_v^j) \leq C_v \quad (8.3)$$

where the notation $\mathbf{1}(\cdot; \cdot)$ is the indicator function.

While every feasible schedule profile will by definition ensure proper operation of the UAM network under normal circumstances, our objective is to ensure the schedule is resilient to interruptions in the network.

8.1.2 Disruption Model

In actual operation, it is expected that unforeseen disruptions that disable one or more nodes, such as adverse weather conditions, will be common. The arrangement for the flights affected by the disabled nodes needs to be considered in advance to ensure safety, that is, when a node is disabled, each flight passing through the disabled node must have a rerouting plan that ensures availability of a landing spot. In this thesis, we postulate the existence of a set of *backup nodes* for the network so that when any node is disabled, the flights can be redirected to the backup nodes depending on the link they are traveling through.

In this subsection, we introduce the assignment of the backup nodes and the operating mechanism once a node is disabled. We assume that only one node may be disabled in the rest of the thesis. In order to guarantee that each disrupted flight will be able to be assigned to a node after the disruption, we assign a backup node to each link in the network.

Naturally, we let the backup node of the link be its head if the head node is not disabled, and let the backup node be the tail if the head node is disabled. Under this assignment of backup node, a flight whose route is potentially being blocked and is traveling on a link e will continue to the head node $\sigma(e)$ on its route if it is functioning, or return to the previous node $\tau(e)$ if the head node is disabled.

We say the j 'th flight is *affected* by some disabled node $v_c \in \mathcal{V}$ at time t_c if $v_c \in V(R_j)$, i.e., the route of the flight travels through node v_c , and the flight has not yet reached v_c by time t_c . The flight is *not affected* when the node v_c is disabled at time t_c otherwise.

Below is a set of rules that all flights need to follow once a node v_c is disabled at time t_c :

1. flights not affected will continue normal operation;
2. any affected flight that has not yet departed ($\delta_j > t_c$) will be canceled (no longer depart);
3. an affected flight j with $\delta_j \leq t_c$ traveling on a link $e \in \mathcal{E}$ with $\sigma(e) \neq v_c$ will continue to the head node $\sigma(e)$ and stop there indefinitely (block a landing spot indefinitely);
4. an affected flight j with $\delta_j \leq t_c$ that is landed at a node at time t_c will remain there indefinitely;
5. an affected flight j with $\delta_j \leq t_c$ traveling on a link $e \in \mathcal{E}$ with $\sigma(e) = v_c$ will return to the tail of the link $\tau(e)$ and stop there indefinitely.

Note that we do not consider the problem of recovering a new schedule after a disabled node becomes operational again, as our focus is on safety. Further, we postulate the above rules to provide a well-defined problem formulation; alternative rules might be also plausible.

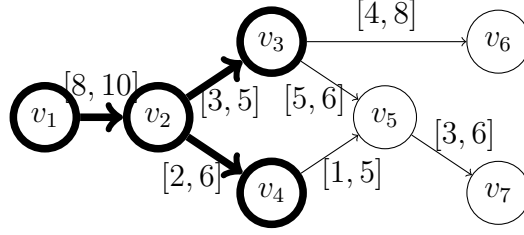


Figure 8.1: (Bold partial graph) The sub-graph consisted of the bold lined 4 nodes and 3 links is used to illustrate the simple network in Example 2.

(Entire graph) The entire graph is used to illustrate the network with 7 nodes and 7 links in the case study.

8.2 Necessary and Sufficient Conditions for Safe Schedule

In this section, we formally define safety and present necessary and sufficient conditions for safety. Given a network $\mathcal{N} = (\mathcal{G}, C, \mathcal{R}, \underline{x}, \bar{x}, w)$ where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a feasible schedule profile $\mathcal{S}^p = \{(R_j, \delta_j)\}_{j \in \mathcal{J}}$, we regard the j 'th flight as *possibly affected* by disabling node v_c at time t_c if $v_c \in V(R_j)$ and $t_c < \sup \mathcal{M}_{v_c}^j$, i.e., the flight may have to travel through the disabled node later than t_c , depending on realized travel times. The closure of a node can affect the set of schedules in different ways. In particular, the set of schedules is:

1. *worst-case (resp., best-case) time-node conditionally safe* for node v_c and time t_c if, supposing that v_c is disabled at time t_c , then all possibly affected flights are able to land at their designated backup nodes while not interfering with any unaffected flights, for all (resp., for some) realization of link travel times.
2. *worst-case (resp. best-case) node conditionally safe* for node v_c if it is worst-case (resp. best-case) time-node conditionally safe for node v_c for all time $t_c \geq 0$.
3. *worst-case (resp. best-case) 1-closure safe* if it is worst-case (resp. best-case) node conditionally safe for any node $v_c \in \mathcal{V}$.

Note that worst-case safety implies best-case safety.

Example 2. We illustrate the model and the safe criteria through the simple network shown in Fig. 8.1 with 4 nodes and 3 links (the bold lined sub-graph). For this example, the set of

nodes $\mathcal{V} = \{v_1, v_2, v_3, v_4\}$ and $\mathcal{E} = \{(v_1, v_2), (v_2, v_3), (v_2, v_4)\}$. We assume that the origin v_1 does not have a capacity constraint, while $C_{v_2} = 2$, $C_{v_3} = 1$ and $C_{v_4} = 1$. The links are indicated in the figure and the corresponding travel time intervals are labeled beside the links, e.g., the interval $[8, 10]$ above the link (v_1, v_2) means that the shortest (resp., longest) possible time for traveling through the link is 8 (resp., 10) time units. We consider two possible routes $R^1 = \{(v_1, v_2), (v_2, v_3)\}$ and $R^2 = \{(v_1, v_2), (v_2, v_4)\}$. Each flight remains at the intermediate nodes or destination along its path for $w = 1$ time unit after landing. Given a feasible schedule profile $\mathcal{S}^p = \{S_1, S_2, S_3\}$, where $S_1 = (R^2, 1)$, $S_2 = (R^2, 8)$ and $S_3 = (R^1, \delta_3)$, where we consider several δ_3 . Assume v_4 is disabled at time $t_c = 15$. Under worst-case travel time realizations, flight S_1 will be traveling on (v_2, v_4) at $t_c = 15$ and needs to be rerouted to v_2 ; Flight S_2 will be traveling on (v_1, v_2) and needs to stay at v_2 upon arrival. If $\delta_3 = 0$, flight S_3 is not affected, and should continue its journey; however, if $\delta_3 = 10$, then v_2 will be short of landing spots upon the arrival of S_3 . Therefore, \mathcal{S} is worst-case time-node conditionally safe for node v_4 at time 15 if $\delta_3 \leq 4$ and is not if $\delta_3 > 4$. In contrast, it is always best-case time-node conditionally safe for node v_4 at time 15 regardless of the choice of δ_3 . Meanwhile, suppose $C_{v_2} = 3$, and we let $\delta_3 = 10$, then obviously, the network will be able to handle the flights after closure no matter when the node v_4 is closed. Therefore, we see that \mathcal{S} is worst-case node conditionally safe for node v_4 in this case. We further check that this is true for all nodes in the network, and thus \mathcal{S} is also worst-case 1-closure safe. \square

To obtain conditions for 1-closure safety, we start by observing that a feasible schedule is trivially node conditionally safe for node $v \in S$, where we recall S the set of source nodes that are not the head of any link.

We next explore safety of a disabled node not in S . Before we study the sufficient and necessary conditions for 1-closure safety when disabling a node $v_c \in \mathcal{V} \setminus S$, we first define several special sets. If $v_c = \ell^{R_j}$, we let $\ell_{v_c, R_j} = \ell$, and $\ell_{v_c, R_j} = 0$ if $v_c \notin V(R_j)$.

We let the set of links with head $v \in \mathcal{V}$ be

$$\mathcal{E}_v := \{e \in \mathcal{E} \mid \sigma(e) = v\}, \quad (8.4)$$

and let b_{e,v_c} be the node that any flight traveling on link e will proceed to if v_c is disabled:

$$b_{e,v_c} = \begin{cases} \sigma(e) & \text{if } \sigma(e) \neq v_c, \\ \tau(e) & \text{if } \sigma(e) = v_c. \end{cases} \quad (8.5)$$

We denote the set of links on which flights will be rerouted to node v when v_c is disabled as

$$\mathcal{B}_{v,v_c} := \{e \in \mathcal{E} \mid b_{e,v_c} = v\}. \quad (8.6)$$

We define the set \mathcal{J}_v as the index set of the flights with routes passing through the node v ,

$$\mathcal{J}_v := \{j \in \mathcal{J} \mid v \in V(R_j)\}, \quad (8.7)$$

and we further define the index set of the flights that might possibly be landed at v after time t as

$$\mathcal{J}_v^*(t) := \{j \in \mathcal{J}_v \mid a_v^j + w > t\}. \quad (8.8)$$

Therefore, the index set of the possibly affected flights when node v_c is closed at time t_c is $\mathcal{J}_{v_c}^*(t_c)$, while the index set for the flights passing through the node v that are not possibly affected when the node v_c is closed at t_c is $\mathcal{J}_p(v, t_c, v_c) = \mathcal{J}_v \setminus \mathcal{J}_{v_c}^*(t_c)$.

We use $\mathcal{J}_{c,v_c}(t_c)$ to represent the set of indices for canceled flights with departure time greater than the closing time t_c :

$$\mathcal{J}_{c,v_c}(t_c) := \{j \in \mathcal{J}_{v_c} \mid \delta_j > t_c\}. \quad (8.9)$$

We then define the index set of rerouting flights $\mathcal{J}_{v_c}^{*\setminus c}(t_c)$ as the possibly affected flights whose not canceled when node v_c is disabled at time t_c , i.e.,

$$\mathcal{J}_{v_c}^{*\setminus c}(t_c) = \mathcal{J}_{v_c}^*(t_c) \setminus \mathcal{J}_{c,v_c}(t_c). \quad (8.10)$$

We let $N_R(v, t_c, v_c)$ be the maximum number of flights that might possibly be landed at node v at the same time once the node v_c is closed at time t_c , which can be computed as

$$N_R(v, t_c, v_c) = \sup_{t \geq t_c} \sum_{j \in \mathcal{J}_p(v, t_c, v_c)} \mathbf{1}(t; \mathcal{M}_v^j). \quad (8.11)$$

In the rest of the thesis, we sometimes drop the notations in the parentheses, t, t_c, v, v_c , when they are clear from the context.

Theorem 8. Consider a network $\mathcal{N} = (\mathcal{G}, C, \mathcal{R}, \underline{x}, \bar{x}, w)$, where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Assume given a feasible schedule profile $\mathcal{S}^p = \{(R_j, \delta_j)\}_{j \in \mathcal{J}}$. Define $N_e(t_c, v_c)$ as the number of flights on link e that stay at b_{e,v_c} if node v_c is disabled at time t_c for all $e \in \mathcal{E}$, i.e.,

$$N_e(t_c, v_c) = \begin{cases} \sum_{j \in \mathcal{J}_{v_c}^{*\setminus c}(t_c)} \mathbf{1}(t_c; [L_e^j, U_e^j]) & \text{if } e \notin \mathcal{E}_{v_c} \\ \sum_{j \in \mathcal{J}_{v_c}^{*\setminus c}(t_c)} \mathbf{1}(t_c; [L_e^j, U_e^j] \setminus [L_{(\ell_{v_c, R_j - 1})^{R_j}}^j, U_{(\ell_{v_c, R_j - 1})^{R_j}}^j]) & \text{if } e \in \mathcal{E}_{v_c} \end{cases} \quad (8.12)$$

where we define the lower and upper bounds of the time interval as

$$L_e^j = \begin{cases} \inf\{\mathcal{M}_{\tau(e)}^j\} + w & \text{if } \tau(e) \neq 0^{R_j} \\ \delta_j & \text{if } \tau(e) = 0^{R_j} \end{cases} \quad (8.13)$$

and

$$U_e^j = \begin{cases} \sup\{\mathcal{M}_{\sigma(e)}^j\} & \text{if } \sigma(e) \neq v_c \\ \sup\{\mathcal{M}_{\sigma(e)}^j\} - w & \text{if } \sigma(e) = v_c. \end{cases} \quad (8.14)$$

Further define $N_v(t_c, v_c)$ as the number of possibly affected flights that may block the node v indefinitely, i.e., $N_v(t_c, v_c) = \sum_{e \in \mathcal{B}_{v, v_c}} N_e(t_c, v_c)$.

Then \mathcal{S}^p is worst-case time-node conditionally safe for node v_c and time t_c if and only if, for all $v \in \mathcal{V}$,

$$C_v - N_v(t_c, v_c) \geq N_R(v, t_c, v_c). \quad (8.15)$$

Further, \mathcal{S}^p is worst-case node conditionally safe for node v_c if and only if (8.15) holds for the finite number of times t_c where the values of $N_v(t_c, v_c)$ and $N_R(v, t_c, v_c)$ possibly change, i.e., at both endpoints of the interval \mathcal{M}_v^j for all $v \in \mathcal{V}$ and at times L_e^j, U_e^j for all $e \in \mathcal{B}_{v, v_c}$.

Proof. To guarantee the schedule profile is time-node conditionally safe for node v_c and time t_c , for any possibly affected flight that is not canceled and may be rerouted to some node $v \in \mathcal{V}$ at time t_c , a landing spot needs to be reserved. Then the problem becomes to ensure the flights surely not affected will have no capacity conflict with any possibly rerouted flights. We then consider the maximum (worst-case) occupation of the node v .

The interval defined as $[L_e^j, U_e^j]$ is the time interval during which flight j might possibly be rerouted to b_{e, v_c} if v_c is closed since the lower bound L_e^j is the earliest time that the flight may leave the previous node $\tau(e)$, and, if $\sigma(e)$ is not disabled, the upper bound U_e^j is the latest time that the flight may leave the head node while, in the case that $\sigma(e)$ is disabled, the upper bound U_e^j for the time interval that the flight may be rerouted to the backup node $\tau(e)$ will be the latest time that the corresponding flight may arrive at the node v_c , since otherwise it will continue its journey without rerouting.

Therefore, $N_e(t_c, v_c)$ in (8.12) is the number of possibly affected flights that may be

rerouted to b_{e,v_c} and will not be canceled. $N_v(t_c, v_c)$ is the number of landing spots needed to be reserved for the rerouted flights. Finally, $N_R(v, t_c, v_c)$ is the maximum number of flights not possibly affected that may park at the node v at any time once v_c is closed at t_c . Therefore (8.15) is a necessary and sufficient condition to avoid the landing-spot conflict between the rerouted flights and those not possibly affected for all realization of link travel times. \square

Theorem 8 provides a finite number of conditions to verify a schedule profile is worst-case node conditionally safe for node v_c . Furthermore, by checking that a schedule profile is node conditionally safe for all $v_c \in \mathcal{V}$, we can conclude the worst-case 1-closure safety. The same checking technique will be applied to the necessary condition below for best-case safety in Theorem 9, which is obtained by observing the best scenario case where we assume that all flights possible to have arrived at or passed the closed node v_c has already arrived or left by the time of failure. We denote the index set of the possibly affected flights that may have arrived at or passed the node v_c disabled at t_c as \mathcal{J}_{v_c}'' , and

$$\mathcal{J}_{v_c}''(t_c) = \{j \in \mathcal{J}_{v_c}^* \mid \inf\{\mathcal{M}_{v_c}^j\} \leq t_c\}. \quad (8.16)$$

Theorem 9. *Consider a network $\mathcal{N} = (\mathcal{G}, C, \mathcal{R}, \underline{x}, \bar{x}, w)$, where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. A feasible schedule profile $\mathcal{S}^p = \{(R_j, \delta_j)\}_{j \in \mathcal{J}}$ is best-case time-node conditionally safe for node v_c and time t_c only if there exists a set of non-negative integers $\{N_v\}_{v \in \mathcal{V}}$ that satisfies*

$$\sum_{v \in \mathcal{V}} N_v = |\mathcal{J}_{v_c}^*(t_c) \setminus \{\mathcal{J}_c(t_c) \cup \mathcal{J}_{v_c}''(t_c)\}| \quad (8.17)$$

$$C_v - N_v \geq \sup_{t \geq t_c} \sum_{j \in \mathcal{J}_v \setminus \mathcal{J}_{v_c}^*(t_c)} \mathbf{1}(t; \mathcal{M}_v^j) \quad \forall v \in \mathcal{V}. \quad (8.18)$$

Further, \mathcal{S}^p is best-case node conditionally safe for node v_c only if, for the finite number of times t_c where the time-varying index sets $\mathcal{J}_{v_c}^(t_c)$, $\mathcal{J}_{c,v_c}(t_c)$ and $\mathcal{J}_{v_c}''(t_c)$, and the endpoints of the interval \mathcal{M}_v^j for all $v \in \mathcal{V}$ and $j \in \mathcal{J}$ possibly change, there exists a set of non-*

negative integers $\{N_v\}_{v \in \mathcal{V}}$ that satisfies the constraints (8.17)–(8.18).

Proof. The proof of Theorem 9 applies the similar logic as in Theorem 8 to the best-case scenario. First of all, the set of all rerouted flights has to be the same as the possibly affected flights $\mathcal{J}_{v_c}^*$ except for the canceled flights, $\mathcal{J}_c(t_c)$, or the flights that are possibly not affected, \mathcal{J}_{v_c}'' , as depicted in (8.17). If we are not able to find a sequence of non-negative integers $\{N_v\}_{v \in \mathcal{V}}$ that satisfies (8.17)–(8.18), then there must exist a conflict of occupation at one or more nodes once v_c is closed at time t_c , and hence (8.17) and (8.18) are necessary conditions for the set of schedules to be best-case time-node conditionally safe for node v_c and time t_c . \square

Theorem 8 is both sufficient and necessary for worst-case 1-closure safety, while Theorem 9 is necessary for best-case 1-closure safety. Since worst-case safety implies best-case safety, satisfaction of the conditions in Theorem 8 implies satisfaction of the conditions in Theorem 9.

8.3 Case Study

In the case study, we first demonstrate the verification algorithm based on Theorem 8 on a UAM network with 7 nodes and 7 links as shown in Fig. 8.1 with a feasible schedule profile of size 20. We then show the relation between the verification time and the size of the schedules being examined through various sets of schedules with different sizes.

To ensure the safety for all realizations of link travel time, we check whether the condition of worst-case time-node conditional safety for node v_c and t_c is satisfied or not over the time interval $t_c \in [0, +\infty)$. As stated in Theorem 8, we only need to verify (8.15) at each point of time that any value may change, i.e., L_e^j , U_e^j and both ends of \mathcal{M}_v^j for any counted flight $j \in \mathcal{J}$ and link $e \in \mathcal{E}$ for some fixed node v , since the inequality (8.15) will not change between these points.

In the network in Fig. 8.1 (the entire graph), the set of all possible origins (resp., desti-

nations) is $S = \{v_1\}$ (resp., $T = \{v_6, v_7\}$). We assume the origin v_1 does not have capacity constraint, while $C_{v_2} = 8$, $C_{v_3} = 4$, $C_{v_4} = 2$, $C_{v_5} = 4$, $C_{v_6} = 3$ and $C_{v_7} = 5$. The links are indicated in the figure and the corresponding travel time intervals are labeled beside the links. We consider three routes $\mathcal{R} = \{R^1, R^2, R^3\}$ with $R^1 = \{(v_1, v_2), (v_2, v_3), (v_3, v_6)\}$, $R^2 = \{(v_1, v_2), (v_2, v_3), (v_3, v_5), (v_5, v_7)\}$ and $R^3 = \{(v_1, v_2), (v_2, v_4), (v_4, v_5), (v_5, v_7)\}$. Each flight remains at the verti-stops along its path for $w = 1$ time unit after landing. We randomly generate a particular feasible schedule profile with 20 flights and consider the inequality (8.15) in Theorem 8 for worst-case time-node conditionally safe for node $v_c = v_5$ and any time $t_c > 0$. The verification is implemented in MATLAB¹.

In Fig. 8.2, the lower white rectangles show the flights being rerouted to the node v_2 if the node v_5 is closed at time t_c , which is $N_{v_2}(t_c, v_5)$ in (8.15); the upper blue rectangles represent the number of flights not affected and continue to v_2 if the node v_5 is closed at time t_c , which is $N_R(v_2, t_c, v_5)$ in (8.15). As a reference, the capacity $C_{v_2} = 8$ is shown as the dotted, horizontal line so that if the height of the entire bar (the sum of both rectangles) exceeds the capacity, then Theorem 8 is not satisfied at time t_c . For example, the schedule in this case study is not worst-case node conditionally safe for node v_5 , as the node v_2 is not able to accommodate to the failure of v_5 in certain time intervals, e.g., $t_c \in [30, 35]$. Similarly, we can check if the other nodes are able to accommodate the failure of node v_5 for any t_c with the same technique. If for time $t_c > 0$, all the nodes are able to accommodate the failure of v_5 , then we would conclude that the schedule profile is worst-case time-node conditionally safe for node v_5 and time t_c . We can also observe the performance of the network and the schedules with the failure of any other node at any time.

The computation for $N_R(v, t_c, v_c)$ in (8.15) implies $O(n^2)$ computational complexity of this verification process. Therefore, we are also able to efficiently verify worst-case safety large schedule profiles. As an example, consider increasing the capacity for each node of the network in Fig. 8.1 by 10 (this does not change the verification time). We generate 10

¹The related MATLAB code can be found in https://github.com/gtfactslab/Wei_NecSys22.

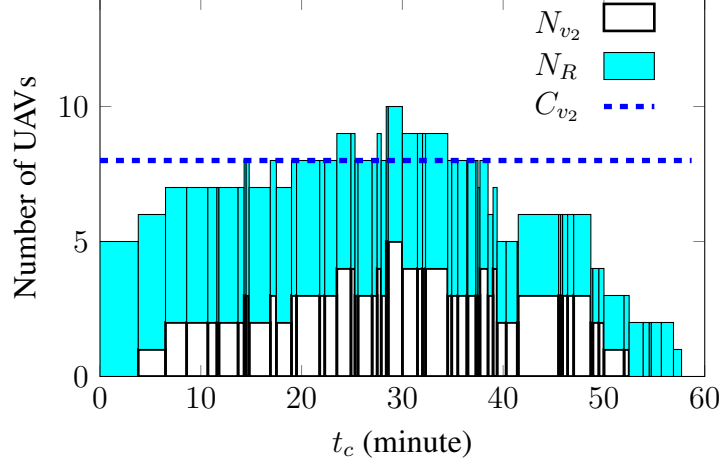


Figure 8.2: The number of flights rerouted to v_2 , $N_{v_2}(t_c, v_5)$ (simplified as N_{v_2}), represented by the upper blue rectangles and the maximum unaffected occupation at v_2 when the node v_5 is disabled at time t_c , $N_R(v_2, t_c, v_5)$ (simplified as N_R), represented by the lower white rectangles. The dotted line corresponds to the capacity at node v_2 .

more sets of random feasible schedules with sizes 100, 200, 300, \dots , 1000 and verify their safety using the same algorithm. Fig. 8.3 confirms the quadratic relation between the size of the set of schedules and the time it takes to check the safety and shows that we are able to verify safety or provide a counterexample for a schedule profile with 1,000 flights in under 30 seconds.

8.4 Concluding Remarks

In this chapter, we explored the safety verification problem for a UAM schedule in a disruption scenario in which a node must close and inbound flights are immediately rerouted. We provided a reasonable link-related backup-node assignment for the UAM network and rules for the flights after a node is disabled. The main impediment to safety is that each flight needs to be provided an available landing spot upon arrival of any intermediate, destination, or backup node. We therefore derived sufficient and necessary conditions for worst-case and best-case safety of a given schedule. These theoretical results provided an efficient safety verification algorithm. We demonstrated through case study that the com-

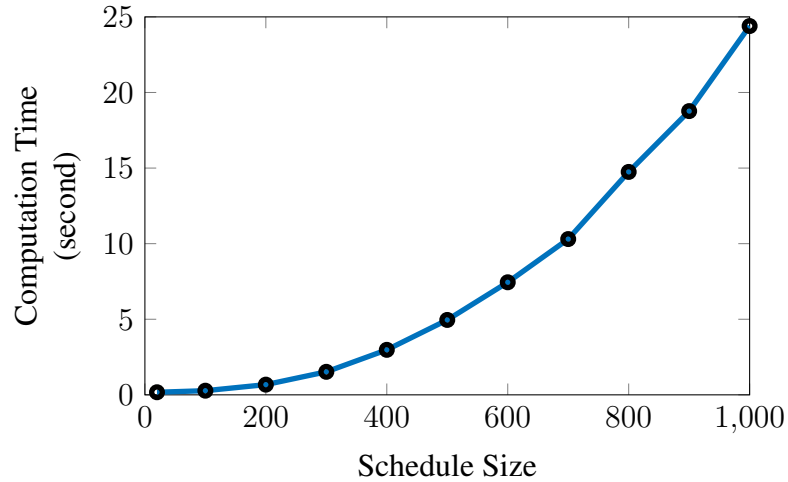


Figure 8.3: The computation time for verifying the worst-case safety of schedules with different sizes. We test on 11 different sets of schedules with sizes from 20 to 1000. The data points constitute a parabola, which demonstrates the $O(n^2)$ computational complexity.

putational time for the algorithm grows quadratically with schedule size. As a result, the safety verification algorithm is applicable to large-scale UAM scheduling problems.

We focused on the case where only one backup node is assigned to each flight in this chapter, while in the next chapter we are going to explore the more general case where multiple backup nodes are considered.

CHAPTER 9

SAFETY VERIFICATION FOR UAM SCHEDULING: GENERAL CASE

In Chapter 8 we studied the safety verification problem for a UAM schedule in a disruption scenario in which a node is disabled and inbound flights are immediately rerouted. We provide a reasonable link-related backup-node assignment for the UAM network where each link is assigned a backup node. In this chapter, we are considering the more general case where multiple backup nodes can be assigned to a link.

9.1 Model Extension with Multiple Backup Nodes

In this section, we revise the assignment of the backup nodes and the operating mechanism once a node is disabled. Similar to Chapter 8, we consider that only one node may be disabled at a time. In order to guarantee that each disrupted flight will be able to be assigned to a node after the disruption, in this chapter, we assign a set of backup nodes \mathcal{B}_e to each link e in the network. We make a natural assumption that the set of backup nodes for any link includes its head node and tail node, i.e., $\tau(e), \sigma(e) \in \mathcal{B}_e$ for any $e \in \mathcal{E}$. Then, a flight traveling on some link e whose route is potentially blocked by a node closure will continue to the head node $\sigma(e)$ on its route if that node is functioning, or reroute to one of its backup nodes if the head node is disabled. We can regard the single-backup assignment in Chapter 8 as a special case of this generalized assignment.

Example 3. We recall graph for the example network from last chapter in Fig. 9.1 with seven nodes and seven links, $\mathcal{V} = \{v_1, v_2, \dots, v_7\}$ and $\mathcal{E} = \{e_1 = (v_1, v_2), e_2 = (v_2, v_3), e_3 = (v_2, v_4), e_4 = (v_3, v_6), e_5 = (v_3, v_5), e_6 = (v_4, v_5), e_7 = (v_5, v_7)\}$. The set of all possible origins (resp., destinations) is $S = \{v_1\}$ (resp., $T = \{v_6, v_7\}$). We assume the origin v_1 does not have a capacity constraint, while $C_{v_2} = 8, C_{v_3} = 6, C_{v_4} = 4, C_{v_5} = 5, C_{v_6} = 3$

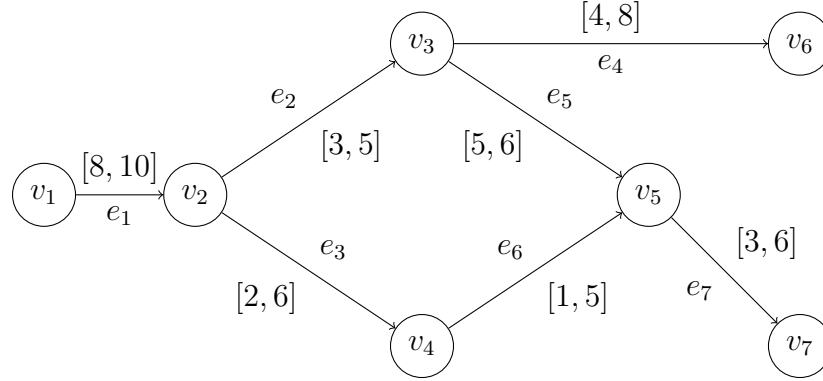


Figure 9.1: A network with 7 nodes and 7 links. This is the same network that appeared in Fig. 7.1.

and $C_{v_7} = 5$. The links are indicated in the figure and the corresponding travel time intervals are labeled beside the links, e.g., the interval $[8, 10]$ above the link e_1 means that the shortest (resp., longest) possible time for traveling through the link is 8 (resp., 10) time units. We consider three routes $\mathcal{R} = \{R^1, R^2, R^3\}$ with $R^1 = \{(v_1, v_2), (v_2, v_3), (v_3, v_6)\}$, $R^2 = \{(v_1, v_2), (v_2, v_3), (v_3, v_5), (v_5, v_7)\}$ and $R^3 = \{(v_1, v_2), (v_2, v_4), (v_4, v_5), (v_5, v_7)\}$. Each UAV remains at the verti-stops along its path for $w = 1$ time unit after landing.

Table 9.1 shows a possible assignment of backup-nodes for the network. The first column represents the link $e \in \mathcal{E}$, the second column is the set of backup nodes assigned to the corresponding link, while the third column shows the node (or nodes) that the UAV on the link can be rerouted to if node v_5 is disabled. We can notice that the UAVs traveling on the link (v_3, v_6) and (v_5, v_7) will not be affected if v_5 fails, hence the corresponding rows in the third column of table is empty. It should be noted that although the link (v_1, v_2) is not directly affected by the closure of node v_5 , other vehicles scheduled through v_5 will have to perform an emergency landing at earlier nodes, something that may also affect journeys not scheduled through via node v_5 .

Again, a realization of the j 'th flight is *affected* by some disabled node $v_c \in \mathcal{V}$ at time t_c if $v_c \in V(R_j)$, i.e., the route of the flight travels through node v_c , and the flight has not yet reached v_c by time t_c . The flight is *not affected* when node v_c is disabled at time t_c .

Table 9.1: Backup nodes for each link in Example 3

Link ($e \in \mathcal{E}$)	\mathcal{B}_e	Possible backup nodes when v_5 is disabled
$e_1 = (v_1, v_2)$	$\{v_1, v_2\}$	v_2
$e_2 = (v_2, v_3)$	$\{v_2, v_3, v_4\}$	v_3
$e_3 = (v_2, v_4)$	$\{v_2, v_3, v_4\}$	v_4
$e_4 = (v_3, v_6)$	$\{v_3, v_5, v_6\}$	Not affected
$e_5 = (v_3, v_5)$	$\{v_3, v_4, v_5\}$	v_3, v_4
$e_6 = (v_4, v_5)$	$\{v_3, v_4, v_5, v_7\}$	v_3, v_4, v_7
$e_7 = (v_5, v_7)$	$\{v_4, v_5, v_7\}$	Not affected

otherwise. The j 'th flight is *possibly affected* by disabling node v_c at time t_c if $v_c \in V(R_j)$ and $t_c < \sup \mathcal{M}_{v_c}^j$, i.e., the flight may have to travel through the disabled node later than t_c and hence is affected for some realization of travel times.

The rules that all flights are assumed to follow once a node v_c is disabled at time t_c are slightly revised compared to the previous chapter, where we include the indeterministic of rerouting to the backup nodes:

1. flights not affected will continue normal operation;
2. any affected flight that has not yet departed ($\delta_j > t_c$) will be canceled (no longer depart);
3. an affected flight j with $\delta_j \leq t_c$ traveling on a link $e \in \mathcal{E}$ with $\sigma(e) \neq v_c$ will continue to the head node $\sigma(e)$ and stop there indefinitely (block the node indefinitely);
4. an affected flight j with $\delta_j \leq t_c$ that is landing at a node at time t_c will remain there indefinitely;
5. an affected flight j with $\delta_j \leq t_c$ traveling on a link $e \in \mathcal{E}$ with $\sigma(e) = v_c$ will be rerouted to one of the backup nodes of the current link in \mathcal{B}_e and stop there indefinitely.

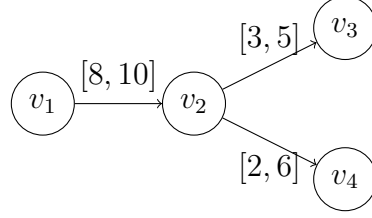


Figure 9.2: The sub-graph consisted of the 4 nodes and 3 links is used to illustrate the simple network in Example 2

9.2 Necessary and Sufficient Conditions for Safe Schedules

We have already defined the worst-case and best-case safety criteria in 8.2. We then illustrate them with an example in the same network as in Example 2 to show how assigning multiple backup nodes may change the safety verification of the network.

Example 4. We illustrate the model and the safety criteria through the simple network shown in Fig. 9.2 with 4 nodes and 3 links, which can be regarded as a sub-network of the UAM network in Fig. 9.1. For this example, the set of nodes $\mathcal{V} = \{v_1, v_2, v_3, v_4\}$ and the set of links $\mathcal{E} = \{(v_1, v_2), (v_2, v_3), (v_2, v_4)\}$. We assume that the origin v_1 does not have a capacity constraint, while $C_{v_2} = 2$, $C_{v_3} = 1$ and $C_{v_4} = 1$. The links are indicated in the figure and the corresponding travel time intervals are labeled beside the links. We consider two possible routes $R^1 = \{(v_1, v_2), (v_2, v_3)\}$ and $R^2 = \{(v_1, v_2), (v_2, v_4)\}$. Each flight remains at the intermediate nodes or destination along its path for $w = 1$ time unit after landing. The backup nodes for each link are the same as in Example 3, where $\mathcal{B}_{(v_1, v_2)} = \{v_1, v_2\}$, $\mathcal{B}_{(v_2, v_3)} = \{v_2, v_3, v_4\}$ and $\mathcal{B}_{(v_2, v_4)} = \{v_2, v_3, v_4\}$. Consider a feasible schedule profile $\mathcal{S}^p = \{S_1, S_2, S_3\}$, where $S_1 = (R^2, 1)$, $S_2 = (R^2, 8)$ and $S_3 = (R^1, \delta_3)$ where we consider several possibilities for δ_3 . Assume v_4 is disabled at time $t_c = 15$. Based on the rerouting rules for the flights, then at time t_c , a flight traveling on the link (v_1, v_2) (resp., (v_2, v_3)) will be rerouted to node v_2 (resp., v_3), while a flight traveling on the link (v_2, v_4) can be rerouted to either v_2 or v_3 . Though it is possible that flight S_1 has already complete its journey by time $t_c = 15$, in the worst case where we count any link that it

may be traveling on, it is possible for flight S_1 to be traveling on (v_2, v_4) and needs to be rerouted to v_2 or v_3 , so that we have to reserve a landing spot at node v_2 or v_3 for S_1 ; flight S_2 must be traveling on (v_1, v_2) and needs to stay at v_2 upon arrival. If $\delta_3 = 0$, then flight S_3 is not affected, and should continue its journey; however, if $\delta_3 = 10$, then either v_2 or v_3 will have insufficient landing spots, since the flight S_1 must have been rerouted to either v_2 or v_3 upon the arrival of S_3 . Therefore, \mathcal{S}^p is worst-case time-node conditionally safe for node v_4 at time 15 if and only if $\delta_3 \leq 4$. In contrast, it is always best-case time-node conditionally safe for node v_4 at time 15 regardless of the choice of δ_3 .

Now, suppose $\delta_3 = 10$ and $C_{v_2} = 3$. Then obviously, the network will be able to accommodate all rerouted flights after closure no matter when node v_4 is closed. Therefore, we see that \mathcal{S}^p is worst-case node conditionally safe for node v_4 in this case. We further check that this is true for all nodes in the network, and thus \mathcal{S}^p is also worst-case 1-closure safe. In contrast, suppose $C_{v_3} = 2$ while $C_{v_2} = 2$, then \mathcal{S}^p will always be worst-case time-node conditionally safe for node v_4 and $t_c = 15$ with any choice of δ_3 . Moreover, we can observe that the schedule is worst-case node-conditionally safe for v_4 if and only if $\delta_3 \geq 4$. \square

We've shown in Section 8.2 of last chapter that a feasible schedule profile is trivially node conditionally safe for node $v \in S$. We next explore safety of a disabled node that is not a source node. Notice some of the special sets we defined in Section 8.2 needs to be redefined here for the general case. If $v_c = \ell^{R_j}$, we let $\ell_{v_c, R_j} = \ell$, and $\ell_{v_c, R_j} = 0$ if $v_c \notin V(R_j)$.

The set of links with head $v \in \mathcal{V}$ be \mathcal{E}_v is define in (8.4). B_{e, v_c} is the set of nodes that any flight traveling on the link e can be rerouted to if v_c is disabled:

$$B_{e, v_c} = \begin{cases} \{\sigma(e)\} & \text{if } \sigma(e) \neq v_c, \\ \mathcal{B}_e \setminus v_c & \text{if } \sigma(e) = v_c. \end{cases} \quad (9.1)$$

\mathcal{B}_{e,v_c} is the set of *possible backup nodes* for link e when v_c is disabled. We then denote b_{e,v_c} as the node that a flight traveling on the link e will be rerouted to if v_c is disabled, so that $b_{e,v_c} \in \mathcal{B}_{e,v_c}$.

We denote the set of links on which flights will possibly be rerouted to node v when v_c is disabled as \mathcal{B}_{v,v_c} , which includes the links with head node as v when $v \neq v_c$ and the links with head node as v_c whose backup nodes include v , i.e.,

$$\mathcal{B}_{v,v_c} := \{e \in \mathcal{E} \mid \sigma(e) = v, \sigma(e) \neq v_c\} \cup \{e \in \mathcal{E} \mid v \in \mathcal{B}_e \setminus v_c, \sigma(e) = v_c\}. \quad (9.2)$$

We then define the set $\mathcal{E}_{v_c}^j$ as the links along the route of flight j whose head node is one of the backup nodes of link $\ell_{v_c, R_j}^{R_j}$, i.e.,

$$\mathcal{E}_{v_c}^j = \{\sigma(\ell^{R_j}) \in \mathcal{B}_{e,v_c} \mid \ell < \ell_{v_c, R_j}\}. \quad (9.3)$$

\mathcal{J}_v , $\mathcal{J}_v^*(t)$, $J_p(v, t_c, v_c)$, $\mathcal{J}_{c,v_c}(t_c)$, $\mathcal{J}_{v_c}^{* \setminus c}(t_c)$, and $N_R(v, t_c, v_c)$ are the same as defined previously in Section 8.2. We sometimes drop the notations in the parentheses, t, t_c, v, v_c , when they are clear from the context.

9.2.1 Necessary and Sufficient Condition for Worst-Case Safe Schedules

Theorem 10. Consider a network $\mathcal{N} = (\mathcal{G}, C, \mathcal{R}, \underline{x}, \bar{x}, w)$, where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and given backup nodes assignment \mathcal{B}_e for all $e \in \mathcal{E}$. Assume given a feasible schedule profile $\mathcal{S}^p = \{(R_j, \delta_j)\}_{j \in \mathcal{J}}$.

The schedule profile \mathcal{S}^p is worst-case time-node conditionally safe for node v_c and time t_c if and only if there exists an integer set $\{N_{e,v}(t_c, v_c)\}_{e \in \mathcal{E}, v \in \mathcal{V}}$ that satisfies the following constraints for all $v \in \mathcal{V}$ and $e \in \mathcal{E}$:

$$C_v - \sum_{e \in \mathcal{B}_{v,v_c}} N_{e,v}(t_c, v_c) \geq N_R(v, t_c, v_c), \quad \forall v \in \mathcal{V}, \quad (9.4)$$

$$\sum_{v \in \mathcal{B}_{e,v_c}} N_{e,v}(t_c, v_c) = \sum_{j \in \mathcal{J}_{v_c}^* \setminus \{c\}} \mathbf{1}\left(t_c; [L_e^j, U_e^j] \setminus \{\cup_{\ell \in \mathcal{E}_{v_c}^j} [L_{\ell^{R_j}}^j, U_{\ell^{R_j}}^j]\}\right), \forall e \in \mathcal{E}_{v_c}, \quad (9.5)$$

$$N_{e,\sigma(e)}(t_c, v_c) = \sum_{j \in \mathcal{J}_{v_c}^* \setminus \{c\}} \mathbf{1}(t_c; [L_e^j, U_e^j]), \quad \forall e \notin \mathcal{E}_{v_c}, \quad (9.6)$$

$$N_{e,v}(t_c, v_c) = 0, \quad \forall e \in \mathcal{E}, v \notin \mathcal{B}_{e,v_c}, \quad (9.7)$$

$$N_{e,v}(t_c, v_c) \geq 0, \quad \forall e \in \mathcal{E}_{v_c}, v \in \mathcal{B}_{e,v_c}, \quad (9.8)$$

where for all $j \in \mathcal{J}$, the lower and upper bounds of the time interval are defined as

$$L_e^j = \begin{cases} \inf\{\mathcal{M}_{\tau(e)}^j\} + w & \text{if } \tau(e) \neq 0^{R_j} \\ \delta_j & \text{if } \tau(e) = 0^{R_j}, \end{cases} \quad (9.9)$$

and

$$U_e^j = \begin{cases} \sup\{\mathcal{M}_{\sigma(e)}^j\} & \text{if } \sigma(e) \neq v_c \\ \sup\{\mathcal{M}_{\sigma(e)}^j\} - w & \text{if } \sigma(e) = v_c. \end{cases} \quad (9.10)$$

Further, S^p is worst-case node-conditionally safe for node v_c if and only if the set $\{N_{e,v}(t_c, v_c)\}_{e \in \mathcal{E}, v \in \mathcal{V}}$ that satisfies (9.4)–(9.8) exists and the conditions holds for the finite number of times t_c where the values of $N_R(v, t_c, v_c)$ and the time-varying index sets $\mathcal{J}_{v_c}^*(t_c)$, $\mathcal{J}_{c,v_c}(t_c)$ possibly change, i.e., at both endpoints of the interval \mathcal{M}_v^j for all $v \in \mathcal{V}$, at times δ_j for all $j \in \mathcal{J}$, and at times L_e^j, U_e^j for all $j \in \mathcal{J}$ and $e \in \mathcal{E}$.

Proof. The schedule profile is worst-case time-node conditionally safe for node v_c and time t_c if and only if, for any possibly affected flight that is not canceled and may be rerouted to some node in \mathcal{V} at time t_c , an available landing spot needs to be reserved. Hence the problem becomes to ensure the flights surely not affected will have no capacity conflict with any possibly rerouted flights. We then consider the maximum (worst-case) occupation

of the node in \mathcal{V} .

We let the set $\{N_{e,v}(t_c, v_c)\}_{e \in \mathcal{E}, v \in \mathcal{V}}$ be the set of variables that denote the number of possibly affected flights that may proceed to node $v \in \mathcal{V}$ when traveling on the link $e \in \mathcal{E}$. Hence, for all $v \in \mathcal{V}, e \in \mathcal{E}$, $N_{e,v}(t_c, v_c)$ is required to be a non-negative integer. The interval defined as $[L_e^j, U_e^j]$ is the time interval during which flight j will possibly be rerouted to b_{e,v_c} if v_c is closed, where the lower bound L_e^j is the earliest time that the flight may leave the previous node $\tau(e)$, and, if $\sigma(e)$ is not disabled, the upper bound U_e^j is the latest time that the flight may leave the head node while, in the case that $\sigma(e)$ is disabled, the upper bound U_e^j for the time interval that the flight may be rerouted to the backup node $\tau(e)$ will be the latest time that the corresponding flight may arrive at node v_c , since otherwise it will continue its normal operation without rerouting. For any $e \in \mathcal{E}$, if $\sigma(e) \neq v_c$, then the possibly affected flights traveling on the link will land at its head node $\sigma(e)$, and thus the number of possibly affected flights rerouting to node $\sigma(e)$ from link e , $N_{e,\sigma(e)}(t_c, v_c)$ is deterministic, which can be simply counted as in (9.6). The constraint (9.7) is preventing flights from proceeding to any node v not in the set of possible backup nodes for link e when node v_c is disabled, B_{e,v_c} .

As a safety requirement, when v_c is disabled at time t_c , any possibly affected flight needs to be rerouted to a node. Consider a fixed $e \in \mathcal{E}_{v_c}$, a flight whose possibly traveling on this link at time t_c is obviously possibly affected flight when node v_c is disabled at time t_c and needs to be rerouted to one of its backup nodes. Therefore, (9.5) is the link safety constraint depicting that all flights possibly traveling on e at t_c needs to be rerouted to one of the possible backup nodes for link e when v_c is disabled. Notice that, suppose the backup nodes of the link e include a node $v' \leq v_c$ that is along the route of the flight, and the flight is also possibly traveling on a link whose head node is v' at t_c , then this means a parking spot at node v' has to be reserved, and we don't need to prepare another one. This situation is reflected through $\{\cup_{\ell=1}^{\ell(v_c, R_j)-1} [L_{\ell R_j}^j, U_{\ell R_j}^j]\}$ in (9.5). Finally, $N_R(v, t_c, v_c)$ is the maximum number of flights not possibly affected that may park at node v at any time once

v_c is disabled at t_c , and the summation $\sum_{e \in \mathcal{B}_{v,v_c}} N_{e,v}(t_c, v_c)$ is the total number of possibly affected flights rerouting to node v . Therefore (9.4) is a necessary and sufficient condition to avoid the capacity conflict between the rerouted flights and those surely not affected for all realization of link travel times. \square

Theorem 10 provides a finite number of conditions to verify a schedule is worst-case node conditionally safe for node v_c . Furthermore, by checking that a schedule profile is worst-case node conditionally safe for all $v_c \in \mathcal{V}$, we can conclude the 1-closure safety. However, we notice that looking for the existence of a integer sequence satisfying the constraints (9.4)–(9.8) in Theorem 10 can be regarded as a Mixed Integer Linear Programming (MILP) problem, which is sensitive to scale and can be time-consuming once the size of the schedule under verification grows. We will recast the MILP as a linear program in Section 9.3, leading to an efficient safety-verification algorithm. In the following subsection, we explore the safety constraints for given UAM schedule profile in the best-case scenario.

9.2.2 Necessary and Sufficient Condition for Best-Case Safe Schedules

Theorem 10 provides a set of constraints that can be served as a necessary and sufficient condition for a feasible schedule profile to be worst-case time-node conditionally, node conditionally or 1-closure safe. In this subsection, we are exploring the constraints for a feasible schedule profile to be best-case safe. In the best-case scenario, we are considering the realization with the least number of rerouting flights and most flexible rerouting plan needed among all possible realizations. Therefore, we assume that all flights possible to have arrived at or passed the closed node v_c has already arrived or left by the time of node failure.

We denote the index set of the *definitely affected* flights as

$$\mathcal{J}_{v_c}^m(t_c) = \{j \in \mathcal{J}_{v_c}^* \setminus \mathcal{J}_{c,v_c}(t_c) \mid \inf\{\mathcal{M}_{v_c}^j\} \geq t_c\}. \quad (9.11)$$

The definitely affected flights are the flights that must be rerouted under any possible realization.

Theorem 11. Consider a network $\mathcal{N} = (\mathcal{G}, C, \mathcal{R}, \underline{x}, \bar{x}, w)$, where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and given backup nodes assignment \mathcal{B}_e for all $e \in \mathcal{E}$. A given feasible schedule profile $\mathcal{S}^p = \{(R_j, \delta_j)\}_{j \in \mathcal{J}}$ is best-case time-node conditionally safe for node v_c and time t_c if and only if there exists a integer set $\{N_{j,v}(t_c, v_c)\}_{j \in \mathcal{J}_{v_c}^m(t_c), v \in \mathcal{V}}$ that satisfies the following constraints for all $j \in \mathcal{J}_{v_c}^m(t_c)$ and $v \in \mathcal{V}$:

$$C_v - \sum_{j \in \mathcal{J}_{v_c}^m} (t_c) N_{j,v}(t_c, v_c) \geq N_R(v, t_c, v_c) \quad \forall v \in \mathcal{V} \quad (9.12)$$

$$\sum_{v \in \mathcal{V}} N_{j,v}(t_c, v_c) = 1 \quad \forall j \in \mathcal{J}_{v_c}^m(t_c) \quad (9.13)$$

$$0 \leq N_{j,v}(t_c, v_c) \leq \max_{e \in \mathcal{R}_j} \mathbf{1}(t_c; [L_e^j, \hat{U}_e^j]) \cdot \mathbf{1}(v; \mathcal{B}_{e,v_c}) \quad \forall v \in \mathcal{V}, j \in \mathcal{J}_{v_c}^m(t_c) \quad (9.14)$$

where L_e^j is defined in (9.9) and

$$\hat{U}_e^j = \begin{cases} \sup\{\mathcal{M}_{\sigma(e)}^j\} & \text{if } \sigma(e) \neq v_c \\ \inf\{\mathcal{M}_{\sigma(e)}^j\} & \text{if } \sigma(e) = v_c. \end{cases} \quad (9.15)$$

Further, \mathcal{S}^p is best-case node-conditionally safe for node v_c if and only if the set $\{N_{j,v}(t_c, v_c)\}_{j \in \mathcal{J}, v \in \mathcal{V}}$ that satisfies (9.12)–(9.14) exists and the conditions holds for the finite number of times t_c where the values of $N_R(v, t_c, v_c)$ and the time-varying index sets $\mathcal{J}_{v_c}^m(t_c)$, $\mathcal{I}_{c,v_c}(t_c)$ possibly change, i.e., at both endpoints of the interval \mathcal{M}_v^j for all $v \in \mathcal{V}$, δ_j for all $j \in \mathcal{J}$ and at times L_e^j, \hat{U}_e^j for all $j \in \mathcal{J}$ and $e \in \mathcal{E}$.

Proof. The proof of Theorem 11 applies the similar logic as in Theorem 10 to the best-case scenario, while from the perspective of flights instead of the links. First of all, we can focus only on the definitely affected flights, since the flights that are possibly affected but

not definitely affected is either canceled or has at least a realization of travel time such that the flight has already passed through or land at node v_c , and does not need to be rerouted.

We regard $N_{j,v}(t_c, v_c)$ as the indicator of j 'th flight to be rerouted to node v if node v_c is disabled at time t_c , for all $j \in \mathcal{J}$ and $v \in \mathcal{V}$. As a result, the non-negative variable $N_{j,v}(t_c, v_c)$ is actually binary. We enforce this binary condition in (9.14), where $N_{j,v}(t_c, v_c) \leq 1$ if there exists $e \in R_j$ such that v is one of its possibly backup nodes when v_c is closed and the flight is definitely affected and possibly traveling on the link e at time t_c and $N_{j,v}(t_c, v_c) = 0$ otherwise. Notice that the upper-bound for the time interval that the flight is possibly aiming at $\sigma(e)$ and will be rerouted to one of its possible backup nodes, \hat{U}_e^j is trimmed comparing to U_e^j defined in (9.10) to include only the definitely affected flights.

Once node v_c is disabled at time t_c , a flight will actually be reroute to exactly one node, which is depicted in (9.13). Moreover, (9.12) is the capacity constraint, where $\sum_{j \in \mathcal{J}_{v_c}^m} N_{j,v}(t_c, v_c)$ is the number of definitely affected flights rerouted to node v .

If we are not able to find a sequence of non-negative integers $\{N_v\}_{v \in \mathcal{V}}$ that satisfies (9.12)–(9.14), then there must exist a conflict of occupation at one or more nodes once v_c is closed at time t_c , and hence (9.12)–(9.14) are sufficient and necessary conditions for the set of schedules to be best-case time-node conditionally safe for node v_c and time t_c . \square

Theorem 10 is both sufficient and necessary for worst-case 1-closure safety, while Theorem 11 is sufficient and necessary for best-case 1-closure safety. Since worst-case safety implies best-case safety, satisfaction of the conditions in Theorem 10 implies satisfaction of the conditions in Theorem 11.

9.3 Simplification for Verification

The necessary and sufficient conditions for safety derived in Section 9.2 involve integer constraints and therefore are inefficient for use in a direct numerical implementation. In this section, we show that these conditions can in fact be translated to efficient LP constraints.

We first establish a lemma explaining the mathematical foundation for our simplification of Theorem 10, followed by a theorem that turns the MILP problem in Theorem 10 into a linear programming (LP) problem.

The proof of 12 makes use of properties of *Totally Unimodular Matrices (TUMs)*.

Definition 11. (*Totally Unimodular Matrix*) A matrix is totally unimodular if every square submatrix has determinant 0, +1, or -1.

TUMs are widely used in the context of optimization problems. In particular, it can be shown that for a large class of linear programs defined via TUMs, the resulting optimal solution takes on integer values [81]. We use this property in the proof of Lemma 12 next.

In particular, the following lemma shows that, for a special set of constraints on a set of variables, the existence of a solution over the real numbers induce the existence of a solution over the integers.

Lemma 12. Given a set $L = \{(l_1, l_2) \in \mathbb{N}_{>0}^2 \mid l_1 \leq N_1, l_2 \leq N_2\}$ for some positive integers N_1, N_2 , and let L_{var} be a subset of L . Let α_{l_1} (resp., β_{l_2}) be non-negative integers for all $l_1 = 1, \dots, N_1$ (resp., $l_2 = 1, \dots, N_2$), and γ_{l_1, l_2} be integers for all $(l_1, l_2) \in L_{var}$. If there exists a set of real numbers $\{n_{l_1, l_2}\}_{l_1=1, l_2=1}^{l_1=N_1, l_2=N_2}$ that satisfies

$$\sum_{l_2=1}^{N_2} n_{l_1, l_2} = \alpha_{l_1} \quad \forall l_1 = 1, \dots, N_1 \quad (9.16)$$

$$\sum_{l_1=1}^{N_1} n_{l_1, l_2} \leq \beta_{l_2} \quad \forall l_2 = 1, \dots, N_2 \quad (9.17)$$

$$n_{l_1, l_2} = \gamma_{l_1, l_2} \quad \forall (l_1, l_2) \in L_{var} \quad (9.18)$$

$$n_{l_1, l_2} \geq 0 \quad \forall (l_1, l_2) \in L \quad (9.19)$$

then there exists a set of integers $\{n'_{l_1, l_2}\}_{l_1=1, l_2=1}^{l_1=N_1, l_2=N_2}$ also satisfying (9.16) – (9.19).

Proof. We first let the vector \vec{n} be the vectorized sequence $\{n_{l_1, l_2}\}_{l_1=1, l_2=1}^{l_1=N_1, l_2=N_2}$, so that

$$\vec{n} = [n_{1,1}, n_{1,2}, \dots, n_{1,N_2}, n_{2,1}, \dots, n_{N_1,1}, \dots, n_{N_1,N_2}]^T. \quad (9.20)$$

We can then simplify the constraint (9.16)–(9.17) as

$$A_1 \vec{n} = \vec{\alpha} \quad (9.21)$$

$$A_2 \vec{n} \leq \vec{\beta}, \quad (9.22)$$

where $\vec{\alpha} = [\alpha_1, \dots, \alpha_{N_1}]^T$ and $\vec{\beta} = [\beta_1, \dots, \beta_{N_2}]^T$, and A_1 (resp., A_2) is a $N_1 \times N_1 N_2$ (resp., $N_2 \times N_1 N_2$) matrix that reflects the matrix form of the multiplication of the constraints. In particular, the row- i -column- j element of A_1 is $A_1(i, j) = 1$ if $(i-1)N_2 < j \leq iN_2$ and $A_1(i, j) = 0$ otherwise, and $A_2(i, j) = 1$ if $j = m \cdot N_2 + i$ for $m = 0, 1, \dots, N_1 - 1$ and $A_2(i, j) = 0$ otherwise.

Since for $(l_1, l_2) \in L_{var}$, $n_{l_1, l_2} = \gamma_{l_1, l_2}$, we can then subtract the corresponding entries from the left sides of (9.21) and (9.22), and subtract the values from their right sides. We let A_3 (resp., A_4) be the resulting matrices, so that A_3 (resp., A_4) is a $N_2 \times N_1 N_2$ (resp., $N_1 \times N_1 N_2$) and the row- i -column- j element of A_3 is $A_3(i, j) = 0$ if $(i, j - (i-1)N_2) \in L_{var}$ and $A_3(i, j) = A_1(i, j)$ otherwise, and $A_4(i, j) = 0$ if $(i, (j-i)/N_2 + 1) \in L_{var}$ and $A_4(i, j) = A_2(i, j)$ otherwise. Let

$$\alpha'_{l_1} = \alpha_{l_1} - \sum_{l_2: (l_1, l_2) \in L_{var}} \gamma_{l_1, l_2} \quad \text{for all } l_1, \quad (9.23)$$

$$\beta'_{l_2} = \beta_{l_2} - \sum_{l_1: (l_1, l_2) \in L_{var}} \gamma_{l_1, l_2} \quad \text{for all } l_2, \quad (9.24)$$

$$\vec{\alpha}' = [\alpha'_1, \dots, \alpha'_{N_1}]^T, \quad (9.25)$$

$$\vec{\beta}' = [\beta'_1, \dots, \beta'_{N_2}]^T. \quad (9.26)$$

We can then reformulate (9.21) and (9.22) together with the constraints (9.18)–(9.19) as

$$\underbrace{\begin{bmatrix} A_3 \\ -A_3 \\ A_4 \\ -I_{N_1 N_2} \end{bmatrix}}_{=:A} \vec{n} \leq \underbrace{\begin{bmatrix} \vec{\alpha}' \\ -\vec{\alpha}' \\ \vec{\beta}' \\ \vec{0}_{N_1 N_2} \end{bmatrix}}_{=: \vec{b}}, \quad (9.27)$$

where $I_{N_1 N_2}$ is the identity matrix with $N_1 N_2$ rows and $\vec{0}_{N_1 N_2}$ is the zero vector of length $N_1 N_2$.

The first part of the lemma is then turned into the standard linear programming problem, which is finding the existence of \vec{n} that satisfies $A\vec{n} \leq \vec{b}$. The next step is to prove that A is a totally unimodular matrix (TUM) as defined in Definition 11.

We first consider the matrix $\begin{bmatrix} A_3 \\ A_4 \end{bmatrix}$. Notice that for each column of A_3 and A_4 , there exists at most one nonzero entry, 1, therefore, for each column of the matrix $\begin{bmatrix} A_3 \\ A_4 \end{bmatrix}$, there exists at most two nonzero entries, and for any column with two non-zero entries, both of them will be 1, and the row of one is in A_3 while the other in A_4 . According to Hoffman's sufficient conditions [82], $\begin{bmatrix} A_3 \\ A_4 \end{bmatrix}$ is a TUM. By the general rule of TUM, $-\begin{bmatrix} A_3 \\ A_4 \end{bmatrix}$ is a

TUM and thus $\begin{bmatrix} A_3 \\ A_4 \\ -A_3 \\ -A_4 \end{bmatrix}$ is also a TUM. According to the definition of TUM, it is obvious

that deleting some rows from a TUM will produce a TUM, as any square non-singular submatrix of the new matrix will still be unimodular. As a result, $\begin{bmatrix} A_3 \\ A_4 \\ -A_3 \end{bmatrix}$ and $-\begin{bmatrix} A_3 \\ A_4 \\ -A_3 \end{bmatrix}$

are TUM. By the general rule of TUM, $\begin{bmatrix} -A_3 \\ -A_4 \\ A_3 \\ I_{N_1 N_2} \end{bmatrix}$ is TUM and $\begin{bmatrix} A_3 \\ A_4 \\ -A_3 \\ -I_{N_1 N_2} \end{bmatrix}$ is also a TUM.

As switching rows doesn't affect the abstract value of the determinant of a matrix, then we conclude from above that A is a TUM.

Therefore, [81] implies that if there exists a solution for the LP in (9.27), then there exists a integral solution for the same LP problem, which concludes the lemma. \square

Further, the remark below can be shown with some trivial revisions to the proof.

Remark 1. *Lemma 12 holds if n_{l_1, l_2} is bounded from above by integer, i.e., (9.19) is changed to $0 \leq n_{l_1, l_2} \leq U_{l_1, l_2}$ for all $(l_1, l_2) \in L$ for some integer number $U_{l_1, l_2} > 0$.*

The simplified corollary below makes use of Lemma 12 above and provides an LP alternative to the MILP problem in Theorem 10.

Corollary 3. *Consider a network $\mathcal{N} = (\mathcal{G}, C, \mathcal{R}, \underline{x}, \bar{x}, w)$, where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Assume given a feasible schedule profile $\mathcal{S}^p = \{(R_j, \delta_j)\}_{j \in \mathcal{J}}$. There exists a set of real numbers $\{N_{e,v}(t_c, v_c)\}_{e \in \mathcal{E}, v \in \mathcal{V}}$ that satisfies the constraints (9.4) – (9.8) if and only if there exists a set of integers $\{N'_{e,v}(t_c, v_c)\}_{e \in \mathcal{E}, v \in \mathcal{V}}$ that satisfies the constraints.*

Proof. We first show that the conditions (9.4) and (9.5) conforms to the form in Lemma 12. For the sake of convenience, we fix t_c and v_c and drop the notation from $N_{e,v}(t_c, v_c)$ and $N_R(v, t_c, v_c)$, i.e., we write them as $N_{e,v}$ and $N_R(v)$ in this proof.

We can observe that, by the definition of \mathcal{B}_{v, v_c} in (9.2), assume $e \notin \mathcal{B}_{v, v_c}$, if $\sigma(e) = v_c$, then $v \notin \mathcal{B}_e \setminus v_c$, otherwise $\sigma(e) \neq v_c$. We can therefore conclude from (9.6) and (9.7) that $N_{e,v}$ is a definite number if $e \notin \mathcal{B}_{v, v_c}$.

By adding the definite terms of $N_{e,v}$ for $e \notin \mathcal{B}_{v, v_c}$ to both sides of (9.4) we can then

obtain that, for all $v \in \mathcal{V}$,

$$\begin{aligned} \sum_{e \in \mathcal{B}_{v,v_c}} N_{e,v} &\leq C_v - N_R(v) \\ \sum_{e \in \mathcal{E}} N_{e,v} &\leq C_v - N_R(v) + \sum_{e \notin \mathcal{B}_{v,v_c}} N_{e,v}. \end{aligned} \quad (9.28)$$

Similarly, if $v \notin \mathcal{B}_{e,v_c}$, then $N_{e,v}$ is a fixed number. By adding $\sum_{v \notin \mathcal{B}_{e,v_c}} N_{e,v}$ to both sides of (9.5), we have

$$\sum_{v \in \mathcal{V}} N_{e,v} = \sum_{j \in \mathcal{J}_{v_c}^*(t_c)} \mathbf{1}(t_c, [L_e^j, U_e^j]) + \sum_{v \notin \mathcal{B}_{e,v_c}} N_{e,v}. \quad (9.29)$$

We then let $L = \{(e, v) \mid e \in \mathcal{E}, v \in \mathcal{V}\}$, and $L_{var} = L \setminus \{(e, v) \mid \sigma(e) = v_c, v \in \mathcal{B}_{e,v_c}\}$.

Then we can combine and rewrite (9.6) – (9.8) as

$$N_{e,v} = \gamma_{e,v} \quad \forall (e, v) \in L_{var} \quad (9.30)$$

$$N_{e,v} \geq 0 \quad \forall (e, v) \in L, \quad (9.31)$$

where $\gamma_{e,v} = \sum_{j \in \mathcal{J}_{v_c}^*(t_c) \setminus \mathcal{J}_c(t_c)} \mathbf{1}(t_c, [L_e^j, U_e^j])$ if $\sigma(e) = v_c$ and $\gamma_{e,v} = 0$ if $\sigma(e) \neq v_c$ and $v \notin \mathcal{B}_{e,v_c}$.

As \mathcal{E} and \mathcal{V} are both finite sets, while the right sides of the inequalities (9.28) and (9.29) are integers, then the conditions (9.28)–(9.31) exactly follows the conditions (9.16)–(9.19) in Lemma 12. Therefore, by Lemma 12, there exists a integral sequence $\{N_{e,v}\}_{e \in \mathcal{E}, v \in \mathcal{V}}$.

As a result, given the schedule profile \mathcal{S}^p for the demands \mathcal{D} , if there exists a set of real numbers $\{N_{e,v}(t_c, v_c)\}_{e \in \mathcal{E}, v \in \mathcal{V}}$ that satisfies the constraints (9.4) – (9.8), then there exist a set of integers $\{N'_{e,v}(t_c, v_c)\}_{e \in \mathcal{E}, v \in \mathcal{V}}$ that satisfies the constraints. The other direction of the corollary is obvious because of the inclusive relation between real number and integer. \square

Combining Theorem 10 and Corollary 3, we are then able to verify 1-closure safety of given feasible schedules by solving a linear program. Similarly, we develop a corollary for

simplification of best-case safety mirroring Corollary 3 given Remark 1.

Corollary 4. *Consider a network $\mathcal{N} = (\mathcal{G}, C, \mathcal{R}, \underline{x}, \bar{x}, w)$, where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Assume given a feasible schedule profile $\mathcal{S}^p = \{(R_j, \delta_j)\}_{j \in \mathcal{J}}$. There exists a set of real numbers $\{N_{j,v}(t_c, v_c)\}_{j \in \mathcal{J}_{v_c}^m, v \in \mathcal{V}}$ that satisfies the constraints (9.12)–(9.14) if and only if there exists a set of integers $\{N'_{j,v}(t_c, v_c)\}_{j \in \mathcal{J}_{v_c}^m, v \in \mathcal{V}}$ that satisfies the constraints.*

The proof for Corollary 4 is immediate from Lemma 12 as we can easily convert the constraints (9.12)–(9.14) to the same form as in Lemma 12 and Remark 1.

9.4 Case Study

In the case study, we demonstrate the verification algorithm based on Theorem 10 and Corollary 3 on the UAM network in the Example 3 with a feasible schedule profile of size 20. We also demonstrate the efficient scaling of the algorithm on examples up to 1000 UAVs.

To ensure the worst-case node conditional safety when v_c is closed, we check whether there exists a set of real numbers $\{N_{e,v}(t_c, v_c)\}_{e \in \mathcal{E}, v \in \mathcal{V}}$ that satisfies the constraints (9.4)–(9.8) over the time interval $t_c \in [0, +\infty)$ so that the worst-case safety is guaranteed. As stated in Theorem 10, we only need to solve the LP feasibility problem at each point of time that any value may change, i.e., L_e^j, U_e^j , both ends of \mathcal{M}_v^j and δ_j for any counted flight $j \in \mathcal{J}$ and link $e \in \mathcal{E}$ for some fixed node v , since the system of linear inequalities (9.4)–(9.8) will not change between these points. We randomly generate a particular feasible schedule profile with 20 flights and consider the constraints (9.4)–(9.8) in Theorem 10 for time-node conditionally safe for node $v_c = v_5$ and any time $t_c > 0$. The verification is implemented in MATLAB¹.

In Fig. 9.3, we observe the worst-case landing-spot occupation at node v_3 (top) and the redistribution of flights on link e_5 and e_6 (middle and bottom) when node v_5 is disabled

¹The related MATLAB code can be found in https://github.com/gtfactslab/Wei_TCNS_ScheduleVerification.git.

at any time t_c . For convenience, we simplify the notation $N_R(\cdot, t_c, v_5)$ and $N_{\cdot,}(t_c, v_5)$ as $N_R(\cdot)$ and $N_{\cdot,}$ in the figure. The top graph of Fig. 9.3 shows the distribution of UAVs that might possibly land at node v_3 . The blue rectangles correspond to flights not affected and continue to v_3 if node v_5 is disabled at time t_c , which is $N_R(v_3, t_c, v_5)$ in (9.4); the pink rectangles correspond to flights rerouted to node v_3 from link $e_2 = (v_2, v_3)$ if v_5 is disabled at time t_c , which is $N_{e_2, v_3}(t_c, v_5)$; the orange (resp., green) rectangles correspond to the number of flights rerouted to node v_3 from link $e_5 = (v_3, v_5)$ (resp., $e_6 = (v_4, v_5)$) if node v_5 is disabled at time t_c , which is $N_{e_5, v_3}(t_c, v_5)$ (resp., $N_{e_6, v_3}(t_c, v_5)$). Notice that $N_{e_2, v_3}(t_c, v_5)$ is fixed and can be computed by (9.6), since any flight traveling on e_2 at time t_c has to land at v_3 if v_5 is disabled at that time; meanwhile, $N_{e_5, v_3}(t_c, v_5)$ (resp., $N_{e_6, v_3}(t_c, v_5)$) is an optimization variable computed through the LP problem (9.4)–(9.8). However, it is possible that there does not exist a solution to the LP problem at certain time instances t_c , that is, the schedule is not worst-case time-node conditionally safe when node v_5 is disabled at time t_c . If that is the case, only the definite parts $N_R(t_c, v_5)$ and $N_{e_2, v_3}(t_c, v_5)$ (represented by blue and pink rectangles) are shown in the corresponding time interval, while the remaining parts are shown as a grey rectangle to demonstrate the failure. As a reference, the capacity $C_{v_3} = 6$ is shown as the dotted, horizontal line so that the height of the entire bar (the sum of all rectangles) must not exceed the capacity for safety.

The redistribution of flights on link e_5 (resp., e_6) shown in the middle (resp., bottom) graph of Fig. 9.3 when node v_5 is disabled at any time t_c provide us a detailed partition of flights onto the nodes to which they are rerouted. Since the head of the link e_5 (resp., e_6), v_5 , is disabled, the flights traveling on the link needs to be rerouted to one of the possible backup nodes, v_3 or v_4 (resp., v_3, v_4 or v_7). We use orange and purple (resp., green, red and blue) rectangles to represent $N_{e_5, v_3}(t_c, v_5)$ and $N_{e_5, v_4}(t_c, v_5)$ (resp., $N_{e_6, v_3}(t_c, v_5)$, $N_{e_6, v_4}(t_c, v_5)$, and $N_{e_6, v_7}(t_c, v_5)$), i.e., the number of affected flights traveling on link e_5 (resp., e_6) rerouted to the backup nodes v_3 and v_5 (resp., v_3, v_4 and v_7). Similar to the top graph of Fig. 9.3, we use grey rectangles to indicate the failure of obtaining the solution to

the LP problem (9.4)–(9.8). The height of the grey rectangles represents the total number of flights that need to be rerouted from link e_5 (resp., e_6) when node v_5 is disabled at time t_c , i.e., $\sum_{v \in \mathcal{B}_{e_5, v_5}} N_{e_5, v}(t_c, v_5)$. Notice that the solution to the LP problem $(N_{\cdot, \cdot}(t_c, v_5))$ for $t_c > 0$, if it exists, is not unique, and hence Fig. 9.3 is only one possible rerouting arrangement. Thus, as an example, the schedule in this case study is not time-node conditionally safe for node v_5 at $t_c = 40$, as the grey rectangle indicates there does not exist a solution to the problem (9.4)–(9.8) at time t_c . Therefore, the network is not able to accommodate the failure of v_5 at time $t_c = 40$.

For the sake of comparison, we increase the capacity of v_4 to $C_{v_4} = 8$ while the other parts of the network remain the same. We then verify the safety of the same schedule with the algorithm, and these results are shown as in Fig. 9.4. As shown in the plots, after increasing the capacity of v_4 , which is a backup node for both e_5 and e_6 , the solution to the problem (9.4)–(9.8) exists all the time. To conclude if the schedule profile is node conditionally safe when v_5 is disabled, we would need to observe all affected nodes and links in the network in the same way.

The computation time for $N_R(v, t_c, v_c)$ in (9.4) increases quadratically with the size of the schedule, and as indicated in [83], solving the LP problem (9.4)–(9.8) with a fixed number of variables can be computed within linear time with respect to the number of constraints, while the number of constraints in the LP problem and the number of times the LP needs to be solved are both linearly growing with the size of schedule profile. We thus conclude that the verification process is completed in $O(n^2)$ time. This efficient scaling implies that we are able to verify worst-case safety with large schedule profiles. As an example, consider increasing the capacity for each node of the network in Fig. 9.1 by 10 to produce feasible schedule profiles more easily. We generate 10 more sets of random schedules with sizes 100, 200, 300, \dots , 1000 and verify their safety using the same algorithm. Fig. 9.5 demonstrates the $O(n^2)$ computation complexity and shows that we are able to verify safety or demonstrate the safety failure for a schedule profile with 1,000 flights



Figure 9.3: Observation of the network when node v_5 is disabled at any time $t_c > 0$. (Top) Expected landing-spot occupation at node v_3 . (Middle) Redistribution of flights on link e_5 to its backup nodes. (Bottom) Redistribution of flights on link e_6 to its backup nodes. We simplify the notation $N_R(\cdot, t_c, v_5)$ and $N_{\cdot, \cdot}(t_c, v_5)$ as $N_R(\cdot)$ and $N_{\cdot, \cdot}$ in the figure.

in under 50 seconds. As a baseline comparison, we also implement the verification algorithm with the naive MILP implied by Theorem 10 without the efficient simplification to a LP derived in Section 9.3. This implementation is solved using the Gurobi [79] solver through with the YALMIP MATLAB toolbox [84]. We test the same 20-flight schedule

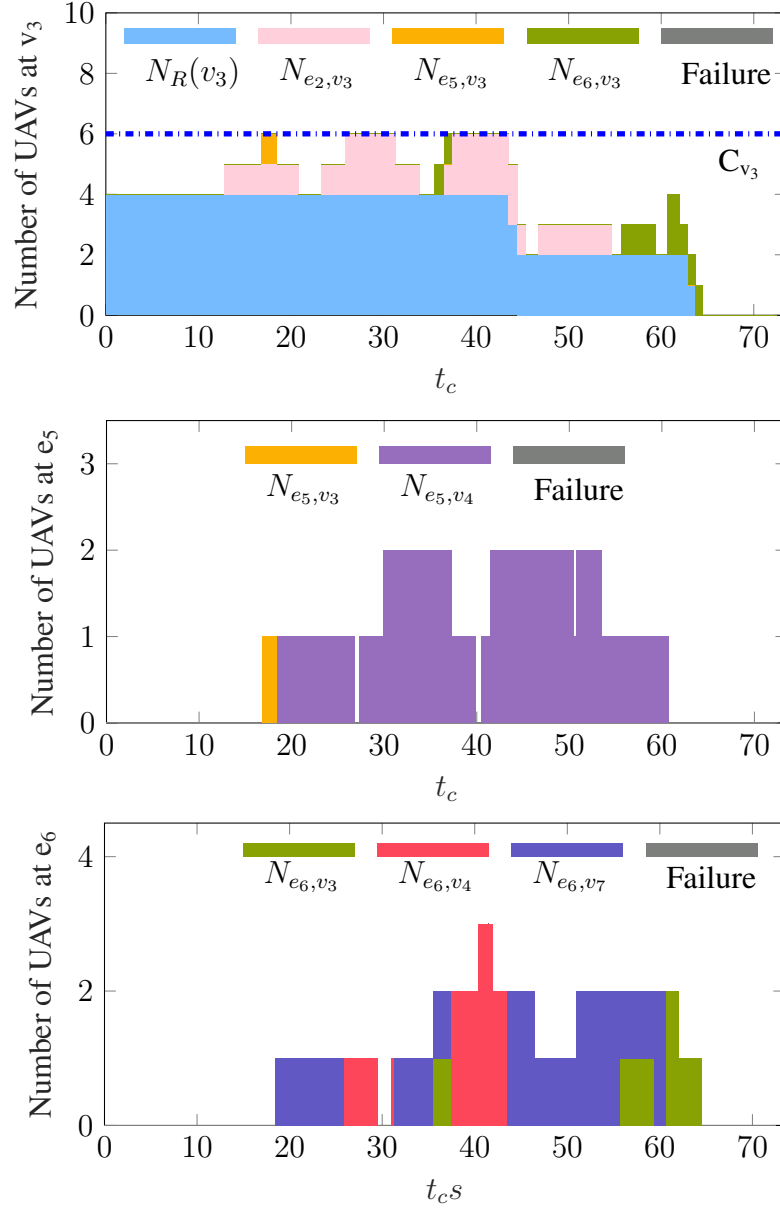


Figure 9.4: Observation of the network when node v_5 is disabled at any time $t_c > 0$ when we adjust the capacity of node v_4 to $C_{v_4} = 8$. (Top) Expected landing-spot occupation at node v_3 . (Middle) Redistribution of flights on link e_5 to its backup nodes. (Bottom) Redistribution of flights on link e_6 to its backup nodes. We simplify the notation $N_R(\cdot, t_c, v_5)$ and $N_{\cdot,}(t_c, v_5)$ as $N_R(\cdot)$ and $N_{\cdot,}$ in the figure.

on this MILP algorithm, which takes 7.3442 seconds to verify, while the algorithm we use with simplification to LP takes only 1.4348 seconds. A 100-flight schedule takes around 40 seconds to verify with the naive MILP formulation, and 8 seconds with the LP algorithm.

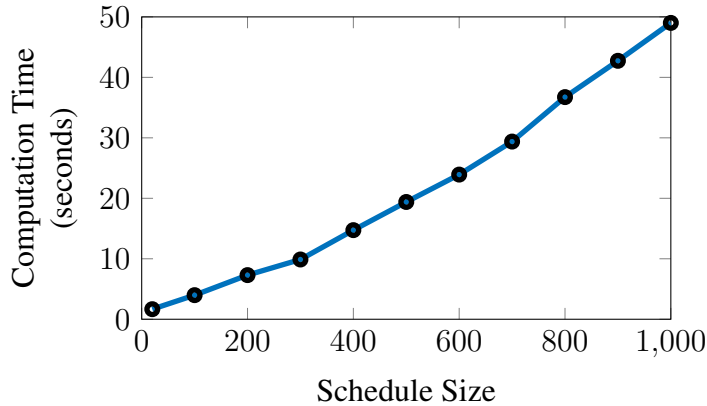


Figure 9.5: The computation time for verifying the worst-case safety of schedules with different sizes. We test on 11 different sets of schedules with sizes from 20 to 1000. The data points demonstrates the $O(n^2)$ computational complexity.

9.5 Concluding Remarks

We studied the safety verification problem for UAM schedules in the face of vertiport closures. We adopted a UAM network model that considers a set of finite-capacity vertiports and links between vertiports with uncertain travel time. If a vertiport is closed at some time, then flights destined for the closed vertiport must be rerouted to one of a set of link-dependent backup nodes. A safety violation occurs if the finite landing capacity at any node is exceeded due to the rerouting.

We considered the travel time uncertainty as a nondeterministic uncertainty, and therefore, we defined appropriate notions of worst-case and best-case safety. We gave necessary and sufficient conditions in both cases. If a given schedule satisfies the conditions for worst-case safety, then it is guaranteed that the schedule will not violate the safety constraints under any possibility of the travel times. On the other hand, if a schedule does not satisfy the conditions for best-case safety, then even if the uncertainty were favorably eliminated from the travel times via, e.g., aggressive low-level motion planning and control schemes, safety violation would still occur, implying the need for a new schedule.

As formulated, these conditions take the form of mixed integer linear programming (MILP) constraints. We then showed that these numerically inefficient MILP constraints

are able to be converted into efficient linear programming (LP) constraints using the theory of totally unimodular matrices (TUMs), resulting in an efficient algorithm for safety verification. We demonstrated our approach through several examples and case studies.

Our modeling approach could further allow other generalizations. For example, we regard a disrupted node as completely malfunctioning, but a partial malfunctioning disruption model, where not all landing spots of the disrupted node are disabled, could also be investigated.

CHAPTER 10

CONCLUSION

Introducing autonomous agents into the existing transportation networks can greatly relieve the congestion problem of ground transportation and enhance the transition efficiency. In this thesis, we focused on two aspects that reasonably integrate autonomy with the existing traffic systems: on one hand, we studied the ride-sharing networks with mixed autonomy where both AVs and HVs are considered; on the other hand, we explored the management of UAVs for UAM networks.

In Chapters 3–5, we considered ride-sharing networks served by HVs and AVs. We proposed a model in Chapter 3 for ride-sharing in this mixed autonomy setting for a multi-location equidistant network under three different ride-assigning schemes: AV, HV and weighted priority assignment. Chapter 4 showed that under equilibrium conditions, for AV and HV priority assignments, the nonconvex profit-maximization problem can be computed efficiently by converting the original problems into alternative convex programs. We were then able to show that all three priority possibilities result in the same maximum profits for the ride-sharing platform in Chapter 5. We also demonstrated that, in certain scenarios, there exist a regime that the maximal profit may only be attained by mixing HVs and AVs, while in other scenarios, HV-only or AV-only operation will be more profitable, depending on the relative cost of AVs. Further, we quantified the conditions for which the mixed autonomy deployment allows for higher profits than a forced AV-only or forced HV-only deployment on star-to-complete networks and demonstrated our results on an example. We observed that the optimal profits for the ride-sharing platform with AV option in the fleet will be the same as that of the human-only network when the cost for operating an AV is relatively high compared to the outside option earnings for drivers' lifetime. Intuitively, we proved theoretically that incorporating the AVs into the fleet would not be optimal for

maximizing the profits for the platform if the cost of operating an AV exceeds the expected compensation to a driver in the system, which was also demonstrated in the case study. The case study also illustrated that even when the cost of operating an AV is less than the expected compensation to a driver in the network, it is not necessarily optimal to use AVs.

We then focused on UAM network in the rest of the thesis. We studied a dynamic model of UAM network with uncertain travel times and limited capacities in Chapter 6, which can be easily turned into a static model. One main challenge for UAM network that makes it substantially different from the general ground transportation network is the limited parking spaces at the nodes in a UAM network. Therefore, a schedule for each flight in the network has to be decided before it takes off to ensure that a parking space is available upon arrival. The key safety constraint in the thesis is to avoid the conflict of parking spaces. For both dynamic network and static network, we presented theoretical results establishing necessary conditions for a schedule to be feasible.

Specifically, in the case of static scheduling, we then demonstrated that the necessary condition is also sufficient for the existence of feasible schedule in star networks, and presented a mixed integer program obtaining an optimal schedule for star-branch networks while satisfying the deadline and safety constraints. However, this static scheduling method could be time-consuming with large demands, and was not suitable for general dynamic scheduling problems. In Chapter 7, we then investigated the dynamic scheduling algorithm for scheduling trips to satisfy safety constraints and arrival deadlines, which is able to update the demands over time and, at the same time, reduce the computation time at each scheduling time.

We then took the intermittent closures of nodes in the UAM networks into account, as similar accidents is likely and it is important to guarantee the safety of the flights under this kind of emergencies. Specifically, we considered the event of vertiport closure. For safety, all in-transit UAVs in the network are required to have backup landing nodes with sufficient landing capacity when a vertiport is disabled. We assigned the link-related backup nodes

for each flight and studied the problem of safety verification of UAM schedules in the face of vertiport closures. Chapter 8 explored the simplest case where only one backup node was assigned to each flight, while Chapter 9 investigated the general case where multiple backup nodes could be assigned to each flight. In both cases, we established safety criteria for best-case and worst-case safety. In the simplest case, we provided sufficient and necessary conditions for a given UAM schedule to be worst-case safe and only necessary conditions for best-case safety. In the meanwhile, we presented sufficient and necessary conditions for both worst-case and best-case safety in the general case. We then developed safety verification algorithm based on the worst-case safety constraints. In the case where multiple backup nodes were allowed for each flight, the indeterminism were introduced and hence the verification process would include the time-consuming MILP solving process. Fortunately, we were able to simplify this MILP problem as a LP problem, which significantly reduce the computation complexity. In both simplest and general cases, the safe verifying algorithm had a computational complexity $O(n^2)$ and hence, allowed for efficient verification even with a large size of UAM schedules. We demonstrated our algorithms and their efficiency on a UAM network with up to 1,000 UAVs.

There are several future directions we are able to explore based on the research methods and results presented in this thesis. First, the UAM network we are currently exploring is taking the interval of uncertain travel time where lower bounds and upper bounds are considered, while the extreme cases are taken into account, which is a key factor that the scheduling problem turns into MILP problems. Solving MILP problem is computationally hard and time consuming, which greatly degrade the performance of the scheduling algorithm. In the contrast, we can consider stochastic travel times, where the travel times falls in a designated distribution. In this stochastic model, the scheduling process may be able to get rid of the MILP problem and instead, can take into account the possibility of safe arrival for each flight. Secondly, for safety verification, we currently consider only one closure of vertiport, while multiple closure can be an interesting expansion in this direction.

Whether the disabled vertiports are connected, strongly connected or not will certainly affect the complexity of the problem. Also, the closure may not happen all at once, hence the order of closure and the exact closing time is another factor that will influence the safety of the given schedule. Another direction, though may be broad, is to explore the regulation of mixed autonomy taking both the ground transportation and the urban airspace into account. As we aim to relieve the congestion of ground transportation and enhance the transit efficiency, it will be useful if we are able to investigate that in which scenario introducing the UAVs or AVs into the transportation system will help.

Appendices

APPENDIX A
PROOF OF THEOREM 5

To prove the theorem, we only need to find the optimal solutions of the optimization problems (4.1), (4.5) or (5.16), divide k values into different regions according to the optimal solutions and find the intersection of k values between different regions.

For convenience, we can first divide the optimal solutions into four possible regions:

1. HV only: $x_i > 0, z_j = 0$ for some i and all j ,
2. Mixed-autonomy: $x_i > 0, z_i > 0$ for some i .
3. AV only without transition: $x = 0, z > 0, r = 0$.
4. AV only with transition: $x = 0, z > 0, r_{ij} > 0$ for some i, j .

Region 1:

Notice that in this region, since $z_j = 0$ for all j , then $d_j \leq x_j$ for all j . As we've shown in the proof of Theorem 1, the optimal solutions for the first region with only HVs can thus be derived from [20] by setting $z_i = r_{ij} = 0$ for all i, j .

Let

$$\beta'_{lim} = \frac{n-1}{2(1-\beta)\beta(n-2)} \left[\beta(1-2\beta) + \sqrt{\frac{\beta^2(n-1) + 4\beta - 4}{n-1}} \right].$$

Obviously, $\beta'_{lim} > 0$ when $(n-1)^{-\frac{1}{3}} < \beta < 1$.

1. When $\xi \in [\frac{\beta(n-1)-1}{\beta(n-2)}, 1]$, $p_i = 1 - \frac{\beta}{2}$, $x_i = 1 - p_i = \frac{\beta}{2}$, $y_{ij} = 0$ for all i, j ; $\delta_1 = \frac{\beta - (n-1)\beta^2}{2}$, $\delta_i = \frac{(n-1)\beta - \beta^2}{4}$ for $i > 1$,

2. When $\xi \in [\max\{\beta'_{lim}, 0\}, \frac{\beta(n-1)-1}{\beta(n-2)}]$, let $Z = -(n-1)c_1$. Then

$$p_i = \frac{1}{2} + \frac{\beta Z(1 + \beta Z + \beta) + (n-1)[1 - \beta c_2(n-2)]}{2(n-1) + 2\beta^2 Z^2}$$

for all $i > 1$, and $p_1 = 1 - \beta c_1(n-1)(1-p_2)$. $x_i = 1 - p_i$, $y_{ij} = 0$ for all i, j ;
 $\delta_1 = 0$, $\delta_i = (1 - \beta c_2 - \beta^2 c_1)(1 - p_i)$ for $i > 1$.

3. When $\xi \in [0, \max\{\beta'_{lim}, 0\})$, $p_1 = \frac{1}{2}$ and $p_i = \frac{1}{2} + \frac{1 - \beta^2 c_1 - \beta(n-2)c_2}{2}$ for $i > 1$.
 $x_i = \sum_j y_{ij} + (1 - p_i)$ for all i . $y_{1j} = \beta c_1(1 - p_2) - \frac{1}{2(n-1)}$ for all $j > 1$ and $y_{ij} = 0$
for all $i > 1$ and all j . $\delta_1 = 0$, $\delta_i = (1 - p_2)(1 - \beta c_2 - \beta^2 c_1)$ for all $i > 1$.

If $\beta \leq \frac{1}{n-1}$, then only the first case exists; if $\frac{1}{n-1} < \beta \leq (n-1)^{-\frac{1}{3}}$, then only first two cases exist; if $(n-1)^{-\frac{1}{3}} < \beta < 1$, then all three cases exist.

For the rest of regions, $z_i > 0$ for some i and given Lemma 4, we can ensure that $y_{ij} = 0$ for all i, j . By Theorem 1, it is without loss of generality for us to compute only the optimal solutions for (4.7) knowing $y_{ij} = 0$ for all i, j . Moreover, under Assumption 4, the optimization problem (4.7) become a quadratic problem with linear constraints. As a result, the KKT conditions are both necessary and sufficient for optimal solutions. Therefore, we first rewrite the simplified optimization problem of (4.7) under Assumption 4 as below

$$\begin{aligned} & \max_{\{p_i, \delta_i, x_i, z_i, r_{ij}\}} \sum_{i=1}^n p_i(1 - p_i) - \omega \sum_{i=1}^n \delta_i - s \sum_{i=1}^n z_i \\ & \text{s.t.} \quad x_i = \beta \sum_{j=1}^n \alpha_{ji} x_j + \delta_i \\ & \quad \quad z_i = \sum_{j=1}^n \alpha_{ji}(1 - p_j - x_j) + \sum_{j=1}^n r_{ji} \\ & \quad \quad \sum_{j=1}^n r_{ij} = z_i - (1 - p_i - x_i) \\ & \quad \quad \delta_i, x_i, z_i, r_{ij} \geq 0 \quad \forall i, j. \end{aligned} \tag{A.1}$$

We denote the dual variables for the three equality constraints as λ_i, μ_i and γ_i , for all $i = 1, \dots, n$; a_i, b_i, c_i and d_{ij} for all $i, j = 1, \dots, n$ are used to denote the four inequality constraints $-\delta_i, -x_i, -z_i, -r_{ij} \leq 0$. Therefore, the KKT conditions for (A.1) can be written as:

$$1 - 2p_i - \sum_{j=1}^n \alpha_{ij} \mu_j - \gamma_i = 0 \quad (\text{A.2})$$

$$-\omega + \lambda_i + b_i = 0 \quad (\text{A.3})$$

$$\sum_{j=1}^n (\beta \lambda_j - \mu_j) - \lambda_i + c_i = 0 \quad (\text{A.4})$$

$$-s - \mu_i - \gamma_i + a_i = 0 \quad (\text{A.5})$$

$$\mu_j + \gamma_i + d_{ij} = 0 \quad (\text{A.6})$$

$$\lambda_i (\beta \sum_{j=1}^n \alpha_{ji} x_j + \delta_i - x_i) = 0 \quad (\text{A.7})$$

$$\mu_i \left(\sum_{j=1}^n \alpha_{ji} (1 - p_j - x_j) + \sum_{j=1}^n r_{ji} - z_i \right) = 0 \quad (\text{A.8})$$

$$\gamma_i \left(\sum_{j=1}^n r_{ij} - z_i + (1 - p_i - x_i) \right) = 0 \quad (\text{A.9})$$

$$a_i z_i = b_i \delta_i = c_i x_i = d_{ij} r_{ij} = 0 \quad (\text{A.10})$$

$$a_i, b_i, c_i, d_{ij} \geq 0 \quad \forall i, j. \quad (\text{A.11})$$

Solving equations (A.2)–(A.11) provides us an optimal solution for (A.1) and thus for (4.2).

Region 2:

$$p_1 = \frac{k(c_1 + 1) + (1 - \beta)c_1 - 1}{2c_1},$$

$$p_i = 1 - \frac{\beta k(c_1 + 1) + \beta c_1}{2} \quad \forall i > 1.$$

$$x_i = \frac{(n-1)c_1(1-p_2)-(1-p_1)}{(n-1)(1-\beta)c_1} \text{ for all } i > 1, x_1 = c_1(n-1)\beta x_2.$$

$$r_{ij} = 0, z_i = 1 - p_i - x_i \text{ for all } i, j.$$

$$\delta_1 = 0, \delta_i = (1 - \beta(1 - c_1))x_2 - \frac{\beta x_1}{n-1} \text{ for } i > 1.$$

Region 3:

$$p_i = \frac{2(n-1)c_1^2 + c_1(k(1-\beta) - 1) + k(1-\beta) + 1}{2[(n-1)c_1^2 + 1]} \quad \forall i > 1,$$

$$p_1 = 1 - (n-1)c_1(1-p_2),$$

and $z_i = 1 - p_i, r_{ij} = 0, x_i = \delta_i = 0$ for all i, j .

Region 4:

$$p_1 = 1/2,$$

$$p_i = \frac{1 + (c_1 + 1)k(1-\beta)}{2} \quad \forall i > 1,$$

and $r_{1j} = c_1(1-p_2) - \frac{1}{2(n-1)} = \frac{c_1 - c_1(c_1+1)k(1-\beta)}{2} - \frac{1}{2(n-1)}$ for all $j > 1$ while $r_{ij} = 0$ for all $i > 1$ and j . $z_1 = (n-1)c_1(1-p_2) = \frac{1-(c_1+1)k(1-\beta)}{2}c_1(n-1)$, $z_i = \frac{1-(c_1+1)k(1-\beta)}{2}$ for all $i > 1$; $\delta_i = 0$ for all i .

Knowing all the optimal solutions of (A.1), we can therefore compute the optimal profits by the objective function. We can then compare the profits of different regions to obtain the boundary values of k that divide different regions. We denote the optimal profits of different regions as π_l and the critical value of k as $k_{l_1 \rightarrow l_2}$, where $l, l_1, l_2 \in 1, 2, 3, 4$ and $l_1 > l_2$. Hence $k_{l_1 \rightarrow l_2}$ represents the lowest value of k that $\pi_{l_2} > \pi_{l_1}$ or optimal solution in region l_1 becomes infeasible. To justify Theorem 5, we need to find out the k values each region will take part in.

$k_{4 \rightarrow 3}$: transition from $r > 0$ to $r = 0$ while $z > 0$.

$$k_{4 \rightarrow 3} = \frac{(n-1)c_1 - 1}{(1-\beta)(n-1)(1+c_1)c_1}. \quad (\text{A.12})$$

There can be profit jump in this transition (since $k_{4 \rightarrow 3}$ is not obtained by profit equality $\pi_3 = \pi_4$ but instead, is the k value when $r = 0$ from region 4).

$k_{3 \rightarrow 2}$, transition of $x = 0$ to $x > 0$ while $z > 0$.

$$k_{3 \rightarrow 2} = \frac{(1 + \beta)c_1 + (n - 1)\beta c_1^3 + 1}{(c_1 + 1)(\beta(n - 1)c_1^2 + 1)}. \quad (\text{A.13})$$

$k_{2 \rightarrow 1}$, transition of $z > 0$ to $z = 0$ while $x > 0$.

$$k_{2 \rightarrow 1} = \begin{cases} 1 & \text{if } \xi \in \left[\frac{\beta(n-1)-1}{\beta(n-2)}, 1\right] \\ \frac{c_1(1+\beta)+(n-1)\beta^2 c_1^3+1}{(c_1+1)((n-1)\beta^2 c_1^2+1)} & \text{if } \xi \in \left[\beta_{lim}, \frac{\beta(n-1)-1}{\beta(n-2)}\right) \\ \frac{1+\beta c_1}{c_1+1} & \text{if } \xi \in [0, \beta_{lim}). \end{cases}$$

$k_{4 \rightarrow 2}$: transition from region 4 to region 2 directly.

$$k_{4 \rightarrow 2} = \frac{1 + \beta c_1}{1 + c_1}. \quad (\text{A.14})$$

$k_{4 \rightarrow 1}$: transition from region 4 to region 1 directly.

$$k_{4 \rightarrow 1} = \frac{1 + \beta c_1}{1 + c_1}. \quad (\text{A.15})$$

Since the $k_{4 \rightarrow 3}$, $k_{4 \rightarrow 2}$ and $k_{4 \rightarrow 1}$ are always real number in $[0, 1]$, then there always exist some values of k such that the optimal solution falls in region 4. Hence region 4 exists for any n, β and ξ . Moreover, when k grows infinitely large so that operating AV is much more expensive than HVs, then obviously the optimal solution will use HVs only for any possible n, β and ξ . Therefore, region 1 also exist for all n, β and ξ . However, as we will show later, there are values of n, β and ξ that the optimal solution of (4.1) will never fall in region 2 or 3 for any k .

Region 3 exists means that region 4's solution transits to region 3 before reaching region

2 or 1. Also, $k_{4 \rightarrow 2} = k_{4 \rightarrow 1}$. Thus region 3 exists only if $k_{4 \rightarrow 3} < k_{4 \rightarrow 2}$. That is,

$$\begin{aligned} \frac{(n-1)c_1 - 1}{(1-\beta)(n-1)(1+c_1)c_1} &< \frac{1+\beta c_1}{1+c_1} \\ (n-1)c_1 - 1 &< (1-\beta)(n-1)c_1(1+\beta c_1) \\ \beta c_1(n-1)(1-c_1+\beta c_1) &< 1. \end{aligned} \tag{A.16}$$

Therefore, if the values of n, β and ξ do not satisfy (A.16), then region 3 does not exist and the optimal solution of (4.1) will transit from region 4 directly to region 2 or 1. With this condition held, region 2 exists only if $k_{4 \rightarrow 2} < k_{2 \rightarrow 1}$ for similar reason as above.

If region 3 exists, then we claim that region 2 must exist because $k_{3 \rightarrow 2} \leq k_{2 \rightarrow 1}$ always.

To demonstrate that, we need to show that $\frac{(1+\beta)c_1+(n-1)\beta c_1^3+1}{(c_1+1)(\beta(n-1)c_1^2+1)} \leq 1$, $\frac{(1+\beta)c_1+(n-1)\beta c_1^3+1}{(c_1+1)(\beta(n-1)c_1^2+1)} \leq \frac{c_1(1+\beta)+(n-1)\beta^2 c_1^3+1}{(c_1+1)((n-1)\beta^2 c_1^2+1)}$ and that $\xi \notin [0, \beta_{lim})$ in this case.

We will show the inequalities through contradiction. Suppose first that

$$\begin{aligned} \frac{(1+\beta)c_1+(n-1)\beta c_1^3+1}{(c_1+1)(\beta(n-1)c_1^2+1)} &> 1 \\ (1+\beta)c_1+(n-1)\beta c_1^3+1 &> (c_1+1)(\beta(n-1)c_1^2+1) \\ \beta c_1 &> \beta(n-1)c_1^2 \\ 1 &> (n-1)c_1. \end{aligned} \tag{A.17}$$

But $(n-1)c_1 = (n-1) + (2-n)\xi \in [0, 1]$, hence $\frac{(1+\beta)c_1+(n-1)\beta c_1^3+1}{(c_1+1)(\beta(n-1)c_1^2+1)} \leq 1$.

Similarly, suppose

$$\begin{aligned} \frac{(1+\beta)c_1+(n-1)\beta c_1^3+1}{(c_1+1)(\beta(n-1)c_1^2+1)} &> \frac{c_1(1+\beta)+(n-1)\beta^2 c_1^3+1}{(c_1+1)((n-1)\beta^2 c_1^2+1)} \\ \beta(\beta c_1+1) &> \beta c_1+1 \\ \beta &> 1. \end{aligned}$$

Hence $\frac{(1+\beta)c_1+(n-1)\beta c_1^3+1}{(c_1+1)(\beta(n-1)c_1^2+1)} \leq \frac{c_1(1+\beta)+(n-1)\beta^2 c_1^3+1}{(c_1+1)((n-1)\beta^2 c_1^2+1)}$.

When region 3 exists, then (A.16) is satisfied. Solving (A.16) gives that

$$\xi > \max \left\{ \frac{n-1}{2(1-\beta)\beta(n-2)} \left[\beta(1-2\beta) + \sqrt{\frac{\beta^2(n-1) + \beta(4\beta-4)}{n-1}} \right], 0 \right\}$$

$$> \beta_{lim}$$

since $\beta \in (0, 1)$. As a result, $\xi \notin [0, \beta_{lim})$ in this case.

As a result, $k_{3 \rightarrow 2} \leq k_{2 \rightarrow 1}$ when (A.16) is satisfied and thus region 2 always exists if region 3 exists.

To summarize, if (A.16) is satisfied, then all four regions exist, while the transition k values are defined above. We then assume that (A.16) is not satisfied, if $k_{4 \rightarrow 2} < k_{2 \rightarrow 1}$, then region 1, 2 and 4 exist while separated by $k_{2 \rightarrow 1}$ and $k_{4 \rightarrow 2}$; if $k_{4 \rightarrow 2} \geq k_{2 \rightarrow 1}$, then only region 1 and 4 exist while separated by $k_{4 \rightarrow 1}$.

Define $k_1 = k_{4 \rightarrow 2}$, $k_2 = k_{2 \rightarrow 1}$, $k_3 = k_{4 \rightarrow 3}$ and $k_4 = k_{3 \rightarrow 2}$.

We can therefore conclude that for the case that $k_3 \geq k_1$, suppose $k_1 < k_2$, if $k \in [0, k_1]$, it is optimal for the platform to use only AVs; if $k \in (k_1, k_2)$, it is optimal to deploy a mixed-autonomy network; if $k \geq k_2$, it is optimal to use HVs only. Suppose $k_1 \geq k_2$, if $k \in [0, k_1]$, it is optimal to use AVs only and if $k > k_1$, it is optimal to use HVs only.

For the case that $k_3 < k_1$, if $k \in [0, k_4]$, it is optimal for the platform to have only AVs; if $k \in (k_4, k_2)$, it is optimal to deploy a mixed-autonomy network; if $k \geq k_2$, it is optimal to use HVs only.

REFERENCES

- [1] W. Mitchell, B. Hainley, and L. Burns, *Reinventing the automobile: Personal urban mobility for the 21st century*. MIT press, 2010.
- [2] S. Feigon and C. Murphy, *Shared Mobility and the Transformation of Public Transit*, Project J-11, Task 21. The National Academies Press, 2016.
- [3] C. Hass-Klau, G. Crampton, and A. Ferlic, *The effect of public transport investment on car ownership: the results for 17 urban areas in France, Germany, UK and North America*. Environmental & Transport Planning, 2007.
- [4] R. Javid, A. Nejat, and M. Salari, “The environmental impacts of carpooling in the United States,” in *Transportation, Land and Air Quality Conference*, Aug. 2016.
- [5] B. McBain, M. Lenzen, G. Albrecht, and M. Wackernagel, “Reducing the ecological footprint of urban cars,” *International Journal of Sustainable Transportation*, vol. 12, no. 2, pp. 117–127, 2018.
- [6] T. Litman, *Autonomous vehicle implementation predictions*. Victoria Transport Policy Institute Victoria, Canada, 2017.
- [7] P. M. Boesch, F. Ciari, and K. W. Axhausen, “Autonomous vehicle fleet sizes required to serve different levels of demand,” *Transportation Research Record*, vol. 2542, no. 1, pp. 111–119, 2016.
- [8] K. Conger, “In a shift in driverless strategy, Uber deepens its partnership with Toyota,” *The New York Times*, Aug. 2018.
- [9] J. Holden and N. Goel, “Fast-forwarding to a future of on-demand urban air transportation,” 2016.
- [10] D. P. Thippavong *et al.*, “Urban air mobility airspace integration concepts and considerations,” in *2018 Aviation Technology, Integration, and Operations Conference*, 2018, p. 3676.
- [11] K. Balakrishnan, J. Polastre, J. Mooberry, R. Golding, and P. Sachs, “Blueprint for the sky,” *The roadmap for the safe integration of autonomous aircraft*. Airbus A, vol. 3, 2018.
- [12] “NextGen Independent Assessment Recommendations,” *McLean, VA: The MITRE Corporation*, 2014.

- [13] B. Lascara, T. Spencer, M. DeGarmo, A. Lacher, D. Maroney, and M. Guterres, “Urban air mobility landscape report: Initial examination of a new air transportation system,” *McLean, VA: The MITRE Corporation*, 2018.
- [14] INRIX, *Electric passenger drones could relieve housing costs and spread growth in nation’s booming cities*, 2019.
- [15] C. Al Haddad, “Identifying the factors affecting the use and adoption of urban air mobility,” 2018.
- [16] E. Ancel, F. M. Capristan, J. V. Foster, and R. C. Condotta, “Real-time risk assessment framework for unmanned aircraft system (UAS) traffic management (UTM),” in *17th AIAA Aviation Technology, Integration, and Operations Conference*, 2017, p. 3273.
- [17] C. Bosson and T. A. Lauderdale, “Simulation evaluations of an autonomous urban air mobility network management and separation service,” in *2018 Aviation Technology, Integration, and Operations Conference*, 2018, p. 3365.
- [18] M. Xue, J. Rios, J. Silva, Z. Zhu, and A. K. Ishihara, “Fe3: An evaluation tool for low-altitude air traffic operations,” in *2018 Aviation Technology, Integration, and Operations Conference*, 2018, p. 3848.
- [19] M. A. Aiello, C. Dross, P. Rogers, L. Humphrey, and J. Hamil, “Practical application of SPARK to OpenUxAS,” in *Formal Methods – The Next 30 Years*, M. H. ter Beek, A. McIver, and J. N. Oliveira, Eds., Cham: Springer International Publishing, 2019, pp. 751–761, ISBN: 978-3-030-30942-8.
- [20] K. Bimpikis, O. Candogan, and D. Saban, “Spatial pricing in ride-sharing networks,” *IDEAS Working Paper Series from RePEc*, 2016.
- [21] Q. Wei, J. A. Rodriguez, R. Pedarsani, and S. Coogan, “Ride-sharing networks with mixed autonomy,” in *American Control Conference*, 2019.
- [22] Q. Wei, R. Pedarsani, and S. Coogan, “Mixed autonomy in ride-sharing networks,” *IEEE Transactions on Control of Network Systems*, vol. 7, no. 4, pp. 1940–1950, 2020.
- [23] Q. Wei, G. Nilsson, and S. Coogan, “Scheduling of urban air mobility services with limited landing capacity and uncertain travel times,” in *2021 American Control Conference (ACC)*, IEEE, 2021, pp. 1681–1686.
- [24] ———, “Capacity-constrained urban air mobility scheduling,” *arXiv:2107.02900*, 2021.
- [25] ———, “Safety verification for urban air mobility scheduling.”

- [26] R. Zhang, K. Spieser, E. Frazzoli, and M. Pavone, “Models, algorithms, and evaluation for autonomous mobility-on-demand systems,” in *2015 American Control Conference (ACC)*, IEEE, 2015, pp. 2573–2587.
- [27] P.-J. Rigole, *Study of a shared autonomous vehicles based mobility solution in Stockholm*, 2014.
- [28] R. Zhang and M. Pavone, “A queueing network approach to the analysis and control of mobility-on-demand systems,” in *2015 American Control Conference (ACC)*, IEEE, Jul. 2015, pp. 4702–4709.
- [29] —, “Control of robotic mobility-on-demand systems: A queueing-theoretical perspective,” *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 186–203, 2016.
- [30] D. J. Fagnant and K. M. Kockelman, “Dynamic ride-sharing and fleet sizing for a system of shared autonomous vehicles in austin, texas,” *Transportation*, vol. 45, no. 1, pp. 143–158, 2018.
- [31] S. Banerjee, R. Johari, and C. Riquelme, “Pricing in ride-sharing platforms: A queueing-theoretic approach,” in *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, ser. EC ’15, ACM, Portland, Oregon, USA, 2015, pp. 639–639.
- [32] G. P. Cachon, K. M. Daniels, and R. Lobel, “The role of surge pricing on a service platform with self-scheduling capacity,” *Manufacturing & Service Operations Management*, vol. 19, no. 3, pp. 368–384, 2017.
- [33] S. Banerjee, D. Freund, and T. Lykouris, “Multi-objective pricing for shared vehicle systems,” *arXiv preprint arXiv:1608.06819*, 2016.
- [34] V. Bulusu, R. Sengupta, E. R. Mueller, and M. Xue, “A throughput based capacity metric for low-altitude airspace,” in *2018 Aviation Technology, Integration, and Operations Conference*, 2018, p. 3032.
- [35] M. Gendreau, G. Laporte, and R. Séguin, “Stochastic vehicle routing,” *European Journal of Operational Research*, vol. 88, no. 1, pp. 3–12, 1996.
- [36] B. C. Dean, “Algorithms for minimum-cost paths in time-dependent networks with waiting policies,” *Networks: An International Journal*, vol. 44, no. 1, pp. 41–46, 2004.
- [37] S. Samaranayake, S. Blandin, and A. Bayen, “A tractable class of algorithms for reliable routing in stochastic networks,” *Procedia-Social and Behavioral Sciences*, vol. 17, pp. 341–363, 2011.

- [38] D. Sun, K. Zhao, Y. Fang, and J. Cui, “Dynamic traffic scheduling and congestion control across data centers based on SDN,” *Future Internet*, vol. 10, no. 7, 2018.
- [39] P. Ioannou and F. d. A. A. Vital, “Optimizing combined truck routing and parking based on parking availability prediction,” No. METRANS Project 17-01, Tech. Rep., 2018.
- [40] A. Kok, E. W. Hans, J. M. Schutten, and W. H. Zijm, “A dynamic programming heuristic for vehicle routing with time-dependent travel times and required breaks,” *Flexible services and manufacturing journal*, vol. 22, no. 1-2, pp. 83–108, 2010.
- [41] K. Albers and F. Slomka, “Efficient feasibility analysis for real-time systems with edf scheduling,” in *Design, Automation and Test in Europe*, IEEE, 2005, pp. 492–497.
- [42] N. Fisher and S. Baruah, “The global feasibility and schedulability of general task models on multiprocessor platforms,” in *19th Euromicro Conference on Real-Time Systems (ECRTS’07)*, IEEE, 2007, pp. 51–60.
- [43] J. K. Lenstra, A. R. Kan, and P. Brucker, “Complexity of machine scheduling problems,” in *Annals of discrete mathematics*, vol. 1, Elsevier, 1977, pp. 343–362.
- [44] G. C. Buttazzo, *Hard real-time computing systems: predictable scheduling algorithms and applications*. Springer Science & Business Media, 2011, vol. 24.
- [45] G. Como, K. Savla, D. Acemoglu, M. A. Dahleh, and E. Frazzoli, “Robust distributed routing in dynamical networks—part I: Locally responsive policies and weak resilience,” *IEEE Transactions on Automatic Control*, vol. 58, no. 2, pp. 317–332, 2013.
- [46] ———, “Robust distributed routing in dynamical networks—part II: Strong resilience, equilibrium selection and cascaded failures,” *IEEE Transactions on Automatic Control*, vol. 58, no. 2, pp. 333–348, 2013.
- [47] M. Løve, K. R. Sørensen, J. Larsen, and J. Clausen, “Disruption management for an airline—rescheduling of aircraft,” in *Workshops on Applications of Evolutionary Computation*, Springer, 2002, pp. 315–324.
- [48] G. Zhu, J. F. Bard, and G. Yu, “Disruption management for resource-constrained project scheduling,” *Journal of the Operational Research Society*, vol. 56, no. 4, pp. 365–381, 2005.
- [49] T. Andersson* and P. Värbrand, “The flight perturbation problem,” *Transportation planning and technology*, vol. 27, no. 2, pp. 91–117, 2004.

- [50] S. Bisailon, J.-F. Cordeau, G. Laporte, and F. Pasin, “A large neighbourhood search heuristic for the aircraft and passenger recovery problem,” *4OR*, vol. 9, no. 2, pp. 139–157, 2011.
- [51] Z. Wu, Q. Cao, B. Li, C. Dang, and F. Hu, “A rapid solving method to large airline disruption problems caused by airports closure,” *IEEE Access*, vol. 5, pp. 26 545–26 555, 2017.
- [52] B. G. Thengvall, G. Yu, and J. F. Bard, “Multiple fleet aircraft schedule recovery following hub closures,” *Transportation Research Part A: Policy and Practice*, vol. 35, no. 4, pp. 289–308, 2001.
- [53] S. Yan and C.-G. Lin, “Airline scheduling for the temporary closure of airports,” *Transportation Science*, vol. 31, no. 1, pp. 72–82, 1997.
- [54] M. F. Argüello, J. F. Bard, and G. Yu, “A grasp for aircraft routing in response to groundings and delays,” *Journal of Combinatorial Optimization*, vol. 1, no. 3, pp. 211–228, 1997.
- [55] M. F. Arguello, *Framework for exact solutions and heuristics for approximate solutions to airlines’ irregular operations control aircraft routing problem*. The University of Texas at Austin, 1997.
- [56] J. M. Rosenberger, E. L. Johnson, and G. L. Nemhauser, “Rerouting aircraft for airline recovery,” *Transportation Science*, vol. 37, no. 4, pp. 408–421, 2003.
- [57] Z. Wu, B. Li, C. Dang, F. Hu, Q. Zhu, and B. Fu, “Solving long haul airline disruption problem caused by groundings using a distributed fixed-point computational approach to integer programming,” *Neurocomputing*, vol. 269, pp. 232–255, 2017.
- [58] Z. Wu, B. Li, and C. Dang, “Solving multiple fleet airline disruption problems using a distributed-computation approach to integer programming,” *IEEE access*, vol. 5, pp. 19 116–19 131, 2017.
- [59] B. Li, C. Dang, and J. Zheng, “Solving the large airline disruption problems using a distributed computation approach to integer programming,” in *2013 IEEE Third International Conference on Information Science and Technology (ICIST)*, IEEE, 2013, pp. 444–450.
- [60] B. Aguiar, J. Torres, and A. J. Castro, “Operational problems recovery in airlines—a specialized methodologies approach,” in *Portuguese Conference on Artificial Intelligence*, Springer, 2011, pp. 83–97.

- [61] K. Novianingsih, R. Hadianti, S. Uttunggadewa, and E. Soewono, “A solution method for airline crew recovery problems,” *INTERNATIONAL JOURNAL OF APPLIED MATHEMATICS & STATISTICS*, vol. 53, no. 4, pp. 137–149, 2015.
- [62] D. Zhang, H. H. Lau, and C. Yu, “A two stage heuristic algorithm for the integrated aircraft and crew schedule recovery problems,” *Computers & Industrial Engineering*, vol. 87, pp. 436–453, 2015.
- [63] B. Zhu, X. L. Cao, Y. Wang, and Q. Gao, “Constraint programming method for crew schedule recovery,” in *Applied Mechanics and Materials*, Trans Tech Publ, vol. 496, 2014, pp. 1788–1791.
- [64] R. Nissen and K. Haase, “Duty-period-based network model for crew rescheduling in european airlines,” *Journal of Scheduling*, vol. 9, no. 3, pp. 255–278, 2006.
- [65] J. Vink, B. F. Santos, W. J. Verhagen, I. Medeiros, *et al.*, “Dynamic aircraft recovery problem-an operational decision support framework,” *Computers & Operations Research*, vol. 117, p. 104 892, 2020.
- [66] C.-H. Chen, F.-I. Chou, and J.-H. Chou, “Multiobjective evolutionary scheduling and rescheduling of integrated aircraft routing and crew pairing problems,” *IEEE Access*, vol. 8, pp. 35 018–35 030, 2020.
- [67] S. Weilant, A. Strong, and B. Miller, “Incorporating resilience into transportation planning and assessment,” RAND Santa Monica, CA, Tech. Rep., 2019.
- [68] Y. Zhou, J. Wang, and H. Yang, “Resilience of transportation systems: Concepts and comprehensive review,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 12, pp. 4262–4276, 2019.
- [69] W. H. Ip and D. Wang, “Resilience and friability of transportation networks: Evaluation, analysis and optimization,” *IEEE Systems Journal*, vol. 5, no. 2, pp. 189–198, 2011.
- [70] A. A. Ganin, M. Kitsak, D. Marchese, J. M. Keisler, T. Seager, and I. Linkov, “Resilience and efficiency in transportation networks,” *Science advances*, vol. 3, no. 12, e1701079, 2017.
- [71] A. A. Ganin, A. C. Mersky, A. S. Jin, M. Kitsak, J. M. Keisler, and I. Linkov, “Resilience in intelligent transportation systems (ITS),” *Transportation Research Part C: Emerging Technologies*, vol. 100, pp. 318–329, 2019.
- [72] A. Cox, F. Prager, and A. Rose, “Transportation security and the role of resilience: A foundation for operational metrics,” *Transport policy*, vol. 18, no. 2, pp. 307–317, 2011.

- [73] G. Loureiro, A. Dias, A. Martins, and J. Almeida, “Emergency landing spot detection algorithm for unmanned aerial vehicles,” *Remote Sensing*, vol. 13, no. 10, p. 1930, 2021.
- [74] D. J. Fagnant and K. Kockelman, “Preparing a nation for autonomous vehicles: Opportunities, barriers and policy recommendations,” *Transportation Research Part A: Policy and Practice*, vol. 77, pp. 167–181, 2015.
- [75] E. Guerra, “Planning for cars that drive themselves: Metropolitan planning organizations, regional transportation plans, and autonomous vehicles,” *Journal of Planning Education and Research*, vol. 36, no. 2, pp. 210–224, 2016.
- [76] B. Grush and J. Niles, *The end of driving: transportation systems and public policy planning for autonomous vehicles*. Elsevier, 2018.
- [77] M. Grant and S. Boyd, *CVX: Matlab software for disciplined convex programming, version 2.1*, <http://cvxr.com/cvx>, Mar. 2014.
- [78] ———, “Graph implementations for nonsmooth convex programs,” in *Recent Advances in Learning and Control*, ser. Lecture Notes in Control and Information Sciences, V. Blondel, S. Boyd, and H. Kimura, Eds., http://stanford.edu/~boyd/graph_dcp.html, Springer-Verlag Limited, 2008, pp. 95–110.
- [79] Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual*, 2022.
- [80] IBM ILOG Cplex, “V12. 1: User’s manual for cplex,” *International Business Machines Corporation*, vol. 46, no. 53, p. 157, 2009.
- [81] A. J. Hoffman and J. B. Kruskal, “Integral boundary points of convex polyhedra,” in *50 Years of integer programming 1958-2008*, Springer, 2010, pp. 49–76.
- [82] I. Heller and C. B. Tompkins, “An extension of a theorem of Dantzig’s,” *Linear inequalities and related systems*, vol. 38, pp. 247–254, 1956.
- [83] N. Megiddo, “Linear programming in linear time when the dimension is fixed,” *Journal of the ACM (JACM)*, vol. 31, no. 1, pp. 114–127, 1984.
- [84] J. Löfberg, “YALMIP: A toolbox for modeling and optimization in MATLAB,” in *In Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.