Column Generation Approaches to the Military Airlift Scheduling Problem

by

Mark J Williams


Submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of

Master of Science in Operations Research

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2014

This material is a declared work of the U.S. Government and is not subject to copyright protection in the United States.
Column Generation Approaches to the Military Airlift Scheduling Problem

by

Mark J Williams

Submitted to the Sloan School of Management on June 2014, in partial fulfillment of the requirements for the degree of Master of Science in Operations Research

Abstract

In this thesis, we develop methods to address airlift scheduling, and in particular the problem of scheduling military aircraft capacity to meet ad hoc demand. Network optimization methods typically applied to scheduling problems do not sufficiently capture all necessary characteristics of this problem. Thus, we develop a new method that uses integer linear programming (IP) with column generation to make the problem more tractable while incorporating the relevant characteristics. In our method, we decompose the problem into two steps: generating feasible aircraft routes, and solving the optimization model. By ensuring that routes are feasible with respect to travel time, ground time, crew rest, and requirement restrictions when we build them, we do not need to encode these characteristics within the IP optimization model, thus reducing the number of constraints. Further, we reduce the number of decision variables by generating only the fraction of feasible aircraft routes needed to find near-optimal solutions.

We propose two methods for generating routes to include in the IP model: explicit column generation and selective column generation. In explicit column generation, all aircraft routes that we could potentially consider including in the model are generated first. Starting with a subset of these routes, we iteratively use reduced cost information obtained by solving a relaxed version of the IP model to choose more routes to add from the original set of routes. In selective column generation, we first generate a small set of feasible aircraft routes. Starting with this set of routes, we iteratively generate more routes by solving a relaxed version of the IP model and then combine routes in the solution together and add those that are feasible to the route set. In both methods, we iterate until there are either no other routes to include or the solution stops improving. Last, we solve the IP model with the final set of routes to obtain an integer solution.

We test the two approaches by varying the number of locations in the network, the number of locations that are wings, and the number of requirements. We show that selective column generation produces a solution with an objective value similar to that of explicit column generation in a fraction of the time. In our experiments, we solve problems with up to 100 requirements using selective column generation. In addition, we test the impact of integrating lines of business while scheduling airlift and show a significant improvement over the current process.

Technical Supervisor: Allison Chang
Title: Technical Staff, MIT Lincoln Laboratory

Technical Supervisor: Mariya Ishutkina
Title: Technical Staff, MIT Lincoln Laboratory

Thesis Supervisor: Cynthia Barnhart
Title: Chancellor
Ford Professor of Engineering
Massachusetts Institute of Technology
Acknowledgments

This thesis would not have been possible without the support of many people. First, and foremost, I must thank my MIT advisor, Cynthia Barnhart, for her guidance and encouragement along the way. I am fortunate to have had the opportunity to learn from such a caring and knowledgeable person and I am grateful for all of her time.

I owe a tremendous amount of credit to my MIT Lincoln Laboratory advisors, Mariya Ishutkina and Allison Chang. I truly appreciate all of their thoughtful feedback and support throughout my research. I have learned a great deal about problem solving from each of them. I hope to see you during my next assignment the next few years.

Thank you to all of the students and faculty that have helped me along the way. I appreciate all of the help in coursework, seminars, and research to get to this point. I am especially grateful to Jon Paynter for his feedback on my thesis.

I thank the U.S. Air Force and MIT Lincoln Laboratory for allowing me the opportunity to pursue this degree. I look forward to serving and hope to make a positive impact.

Last, and certainly not least, I am forever indebted to my family, especially my parents, for their love and support in all my endeavors. I especially thank Kristen for her love and patience the past several years.

The views expressed in this article are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.
# Contents

1 Introduction .................................................. 10  
1.1 Contributions ............................................... 11  
1.2 Thesis Structure ............................................. 11  

2 Background and Literature Review ......................... 13  
2.1 Background .................................................. 13  
2.2 Current Planning Process ................................. 16  
  2.2.1 Strategic Planning ..................................... 16  
  2.2.2 Mission Planning ....................................... 17  
  2.2.3 Sources of Uncertainty ................................ 18  
2.3 Proposed Planning Process ............................... 18  
  2.3.1 Problem Definition and Assumptions .................. 20  
    2.3.1.1 Network ........................................... 21  
    2.3.1.2 Requirements .................................... 21  
    2.3.1.3 Aircraft and Aircrew ............................. 22  
2.4 Literature Review .......................................... 22  
  2.4.1 The Network Design Problem ........................... 23  
  2.4.2 The Pickup and Delivery Problem ..................... 23  
  2.4.3 Airlines ............................................... 24  
  2.4.4 Related Efforts ....................................... 25  

3 Arc-based Formulation ....................................... 27  
3.1 Related Work .............................................. 27  
3.2 Time-expanded Network ................................... 28  
3.3 Arc-based Model .......................................... 29  
  3.3.1 Sets ................................................... 30  
  3.3.2 Parameters ............................................ 31  
  3.3.3 Decision Variables ................................... 32
5.1.3 Aircraft and Aircrew ........................................ 67
5.1.4 Costs ....................................................... 67
5.2 Route Generation ............................................... 68
5.3 Model Scaling .................................................. 71
5.4 Integrated versus Segregated Scheduling ...................... 73

6 Conclusion ......................................................... 75
  6.1 Summary of Results and Contributions ...................... 75
  6.2 Future Work .................................................. 76

A Logic for POTENTIAL ............................................. 78
B Logic for SEQUENCE ............................................ 79
C Logic for FEASIBLE ............................................. 81
D Logic for DOMINANT ............................................ 92
E Port Locations .................................................... 93
List of Figures

2-1 US Transportation Command organization .................................................. 14
2-2 Organization of the 618th Air and Space Operations Center (TACC) .............. 15
2-3 Scheduling timeline for a typical requirement ................................................ 17
2-4 Hypothetical probability density function and cumulative density function describ-
ing the likelihood a requirement will change depending on the number of days until
execution. ............................................................................................................. 19
2-5 Hypothetical histogram of the number of requirements known based on the number
of days until execution .......................................................................................... 20
3-1 An example of a dynamic network where the labels on the arc represent travel times 28
3-2 Time-expanded version of the network in Figure 3-1 ...................................... 29
3-3 Time-expanded network with a sink node and artificial arcs for one commodity and
one wing at port D ................................................................................................. 32
4-1 This path covers requirements 1 and 2 and is not empty until the final stop ........ 38
4-2 This route consists of two paths with an empty flight segment in between and it begins
and ends at the same wing ................................................................................... 38
4-3 Sequence of a possible path with the travel times between ports ..................... 42
4-4 A simplified example of the impact limiting paths by the number of requirements has
in the case of delays ............................................................................................... 45
4-5 Limiting the number of possible route combinations based on proximity. In this
example, there are only five possible locations reachable in one move from the location
in the center. ......................................................................................................... 46
4-6 Once a partial route length exceeds the threshold, we add the original route if its
origin and destination are closer together than the origin and destination of the new
route. Otherwise, we continue to add paths to the new route. .............................. 49
4-7 We will try to complete a partial route if the origin and destination are in close
proximity regardless of partial route length ......................................................... 49
4-8 Column generation approach .................................................. 54

5-1 Ports considered for use in our experimental network .................. 65

5-2 Average time to build paths based on the number of requirements in the path .... 68

5-3 Average time required to generate paths and routes based on how we limit paths .. 70

C-1 Sequence of a possible path with travel times ............................ 89
List of Tables

4.1 Requirement data ................................................................. 41
4.2 Earliest arrival and departure times ........................................ 42
4.3 Inflated latest arrival and departure times .................................. 43
4.4 Latest arrival and departure times ............................................ 43

5.1 Per unit cargo type specifications ............................................. 66
5.2 Partial aircraft cargo mix ...................................................... 67
5.3 Limiting paths has minimal affect on solution quality and reduces the computational
time necessary to generate routes dramatically .............................. 69
5.4 Comparison of explicit and selective column generation for small test problems ... 72
5.5 Summary of results for selective column generation in large problems .............. 73
5.6 Comparison of scheduling requirements aggregated with scheduling requirements by
LOB ................................................................................. 74

C.1 Requirements data ............................................................. 89
Chapter 1

Introduction

The United States Transportation Command (USTRANSCOM) is responsible for a global supply chain comprising multiple modes of transportation and numerous stakeholders. The airlift network is a vital portion of their supply chain. Airlift requests, called requirements, vary from Presidential movements to natural disaster relief and the lead time to schedule such requests range from hours to months. Each requirement specifies a pick up location, delivery location, time windows restricting when the cargo can be picked up and delivered at each location, as well as size and type of cargo to be transported.

Scheduling these requirements is challenging for a couple of reasons. First, the problem is highly dynamic in terms of both demand and resources. Second, the volume of requirements makes it difficult to schedule. Third, identifying requirements with potential to be scheduled together is challenging for a human because the opportunities depend on the time and locations of the pick up and delivery location as well as the size of the cargo to be transported [15].

Requirements are currently scheduled in a manual process by flight planners. Flight planners are divided by lines of business (LOBs), each of which schedule requirements of one type because of the large number of requests. As a result, flight planners have no knowledge of other LOBs even though they are utilizing the same set of limited aircraft, crew, and airport resources. In addition, flight planners wait to schedule requirements until two weeks prior to execution because the requirement data tends to stabilize at this point.

In this thesis, we propose a scheduling framework based on optimization in order to produce a single airlift schedule and utilize the limited resources efficiently. We develop an integer linear programming formulation that is able to aggregate requirements both within and between LOBs to schedule all requirements at once. Furthermore, we implement two column generation approaches to reduce the size and complexity in order to address the large number of airlift requirements.
1.1 Contributions

We make the following contributions in this thesis:

- We develop an algorithm to generate aircraft routes that are feasible in terms of travel time, ground time, and requirement time restrictions in order to reduce the size and complexity of the optimization model. We demonstrate that the use of dominance criteria to eliminate routes that serve the same requirements and utilize the same aircraft type does not have a significant impact on solution quality.

- We develop the deterministic Route-based model, an integer linear programming (IP) optimization model that is able to aggregate all types of requests to create a baseline schedule. In addition, we present an alternative, the Flexible model, by modifying the Route-based model to allow for flexibility in the departure and arrival days for aircraft.

- We implement two column generation approaches to further reduce the size of the optimization model: explicit column generation and selective column generation. We show that neither of the two approaches consistently performs better in terms of solution cost, but selective column generation produces a solution in a fraction of the time compared to explicit column generation. As a result, we are able to solve instances with up to 100 requirements using selective column generation.

- While we implement techniques to produce a schedule resilient to change, we provide the framework to develop a robust Route-based model to incorporate the level of uncertainty in the problem.

1.2 Thesis Structure

The organization of this thesis is as follows.

In Chapter 2, we motivate the problem by describing the current scheduling process and propose a scheduling framework suitable for the characteristics of the problem while allowing the opportunity for optimization. We introduce relevant terminology while describing the available data and the necessary constraints that characterize the military airlift scheduling problem. In addition, we review the pertinent literature and relate it to our problem.

We introduce traditional network formulations and relate relevant work in the literature to our problem in Chapter 3. We develop an optimization model based on this using a time-expanded network. We conclude by discussing the tradeoffs with using this type formulation.

In Chapter 4, we reformulate our model to tackle the challenges from using a traditional network formulation. We begin by relating relevant approaches taken in the literature to our problem. Next,
we describe our algorithm to generate feasible aircraft routes and the corresponding optimization model. In addition, we explain the two column generation approaches we employ and formulate an alternative optimization model.

We design computational experiments to test the performance of our approaches in Chapter 5. We report on how well our model scales and discuss the tradeoffs in using different parameters to reduce the solve time. In addition, we evaluate the benefit of aggregating request types in creating aircraft routes.

Finally, we conclude by summarizing our keys results and suggest future work to build upon our contributions.
Chapter 2

Background and Literature Review

The United States Transportation Command (USTRANSCOM) is responsible for a global supply chain comprising multiple modes of transportation and numerous stakeholders. The airlift network is a vital portion of their supply chain. Airlift requests vary from Presidential movements to natural disaster relief and the lead time to schedule such requests range from hours to months. Scheduling is challenging due to the number of requests, the size of the transportation network, and the many practical constraints unique to the problem. Creating an efficient schedule is necessary to provide timely airlift to support these critical missions.

The remainder of the chapter is organized as follows. We provide general background information on USTRANSCOM in Section 2.1. In Section 2.2, we detail their current planning process and the challenges presented by the military airlift scheduling problem. We propose a planning process to overcome those challenges in Section 2.3. We conclude with a brief review of the pertinent research in the literature describing solution approaches to similar problems in Section 2.4.

2.1 Background

USTRANSCOM is a Department of Defense (DOD) unified command. A unified command is a military organization with a “broad continuing mission under a single commander and composed of significant assigned components from two or more Military Departments” [18]. The DOD organizes unified commands by their geographic responsibility or by the function they provide to the geographic Combatant Commands (COCOMs). USTRANSCOM is a functional command with a mission to “provide full-spectrum global mobility solutions and related enabling capabilities for supported customers’ requirements in peace and war” [34].

USTRANSCOM relies on multiple modes of transportation to provide those mobility solutions. Figure 2-1 depicts how USTRANSCOM is organized. It consists of two subordinate commands—the Joint Enabling Capabilities Command (JECC) and the Joint Transportation Reserve Unit (JTRU)—
as well as three component commands—Surface Deployment and Distribution Command (SDDC), Military Sealift Command (MSC), and Air Mobility Command (AMC). This thesis focuses on the airlift portion of operations, which is largely the purview of the 618th Air and Space Operations Center, primarily known as the Tanker Airlift Control Center (TACC), and Air Force wings.

USTRANSCOM partners with commercial carriers to provide additional capacity. This is done using the Civil Reserve Airlift Fleet (CRAF) for air transportation. Commercial airlines participating in CRAF commit a portion of their passenger and/or cargo fleet to augment military airlift in emergencies [25]. Once notified by TACC, the carriers have 24 to 48 hours to have their aircraft ready. In exchange, the government makes peacetime business available to those airlines.

USTRANSCOM is responsible for the distribution and deployment of DOD personnel and cargo. Because we focus on airlift operations, the relevant USTRANSCOM responsibilities include [34]:

- Partner with the geographic COCOMs in the planning and execution of military support to stability operations, humanitarian assistance, and disaster relief, as directed;

- DOD’s single manager for transportation responsible for providing common-user (i.e., other than service-unique or theatre-assigned assets) and commercial air, land, and sea transportation; terminal management; and aerial refueling to support the global deployment, employment, sustainment, and redeployment of US forces; and

- DOD Distribution Process Owner (DPO) responsible for coordinating and overseeing the DOD distribution system to provide interoperability, synchronization, and alignment of DOD-wide, end-to-end distribution and developing and implementing distribution process improvements that enhance the defense logistics and Global Supply Chain Management System.
In fiscal year 2012 (FY12), AMC executed 31,181 airlift missions delivering 1,882,495 passengers and 659,176 short tons (stons) of cargo to carry out those responsibilities [34]. Distribution involves moving sustainment supplies and accounts for approximately sixty percent of all AMC movements [26]. Although the distribution process can be difficult to forecast, it is much more similar to commercial supply chain networks than deployment. AMC classifies distribution movements as Channel missions. Deployment encompasses moving forces and equipment. Forecasting deployment movements is nearly impossible due to the highly dynamic environment. In general, AMC classifies deployment movements as either Contingency missions or Special Airlift Assignment Missions (SAAMs).

TACC is the AMC unit that is responsible for planning and scheduling all missions in support of USTRANSCOM customers. Their mission is “to make Global Reach a reality by transforming requirements into executable and effective missions, through efficient planning, tasking, execution, and assessment of global air mobility operations” [27]. TACC is comprised of eight directorates as shown in Figure 2-2. The directorates responsible for scheduling airlift missions include the Global Channel Operations Directorate (XOG), Current Operations Directorate (XOO), Global Readiness Directorate (XOP), and Mobility Management Directorate (XOB). The Command and Control Directorate (XOC) oversees all missions beginning 24 hours prior to execution through completion. The Director of Operations (XOZ) provides oversight and the single point of contact for AMC. The Mission Support Directorate (XON) and Global Mobility Weather Operations Directorate (XOW) provide support services.

The military aircraft TACC employs to provide the necessary airlift capacity include the C-5
Galaxy, KC-10 Extender, C-17 Globemaster III, C-130 Hercules, and KC-135 Stratotanker. These aircraft and the crews that fly them are located at individual Air Force bases called wings. There is an AMC wing of C-17’s, for example, located at Travis Air Force Base in California. Each wing comprises one specific aircraft type.

AMC had a planned operating budget of over $10B in FY12, making it comparable to some of the largest US companies [34]. In addition, AMC’s budget accounted for approximately 75% of the entire USTRANSCOM budget that year. Air transportation provides a critical service to facilitate timely US response to geopolitical and natural disaster events around the world. In FY12, AMC delivered humanitarian aid to Turkey to assist in earthquake recovery efforts, evacuated wounded Libyan personnel to medical facilities in Europe and the US, and moved cargo to support contingency operations in Northern Africa. In addition to these, AMC continued to support deploying, redeploying, and sustaining US and coalition troops around the globe.

2.2 Current Planning Process

Planning at USTRANSCOM can be described in two distinct, yet dependent processes. We refer to the process by which USTRANSCOM uses projected demand to develop a long-term transportation plan that includes contracting commercial partners as strategic planning. We refer to the process of creating an executable schedule using military aircraft based on realized demand as mission planning.

2.2.1 Strategic Planning

The planning cycle is approximately six months long and begins with the Force Flow Conference. Service leaders, COCOMs, and planners from USTRANSCOM meet to create a fluid transportation plan based on the Joint Chiefs of Staff (JCS) current priorities. Each commander comes to the meetings with generic data of the requirements they need based on their expectations for the next six months. Given the limited resource capacity and infrastructure, not all commanders’ requirements are feasible for USTRANSCOM to handle. As a result, the commanders must negotiate USTRANSCOM support for as many of their requirements as possible. USTRANSCOM uses data visualization software and various simulation tools to analyze the aggregated requirements and what-if scenarios to determine the feasibility of potential plans. This thesis focuses on the airlift portion of operations, so we only describe the airlift planning process.

At the conclusion of the Force Flow Conference, USTRANSCOM has a transportation plan for the next six months. COCOMs must validate airlift requirements at least twenty-one days prior to the earliest arrival date (EAD) at the port of debarkation (POD). Once this occurs, an Air Feasibility Officer at USTRANSCOM determines if the requirement is transportation feasible. This decision involves determining whether aircraft and airfields are capable of transporting the requirement by
air. This usually occurs anywhere between one week and one month prior to the start of the mission depending upon when the requirement is validated. Once the mission is determined to be transportation feasible, AMC flight planners at the TACC coordinate scheduling the mission.

### 2.2.2 Mission Planning

Scheduling a mission involves several different planners depending upon the type of mission and the type of aircraft required. Figure 2-3 shows the progression of a typical requirement in the scheduling process and the various stakeholders involved. Currently, the division of flight planners is as follows: the XOG directorate plans all channel missions, the XOP directorate plans all contingency missions, and the XOO directorate plans all SAAM missions. Furthermore, each of these directorates only has knowledge of the requirements coming through their respective directorates—they do not know the volume or size of requirements entering the other two directorates.

Each of these offices works with a barrel planner in the XOB directorate, who tasks aircraft to the requirements. Each barrel is responsible for one specific aircraft type; therefore, they have knowledge of all types of requirements requesting their type of aircraft but do not have any knowledge of requirements requesting other types of aircraft.

The flight planners are responsible for planning the mission from the port of embarkation (POE) to the port of debarkation (POD) including all intermediate stops and/or air refuelings. The flight planners ensure the plan includes sufficient time to perform all necessary work on the ground at intermediate stops to include (un)loading, refueling, maintenance, crew rest, etc. In addition, they ensure the maximum on ground (MOG) requirements at each stop are feasible at that time and location, including parking and taxiing availability. Finally, the flight planners coordinate any lead time requirements including prior permission required (PPR) and diplomatic clearances (DIPs). Some airports have a PPR requirement that requests planners to give notice up to 72-hours prior to landing. The rules regarding the lead time and whether a country requires a DIP vary depending on the POE/POD, type of cargo the aircraft is transporting, and the country. If the flight planners do not request the DIP in time, they must either re-route the mission around the country or the
mission is automatically delayed by the number of days the DIP was late.

Requirements filter into the flight planners’ offices once they have been deemed transportation feasible. Generally, flight planners will not begin to plan the mission until approximately two weeks prior to the start of the mission regardless of when it enters their office because the requirement data tends to stabilize after this point. After this point, however, the flight planners schedule missions on a first in, first out (FIFO) basis. After analyzing the requirement data, the flight planner will request the number and type of aircraft from the corresponding barrel in the XOB directorate. The XOB process could take anywhere from a couple of hours to a few days depending on the current work load. If the aircraft is available, the flight planner schedules the rest of the mission while the individual Air Force wings assign the actual tail number of the aircraft and crews to the mission. If the aircraft is not available, the flight planner could try to request a different type of aircraft or send it back to the COCOM. The COCOM can attempt to raise the priority of the mission in order to take aircraft from another mission. At any point in time, if a higher priority mission comes up that requires the assets the flight planner will decide which mission to take the assets from. If a requirement comes up within 24 hours of execution, the TACC command and control directorate (XOC) will coordinate with the XOB and flight planners to determine how to allocate assets.

2.2.3 Sources of Uncertainty

The mission planning process contains many sources of uncertainty. These sources manifest themselves in two types of uncertainty. The first uncertainty is in demand. World events often lead to changes in the JCS priorities and COCOM requirements, creating a highly dynamic environment for AMC. These changes lead to changes in requirement demand at all points in the mission planning process—both in volume and additions or deletions. Figure 2-4 presents a hypothetical probability density function and cumulative density function describing the likelihood that a requirement will change based on the number of days until execution that is consistent with flight planners scheduling requirements two weeks prior to execution.

The second type of uncertainty is in the physical network. This uncertainty lies in the time to fly between ports and the time required to complete ground activities. Delays can be attributed to maintenance issues, weather, or a myriad of other possibilities.

2.3 Proposed Planning Process

In the current mission planning process, flight planners are segregated by requirement type and barrel planners are segregated by aircraft type. Stovepiping each of these directorates yields a more manageable problem for planners, but creates opportunities for inefficiency. Planning missions in this way makes it difficult to take advantage of synergies between requirement types. A single
Figure 2-4: Hypothetical probability density function and cumulative density function describing the likelihood a requirement will change depending on the number of days until execution.

Aircraft could not, for example, deliver a channel requirement with a POE in New York and a POD in Los Angeles and a SAAM requirement with a POE in Los Angeles and a POD in New York even if one began after the other. In addition, flight planners generally schedule requirements in a FIFO manner because manually scheduling multiple missions at once is cumbersome. Even though they look for requirements to schedule for the return trip, it is challenging to revisit past decisions to identify opportunities for improvement in a manual fashion.

We propose to combine the XOB, XOG, XOO, and XOP in a central planning process in order to aggregate all requirement types and create one schedule. In this way, we can utilize optimization to create a baseline schedule that is otherwise challenging for human planners to generate, especially in a limited timeframe. Flight planners can then use their expert knowledge to adapt the baseline schedule to identify flexible solutions that can meet the changing environment rather than identifying a baseline schedule manually.

A baseline schedule consists of a set of routes where each route serves one or more requirements. Each route has a minimum lead time necessary for flight planners and wings to finalize the route and task resources. However, a route may require additional lead time due to PPR and DIP stipulations depending on the cargo and ports visited on the route. The lead time for a route determines when we must ‘lock’ the requirements it serves and the resources it uses. While solving subsequent optimization models, we do not consider locked requirements and we adjust the necessary resources to account for locked resources. As a result, we lock routes with a minimum lead time threshold before creating the next baseline schedule. This process allows us to address the uncertainty in demand by scheduling a requirement as late as possible.

Because it is such a dynamic problem, we could solve the optimization model continuously. This is not practical from an operational or a computational standpoint. Instead, we propose solving the optimization model on a recurring (e.g., daily) basis. Solving the optimization model on a recurring
schedule provides flight planners a convenient way to translate the baseline schedule into executable missions based on necessary lead times.

The value of optimization is the ability to schedule many requirements at once while taking advantage of synergies in requirement data. If there were no uncertainty in demand, we would use all known requirement data to solve the optimization model. Because there is uncertainty in demand, we must determine which requirements to include. Figure 2-5 presents a possible histogram showing the number of requirements known to the flight planners with the number of days until their execution. We must balance the value of using requirement data to create efficient routes with the cost of replanning if the data changes. For example, using the probability function in Figure 2-4 it might make sense to use the requirement data for those executing in the next ten days and ignoring the other requirement data because those requirements are likely to change. In addition, including more requirements makes it more difficult to solve the optimization model computationally.

2.3.1 Problem Definition and Assumptions

We refer to the problem of creating a schedule based on airlift requirement data as the military airlift scheduling problem. The goal of the military airlift scheduling problem is to assign aircraft capacity to routes in order to meet demand at minimum cost. Therefore, the output of our optimization model will consist of the number of aircraft assigned to each route, the sequence of stops along the route, the departure time at each stop along the route, and the amount of all requirements assigned to each route. The key aspects of the military airlift scheduling problem are the network, demand, and resources.
2.3.1.1 Network

The airlift network consists of a series of interconnected aerial ports. Some of these ports are the locations for Air Force wings. A wing is the home location for a specific aircraft type. Each aircraft type has multiple wings and some ports may be home to multiple wings.

Aircraft land and take off at ports to (un)load cargo and refuel. We assume that the capacity for storing cargo at a port is infinite. Each ground activity requires a minimum amount of time depending on the aircraft type. The number of aircraft at each port is restricted by the number that port can service in terms of taxiing and parking space. These restrictions are known as maximum on ground (MOG) constraints. In addition, each port may only operate during certain hours. For simplicity, we assume each port is open 24 hours per day in this thesis. However, constraining the operating hours for ports would likely reduce the solution space as the number of feasible aircraft routes would be reduced.

The minimum travel time between ports depends on the aircraft type and we assume this is known from historical data. Intermediate stops are necessary to refuel when the distance between two ports exceeds an aircraft’s range. Aerial refueling is a possibility for cargo aircraft making their effective range global. However, we do not consider aerial refueling to limit the size of the solution space and create a more tractable problem. Instead, we leave aerial refuelings as an option for flight planners to add to the baseline schedule created by the optimization model to shape it into an executable schedule with flexibility to withstand possible changes.

2.3.1.2 Requirements

Customer demand in the airlift scheduling setting is referred to as a requirement. Each requirement specifies a POE and POD in the airlift network as well as time restrictions. We assume a requirement stipulates an available to load date (ALD) at the POE and an earliest arrival date (EAD) and latest arrival date (LAD) at the POD. In addition, each requirement specifies the quantity and type of cargo to be transported.

In military airlift, cargo is described as either bulk, oversized, outsized, or passenger. Bulk cargo fits on a standard 463L pallet (104” x 84” x 96”) and can be transported on any cargo aircraft type. Oversized cargo is a single item that exceeds the dimensions of a standard pallet, while outsized cargo is a single item that exceeds 1000” in length, 117” in width, and 105” in height [3]. A cargo type, therefore, is any cargo with the same dimensions and weight. A requirement could demand multiple cargo types. We refer to a requirement-cargo type pair as a commodity. For example, an AH-64 Apache helicopter and a M1 Abrams tank are both described as outsized cargo, but each has distinct weight and dimensions. As a result, they are distinct commodities. Oversized and outsized cargo can only be transported on certain aircraft depending on their dimensions.

In general, flight planners know the total weight of each cargo type for a requirement. However,
aircraft capacity depends on the total number of passengers, the total weight measured in short tons (stons), and the total footprint (area) measured in pallet positions. For example, an aircraft transporting tank treads is likely to run out of weight capacity before footprint capacity while an aircraft transporting helicopters is likely to run out of footprint capacity before weight capacity. As a result, we assume flight planners are able to translate requirement cargo data to commodities each with a weight and footprint.

Certain commodities can be classified as hazardous material, like ammunition and explosives. Flight planners must ensure that all cargo onboard an aircraft is compatible. We assume we know whether every pair of requirements are compatible to be onboard an aircraft simultaneously.

2.3.1.3 Aircraft and Aircrew

Aircraft and aircrew are the resources that carry out the airlift missions. Because we are not considering aerial refuelings, the relevant aircraft types include the C-130, C-17, and C-5. Each aircraft type has a specific cargo capacity and range. The capacity is in terms of passengers, weight in stons, and footprint in pallet positions.

Each AMC wing allocates a certain number of its aircraft on a daily basis for mobility purposes while reserving the rest for other uses including training. As a result, the number of aircraft assigned to a route on a specific day must be less than the number allocated. In addition, each aircraft must return to its home wing at the end of its route.

There are two types of aircrew. A basic aircrew is the minimum aircrew required and an augmented aircrew is a combination of two basic crews. There are two types of restrictions on an aircrew. An aircrew is limited to the number of working hours including ground activities by the crew duty day. An aircrew must take the minimum rest time called crew rest at a port to reset their crew duty day. An augmented crew extends the maximum crew duty day of a basic crew. Furthermore, aircrew tend to stay with the same aircraft throughout the route. In this thesis, we assume all aircrews are augmented and that we are limited by available aircraft.

2.4 Literature Review

We introduce the general approaches to solving problems similar to the military airlift scheduling problem in this section and further discuss their applications in the relevant chapters. In addition, we compare the military airlift scheduling problem to the scheduling problem airlines typically encounter. We finish by describing solution approaches (in the literature) for variants of the military airlift scheduling problem.
2.4.1 The Network Design Problem

The network design problem arises in many applications like transportation and communications. Magnanti and Wong (1984) describe the general network design problem and survey the variations and solution approaches in the literature. The network design problem is applicable in making strategic decisions like fleet acquisitions as well as tactical decisions like scheduling the effective use of the fleet similar to the military airlift scheduling problem.

In the network design problem, there are two types of decisions to be made. The design decisions assign capacity to edges in the network and the operating decisions assign commodities to those edges facilitating the necessary commodity flow. The objective then is to minimize the fixed cost of the design decisions and the variable cost of the operational decisions. The general network design problem includes the classic conservation of flow constraints as well as constraints that couple the design and operational decisions ensuring commodities only flow on the capacity assigned to each arc. The general network design problem assumes the operational decisions are continuous which is not the case in the military airlift scheduling problem because the types of cargo cannot be split arbitrarily.

The general network design problem can be formulated as a multicommodity flow problem where each commodity has a given origin and destination like the requirements in the military airlift scheduling problem. In addition, the military airlift scheduling problem is capacitated because there is not infinite aircraft capacity to transport requirements.

One of the successful optimization techniques to solve the network design problem uses Benders decomposition. One applies Benders decomposition iteratively by first setting the design variables to create a network, solving a routing problem to assign the operating decisions, and using the routing solution to create a new network. This solution provides a valid upper bound. After solving the routing problem, one adds Benders cuts that are inequalities representing the greatest improvement to the design problem based on the dual variables, which provides a valid lower bound. This process iterates until the gap between the upper and lower bounds is within some tolerance threshold.

There are numerous applications of the network design problem in the literature, especially in transportation. We discuss a few of these and relate them to the military airlift scheduling problem in Section 3.1.

2.4.2 The Pickup and Delivery Problem

There has been a considerable amount of work done on the pickup and delivery problem in recent years. Savelsbergh and Sol (1995) present a general pickup and delivery problem (GPDP) model that can be applied to many complex problems in practice, and provide a survey of the special cases of the general problem and solution approaches for each of these. In the GPDP, a set of vehicles,
each with a given start and end location and capacity, must satisfy a set of transportation requests, each with a given start and end location and load size. In addition, each request must be transported by a single vehicle without transshipment at intermediate locations. The objective of the GPDP is to determine the minimum cost pickup and delivery plan made up of multiple pickup and delivery routes.

There are three special cases of the GPDP in the literature: the pickup and delivery problem (PDP), the dial-a-ride problem (DARP), and the vehicle routing problem (VRP). In the PDP, all vehicles depart from and return to a central depot. The DARP is a PDP in which the requests represent people so every load size is one. The VRP is a PDP in which either all of the requests’ origins or destinations are located at the depot. The military airlift scheduling problem is best described as a PDP with multiple depots.

Transportation requests can be characterized as either static or dynamic. In the static case, all transportation requests are known prior to making the routes. In contrast, in the dynamic case, some requests are not known until during execution. As a result, one or more routes must be adjusted to serve the new requests. Savelsbergh and Sol (1995) note that dynamic problems are often solved as a sequence of static problems in practice. The military airlift scheduling problem is a dynamic problem and we propose solving a sequence of static problems.

Many practical applications of the PDP have constraints related to time. The pickup and delivery problem with time windows (PDPTW) is one in which the service at a location must occur in a given time window. In many practical applications, there are multiple time windows at each location because service can only occur during normal operating hours. In the military airlift scheduling problem, service at a pickup location must occur on or after ALD and the service at the delivery location must occur between EAD and LAD; furthermore, many airports do not operate 24 hours per day creating multiple time windows at each location.

In Section 4.1, we relate solution approaches to the PDP in the literature to the military airlift scheduling problem.

2.4.3 Airlines

Optimization has been applied with much success in the airline industry. In general, the airlines solve a series of optimization models sequentially to design the network, assign the fleet to the network, and assign crew to flights for planning purposes months ahead of execution. In execution, they rely on different strategies to adapt the optimal schedules in case of irregular operations.

Barnhart, Belobaba, and Odoni (2003) highlight the successful use of operations research techniques in various air transportation planning areas and address areas for future research. However, there are a number of differences between the military airlift scheduling problem and airline scheduling. First, airlines create their schedules well in advance of execution. They rely on forecasts and
simulate demand to test alternative network designs. Second, airlines typically create a schedule for each day of the week that repeats weekly. Finally, crew are able to swap between aircraft to allow greater utilization of aircraft.

The airline scheduling problem is usually treated as deterministic because it is large and complex. Optimal schedules make efficient use of aircraft which often leads to less slack in the schedule. In execution, this reduced slack could create significant increased cost in the event of changes or delays. Recently, there has been an effort to create schedules that are robust to these changes. Ahmadbeygi, Cohn, and Lapp (2009) and Lan, Clarke, and Barnhart (2006) present a strategy to reduce the realized cost of a schedule without increasing the planned cost by re-allocating scheduled slack to areas most prone to delay. This type of robust planning has potential to apply to the military airlift scheduling problem, but it is unclear whether it would be as successful because flight segments in military airlift schedules are not interconnected like airline schedules. Aircrew and cargo stay with the same aircraft eliminating the delay from propagating to other routes.

Optimal schedules are rarely executed in practice because of disruptions like weather and maintenance. As a result, there has been considerable research focused on schedule recovery in the literature. Similar to planning, airlines often take a sequential approach by first recovering aircraft, followed by crew, and finally recovering passenger itineraries [6]. Airline controllers can swap aircraft or cancel/delay flights in order to minimize disruption. The difficulty in schedule recovery is the need for solutions quickly. The XOC directorate in the TACC manage schedule recovery in the airlift network. While we are not explicitly trying to account for schedule recovery, we attempt to create a schedule that lends itself well to managing disruptions.

2.4.4 Related Efforts

The military airlift scheduling problem is not new. Research on military airlift problems have been underway for several decades [15, 33]. We provide an overview of a few of the recent efforts.

Scheduling AMC channel missions distributing sustainment cargo is most similar to commercial supply chain networks. Nielsen, Armacost, Barnhart, and Kolitz (2004) utilize composite variables to design the AMC channel network. Composite variables combine different variable types (aircraft flow and cargo flow) into a single aircraft route variable that implicitly captures the feasible cargo flow. The result is an equivalent formulation that is computationally superior because it yields a tighter LP-IP gap. Channel missions occur when there is enough cargo to warrant an aircraft or enough time passes requiring new supplies. As a result, a channel mission utilizes a full or partial aircraft. The cargo for contingency and SAAM missions, on the other hand, can utilize multiple aircraft. Therefore, we would rather model cargo flow explicitly to allow the optimization to decide how best to split cargo between routes.

address this issue by developing a tool called the AMC allocator to aid the XOB in continuously tasking aircraft to missions scheduled by the flight planners. In addition, the AMC allocator is able to combine missions to utilize non-productive flying time. When missions cannot feasibly be accommodated, the tool uses heuristics to assign higher priority missions while minimizing disruptions to missions currently scheduled. Our goal is to create a baseline schedule for the military airlift scheduling problem that aggregates requirement types and makes a single decision. The AMC allocator is still essential to allow decision-makers to adapt the baseline schedule to create an executable schedule in the continuously executing and oversubscribed environment. Our objective is to provide a better baseline schedule for the AMC allocator.

Few efforts have considered uncertainty because of the size of the problem. Baker et al. (2012) formulate a stochastic mixed-integer linear programming model using Benders decomposition. This approach incorporates uncertainty by solving the routing problem on a series of random scenarios. The Benders cuts are weighted by the probability of each scenario happening. The goal of this formulation was to provide the TACC with a 30-day aircraft allocation plan that is robust to changes so as to reduce the number of commercial aircraft leased in the short-term for a much higher cost than leasing them ahead of time. This formulation estimates demand based on forecasts and models at a high level-time is discretized into days and crew rest is not taken into account. On the other hand, our goal is to assign aircraft and cargo to routes based on that allocation to reduce the number of commercial aircraft leased in the short-term. Our formulation uses actual demand and models ground activity, travel time, and crew rest to ensure a feasible solution. While these problems are highly related and depend on one another, they differ in the data and assumptions they use as well as the goal of the problem. As a result, they require distinct approaches.
Chapter 3

Arc-based Formulation

The military airlift scheduling problem can be formulated as a multicommodity flow problem over a dynamic network. The goal is to flow a set of commodities from their POEs, beginning at a given time across a network to their PODs, by a given time. We must decide how best to design the network over which the cargo will flow by assigning aircraft capacity to each arc as well as commodities to each arc. To further complicate this challenge, we must include the numerous constraints specific to the military airlift scheduling problem like crew rest and cargo compatibility.

We begin the chapter reviewing the relevant research in the literature in Section 3.1 and relate it to the military airlift scheduling problem. We introduce the time-expanded network in Section 3.2. In Section 3.3, we formulate the military airlift scheduling problem as a multicommodity flow problem. We conclude with a discussion of the formulation’s merits in Section 3.4.

3.1 Related Work

The military airlift scheduling problem is a network design problem in which both the design decisions and the operational decisions are integral. In particular, it is a multicommodity flow problem where we must couple the aircraft route decisions with the cargo flow decisions to ensure sufficient capacity. Kotnyek (2003) surveys dynamic network flows in the literature and provides a few solution approaches. Kotnyek defines a problem as dynamic if the input is constantly changing or the input is time-dependent but known in advance. The military airlift scheduling problem is dynamic because the input is constantly changing as discussed in Section 2.2.3.

Most solution methods in the literature reduce a dynamic problem to a static one. In practical applications, the solution methods discretize time to create a static network using a time-expanded graph. In a time-expanded graph, there is a copy of each node for every time step in the time horizon and the arcs are drawn between them to represent the traversal time. The challenge using time-expanded graphs is determining a reasonable time step in order to ensure computational tractability.
The goal of any multicommodity flow problem is to satisfy the demands. There are many possible objectives to satisfy this goal. One can maximize total flow, minimize the time required to flow a given amount of commodities, minimize the cost to flow a given amount of commodities, or minimize the cost to flow the maximum flow. In the military airlift scheduling problem, we assume there is a cost for delivering a commodity late. We are interested in minimizing the cost to design the network and the cost to flow all commodities from their source to their sink over this network.

Belobaba, Odoni, and Barnhart (2009) uses a variant of the time-expanded graph called a time-space network to solve the fleet assignment problem in the airlines. The time-space network includes a wrap-around arc for every node to ensure the balance of aircraft. For example, if an airport begins the day with five aircraft of a particular type, the wrap-around arcs ensure there are five of those type of aircraft at the location at the beginning of the next day. The military airlift scheduling problem is more constrained than this, however, because the aircraft not only have to be of the same type but also start and end at certain wing locations as well.

### 3.2 Time-expanded Network

The military airlift scheduling problem is dynamic because the input is constantly changing both in terms of requirement demand and the physical network. For example, the cargo for a requirement might double or the travel time between two ports might increase due to weather. We assume the input to our problem is known and unchanging. However, ignoring these uncertainties still results in a dynamic problem because the input is time-dependent. The travel time between ports is not instantaneous and requirements have time restrictions at the POE and POD. Figure 3-1 shows an example of a small dynamic network with the travel times labeled on each arc. In this example, port C is a hub connecting ports A and B with port D and the arcs are directed, meaning the travel time between ports depends on the direction.

We reduce our dynamic network to a static representation, similar to the literature, by employing a time-expanded graph. Figure 3-2 is the time-expanded version of Figure 3-1 over four time periods.
We take travel time into account while constructing the network. As a result, the flow through an arc in the time-expanded version as instantaneous, resulting in a static network. The dashed lines represent ground arcs while the solid lines represent flight segments. Even in this small example with only four time periods, the number of arcs increased from 8 to 45. In general, the number of arcs leaving a node at each time period equals the number of ports within range of all aircraft plus an additional ground arc. The number of edges leaving a node is smaller near the end of the time horizon for arcs that require multiple time periods. For example, there are two arcs leaving port B in the dynamic network which results in three arcs leaving port B in each time period except for the last because the travel time from port B to port C is two time periods.

The goal of the military airlift scheduling problem is to create a baseline schedule that flight planners can implement immediately or adapt to create a more flexible schedule in case of changes. Therefore, we must discretize time with enough precision to capture accurate flight times and ground activities. In a continuous time setting, modeling exact times would lead to an intractable number of time periods. Thus, we must balance precision with performance. Published flight times are often estimated to the minute and ground activity times estimated to the nearest fifteen minutes. Discretizing a time period by fifteen minutes results in 96 time periods per day in the scheduling horizon. Because our scheduling horizon is likely to be ten to fourteen days, a more tractable discretization is a time period of an hour.

3.3 Arc-based Model

We formulate the military airlift scheduling problem using a time-expanded network with time periods \( \{0, 1, \ldots, T\} \). In this formulation, the design variables represent aircraft flow across the network while the operating variables represent the flow of cargo on the aircraft. Each aircraft
type has its own dynamic network and we transform each of these dynamic networks into one time-expanded network. As a result, each aircraft type has its own network defined over a subset of the time-expanded network.

We make some simplifying assumptions for this model. First, the ground activity time required at the next port is included in the traversal time. As a result, all flight arcs include the minimum ground time to (un)load an aircraft at the next port. Second, we do not model crew rest. Including these extra constraints would increase the size of the model because it would require modeling each individual aircraft. In addition, it would complicate the model because determining if crew rest is necessary depends on the crew’s current working time. A third simplification is to aggregate design variables by aircraft type. Aggregating in this way reduces the size of the problem significantly, but it means we cannot check individual aircraft capacity. It is possible to have enough capacity in aggregate, but not enough capacity for individual aircraft because our cargo cannot be split arbitrarily. Furthermore, we cannot ensure an aircraft returns to its home wing at the end of its route. Instead, we can only ensure there is the proper balance of aircraft by type at each wing similar to the airlines. These assumptions yield the problem of determining the optimal aircraft schedule, one similar to the airlines fleet assignment problem.

3.3.1 Sets

Requirements may consist of multiple cargo types (e.g., vehicle, pallet, passenger), and we refer to a requirement-cargo type pair as a commodity. The nodes in our network represent an airport at a specific time. The edges in our network represent either a flight segment or a ground segment. The sets we use in our model are the following:

- \( K = \{1, 2, \ldots, |K|\} \): set of commodities;
- \( N = \{1, 2, \ldots, |N|\} \): set of nodes;
- \( Wing \subseteq N \): subset of nodes that are wings;
- \( M = \{1, 2, \ldots, |M|\} \): set of aircraft types;
- \( K^m \subseteq K \): subset of commodities able to be transported on an aircraft type \( m \);
- \( E = \{1, 2, \ldots, |E|\} \): set of edges;
- \( F \subseteq E \): subset of edges that are flight segments;
- \( I(n) \subseteq E \): subset of edges entering node \( n \); and
- \( O(n) \subseteq E \): subset of edges leaving node \( n \).
A commodity can either arrive at the POD by LAD, after LAD, or not at all. To allow any of these possibilities, we define the node, \( Sink^k \in N \), for every commodity \( k \) and artificial arcs with infinite capacity from the source node of the commodity, \( POE^k \), to \( Sink^k \) and from the node associated with the POD of the commodity at \( t = T \) to \( Sink^k \). The commodity must arrive to the sink using one of the artificial arcs. Commodity flow through the first artificial arc does not reach the POD at all and commodity flow through the second artificial arc reaches the POD. The node, \( Early^k \in N \), represents the \( k \)th commodity’s POD in the time period prior to EAD.

### 3.3.2 Parameters

The parameters we use in our model are the following:

- **Alloc**\(_n\): demand for aircraft at wing \( n \);
- **WCAP**\(_m\): capacity of aircraft type \( m \) in weight (stons);
- **VCAP**\(_m\): capacity of aircraft type \( m \) in pallet positions;
- **PCAP**\(_m\): capacity of aircraft type \( m \) in passengers;
- **MOG**\(_n\): aircraft capacity of port \( n \) in number of aircraft parking spaces;
- **q**\(_m\): number of aircraft parking spaces required for aircraft type \( m \);
- **D**\(_k\): demand for commodity \( k \) in quantity (number of items);
- **\( \omega^k \)**: weight of commodity \( k \) in stons;
- **\( \nu^k \)**: footprint of commodity \( k \) in pallet positions;
- **\( C^m_e \)**: fixed cost of aircraft type \( m \) flying edge \( e \);
- **\( W^k_m \)**: variable cost of transporting cargo of commodity \( k \) on aircraft type \( m \) on edge \( e \); and
- **\( \delta^m_e \)**:
  \[
  \begin{cases}
  1, & \text{if edge } e \text{ is in the network of an aircraft of type } m \\
  0, & \text{otherwise}
  \end{cases}
  \]

Each wing allocates the number of aircraft available each day. The node associated with a wing on the first time period each day will be a source if the number of allocated aircraft increases from the previous day. Similarly, the node associated with a wing on the last time period each day will be a sink if the number of allocated aircraft decreases from the previous day.

Figure 3-3 includes the sink node, \( Sink^1 \), and artificial arcs, represented by the dotted lines, for commodity 1. In this example, commodity 1 has a POE at port B with an ALD at the beginning of the first time period and a demand of one unit. Figure 3-3 also shows the supply and demand of
aircraft for the wing at port D. It supplies five aircraft at the beginning of the first period, demands an aircraft at the end of the second, and demands the remaining four aircraft at the end of the final time period.

We include in the variable cost for transporting cargo on an edge a per unit penalty for either delivering a commodity late or not at all. In this way, the penalty for delivering a commodity depends on the time of delivery and the type of commodity.

### 3.3.3 Decision Variables

We have two sets of decision variables:

- \(x^m_e\) = number of aircraft of type \(m\) assigned to edge \(e\); and

- \(y^{k,m}_e\) = number of items of commodity \(k\) assigned to aircraft type \(m\) on edge \(e\).

### 3.3.4 Formulation

We formulate our model in the following way:

\[
\min_{x^e, y^{k,m}_e} \sum_{m \in M} \sum_{e \in E} C^m_e \delta^m_e x^m_e + \sum_{m \in M} \sum_{k \in K} \sum_{e \in E} W^{k,m}_e y^{k,m}_e
\]  

(3.1)
Our objective is to minimize the fixed cost of assigning an aircraft to a flight segment and the variable cost (including penalty costs for late cargo) of transporting cargo on a flight segment. Constraints (3.2) conserve the flow of aircraft, ensuring aircraft are supplied and demanded where appropriate. Similarly, constraints (3.3) conserve the flow of each commodity from its source, \( POE^k \), to its sink, \( Sink^k \), ensuring that it does not leave its POE until ALD and it enters the sink node.
through one of the artificial arcs (allowing the commodity to arrive at the POD on time, late, or not at all). We ensure a commodity does not arrive prior to its EAD in constraints (3.4) and that there are not more aircraft at a port at any time than there is capacity in constraints (3.5). We couple aircraft and cargo flow together by ensuring there is enough aggregate aircraft capacity for each aircraft type on every flight segment in constraints (3.6)-(3.8). Finally, constraints (3.9) restrict aircraft and commodity flow to be non-negative integers.

### 3.4 Summary

Modeling the military airlift scheduling problem using a time-expanded network is natural. It is intuitive to interpret the flow of aircraft and cargo over the time-expanded network. The suitability of this formulation, however, is not practical because of the simplifying assumptions and the size.

We make several simplifying assumptions to formulate the military airlift scheduling problem similar to an airline fleet assignment problem. Scheduling aircraft and crew separately and removing ownership of an aircraft from a wing would require AMC to institute significant changes to their business process. Furthermore, it is not clear that these simplifying assumptions would yield a model that is tractable in realistic-sized instances.

AMC operates a global network that results in a large time-expanded network. Estimating the size of the network is challenging because military aircraft can utilize commercial airports and not all military bases have an airfield. We use the number of large or medium (replacement value of $915M or greater) DOD sites to estimate the number of ports in the network. At the start of FY12, the DOD controlled 257 large or medium sites around the world [12]. This does not include the 4,444 small sites or commercial airports AMC aircraft may utilize, so assuming a dynamic network of 257 nodes is reasonable. Assuming $T = 240$ (discretizing time per hour for ten days), this results in a time-expanded network of 6,168 nodes. Assuming that each aircraft type can reach ten airports from any given airport, this results in a dynamic network of 2,570 flight segments for each aircraft type. The time-expanded network results in approximately 1,850,400 edges (i.e. the number of $x_e^m$ decision variables). Even a rougher discretization of time to time steps of six hours results in over 300,000 edges.

The global nature of the AMC air network makes solving the military airlift scheduling problem challenging. Furthermore, our problem requires an integral solution because aircraft and cargo cannot be split arbitrarily. Kotnyek (2003) points out that the integral multicommodity problem is not solvable in polynomial time. Therefore, we must reformulate to yield a more tractable model while incorporating the constraints characteristic of the problem.
Chapter 4

Route-based Formulation

Solving the military airlift scheduling problem using traditional network optimization is not tractable due to its size and characteristics as discussed in Section 3.4. Instead, we reformulate the problem as a route-based optimization model and develop column generation techniques to address it.

In our reformulation, the decision variables correspond to aircraft routes. We build feasible aircraft routes in steps separate from solving the optimization model. By ensuring that routes we build are feasible with respect to travel time, ground time, crew rest, and requirement restrictions, we do not need to encode these characteristics within the optimization model, thus reducing the number of constraints. Further, we reduce the number of decision variables by generating only a fraction of feasible aircraft routes.

The remainder of the chapter is organized as follows. In Section 4.1, we give a brief review of related work in the literature. In Section 4.2, we explain our route generation algorithm and illustrate with simple examples. In Section 4.3, we detail our Route-based optimization model. We explain our two column generation approaches in Section 4.4. Finally, we finish in Section 4.5 with an alternative optimization model that incorporates departure time flexibility.

4.1 Related Work

4.1.1 The Pickup and Delivery Problem

In Section 2.4.2 we introduced the PDPTW and provided the general framework. Numerous efforts have formulated the PDPTW using a column generation approach on a set partitioning problem. Cordeau and Ropke (1990) provide a branch and cut algorithm to solve the set partitioning problem to optimality while Arunapuram et al. (2003) use dynamic programming to solve their problem to global optimality. These solution techniques do not work on the large-scale problems characterized by our application. Much research has focused on heuristic methods to generate feasible vehicle
routes [5, 7, 13, 14, 19, 24]. Many of these approaches are too simplistic for the military airlift scheduling problem because they do not address the practical constraints often encountered in reality.

Arunapuram et al. (2003) solve a practical pickup and delivery problem using a column generation approach with a set partitioning problem as the master problem and a heuristic to generate potential vehicle routes as the subproblems. We take a similar approach in that we are generating our routes outside of the optimization model, taking into account the practical constraints unique to the problem, many of which prior solutions in the literature ignore. The practical constraints considered in the literature include multiple time windows to either pickup or deliver an order; Department of Transportation rules for driving time, working time, rest time, and trip time; compatibility constraints for assigning orders to vehicles and for servicing orders together; nested precedence constraints that ensure that the last order picked up is the first order delivered; and, finally, a cost structure that takes into account the cost of waiting and resting as well as the transportation costs.

Our problem has many of these same constraints, except our application is in air transportation rather than ground transportation. Complexity of our problem results because rest stops and refueling stops cannot occur anywhere along the route; and our problem is a military application that does not require cargo to be loaded and unloaded in a last-in-first-out manner, allowing more feasible route combinations.

Due to these differences, the way in which we create routes is slightly different. During column generation, Arunapuram et al. (2003) create new routes by combining routes and rearranging the sequence of pickups and deliveries to create the cheapest feasible route. In contrast, we generate all possible combinations of pickups and deliveries up front, and during column generation, create new routes by either identifying one of the sequences we generated ahead of time or simply combining two routes together.

4.1.2 Column Generation

Column generation is a widely used technique to reduce the computing resources needed to solve large linear programs (LP). Dantzig and Wolfe (1960) first introduced the idea of using only a subset of columns (decision variables) because only a small number of columns will actually be in the optimal LP solution. Rather, columns with positive reduced cost will not be in the optimal solution and, thus, can be ignored.

Lubbecke and Desrosiers (2004) survey recent applications of column generation and present the general framework. The minimization model to be solved using all possible columns and relaxed integer constraints is referred to as the master problem. The restricted master problem uses a small subset of all possible columns. In each iteration of the simplex method, there is a pricing step in which we find the non-basic variable with the most negative reduced cost. In each iteration of column generation, the subproblem is to find, among the variables not currently in the restricted
master problem, one or more variables with negative reduced cost. In addition, there are many strategies to decide the set of variables, or columns, with negative reduced cost to add to the restricted master problem.

There are two types of column generation: explicit and implicit. In explicit column generation, one prices out each column, that is, computes the reduced cost of each column, given the current solution. In implicit column generation, one formulates a mathematical program to solve the pricing problem and identify the variable with the minimum reduced cost. Barnhart and Schneur (1996) note that explicit column generation is preferred to implicit column generation when this mathematical program is difficult to formulate. This is the case for the military airlift scheduling problem due to the unique practical constraints. As a result, we focus on explicit column generation to identify routes that can potentially improve the current solution.

4.2 Aircraft Route Generation

By manually generating aircraft routes, we ensure the routes are feasible in terms of travel and ground time, crew rest, and requirement time restrictions. Thus, we do not need to include these same constraints in the optimization model. This greatly reduces the number of constraints in our model. However, there are an exponential number of routes possible. Including departure times at each location further exacerbates the problem. Therefore, our goal is to develop a method to generate ‘good’ routes.

We take an algorithmic approach to generating aircraft routes in a reasonable amount of time. A reasonable amount of time in the context of our problem is on the order of hours. First, we observe that a route can be decomposed into smaller components that we call paths.

Definition A path is a sequence of pickup and deliveries for a specific aircraft type such that the aircraft is not empty until the last stop.

Definition A route is one or more paths combined together that begins and ends at the same wing.

Figure 4-1 illustrates a path that covers two requirements. We use the notation $1^+$ to represent the POE of requirement 1 while $1^-$ represents the POD of requirement 1. In this example, the aircraft picks up cargo for requirement 1 and then requirement 2 before delivering requirement 1 and then requirement 2. Figure 4-2 illustrates a route with two paths that cover one requirement each. The dashed lines represent empty flight segments. To highlight the combinatorial challenge in create aircraft routes, there are six sequences possible to pick up and deliver two requirements. There are two possible sequences with only one requirement onboard at a time like in Figure 4-2 and there are four sequences possible with both requirements simultaneously onboard like in Figure 4-1. However, there are 66 possible sequences to pick up and deliver three requirements.
We then generate aircraft routes in phases:

1. Generate feasible aircraft paths
2. Combine aircraft paths together to create feasible partial aircraft routes
3. Assign a wing to each partial route to complete an aircraft route

We use the following data while building aircraft routes:

- Requirement POE and POD;
- Requirement time restrictions: ALD, EAD, and LAD;
- Requirement type and weight;
- Compatibility between pairs of requirements;
- Minimum travel time between ports;
- Minimum ground time at ports; and
- Crew rest restrictions.

These data provide practical constraints to guide route generation. We can eliminate potential combinations based on the ALD, EAD, and LAD of requirements. In addition, the practical constraints relating to travel time and ground time can eliminate potential paths/routes as well. We must ensure requirements simultaneously onboard are compatible. Finally, we must ensure a path/route is feasible in terms of crew rest restrictions. Because we create aircraft routes that satisfy these constraints, we do not need to include these constraints in the optimization model. This simplifies and reduces the size of the optimization model a great deal.

The general idea of our algorithm is to build paths (and then, routes) beginning with one requirement (corresponding to a path) first and adding a requirement (another path) at each iteration.
Given a sequence of stops, we are able to calculate time windows (earliest arrival time, earliest departure time, latest arrival time, and latest departure time) for each stop based on the sequence, aircraft type, and practical constraints. As such, each route corresponds to a specific aircraft type and we generate aircraft routes for all aircraft types. Furthermore, we can define the following “delay allowable” term to differentiate similar paths:

**Definition** Delay allowable is the difference between the latest arrival time and the earliest arrival time at a pickup or delivery location

### 4.2.1 Path Generation

The first step in generating aircraft routes is to generate all feasible paths for each aircraft type. We define the following notation that we use to describe the path generation algorithm:

- **Paths**: set of all feasible paths that are possible
- **NewPaths**: set of all paths to potentially merge
- **TempPaths**: temporary set of new paths that will replace NewPaths
- **Requirements**: set of all requirements
- **C(r) ⊆ Requirements**: subset of requirements that are compatible with requirement r

We summarize this process in Algorithm 1. In Step 1 of the algorithm, we validate the requirement time restrictions by checking to see if it is possible based on travel time, ground time, and crew rest to get from POE to POD by LAD for each requirement. In addition, we determine the set of single requirement paths that are compatible with at least one other requirement to be the basis for creating multiple requirement paths.

Step 2 is the main part of the algorithm. In the first iteration of Step 2, we build paths with two requirements and on each subsequent iteration we build paths with one additional requirement. We begin by filtering requirements with potential to merge with the path in Step 2a. We describe *POTENTIAL* in more detail in Appendix A.

Next, we insert the requirement into the path and check for feasibility using *SEQUENCE* in Step 2(b)i. In order to limit the number of possible combinations, we assume the sequence of the current path must remain the same. We provide an example of the possible sequences to insert a requirement into a path in Section 4.2.1.1 and we provide further details and an additional example in Appendix B. A critical part of *SEQUENCE* is checking each of the possible sequences for feasibility in terms of travel time, ground time, crew rest, and requirement time restrictions. We do so using the function, *FEASIBLE*. We highlight the details of this function with an example in Section 4.2.1.1 and provide details, pseudocode, and an additional example in Appendix C.
Algorithm 1 Path generation

1. For each $r \in \text{Requirements}$,
   (a) IF it is possible to get from POE to POD by LAD, THEN
      i. Add the single requirement path to Paths
         A. IF $C(r) \neq \emptyset$, THEN add the single requirement path to NewPaths
   (b) ELSE flag for revision

2. For each path, $L \in \text{NewPaths}$
   (a) $P = \text{POTENTIAL}(L)$: Determine the subset, $P$, of requirements with potential to merge with the path
   (b) For each requirement, $p \in P$
      i. $S = \text{SEQUENCE}(p, L)$: Determine the sequence of all feasible paths
      ii. IF $S = \emptyset$, THEN go to the next requirement
      iii. ELSE $D = \text{DOMINANT}(S)$: Determine the dominant sequence
      iv. IF $D \notin \text{Paths}$, THEN Add $D$ to $\text{Paths}$ and $\text{TempPaths}$

3. IF $\text{TempPaths} = \emptyset$, THEN exit the algorithm

4. Set $\text{NewPaths} = \text{TempPaths}$

5. Set $\text{TempPaths} = \emptyset$

6. Repeat steps 2-5
It is possible for multiple sequences to be feasible, so we pick one sequence that dominates the others in order to limit the number of paths and potential combinations in subsequent iterations. Many different fields use dominance criteria to limit the number of combinations to either reduce time or memory or both. Wilhem (1999) applies dominance to eliminate possible paths in solving the assembly system design problem with tooling changes and Petersen (2011) uses dominance to reduce the number of labels used in solving a resource constrained shortest path problem. In our application, we use dominance criteria to eliminate sequences that use the same aircraft type and serve the same exact requirements.

Because there are numerous sources of uncertainty in the military airlift scheduling problem, we define dominance as the sequence with the greatest minimum delay allowable. We expect this sequence to be the most resilient to delays or changes. We provide additional dominance criteria in Appendix D.

We use all of the new dominant paths on the next iteration of the loop in Step 2. We continue until it is not possible to create any new dominant cargo paths.

4.2.1.1 Examples

We use the following examples to highlight the main portions of the algorithm. Table 4.1 contains data for requirements.

We assume all requirements are compatible, the minimum ground time for (un)loading is three hours, the maximum range of the aircraft type is eight hours, the maximum crew duty day is fifteen hours, and the minimum crew rest is twelve hours. A crew duty day begins at the departure time after a period of crew rest and ends at the arrival time at the next crew rest location.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Requirement data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
</tr>
</tbody>
</table>

**Sequence** In this example, we want to insert requirement 2 into the path \( \{1^+, 1^-\} \). There are four possible sequences to insert a requirement into a single requirement path while satisfying our definition of a path:

1. \( \{2^+, 1^+, 2^-, 1^-\} \)
2. \( \{2^+, 1^+, 1^-, 2^-\} \)
3. \( \{1^+, 2^+, 2^-, 1^-\} \)
4. \( \{1^+, 2^+, 1^-, 2^-\} \)

**Feasible** In this example, we will check the first sequence for feasibility. Figure 4-3 shows the sequence of the possible path with travel times between each port. Note that all of the travel times are less than the maximum range of the aircraft type, so no intermediate refueling stops are necessary. In general, however, this is not always the case.

![Figure 4-3: Sequence of a possible path with the travel times between ports](image)

We begin by calculating the earliest arrival and departure times at each stop in the order of the sequence. We set the earliest arrival time of the first stop as the ALD of requirement 2 because the earliest time we can begin loading an aircraft at a pickup location is ALD. The difference between the earliest departure time and the earliest arrival time at the same stop will depend on many factors including: ground time, crew rest, and arrival time restrictions at the next port. In this case, the earliest departure time only depends on the minimum ground time.

**Table 4.2: Earliest arrival and departure times**

<table>
<thead>
<tr>
<th>Location</th>
<th>2⁺</th>
<th>1⁺</th>
<th>2⁻</th>
<th>1⁻</th>
</tr>
</thead>
<tbody>
<tr>
<td>Earliest arrival time</td>
<td>24</td>
<td>32</td>
<td>48</td>
<td>54</td>
</tr>
<tr>
<td>Earliest departure time</td>
<td>27</td>
<td>45</td>
<td>51</td>
<td>57</td>
</tr>
</tbody>
</table>

Because there are no intermediate refueling locations necessary, the difference between the earliest arrival time and the earliest departure time at the previous stop will be the travel time between the ports. Table 4.2 shows the results of these calculations. We must wait at 1⁺ before delivering requirement 2 because of the arrival time restriction at this port. Because the waiting time at the second stop is greater than crew rest, this becomes a crew rest location even though the crew duty day did not require it.

We ensure the sequence is feasible by checking if the earliest arrival time at the delivery locations of each requirement plus the unloading time is less than the LAD for that requirement. It is sufficient to only calculate the earliest arrival and departure times for a sequence to ensure feasibility. However, we calculate the latest arrival and departure times at each sequence to differentiate between similar sequences and to filter potential sequences by earliest and latest departure times.
Because the latest arrival and departure times depend on the LAD of requirements later in the sequence while crew rest depends on information obtained earlier in the sequence, we calculate the latest arrival and departure times in two steps.

First, we calculate inflated latest arrival and departure times based only on travel time, ground time, and crew rest in the order of the sequence where the latest arrival time at the first stop is the maximum LAD over all requirements covered in the path. As shown in Table 4.3, the crew duty day did not require a crew rest until the third location by calculating arrival and departure times without considering requirement time restrictions. It was necessary to crew rest at the third stop because the additional unloading time and travel time exceeds the maximum crew duty day. Using these times we calculate the minimum travel time defined as follows:

**Definition** The *minimum travel time* for a sequence is the difference between the latest arrival time at the final stop and the latest departure time at the first stop when these times only depend on travel time, ground time, and crew rest.

<table>
<thead>
<tr>
<th>Location</th>
<th>2⁺</th>
<th>1⁺</th>
<th>2⁻</th>
<th>1⁻</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inflated latest arrival time</td>
<td>72</td>
<td>80</td>
<td>86</td>
<td>101</td>
</tr>
<tr>
<td>Inflated latest departure time</td>
<td>75</td>
<td>83</td>
<td>98</td>
<td>104</td>
</tr>
</tbody>
</table>

The minimum travel time for this sequence is 26 hours. We use the minimum travel time as one of the criteria for determining the dominant sequence. In the second step, we adjust the latest arrival and departure times to get valid times based on the LAD of all delivery locations. We begin by adjusting all latest arrival and departure times by 32 to ensure we deliver requirement 1 with the minimum ground time necessary by LAD. Next, we check all other delivery locations to see if we need to adjust further. In this case, we reach both delivery locations by LAD after the first adjustment. Table 4.4 shows the latest arrival and departure times. We can calculate the delay allowable by taking the difference between the latest arrival time and earliest arrival time at all locations and determine that the minimum is six hours.

<table>
<thead>
<tr>
<th>Location</th>
<th>2⁺</th>
<th>1⁺</th>
<th>2⁻</th>
<th>1⁻</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latest arrival time</td>
<td>40</td>
<td>48</td>
<td>54</td>
<td>69</td>
</tr>
<tr>
<td>Latest departure time</td>
<td>43</td>
<td>51</td>
<td>66</td>
<td>72</td>
</tr>
</tbody>
</table>

**4.2.1.2 Algorithm Tactics**

Implementing this algorithmic method, practical constraints, and dominance greatly reduces the number of feasible paths generated. However, the number of total paths depends on the number of
requirements, the length of requirement time windows, and the distribution of POE and POD’s for requirements across the network. Furthermore, we still need to combine the paths together to create routes. This is a combinatorially difficult task, especially when there are many feasible paths. As a result, we have the option of reducing the number of feasible paths that we consider by aircraft capacity and/or number of requirements.

**Aircraft capacity**  Even though we are using the optimization model to decide how much cargo to put on each route, we can use aircraft capacity to limit the feasible paths to those likely to be in the solution. We are motivated by the fact that if a path serves a requirement, we must pick up part of its cargo because its pickup and delivery location are in the sequence. As a result, we would like to avoid building paths that pickup a small amount of cargo at each stop because those paths are not efficient and not likely to be in the solution. We do this by limiting feasible paths to those that when any one requirement onboard is removed, all other requirements fit on the aircraft in their entirety. Because the cargo for a requirement might exceed an aircraft capacity, we use an adjusted cargo amount for each aircraft type. For example, if the cargo for a requirement would fill 2.4 C-17’s, we use an adjusted cargo amount of 0.4 when building paths for the C-17 because it is likely that two aircraft will directly deliver the remaining cargo. Limiting feasible paths in this way eliminates paths in which we continuously pick up only a fraction of each requirement. Because we build the routes outside of the optimization, the total cost of a route includes the cost for stopping at each location so we would expect to pick up at least a portion of each requirement covered in a route. This means a path with three requirements onboard with adjusted cargo amounts of 0.6, 0.3, and 0.3 is considered feasible, but a path with three requirements onboard with adjusted cargo amounts of 0.8, 0.4, and 0.2 is not considered feasible. In the first example we were able to pick up any two of the requirements in their entirety. However, in the second example there is an opportunity to not pick up any of a requirement even though the path is designed to stop at its POE and POD. In this case, a path that only serves the two requirements with cargo onboard dominates a path that serves all three requirements because it will not include the cost to visit the pickup and delivery location of the requirement without cargo onboard.

**Number of requirements**  Another way in which we can reduce the number of feasible paths is by limiting the total number of requirements in a path or, equivalently, the number of times we perform Step 2. Because the number of times we perform this step depends on the requirement data, we can choose the number of requirements in such a way that ensures the completion of path generation in a reasonable amount of time. Limiting the feasible paths in this way may lead to the additional benefit that the path allows planners to be more responsive to changes such as delays due to weather or maintenance. We provide a simplified example based on two requirements in Figure 4-4. The route in Figure 4-4a serves both requirements in a single path. If there is a delay
due to maintenance after the second flight segment, both requirements are affected. The route in Figure 4-4b serves both requirements as well, but it is less efficient because it limits the number of requirements in a path to one. However, only one requirement is affected if there is a delay after the second flight segment. Furthermore, even if a new aircraft positions itself to serve the requirements affected, it can respond quicker to the route with single-requirement paths because there is one fewer flight segments.

Figure 4-4: A simplified example of the impact limiting paths by the number of requirements has in the case of delays

4.2.2 Route Generation

After we complete the path generation portion of the algorithm, we have the problem of combining paths together to create routes.

Again, we utilize travel time, ground time, crew rest, and requirement time restrictions to limit combinations to those that are feasible. Similar to path generation, we begin with one path and at each iteration, add an additional path to the end of the route. However, we need additional criteria to limit the number of possible combinations of paths and ensure the algorithm converges in a reasonable amount of time. As a result, we employ common-sense rules to only generate routes likely to be in the final solution. We define the following parameters that we tune to reduce the number of potential routes and the time required for this portion of the algorithm:

- $Max.Prox$: the maximum travel time between two ports to be considered in close proximity;
- $Max.Wait$: the maximum waiting time between paths including crew rest;
• *Max.Total*: the maximum total travel and wait time between paths; and
• *Max.Route*: threshold route length to relax route quality and quit adding additional paths.

Because feasible solutions to our problem depend on both time and space, we chose parameters that capture both of these dimensions. The first parameter, *Max.Prox*, is meant to limit the length of positioning, repositioning, and depositioning flights. Routes likely to be in the final solution should be efficient, thus, the length of their empty flights should be small. Figure 4-5 depicts how we can limit the number of possible locations for the next stop by proximity.

Figure 4-5: Limiting the number of possible route combinations based on proximity. In this example, there are only five possible locations reachable in one move from the location in the center.

Similarly, routes in the final solution are likely to utilize aircraft as much as possible. Therefore, we limit the time it can be on the ground by *Max.Wait*. We included the parameter, *Max.Total*, to allow the user more control in limiting the number of possible combinations. A value less than the sum of *Max.Prox* and *Max.Wait* will eliminate those routes that are close to both the limit of proximity and waiting time.

Finally, we realize that we must include routes with inefficient flights as some inefficiency may be necessary to achieve global optimality. For example, it may be necessary to fly empty halfway across the globe to get a requirement if there are no closer aircraft available, if the cost of delivering the requirement is less than the penalty for not delivering it. For this reason, we relax the proximity parameter for positioning and depositioning flights once the total time for a route exceeds the threshold. This also limits the time an aircraft and crew can be away from their home location at any time. This aligns well with the current way aircraft are scheduled for maintenance upkeep and reserve and national guard crew are scheduled to be active.
Algorithm 2 describes this process. We use the following notation to describe the route generation algorithm:

- **Candidates**: set of all partial routes to consider to create complete routes
- **Partial**: set of partial routes to consider to add an additional path
- **TempPartial**: temporary set of partial routes that will replace **Partial**

### Algorithm 2 Route Generation

1. For each path in **Paths**
   
   (a) For each wing in **Wings**
   
   i. IF the travel time from any wing to the origin is less than Max.Prox, THEN add the path to **Partial**
   
   ii. Add the path to **Candidates**

2. For each partial route in **Partial**
   
   (a) For each path in **Paths**
   
   i. IF the path and route serve any of the same requirements, THEN try the next path
   
   ii. IF the latest departure time to begin the path is less than or equal to the earliest departure time at the end of the route, THEN try the next path
   
   iii. IF the travel time from the end of the candidate route to the beginning of the path is greater than Max.Prox, THEN try the next path
   
   iv. IF the time to wait for the start of the path is greater than Max.Wait, THEN try the next path
   
   v. IF the travel time plus the wait time is greater than Max.Total, THEN try the next path
   
   vi. IF the new route is infeasible, THEN try the next path
   
   vii. IF the new partial route time is greater than Max.Route AND the distance between the beginning and end of the new route is greater than the distance between the beginning and end of the candidate route (getting further from the first stop), THEN add the original partial route to **Candidates**
   
   viii. ELSE IF the total new route time is greater than Max.Route, THEN add the new partial route to **TempPartial**
   
   ix. ELSE IF the distance between the beginning and end of the new partial route is greater than Max.Prox, THEN add the new partial route to **TempPartial**

   x. ELSE, add the new partial route to **TempPartial** and **Candidates**

3. IF **TempPartial** = Ø, THEN exit the algorithm

4. Set **Partial** = **TempPartial**

5. Set **TempPartial** = Ø

6. Repeat steps 2-5

The first step in creating a route is to identify which paths can begin a route. A path qualifies to begin a route if the first stop is within Max.Prox of a wing. In addition, we add every path
to Candidates to try to guarantee every requirement is covered. However, we can only guarantee this if there is a wing close enough to deliver a commodity from POE to POD by LAD for every requirement. The best way to do this is to make each individual path a route because these include single requirement paths.

The next section of logic is where we attempt to combine each path to the end of the partial routes. We begin by checking a series of conditions to limit the number of routes we test for feasibility. We filter out those that would serve the same requirement again, those where the path cannot possibly begin after the route due to departure times, and those that do not meet the rules we employ to identify quality routes. Because we are adding each path to the end of a partial route and we know the latest arrival and departure times for every path, we only need to calculate the earliest arrival and departure times for a partial route to filter paths by time. Therefore, we use a limited version of FEASIBLE to calculate the earliest arrival and departure times at each stop to ensure feasibility.

If a new route is feasible, we use Max.Route and Max.Prox to determine to which set(s) to add the new route. Because we do not calculate the latest departure and arrival times, we must define how we calculate partial route length.

**Definition** Partial route length is the difference between the earliest arrival time at the last stop and the earliest departure time at the first stop.

If the new partial route length is greater than Max.Route, we will add it to Candidates in order to try to make a complete route. However, we will try to add the best route possible. We will continue to try to string additional paths to the route if the new partial route is moving closer to the first stop. If it is moving away, we will add the original partial route to Candidates. We layout this process in Figure 4-6. In this example, the sink of the new partial route is closer to the source than the sink of the original partial route so we will try to add paths.

If the new partial route length is not greater than the threshold, we check the distance between the first and last stop. If they are not within Max.Prox, we will add the new partial route to TempPartial to continue stringing paths to the end of it. However, if they are in close proximity, we will add the new partial route to Candidates to try to make a complete route and we will add the new partial route to TempPartial to continue stringing paths to the end of it. The source and sink of the partial route depicted in Figure 4-7 are within Max.Prox of one another, so we add it to Candidates and we continue adding paths to it.

The last step to generate an aircraft route is to assign a wing. We assign the closest wing to each partial route in Candidates and check feasibility by using FEASIBLE. If it is feasible, we calculate the information needed for the optimization model. This information includes the time windows, the commodities covered, the sets of commodities on board simultaneously, the number of takeoffs, the total flying time, and the flying time per commodity.
Figure 4-6: Once a partial route length exceeds the threshold, we add the original route if its origin and destination are closer together than the origin and destination of the new route. Otherwise, we continue to add paths to the new route.

Figure 4-7: We will try to complete a partial route if the origin and destination are in close proximity regardless of partial route length.
4.3 Route-based Model

In this section, we develop a route-based model to assign aircraft and requirements to routes. The aircraft route generation algorithm in Section 4.2 begins by generating a set of feasible paths that cover one or more requirements for every aircraft type. We then combine two or more of these cargo paths together based on spatial and temporal rules and then check for crew feasibility. Assigning a wing to each string of paths represents an aircraft route that covers one or more requirements, begins and ends at its wing, and is crew feasible. In addition, we know which days in the scheduling horizon each aircraft route can be active so we ensure that no more aircraft are used than allocated.

We make a number of simplifying assumptions for this model. First, for each route, we specify the aircraft to depart and return on the earliest days feasible. This restriction limits the number of routes used in the model and thus allows for larger problem instances, though including departure flexibility might improve the solution quality through increasing the space of feasible solutions. We formulate an alternative, Flexible Model, that allows flexibility in which days an aircraft route departs and returns in Section 4.5.

We do not consider MOG constraints in our model for a number of reasons. First, obtaining this information during route generation costs significant computational time. Second, including these types of constraints increases the number of constraints immensely. The number of constraints increases by the number of time periods to check (at least 4 per day multiplied by the number of days in the scheduling horizon) for every location in the model. Third, these constraints add another layer of complexity to the model further complicating the pricing problem in column generation. Importantly, we believe that there is potential to satisfy these constraints without including them all in the model. There will be slack in the schedule allowing route departure and arrival times to be post-processed and adjusted to satisfy MOG requirements, potentially for all routes. If there are still MOG constraints violated, we can add constraints to eliminate these MOG violations and re-solve the model.

Finally, we limit the size of our model by aggregating our decision variables by aircraft type. Instead of assigning a specific aircraft to a route, we assign the number of aircraft of a specific aircraft type to a route. This allows us to have one decision variable for each route rather than a decision variable for every aircraft allocated at each wing. Furthermore, we obtain this same magnitude in reduction of variables for the decision variables that assign cargo to a route. This results in a significant reduction in the size of our problem. However, the downside is that the capacity constraints are now based on aggregate capacity. It is possible to have enough capacity in aggregate, but not enough capacity for individual aircraft because our cargo cannot be split arbitrarily. We can address this by adding identical routes and constrain them to at most one aircraft.

After solving the model, the solution may be post-processed into a MOG-feasible schedule with
specific departure times.

4.3.1 Sets

Requirements may consist of multiple cargo types (e.g., vehicle, pallet, passenger), and we refer to a requirement-cargo type pair as a commodity. The edges in our network represent aircraft routes while the nodes represent a wing on a particular day. We define the set of critical legs as:

Definition The set of critical legs are those flight segments that are the most constrained in terms of aircraft capacity.

Checking only the flight segments in the set of critical legs is necessary and sufficient to ensure enough capacity for the entire route. For example, the sequence of a route could be \{Wing, 1^+, 2^+, 2^-, 1^-, Wing\}. In this case, the set of critical legs consists of the flight segment that covers both requirements 1 and 2. Checking the capacity of this flight segment implies checking the capacity of the flight segments that only cover requirement 1. The sets we use in our model are the following:

- $K = \{1, 2, ..., |K|\}$: set of commodities;
- $N = \{1, 2, ..., |N|\}$: set of nodes;
- $E = \{1, 2, ..., |E|\}$: set of aircraft routes;
- $E_n \subseteq E$: subset of aircraft routes that are active from node $n$;
- $E_k \subseteq E$: subset of aircraft routes that cover commodity $k$;
- $K_l \subseteq K$: subset of commodities covered by route $l$; and
- $J(l)$: set of critical legs in terms of capacity for route $l$.

4.3.2 Parameters

The parameters we use in our model are the following:

- $ALOC_n$: number of aircraft allocated from wing $n$;
- $WCAP_l$: capacity of aircraft type for route $l$ in weight (stons);
- $VCAP_l$: capacity of aircraft type for route $l$ in pallet positions;
- $PCAP_l$: capacity of aircraft type for route $l$ in passengers;
- $D^k$: demand for commodity $k$ in quantity (number of items);
- $\omega^k$: weight of commodity $k$ in stons;
• \( \nu^k \): footprint of commodity \( k \) in pallet positions;
• \( C_l \): fixed cost of flying on route \( l \);
• \( W^k_l \): variable cost of transporting cargo of commodity \( k \) on route \( l \);
• \( P^k \): penalty associated with not covering a unit of commodity \( k \); and
• \( \delta_{l,j}^k = \begin{cases} 
1, & \text{if leg } j \text{ of route } l \text{ contains commodity } k \\
0, & \text{otherwise}
\end{cases} \)

Note that parameters indexed with an \( l \) depend on the routes generated.

### 4.3.3 Decision Variables

We have three sets of decision variables:
• \( x_l \) = number of aircraft assigned to route \( l \);
• \( y^k_l \) = number of items of commodity \( k \) assigned to route \( l \); and
• \( u^k \) = number of items of commodity \( k \) unassigned.

### 4.3.4 Formulation

We formulate our model in the following way:

\[
\min_{x_l, y^k_l, u^k} \sum_{l \in E} C_l x_l + \sum_{l \in E} \sum_{k \in K} W^k_l \omega^k y^k_l + \sum_{k \in K} P^k u^k \quad (4.1)
\]

s.t. \( \sum_{l \in E_k} y^k_l + u^k \geq D^k \), \( \forall k \in K \) \quad (4.2)

\( \sum_{l \in E_n} x_l \leq ALOC_n \), \( \forall n \in N \) \quad (4.3)

\( \sum_{k \in K_l} \delta_{l,j}^k \omega^k y^k_l \leq WCAP_l x_l \), \( \forall l \in E, j \in J(l) \) \quad (4.4)
\[
\sum_{k \in K} \delta^k_{l,j} \nu^k y^k_l \leq VCAp_l x_l, \quad \forall l \in E, j \in J(l)
\] (4.5)

\[
\sum_{k \in K} \delta^k_{l,j} y^k_l \leq PCAp_l x_l, \quad \forall l \in E, j \in J(l)
\] (4.6)

\[u^k \geq 0 \quad \forall k \in K\] (4.7)

\[x_l, y^k_l \in \mathbb{Z}^+ \quad \forall l \in E, k \in K\] (4.8)

Our objective is to minimize the total fixed cost of using a route, the variable cost of transporting a commodity on a route, and the penalty for not covering a commodity. Constraints (4.2) ensure that the entire commodity is either served or unassigned, and constraints (4.3) ensure each wing does not use more aircraft than there are available. Constraints (4.4), (4.5), and (4.6) restrict the amount of cargo assigned to a route to the capacity available in weight, footprint, and passengers respectively. For each route, it is sufficient to only add the set of critical legs \( J(l) \) for a route. Finally, constraints 4.7 ensure unassigned commodities are non-negative and (4.8) restrict the number of aircraft assigned to routes and the number of commodities assigned to routes to be non-negative integers.

4.4 Column Generation

The decision variables in our model correspond to aircraft routes. Because there are an exponential number of possible aircraft routes, it is impossible to include decision variables for every aircraft route. Instead, we divide our problem into two parts: the master problem and a subproblem. For the master problem, we relax the integrality constraints, 4.8, of the Route-based formulation and the set \( E \) represents every feasible aircraft route. We solve the restricted master problem using a subset of these routes. The subproblem is to identify routes not in the restricted master problem with potential to improve the solution. Figure 4-8 highlights the overall column generation approach.

We utilize column generation in two different approaches: explicit column generation and selective column generation. The approaches differ in two important ways:

1. Generating the initial set of aircraft routes; and
2. Identifying new routes by solving the subproblem.

4.4.1 Explicit Column Generation

In explicit column generation, we generate a set of feasible routes, \( \tilde{E} \subseteq E \), using the route generation algorithm described in Section 4.2. We then solve the restricted master problem using a subset of these routes, \( E' \subseteq \tilde{E} \). The subproblem in this case is to identify routes not currently in the model \( (l \in \tilde{E} \setminus E') \) with at least one of their associated dual constraints violated. If there are none, by strong duality we know we have the optimal solution. We do this by pricing out (i.e. calculating the reduced cost of) every decision variable not in the model. We price out the decision variables by using the dual variables obtained from solving the restricted master problem. We formulate the dual, define the reduced costs associated with our master problem decision variables, and discuss the tradeoffs in approximating these reduced costs for column generation in the following sections. We also list the factors that influence the solution quality.

4.4.1.1 Dual Formulation

We use duality to motivate our explicit column generation. The dual model decision variables are associated with constraints from the master problem. When we solve the master problem, we obtain dual variable values for every constraint in the master problem. The dual decision variables are:

- \( \pi^k \): Associated with the demand constraints
- \( \sigma_n \): Associated with the allocation constraints
- \( \gamma_{l,j} \): Associated with the weight capacity constraints
- \( \mu_{l,j} \): Associated with the footprint capacity constraints
- \( \theta_{l,j} \): Associated with the passenger capacity constraints

Similarly, the dual model constraints are associated with decision variables from the master problem.

The following is the dual formulation associated with the master problem:
\[
\max_{\pi^k, \sigma_n, \gamma_{l,j}, \mu_{l,j}, \theta_{l,j}} \sum_{k \in K} D^k \pi^k + \sum_{n \in N} ALOC_n \sigma_n
\]

s.t.
\[
\sum_{n \in \mathcal{E}_n} \sigma_n - \sum_{j \in \mathcal{J}_l(l)} WCAP_l \gamma_{l,j} - \sum_{j \in \mathcal{J}_l(l)} VCAP_l \mu_{l,j} - \sum_{j \in \mathcal{J}_l(l)} PCAP_l \theta_{l,j} \leq C_l, \quad \forall l \in \mathcal{E}
\]

\[
\pi^k + \sum_{j \in \mathcal{J}_l(l)} \delta^k_{l,j} \omega^k \gamma_{l,j} + \sum_{j \in \mathcal{J}_l(l)} \delta^k_{l,j} \nu^k \mu_{l,j} + \sum_{j \in \mathcal{J}_l(l)} \delta^k_{l,j} \theta_{l,j} \leq W^k_l \omega^k, \quad \forall l \in \mathcal{E}, k \in \mathcal{K}
\]

\[
\pi^k \leq P^k, \quad \forall k \in \mathcal{K}
\]

\[
\pi^k \geq 0, \quad \forall k \in \mathcal{K}
\]

\[
\sigma_n, \gamma_{l,j}, \mu_{l,j}, \theta_{l,j} \leq 0, \quad \forall n \in \mathcal{N}, l \in \mathcal{E}, j \in \mathcal{J}(l)
\]

### 4.4.1.2 Reduced Cost

We are interested in violations to the dual constraints. Thus, we define the reduced costs associated with each decision variable in the master problem:

\[
r_l = C_l - \sum_{n \in \mathcal{E}_n} \sigma_n + \sum_{j \in \mathcal{J}_l(l)} WCAP_l \gamma_{l,j} + \sum_{j \in \mathcal{J}_l(l)} VCAP_l \mu_{l,j} + \sum_{j \in \mathcal{J}_l(l)} PCAP_l \theta_{l,j}, \quad l \in \mathcal{E}
\]

\[
\hat{r}^k_l = W^k_l \omega^k - \pi^k - \sum_{j \in \mathcal{J}_l(l)} \delta^k_{l,j} \omega^k \gamma_{l,j} - \sum_{j \in \mathcal{J}_l(l)} \delta^k_{l,j} \nu^k \mu_{l,j} - \sum_{j \in \mathcal{J}_l(l)} \delta^k_{l,j} \theta_{l,j}, \quad l \in \mathcal{E}, k \in \mathcal{K}
\]

\[
\hat{r}^k = P^k - \pi^k, \quad k \in \mathcal{K}
\]
We can use the dual variable values to determine if any of the dual constraints are violated. A violation of a dual constraint equates to one of 4.15, 4.16, or 4.17 being negative. The challenge that we face is that the reduced cost of routes not in the model depend on the dual variables corresponding to their capacity constraints. Because the route is not in the model, neither are its capacity constraints and we do not get the dual variable values associated with them. Furthermore, a route not being in the model means we do not assign capacity to that route and its corresponding capacity constraints are binding. Therefore, we cannot use complimentary slackness to determine the dual variable values because they are not necessarily zero.

We know \( \pi^k \) is bounded by \( P^k \), so the reduced cost \( \bar{r}^k \) is always non-negative and we need not calculate (4.17). However, we must approximate the unknown dual variables \( (\gamma_{l,j}, \mu_{l,j}, \text{and } \theta_{l,j}) \) to calculate the reduced cost \( r_l \) and \( \hat{r}_l^k \) to identify new routes to add to the model.

### 4.4.1.3 Reduced Cost Approximations

Because the dual variables associated with the capacity constraints on flight segment \( j \) relate to a specific route \( l \), the value of the dual variable represents the change in the objective value if we had an additional unit of capacity for flight segment \( j \) on route \( l \). The simplest way to approximate the unknown dual values is to assume they are zero. In this case, the reduced cost \( r_l \) will always be non-negative and we need not calculate (4.15). As a result, we would only need to calculate \( \hat{r}_l^k \) for every commodity on a route to check for violations.

The dual variable \( \pi^k \) represents the change in the objective value if we had to cover an additional unit of commodity \( k \). The value of \( \pi^k \) will be equal the cheapest cost of the additional weight, footprint, or passenger capacity \( (x_l) \) needed and it will include both the fixed cost \( (C_l) \) and the variable cost \( (W_l^k \omega^k) \). Therefore, it is likely that \( \pi^k > W_l^k \omega^k \) and \( \hat{r}_l^k \) will be negative. As a result, we add nearly all routes to our model.

We observe that the reduced cost of interest, \( \hat{r}_l^k \), is actually only for one commodity. In reality, the route might cover additional commodities as well. One way to approximate the reduced cost is to sum the reduced cost of all commodities on a particular route. If this aggregate reduced cost is negative, we add the route to the model.

We could approximate \( \hat{r}_l^k \) by adjusting \( W_l^k \omega^k \) by a fraction of the fixed cost of the route. We could calculate the fraction of fixed cost based on the proportion of weight of the commodity compared to the total weight of all commodities the routes covers.

Another approach is to instead try to approximate the dual variables using routes in the model. Note that the flight segment, \( j \), represents one of the critical legs on the route. We know the set of commodities covered by each of these legs and the aircraft type for the route. As a result, we can estimate the dual variable value by using the dual variable value of a route using an aircraft of the same type that covers the same set of commodities if any exist. If none exist, then we assume it
is zero. If multiple dual variables exist, we must decide which to choose. Choosing a conservative
estimate in this sense means picking the dual variables that decrease reduced cost. This translates
to choosing the minimum value for $\gamma_{l,j}$, $\mu_{l,j}$, and $\theta_{l,j}$ in $r_l$ and the maximum value for $\gamma_{l,j}$, $\mu_{l,j}$, and $\theta_{l,j}$ in $\hat{r}_k^l$. Being aggressive in the way we price out columns means that we choose the maximum
value for $\gamma_{l,j}$, $\mu_{l,j}$, and $\theta_{l,j}$ in $r_l$ and the minimum value for $\gamma_{l,j}$, $\mu_{l,j}$, and $\theta_{l,j}$ in $\hat{r}_k^l$ to limit the
number of negative reduced cost routes.

### 4.4.1.4 Solution Strategy

In our explicit column generation algorithm, how we approximate the reduced cost of a route is not
the only thing that affects solution quality. In addition, the following factors affect solution quality:

- Initial set of routes
- Routes to enter on each iteration
- Stopping criteria

**Initial set of routes** The initial set of routes is critical to the success of our model. The initial
set of routes must be characteristic of the true solution in order for column generation to progress
towards an optimal solution [20]. As a result, we ensure the initial set of routes covers every
commodity and there is a minimum number of routes. For example, say we want at least 300 routes
in the initial set. First, we sort the routes in ascending order based on cost per unit weight and
choose the 300 cheapest routes. If this set does not cover a commodity, we add the cheapest route
that covers that commodity.

**Routes entering** We need to decide which routes we add to the model on each iteration. We can
either add every route with a negative reduced cost or we can add a certain “cut-off” number of the
negative reduced cost routes. Depending on how we approximate reduced cost, it is likely that most
of the routes have negative reduced cost so we only want to bring in a fraction at each iteration.
However, if we are aggressive in the way we approximate reduced cost, many routes may not have
negative reduced cost.

We can either keep adding routes with negative reduced cost until we reach the cutoff or we can
calculate the reduced cost for every route and add the most negatives ones. In order to add the
most negative reduced cost routes, we must calculate (4.15) for every route not in the model and
(4.16) for every commodity on every route not in the model. In practical instances, this could take
significant time. If, on the other hand, we only add routes with negative reduced cost, we only need
to calculate (4.15) and (4.16) for each route until one is negative and we can quit once we reach the
cut-off number of routes. We calculate the reduced cost of routes in random order to ensure we are
not only adding the cheapest routes, or otherwise bias which routes we add.
Stopping Criteria  The final parameter we need to decide is when to stop iterating. Naturally, we will stop iterating if either all the routes are in the model or there are no routes with negative reduced cost. Often, the total number of routes is too large to solve the Route-based model in a reasonable amount of time and most routes could have negative reduced costs due to degeneracy. As a result, we might want to stop iterating prior to bringing in every route or waiting until there are not any more routes with negative reduced cost. We can do this a couple of different ways: stop iterating after a certain number of iterations without improvement in the solution, or stop iterating when specified time limit is reached.

4.4.2 Selective Column Generation

Selective column generation differs from explicit column generation in two ways: we do not completely enumerate all feasible routes using our route generation algorithm in the initial step and we generate new aircraft routes at each iteration. The main idea in our selective column generation approach is to build routes with a single path in the initial step and let the solution to the restricted master problem guide how we combine aircraft paths together to form aircraft routes. In this way we are able to realize significant computational time savings by not performing the second phase of route generation and we do not rely on rules of thumb to decide how to combine paths together to create feasible routes.

4.4.2.1 Generating Initial Routes

The first step in our column generation approach as depicted in Figure 4-8 is to generate an initial set of routes, $E' \subseteq \tilde{E}$. In selective column generation, we modify our route generation algorithm by creating aircraft routes with only one path. In this way, we eliminate combining paths together as described in Section 4.2.2; a step that took the majority of time in our preliminary computational tests. Instead, we create aircraft routes by assigning a wing to individual paths generated using the path generation algorithm in Secion 4.2.1 and check for feasibility. Generating the initial set of routes in this way results in a much smaller initial set of routes, but in a significant reduction in computational time to initialize our approach.

4.4.2.2 Solving the Subproblem

At each iteration, we solve the restricted master problem using all routes generated thus far. Next, we try to create new routes ($l \in E \setminus E'$) by combining the routes in the solution that use the same type of aircraft together and checking each for feasibility. Intuitively, it is better to use the new route than the two original routes if the cost of the new route is less than the sum of the original routes. To relate this intuition to explicit column generation, a new route will have negative reduced
cost and, thus, should be added to the model if the cost of the new route is less than the sum of the cost of the two original routes.

The final step in generating an aircraft route is to assign a wing and check for feasibility. We choose the wing to minimize travel time and give the best chance of being feasible. One of the important constraints in our model is restricting the number of aircraft assigned from a wing to the number allocated. Assigning wings in this manner will not provide alternative routes for those wings that are highly constrained. To combat this problem, every few iterations we create new routes by simply choosing the second best wing for every route in the solution. This strategy provides alternative routes to those using highly constrained wings and aids in generating better routes.

We iterate in this fashion until we satisfy the stopping criteria. The natural stopping criteria is when we cannot create any new routes. However, we must provide an additional criterion to ensure tractability. Therefore, we stop if the objective value does not improve.

4.4.2.3 Advantages

Selective column generation has a few advantages over explicit column generation. First, the computation time needed up front to create the initial set of routes is dramatically reduced. The majority of time in explicit column generation is spent combining paths together during aircraft route generation for the majority of instances. In selective column generation we only need to generate paths and simply assign a wing to each to create the initial set of routes.

Second, in explicit column generation we rely on rules of thumb to limit the number of feasible combinations to reduce the time spent combining paths together. These rules of thumb include limiting the length of the repositioning flight and the waiting time between paths. However, the power of optimization is often realized by challenging the rules of thumb humans rely on to create aircraft routes. In selective column generation, we do not use any of these rules of thumb to determine which paths to combine together. Instead, we allow the solution to the master problem to determine which paths to try to combine together.

Using our route generation algorithm, we are unable to generate aircraft routes in which the same path is included in the route multiple times. For example, consider a requirement with enough cargo for multiple loads. The best solution may require an aircraft to fly to the pickup location, deliver a load, fly back to the pickup location and deliver another load. We do not allow this in our algorithm because the number of possible combinations is too great. We are able to do this with selective column generation if the same path is in two different routes of the solution to the master problem.

In our experiments, selective column generation is consistently faster at reaching a solution. This advantage is important for a number of reasons. The obvious reason is that we can employ
this approach on larger, more realistic instances. In addition, this approach lends itself better if one wants to change a part of the problem. For example, relax a delivery deadline, and re-solve.

Finally, the most important difference is that the selective column generation approach is simpler. There are far fewer parameters we need to worry about tuning. We do not need to create a complex series of experiments to determine the tradeoffs in the different parameters under different conditions. Furthermore, it is much easier to explain this approach and how we reach a final solution.

We provide a brief analysis of the performance of our Route-based model using selective column generation and discuss the benefits and tradeoffs in Section 5.3.

### 4.5 Flexible Model

As we stated earlier, our Route-based model is inflexible in terms of departure and return times for routes. However, it is likely that there are multiple days on which a given route could feasibly depart and return. We adjust our Route-based model to include this flexibility in order to discuss the tradeoffs between model complexity and solution quality.

#### 4.5.1 Sets

We must adjust our definition of a node to separate a location from the particular day since our decision variables must include departure and return information. The sets we use in our Flexible model are:

- $K = \{1, 2, \ldots, |K|\}$: set of commodities
- $N = \{1, 2, \ldots, |N|\}$: set of nodes
- $T = \{1, 2, \ldots, |T|\}$: set of days in the scheduling horizon
- $E = \{1, 2, \ldots, |E|\}$: set of aircraft routes
- $E_{n,t} \in E$: subset of edges that are active from node $n$ on day $t$
- $E_k \subseteq E$: subset of aircraft routes that cover commodity $k$
- $K_l \subseteq K$: subset of commodities covered by route $l$
- $J(l)$: set of critical legs in terms of capacity for route $l$

#### 4.5.2 Parameters

The parameters we use in the Flexible model are the following:

- $ALOC_{n,t}$: number of aircraft allocated from wing $n$ on day $t$
• $W_{CAP_l}$: capacity of aircraft type for route $l$ in weight (stons)
• $V_{CAP_l}$: capacity of aircraft type for route $l$ in pallet positions
• $P_{CAP_l}$: capacity of aircraft type for route $l$ in passengers
• $D^k$: demand for commodity $k$ in quantity (number of items)
• $\omega^k$: weight of commodity $k$ in stons
• $\nu^k$: footprint of commodity $k$ in pallet positions
• $C_l$: fixed cost of flying on route $l$
• $W^k_l$: variable cost of transporting cargo of commodity $k$ on route $l$
• $P^k$: penalty associated with not covering a unit of commodity $k$
• $\delta^k_{i,j} = \begin{cases} 1, & \text{if leg } j \text{ of route } l \text{ contains commodity } k \\ 0, & \text{otherwise} \end{cases}$
• $\alpha^+_l$: earliest day an aircraft can begin route $l$
• $\beta^+_l$: latest day an aircraft can begin route $l$
• $\alpha^-_l$: earliest day an aircraft can finish route $l$
• $\beta^-_l$: latest day an aircraft can finish route $l$
• $\tau_{l,min}$: minimum travel time in days for route $l$
• $\Gamma_l$: maximum number of aircraft that we can assign to route $l$

In the Flexible model, many more of our parameters depend on the route $l$.

### 4.5.3 Decision Variables

We now have four sets of decision variables:

• $x^+_{i,t} =$ number of aircraft assigned to begin route $l$ on day $t$
• $x^-_{i,t} =$ number of aircraft assigned to finish route $l$ on day $t$
• $y^k_l =$ number of items of commodity $k$ assigned to route $l$
• $w^k =$ number of items of commodity $k$ unassigned
4.5.4 Formulation

We formulate our model in the following way:

\[
\begin{align*}
\min_{x, y, u} & \sum_{t \in T} \sum_{l \in E} C_l x_{l,t}^+ + \sum_{l \in E} \sum_{k \in K} W_l^k \omega^k y_{l}^k + \sum_{k \in K} P^k u^k \\
\text{s.t.} \quad & \sum_{l \in E_k} y_{l}^k + u^k \geq D^k, \quad \forall k \in K \\
& \sum_{t = \alpha^+} \beta^+ x_{l,t}^+ = \sum_{t = \alpha^-} \beta^- x_{l,t}^-, \quad \forall l \in E \\
& \sum_{t' = \max\{t + \tau_{l,\min}, \alpha_t\}} \beta^- x_{l,t'} \geq x_{l,t}^+, \quad l \in E, t \in \{\alpha_t^+, \alpha_t^+ + 1, \ldots, \beta_t^+\} \\
& \sum_{s=1}^t \sum_{n \in E_{n,t}} \left( x_{l,s}^+ - x_{l,s}^- \right) \leq ALOC_{n,t}, \quad \forall n \in N, t \in T \\
& \sum_{k \in K} \delta^k_{l,j} \omega^k y_{l}^k \leq \sum_{t \in T} WCAP_l x_{l,t}^+, \quad \forall l \in E, j \in J(l) \\
& \sum_{k \in K} \delta^k_{l,j} \nu^k y_{l}^k \leq \sum_{t \in T} VCAP_l x_{l,t}^+, \quad \forall l \in E, j \in J(l) \\
& \sum_{k \in K} \delta^k_{l,j} y_{l}^k \leq \sum_{t \in T} PCAP_l x_{l,t}^+, \quad \forall l \in E, j \in J(l)
\end{align*}
\]
\[ u^k \geq 0 \quad \forall k \in K \tag{4.27} \]

\[ x^+_{l,t}, x^-_{l,t}, y^k_l \in \mathbb{Z}^+ \quad \forall l \in E, k \in K, t \in T \tag{4.28} \]

In the Flexible model, the majority of our model remains the same. The objective is still to minimize the fixed cost of using a route, the variable cost to transport cargo on a route, and the penalty for not serving a commodity. Constraint (4.19) ensures demand is either served by a feasible route or it is penalized. In constraint (4.23) we restrict the number of aircraft active on a route on a given day by the number of allocated for that day. Finally, constraints (4.24)-(4.26) limit the total amount of cargo assigned on all parts of a route to the capacity of the route.

Where this model differs is providing a choice in when aircraft can begin and end a route. The additional constraints ensure that we are assigning valid aircraft flows. Constraint (4.20) requires all aircraft that begin a route to begin on one of the feasible days. We limit the total number of aircraft assigned to a route by the parameter \( \Gamma_l \). For most routes this parameter will be one, but we can relax this for the few routes that require multiple aircraft. Because we are calculating fixed cost based on the number of aircraft that begin a route, we will only assign the fewest number to begin a route to cover demand. Constraint (4.21) requires the number of aircraft that begin a route to equal the number that finish the route. This ensures we do not have more aircraft finishing a route and, thus, artificially increasing the number of aircraft allocated. Finally, we ensure travel time is valid in constraint (4.22) by restricting arrival time to be later by at least the minimum travel time.
Chapter 5

Computational Experiments

We performed a number of experiments to evaluate the performance of our models. The purpose of our computational experiments was to answer two questions: First, how well do each of our solution approaches scale in terms of network size and demand volume? Second, what benefit does aggregating requirements between mission type provide?

In Section 5.1, we begin by describing how we generated our data and what assumptions we made. We test the performance of our route generation algorithm in Section 5.2 and show how each solution approach scales in Section 5.3. In Section 5.4, we investigate the value of integrating lines of business in the flight planning process.

5.1 Data and Assumptions

USTRANSCOM, like many commercial companies, cannot publicly share data for confidentiality and national security reasons. For the purpose of this research, we generated artificial data in order to provide a general idea of the performance of our models.

5.1.1 Network

AMC utilizes a global network. To represent this network, we used 40 of the busiest international airports. The map in Figure 5-1 depicts the location of each of these ports. There are 18 ports in North America, 8 ports in Europe, 2 ports in the Near East, 11 ports in the Far East, and 1 port in Australia. Appendix E lists all ports with their airport code, city, country, latitude, and longitude.

Depending upon the experiment, we randomly choose a subset of these ports for consideration as POEs, PODs, and wings. For example, we might choose a random subset of 20 as the AMC network, and then we choose among those 20 the locations of each requirement’s POE and POD as well as the wing locations.
Figure 5-1: Ports considered for use in our experimental network

We approximate the travel time by calculating the great circle distance between each pair of ports, assuming an average flight speed of 500 miles per hour, and adding 30 minutes for take-off and landing [1]. We round each estimate to the nearest half hour. For simplicity, we assume there is only one aircraft type. Extending our experiments to include multiple aircraft types is straightforward because route generation can be done in parallel. In addition, if the travel time between two ports is greater than aircraft range, we assume an artificial refueling location at the maximum range. For example, if the travel time between two ports is ten hours and the aircraft maximum range is eight hours, the first flight segment is eight hours, followed by a refueling ground activity and a two hour flight segment. As a result, the intermediate refueling locations between two points are different depending on flight direction, and the effective size of the network is larger than the 40 locations.

5.1.2 Requirements

For each requirement, we must determine the time restrictions at the POE and POD, the number of units of each cargo type demanded, and whether it is hazardous or not.

We set the ALD of a requirement to a random day uniformly distributed in a week. The EAD can be zero to three days later with the following probability mass function:

\[
\Pr(EAD = x) = \begin{cases} 
0.1, & x = ALD, \\
0.8, & x = ALD + 1, \\
0.05, & x = ALD + 2, \\
0.05, & x = ALD + 3.
\end{cases}
\]
Table 5.1: Per unit cargo type specifications

<table>
<thead>
<tr>
<th>Cargo Type</th>
<th>Weight (lbs)</th>
<th>Footprint (pallet positions)</th>
<th>Seats</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bulk</td>
<td>5,000</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Oversized</td>
<td>15,000</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Outsized1</td>
<td>30,000</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Outsized2</td>
<td>50,000</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Outsized3</td>
<td>6,000</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>PAX</td>
<td>210</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Similarly, the LAD can be one to four days after the EAD with the following probability mass function:

\[
\Pr(LAD = x) = \begin{cases} 
0.1, & x = EAD + 1, \\
0.75, & x = EAD + 2, \\
0.1, & x = EAD + 3, \\
0.05, & x = EAD + 4.
\end{cases}
\]

We defined five cargo types in addition to a passenger (PAX). Table 5.1 describes each cargo type’s per unit capacity in terms of weight, footprint, and seats.

We determine the mix of cargo types and the number of units demanded of each in the following way. First, we determine the volume of cargo in terms of the number of full and partial aircraft necessary. In our experiments, a requirement uses between zero and three full aircraft and either zero or one partial aircraft. We define the random variable, \(X\), as the number of partial aircraft necessary, the random variable, \(Y\), as the number of full aircraft necessary, and the joint probability mass function as follows:

\[
\Pr(X = x, Y = y) = \begin{cases} 
0.2, & x = 0, y = 1, \\
0.01, & x = 0, y = 2, \\
0.01, & x = 0, y = 3, \\
0.6, & x = 1, y = 0, \\
0.09, & x = 1, y = 1, \\
0.06, & x = 1, y = 2, \\
0.03, & x = 1, y = 3.
\end{cases}
\]

Second, we determine the mix of cargo for each requirement in terms of cargo type and units demanded. The cargo mix for each full aircraft is always ten units of Bulk, two units of Oversized, and 40 units of PAX. Each requirement in need of a partial aircraft demands a random integer distributed between a lower bound (LB) and an upper bound (UB) of either Bulk, Oversized, Out-
Table 5.2: Partial aircraft cargo mix

<table>
<thead>
<tr>
<th>$x$</th>
<th>$\Pr(X = x)$</th>
<th>LB</th>
<th>UB</th>
<th>$\Pr(PAX)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bulk</td>
<td>0.5</td>
<td>3</td>
<td>15</td>
<td>0.4</td>
</tr>
<tr>
<td>Oversized</td>
<td>0.25</td>
<td>2</td>
<td>5</td>
<td>0.3</td>
</tr>
<tr>
<td>Outsized1</td>
<td>0.1</td>
<td>1</td>
<td>2</td>
<td>0.15</td>
</tr>
<tr>
<td>Outsized2</td>
<td>0.075</td>
<td>1</td>
<td>2</td>
<td>0.15</td>
</tr>
<tr>
<td>Outsized3</td>
<td>0.075</td>
<td>1</td>
<td>1</td>
<td>0.15</td>
</tr>
</tbody>
</table>

sized1, Outsized2, or Outsized3 cargo. Depending on the cargo type demanded, a requirement demands between 10 and 55 passengers uniformly with some probability, $\Pr(PAX)$. Table 5.2 lists the probability of each cargo type being demanded, the lower bound (LB) and upper bound (UB) of the number of units demanded, and the probability passengers are demanded.

Finally, we categorize each requirement as either hazardous or not. The probability of each requirement being hazardous is 0.05. Hazardous requirements cannot be onboard an aircraft with cargo that is not hazardous.

5.1.3 Aircraft and Aircrew

We assume that there is only one aircraft type and that the allocation at each wing is restricted by the number of aircraft. We model the change in aircraft allocation for mobility missions between days at any wing as a random walk. The number of aircraft allocated on the first day is a random integer between four and seven. On a given day, the number of aircraft allocated at a wing increases or decreases by two with probability 0.05, increases or decreases by one with probability 0.1, and remains the same with probability 0.7. The aircraft has a capacity of 150,000 pounds, 18 pallet positions, and 100 seats. The maximum range is ten hours and the minimum refueling time is two hours while the minimum (un)loading time is three hours. The maximum crew duty day is 24 hours and the minimum crew rest period is 16 hours.

5.1.4 Costs

Our objective function includes a fixed cost for using an aircraft route, a variable cost for transporting cargo on a route, and a penalty for not delivering a commodity. We calculated the fixed cost per hour to use a route as approximately $5000/hour by estimating the fuel consumption as 1722 gallons per hour and the price of jet fuel as $2.86 per gallon [8, 17]. We calculated the variable cost per hour to transport cargo on a route as approximately $0.0625 per pound per hour. We estimated this by using published data from the World Bank [36]. Finally, we calculated the penalty for not delivering a commodity as five times the cost to directly deliver the commodity from its POE to POD.
5.2 Route Generation

A significant portion of both the explicit column generation and selective column generation approaches is generating an initial set of feasible routes. We designed experiments to evaluate how limiting paths using dominance, aircraft capacity, and number of requirements affects the time required to generate paths and routes and the final solution.

In our first experiment for route generation, we determine an appropriate threshold for the number of requirements in a path. We generate paths using dominance and aircraft capacity. Figure 5-2 shows the average time required to build paths based on the number of requirements in the path. The algorithm builds all feasible paths within a reasonable amount of time when the total number of requirements is 20 or fewer. Building paths with four or more requirements requires more than six hours when there are 50 or 100 total requirements.

Limiting the number of requirements in a path to be at most three allows us to generate an initial set of routes for selective column generation in a reasonable amount of time. In addition, the final set of paths becomes the starting point to combine paths together to create routes for explicit column generation. In order for this portion of the algorithm to converge, limiting the number of requirements is necessary.

The second experiment generates an initial set of routes in four different ways and solves the Route-based model using all of the routes generated. We label and describe each of the four route generation methods in the following way:

- None: paths are not limited in any way;
Table 5.3: Limiting paths has minimal affect on solution quality and reduces the computational time necessary to generate routes dramatically

<table>
<thead>
<tr>
<th>Limit</th>
<th>Avg time (s)</th>
<th>Avg ( \Delta ) (%)</th>
<th>Max ( \Delta ) (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>248.5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Dom</td>
<td>20.5</td>
<td>0.93</td>
<td>3.78</td>
</tr>
<tr>
<td>Cap</td>
<td>20.2</td>
<td>1.10</td>
<td>4.08</td>
</tr>
<tr>
<td>Req</td>
<td>16.9</td>
<td>1.11</td>
<td>4.16</td>
</tr>
</tbody>
</table>

- Dom: paths are limited using dominance;
- Cap: paths are limited using dominance and aircraft capacity; and
- Req: paths are limited using dominance, aircraft capacity, and restricting the maximum number of requirements in a path to be three.

We experiment with 10, 15, and 20 requirements using a network with 15 ports and 3 wings and perform 10 trials of each experiment. Figure 5-3 shows the average time required to generate paths and routes using the different route generation methods for 10, 15, and 20 requirements. Table 5.3 shows the performance of each method for 15 requirements. We compare the performance of the three methods of limiting paths with the performance of not limiting paths in terms of total route generation time, average objective difference, and maximum objective difference. We define the objective difference as

\[ \Delta = \frac{(z_{\text{limited}} - z_{\text{none}})}{z_{\text{none}}} \]

where \( z_{\text{limited}} \) is the objective value of the solution using routes limited by one of the three methods and \( z_{\text{none}} \) is the objective value of the solution using all routes.

Figure 5-3 shows that limiting the number of paths we generate greatly reduces the computational time necessary for generating paths and routes. Although it requires more computational time to generate paths limiting the number using dominance and capacity compared to only dominance, it reduces the computational time necessary to generate routes as the number of requirements increases. As a result, the additional investment in computational time to reduce the number of paths is worthwhile as the number of requirements increases for explicit column generation, but it is not worth the additional time for selective column generation. Limiting the number of paths is crucial because generating routes is only the first step in both column generation approaches.

Table 5.3 shows that limiting the number of paths using any of the three methods does not result in a significant difference in the objective value, but the time reduces by over 90%. The average objective difference between the most limited solution and no limitations is 1.11% while the maximum objective difference over all ten trials is 4.16%.
Figure 5-3: Average time required to generate paths and routes based on how we limit paths.
5.3 Model Scaling

The success of any model depends on the practicality of its application. The initial computational results of route generation motivated the selective column generation approach. Even with limiting paths using dominance, aircraft capacity, and number of requirements in a path, generating routes for 20 requirements did not converge for two out of the ten trials. In practice, each of the three flight planning directorates could schedule dozens of requirements each week.

We designed two sets of experiments to evaluate the effectiveness of each solution approach depending on the size of the network and the volume of requirements. We divided these experiments by size and ran ten instances of each problem size. We use the first set of experiments to compare the performance of explicit column generation with selective column generation on a small network with few requirements. We use the second set of experiments to show the performance of selective column generation on instances with a greater number of requirements and discuss the limitations.

We use dominance and restrict the maximum number of requirements in a path to be three to limit the number of paths in generating an initial set of routes for both explicit and selective column generation. In addition, we use aircraft capacity to limit paths in explicit column generation as well. Finally, we limit the total solve time for all trials to six hours.

In explicit column generation, we use the dual variable values of routes in the model utilizing the same aircraft type and serving the same commodities on a critical leg to approximate the unknown dual variables for routes not in the model. We use a conservative estimate of the unknown dual variables to calculate the reduced cost of a route not in the model. We ensure each commodity is served by at least one route in the initial set and the number of routes is at least 30 times the number of requirements. On each iteration, we calculate the reduced cost of routes not in the model in random order and add those that are negative until the number of routes added is 10 times the number of requirements. We stop generating columns once there are none with negative reduced cost or the solution does not improve in two iterations.

In selective column generation, we use all routes in each iteration of column generation. We generate new routes by combining routes in the solution, assigning the closest wing, and adding those that are feasible. We generate routes in this way three out of every four iterations. On every fourth iteration, we generate new routes by assigning each route in the solution to its second closest wing in order to expand the solution search in case a wing is constrained in the number of aircraft available. We iterate in this way until no new routes are feasible or the solution does not improve in two iterations.

Table 5.4 summarizes the main results of the performance of selective column generation using explicit column generation as a baseline on small test problems. We report the average time to determine which routes to include in the Route-based model (time to generate initial routes plus column generation), the number of routes in the Route-based model, and the difference in objective
Table 5.4: Comparison of explicit and selective column generation for small test problems

<table>
<thead>
<tr>
<th>Test Problem</th>
<th>Explicit</th>
<th>Selective</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Req</td>
<td># of ports</td>
<td>Avg time (s)</td>
</tr>
<tr>
<td>10</td>
<td>15 (3)</td>
<td>6.9</td>
</tr>
<tr>
<td>10</td>
<td>30 (3)</td>
<td>6.6</td>
</tr>
<tr>
<td>10</td>
<td>30 (10)</td>
<td>17.8</td>
</tr>
<tr>
<td>20</td>
<td>15 (3)</td>
<td>526.8</td>
</tr>
<tr>
<td>20</td>
<td>30 (3)</td>
<td>450.1</td>
</tr>
<tr>
<td>20</td>
<td>30 (10)</td>
<td>610.8</td>
</tr>
</tbody>
</table>

value between the two approaches. We reported the average number of routes in the IP as a proxy for the time necessary to solve the IP. We found that in general, the length of time to find the optimal integer solution depends on the number of routes (proportional to the number of columns and rows) for problems. However, the length of time to prove its optimality was problem specific. As a result, we report the average number of routes in the IP. In this case, we define the difference in objective values as

\[
\Delta = \frac{(z_{\text{Selective}} - z_{\text{Explicit}})}{z_{\text{Explicit}}},
\]

where \(z_{\text{Selective}}\) is the objective value of the solution using selective column generation and \(z_{\text{Explicit}}\) is the objective value of the solution using explicit column generation.

In the instances with ten requirements, explicit and selective column generation were roughly equivalent in the length of time necessary to determine which routes to include in the final model. There is a significant difference, however, in the time necessary to determine which routes to include in the final model in the instances with 20 requirements. This difference is due to the time required to generate all feasible routes to consider in the initial step of explicit column generation.

The variable performance in objective cost between the two approaches is an interesting result. We attribute the difference in performance, in the cases where selective column generation provides a better solution, to relying on rules of thumb when combining paths together to generate routes for explicit column generation. Using time and space to limit the number of feasible routes while generating routes failed to identify routes that lowered the solution cost. On the other hand, we attribute the difference in performance, in the cases where explicit column generation provides a better solution, to relying on the routes in the solution in the first iteration of column generation to determine which paths to combine to generate routes for selective column generation. We can only improve the solution locally in selective column generation because we are only considering paths in the solution to combine to generate new routes. If a path is not in the initial solution, it is unlikely it will enter the solution in subsequent iterations, thus there is no opportunity to combine it with other paths to generate a new route.

We show the performance of selective column generation in test problems with a large volume of requirements in Table 5.5. We report the average time for each phase of selective column generation.
Table 5.5: Summary of results for selective column generation in large problems

<table>
<thead>
<tr>
<th>Test Problem</th>
<th>Selective</th>
<th>Initial</th>
<th>Column generation in IP</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Ports</td>
<td>Avg time (s)</td>
<td>Avg routes in IP</td>
<td></td>
</tr>
<tr>
<td>Req (Wings)</td>
<td>Avg routes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50 20 (4)</td>
<td>288.2</td>
<td>155.3</td>
<td>6,689</td>
</tr>
<tr>
<td>50 40 (4)</td>
<td>323.8</td>
<td>130.6</td>
<td>7,799</td>
</tr>
<tr>
<td>50 40 (10)</td>
<td>320.1</td>
<td>125.6</td>
<td>7,830</td>
</tr>
<tr>
<td>100 20 (4)</td>
<td>3,274.1</td>
<td>12,494.3*</td>
<td>60,131*</td>
</tr>
<tr>
<td>100 40 (4)</td>
<td>2,852.2</td>
<td>11,941.2*</td>
<td>54,826*</td>
</tr>
<tr>
<td>100 40 (10)</td>
<td>2,904.7</td>
<td>2,474.1</td>
<td>55,258</td>
</tr>
</tbody>
</table>

To highlight its strengths and weaknesses. In problems with 100 requirements and a small number of wings, fewer than half of the instances exceeded the total time threshold of six hours while generating new columns. The asterisk denotes those phases where we report the average of those that did complete generating new columns.

In all cases, we were able to generate an initial set of routes in less than an hour. Furthermore, over 90% of the number of routes in the IP were generated in the initial step and only contained one path. In the cases with only 50 requirements, on average, it took less than three minutes to solve the relaxed Route-based model to optimality multiple times with over 5,000 routes. However, we were not able to solve the relaxed Route-based model efficiently with twice as many requirements. Interestingly enough, we were able to solve the relaxed Route-based model efficiently with 100 requirements and a similar number of routes when there was a higher number of wings. Providing more choice in wings results in the same number of initial routes but with shorter empty legs. In addition, increasing the number of wings increased the number of total available aircraft, reducing the total demand for aircraft at any one wing. This resulted in vastly quicker times to find and prove an optimal solution.

5.4 Integrated versus Segregated Scheduling

Currently, TACC schedules missions independently based on lines of business (LOBs). Each of the three flight planning directorates is responsible for scheduling missions of one line of business and there is not a central planning authority with knowledge of all requirements. We show our model can be used to evaluate business process and policy decisions such as the value of integrating LOBs to schedule requirements.

In these experiments, we compare the solution of scheduling requirements independently based on mission type with scheduling all missions. We use selective column generation as the solution method for both cases with the exception that routes in the segregated case must serve requirements of the same type. For simplicity, we only assume two mission types and that each is equally likely. We use a network of 40 ports, 10 of which are wings. We limit the total time to solve each instance
Table 5.6: Comparison of scheduling requirements aggregated with scheduling requirements by LOB

<table>
<thead>
<tr>
<th>Test Problem</th>
<th>Integrated</th>
<th>Segregated</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Req</td>
<td>Avg routes</td>
<td>Avg AC used</td>
</tr>
<tr>
<td>25</td>
<td>1012.0</td>
<td>13.9</td>
</tr>
<tr>
<td>50</td>
<td>7223.4</td>
<td>23.4</td>
</tr>
</tbody>
</table>

To the earlier of six hours, or the time to prove a feasible integer solution is within 5% of the optimal solution.

We also test the impact that the number of requirements has on the benefit of integrating LOBs. We hypothesize that the benefit of integrating LOBs decreases as the number of requirements of each type increases compared to the number of resources because there will be more opportunities to combine requirements together to create efficient routes.

In both cases, integrating LOBs produces a better solution than building routes based on type, as summarized in Table 5.6. Integrating LOBs produced a solution that was 11.09% lower and used 1.8 fewer aircraft, on average, for instances with 25 requirements. The average improvement in the objective value for instances with 50 requirements decreased to 5.54%, but it used an average of 2.5 fewer aircraft. These results are significant given AMC had a FY12 budget of over $10B and attributed greater efficiency in one LOB to saving over $200M [34]. Furthermore, creating a schedule that uses fewer aircraft allows flight planners to use the additional capacity to react to unanticipated changes in demand or delays in the network.
Applying optimization methods to the military airlift scheduling problems has been and continues to be a challenging task. We have demonstrated that applying column generation techniques and checking aircraft routes for feasibility separate from the optimization model proves successful in reducing the computational time necessary to solve realistic-sized problem instances. However, there is much work to be done to improve our approach as well as further investigate new methods to solve the military airlift scheduling problem.

6.1 Summary of Results and Contributions

We summarize the major results and contributions in the following:

- We develop an algorithm to generate aircraft routes that are feasible in terms of travel time, ground time, and requirement time restrictions in order to reduce the size and complexity of the optimization model. As a result, we do not need to include these constraints in the Route-based model. We demonstrate that the use of dominance criteria to eliminate routes that serve the same requirements and utilize the same aircraft type does not have a significant impact on solution quality.

- We develop the deterministic Route-based model, an integer linear programming (IP) optimization model that is able to aggregate all types of requests to create a baseline schedule. The model ensures each commodity is served in its entirety or is penalized, each wing does not assign more aircraft than allocated, and that there is sufficient aircraft capacity assigned to routes transporting cargo. In addition, we present an alternative, the Flexible model, by modifying the Route-based model to allow for flexibility in the departure and arrival days for aircraft.
• We implement two column generation approaches to further reduce the size of the optimization model: explicit column generation and selective column generation. Explicit column generation enumerates all possible routes using the route generation algorithm in the initial step and identifies which of those routes to include in the model during column generation. Selective column generation generates routes with one path in the initial step and builds routes with multiple paths during column generation. We show that neither of the two approaches consistently performs better in terms of solution cost, but selective column generation produces a solution in a fraction of the time compared to explicit column generation. As a result, we are able to solve instances with up to 100 requirements using selective column generation.

• While we implement techniques to produce a schedule resilient to change, we provide the framework to develop a robust Route-based model to incorporate the level of uncertainty in the problem.

6.2 Future Work

This thesis presents a successful method to solve the military airlift scheduling problem, but there are many areas for future work. We summarize some specific recommendations for future work.

• Improve on the heuristics for generating feasible routes efficiently. On each iteration of selective column generation, we explore the solution space locally by generating new routes by combining routes in the solution together. Every few iterations, we attempt to add variety to the solution space by assigning the second closest wing for those routes in the solution. However, we could better identify paths to combine together to explore the solution space globally.

• Implement a hybrid approach that includes explicit column generation techniques to reduce the number of columns used in selective column generation. As the number of requirements increases, using all of the routes on each iteration of selective column generation becomes impractical. Instead, we can develop a hybrid approach to identify a subset of all routes in order to be able to solve the relaxed Route-based model efficiently on each iteration of column generation and to identify an integer solution.

• Develop a heuristic that schedules requirements in the way flight planners currently schedule missions. We are unable to quantify the improvement optimization provides in scheduling requirements without a baseline to which to compare. Furthermore, we must develop a simulation model to determine the impact optimization has in execution. It is possible that optimization creates a schedule that is significantly better than the current method in planning, but does not perform as well when uncertain data varies from the nominal case in execution.
- Develop a model that transforms the optimal solution into one with specific departure times for each flight segment in a route that is MOG feasible and still feasible in terms of travel time, ground time, crew rest, and requirement time restrictions.

- Extend the Route-based formulation and route generation to include aerial refueling. Aerial refueling is a vital capability of airlift operations as it means an aircraft range is only limited by the crew’s duty day. Considering aerial refueling while scheduling requirements may lead to further improvements in solution cost.

- Adapt the Route-based model to create a robust optimization model that includes the level of uncertainty inherent to the military airlift scheduling problem. We can take uncertainty into account in the way we build feasible routes, assign aircraft and cargo to those routes, and in how we choose the departure times for each flight segment in the final solution. It may be possible to develop schedules that are slightly more expensive in planning, but result in significant cost savings in execution.

- Investigate the impact of transferring ownership of aircraft from individual wings to AMC in general. This change eliminates the need for an aircraft to return to a specific wing and allows TACC to schedule aircraft and crew separately, similar to commercial practices. Scheduling in this way will allow AMC to better utilize aircraft because crews can rotate between an aircraft, thus limiting the time an aircraft spends on the ground.
Appendix A

Logic for \textit{POTENTIAL}

This function will filter out requirements that we know are unable to merge with a given path in \textit{NewPaths}. A requirement cannot merge with a path if the path already covers it or if the requirement completely fills the minimum aircraft required. Furthermore, a requirement whose departure (ALD) or arrival (LAD) times do not overlap with the path is impossible to merge because it will only result in two unique paths. Finally, a requirement must be compatible with at least one requirement covered by the path in order for there to be a potential feasible sequence. The requirement does not need to be compatible with all of the requirements covered by a path because it is possible not to be on board with some requirements simultaneously and satisfy the definition of a path. Therefore, some of this filtering will occur when finding a feasible sequence of stops.

Let

\( \delta_r \) be the earliest departure time for requirement \( r \)
\( \alpha_r \) be the latest arrival time for requirement \( r \)
\( \delta_L \) be the earliest departure time for cargo path \( L \)
\( \alpha_L \) be the latest arrival time for cargo path \( L \)
\( O(L) \) be the set of requirements on cargo path \( L \)
\( C(r) \) be the set of requirements compatible with requirement \( r \)

function \( P = \text{POTENTIAL}(L) \)

1. For each \( r \in \text{Requirements} \)
   
   (a) IF \( \alpha_r \geq \delta_L \) AND \( \alpha_L \geq \delta_r \) AND \( C(r) \cap O(L) \neq \emptyset \) AND \( r \not\in O(L) \), THEN \( P = P \cup r \)
Appendix B

Logic for \textit{SEQUENCE}

This function will find every possible sequence such that it satisfies the definition of a path, pickups occur before the corresponding delivery, and all pickups occur. In order to limit the number of possible combinations, we assume the order of the cargo path must remain the same. As a result, this ends up being equivalent to inserting the pickup and delivery location in every possible position except for directly delivering the new requirement before or after the original path. Since we know the time windows for the path and the requirement time restrictions, we can further limit the possible sequences depending on whether these times overlap. This is the same as part of the filtering logic in \textit{POTENTIAL}. Next, we filter out sequences that are not feasible in terms of travel time, ground time, crew rest, and commodity time restrictions using the \textit{FEASIBLE} function.

Let

\begin{align*}
\text{Result}(i) & \text{ be a vector of length } 2 \times |O(L)| + 2 \text{ indicating the location of stop } i \\
\end{align*}

function \( S(p) = \text{SEQUENCE}(p, L) \)

1. Let \( count = 1 \)

2. For each \( i = 1 : 2 \times |O(L)| \)

   (a) For each \( j = i + 1 : 2 \times |O(L)| + 2 \)

   i. IF \( i = 1 \) AND \( j = 2 \), THEN NEXT
   ii. \( \text{Result}(i) = p \)
   iii. \( \text{Result}(j) = p \)
   iv. Let \( position = 1 \)
   v. For each \( k = 1 : 2 \times |O(L)| + 2 \)
A. IF \( Result(k) \neq p \), THEN \( Result(k) = L(position) \) AND let \( position = position + 1 \)

vi. IF \( FEASIBLE(Result) = TRUE \), THEN \( S(count) = Result \) AND let \( count = count + 1 \)

**Example**  Let’s take a look at how many potential sequences could result if we merge a requirement with a cargo path servicing two requirements. Even though it is impossible, we will attempt to merge requirement 3 with the dominant sequence \( \{1^+, 2^+, 1^-, 2^-\} \). This results in 13 potential sequences:

1. \( \{3^+, 1^+, 3^-, 2^+, 1^-, 2^-\} \)
2. \( \{3^+, 1^+, 2^+, 3^-, 1^-, 2^-\} \)
3. \( \{3^+, 1^+, 2^+, 1^-, 3^-, 2^-\} \)
4. \( \{3^+, 1^+, 2^+, 1^-, 2^-, 3^-\} \)
5. \( \{1^+, 3^+, 3^-, 2^+, 1^-, 2^-\} \)
6. \( \{1^+, 3^+, 2^+, 3^-, 1^-, 2^-\} \)
7. \( \{1^+, 3^+, 2^+, 1^-, 3^-, 2^-\} \)
8. \( \{1^+, 3^+, 2^+, 1^-, 2^-, 3^-\} \)
9. \( \{1^+, 2^+, 3^+, 3^-, 1^-, 2^-\} \)
10. \( \{1^+, 2^+, 3^+, 1^-, 3^-, 2^-\} \)
11. \( \{1^+, 2^+, 3^+, 1^-, 2^-, 3^-\} \)
12. \( \{1^+, 2^+, 1^-, 3^+, 3^-, 2^-\} \)
13. \( \{1^+, 2^+, 1^-, 3^+, 2^-, 3^-\} \)

As you can see the number of potential sequences grows exponentially as the number of requirements increases. There are four potential sequences to merge two requirements, thirteen potential sequences to merge three requirements, 26 potential sequences to merge four requirements and so on. This requires significant computational time to check the feasibility of each.
Appendix C

Logic for FEASIBLE

In this function, we calculate the time windows for the sequence and determine if it is feasible. We begin by calculating the earliest arrival time and the earliest departure time at each stop in the sequence. We can do this in the order of the sequence because they only depend on the travel time, ground time, crew rest, and requirement time restrictions up to that stop in the sequence so far. At each delivery location, we can check feasibility by comparing the earliest arrival time to the LAD for the requirement being delivered. It is sufficient to only calculate the earliest arrival and departure times for the sequence to determine feasibility. However, we must calculate the latest arrival and departure times for the sequence to calculate the delay allowable at each stop in the sequence which we use to differentiate between feasible paths. We also need the latest arrival time of a path to filter requirements in POTENTIAL and to filter sequences in SEQUENCE.

It is more difficult to calculate the latest arrival and departure times than the earliest arrival and departure times for a sequence because they depend on the LAD of requirements later in the sequence while crew rest depends on information obtained earlier in the sequence. Therefore, we calculate the latest arrival and departure times in two steps. First, we calculate inflated latest arrival and departure times based only on travel time, ground time, and crew rest in the order of the sequence where the latest arrival time at the first stop is the maximum LAD over all requirements covered in the path. We specify the latest arrival time at the first step in this way in order for the second step to work. Using these times we calculate the minimum travel times defined as follows:

**Definition** The *minimum travel time* for a sequence is the difference between the latest arrival time at the final stop and the latest departure time at the first stop when these times only depend on travel time, ground time, and crew rest.

In the second step, we adjust the latest arrival and departure times to get valid times based on the LAD of all delivery locations. We can calculate the difference between the inflated latest arrival time and the true latest arrival time for all delivery locations. We begin by adjusting all latest arrival
and departure times by the difference calculated for the last stop. We can proceed in reverse order for all delivery locations in the sequence. If the latest arrival time at a delivery location still exceeds the true latest arrival time, then we adjust the latest arrival time for the current stop and all prior stops and the latest departure time for all prior stops further.

Let

- \( On \) be the set of requirements currently on the aircraft
- \( E.Arrive(i) \) be the earliest arrival time at stop \( i \)
- \( E.Depart(i) \) be the earliest departure time at stop \( i \)
- \( L.Arrive(i) \) be the latest arrival time at stop \( i \)
- \( L.Depart(i) \) be the latest departure time at stop \( i \)
- \( Last \) be the location of the prior stop
- \( Next \) be the location of the next stop
- \( POE(i) \) be a function that returns the POE location for stop \( i \)
- \( POD(i) \) be a function that returns the POD location for stop \( i \)
- \( Ground \) be the minimum ground time for loading/unloading at a pickup or delivery location
- \( Ground.Fuel \) be the minimum ground time for refueling at an intermediate location
- \( Duty.Time \) be the total duty time since the last crew rest
- \( FDP \) be the maximum time a crew can work
- \( Crew.Rest \) be the minimum time a crew needs between shifts
- \( Routing \) be a function that takes the origin and destination as input and outputs the number of legs between the two points and the length of each of the legs
- \( \tau_{a,b} \) be the minimum travel time from \( a \) to \( b \)

function \( FEASIBLE(L) \)

1. \( On = L(1) \)
2. \( E.Arrive(1) = ALD(1) \)
3. \( E.Depart(1) = ALD(1) + Ground \)
4. \( Last = POE(1) \)
5. \( Duty.Time = 0 \)
6. For each \( i = 2 : 2 *|O(L)| \)
(a) IF $L(i) \notin On$, THEN

i. For each $r \in On$
   A. IF $L(i) \notin C(r)$, THEN return $FALSE$

ii. $On = On \cup L(i)$

iii. $R = ROUTING(Last, POE(i))$

iv. IF $Duty.Time + \tau_{Last.POE(i)} < FDP$, THEN
   A. $E.Arrive(i) = E.Depart(i - 1) + \tau_{Last.POE(i)}$
   B. $E.Depart(i) = \max\{ALD(i) + Ground, E.Arrive(i) + Ground\}$
   C. IF $E.Depart(i) - E.Arrive(i) \geq Crew.Rest$, THEN
       • $Duty.Time = 0$
   D. ELSE
       • $Duty.Time = Duty.Time + E.Depart(i) - E.Depart(i - 1)$

v. ELSE IF $R(1) = 1$, THEN
   A. $E.Depart(i - 1) = \max\{E.Depart(i - 1), E.Arrive(i - 1) + Crew.Rest\}$
   B. $E.Arrive(i) = E.Depart(i - 1) + \tau_{Last.POE(i)}$
   C. $E.Depart(i) = \max\{ALD(i) + Ground, E.Arrive(i) + Ground\}$
   D. IF $E.Depart(i) - E.Arrive(i) \geq Crew.Rest$, THEN
       • $Duty.Time = 0$
   E. ELSE
       • $Duty.Time = \tau_{Last.POE(i)} + E.Depart(i) - E.Arrive(i)$

vi. ELSE
   A. IF $Duty.Time + R(2) \geq FDP$, THEN
       • $E.Depart(i - 1) = \max\{E.Depart(i - 1), E.Arrive(i - 1) + Crew.Rest\}$
       • $Duty.Time = R(2) + Ground.Fuel$
   B. ELSE
       • $Duty.Time = Duty.Time + R(2) + Ground.Fuel$
   C. $Rest = 0$
   D. for $i = 3 : length(R)$
       • IF $Duty.Time + R(i) < FDP$, THEN
       • ELSE
           - $Rest = Rest + 1$
- Duty.Time = R(i) + Ground.Fuel

E. E.Arrive(i) = E.Depart(i-1) + τLast,POE(i) + Crew.Rest + Rest + Ground.Fuel + (R(1) - Rest - 1)

F. E.Depart(i) = max {ALD(i) + Ground, E.Arrive(i) + Ground}

G. IF E.Depart(i) - E.Arrive(i) ≥ Crew.Rest, THEN
   • Duty.Time = 0

H. ELSE
   • Duty.Time = Duty.Time + E.Depart(i) - E.Arrive(i) - Ground.Fuel

vii. Last = POE(i)

(b) ELSE

i. On = On \ L(i)

ii. R = ROUTING(Last, POD(i))

iii. E.Arrive(i) = E.Depart(i-1) + τLast,POD(i)

iv. IF E.Arrive(i) + Ground ≥ LAD(i), THEN return FALSE

v. IF Duty.Time + τLast,POD(i) < FDP AND E.Arrive(i) ≥ EAD(i), THEN
   A. E.Depart(i) = E.Arrive(i) + Ground
   B. Duty.Time = Duty.Time + E.Depart(i) - E.Depart(i-1)

vi. ELSE IF Duty.Time + τLast,POD(i) < FDP AND Duty.Time + EAD(i) - E.Depart(i-1) < FDP, THEN
   A. IF EAD(i) - E.Arrive(i) ≥ Crew.Rest, THEN
      • E.Arrive(i) = EAD(i)
      • E.Depart(i) = E.Arrive(i) + Ground
      • Duty.Time = R(length(R)) + Ground
   B. ELSE
      • E.Arrive(i) = EAD(i)
      • E.Depart(i) = E.Arrive(i) + Ground
      • Duty.Time = Duty.Time + E.Depart(i) - E.Depart(i-1)
   C. IF R(1) = 1, THEN, E.Depart(i−1) = EAD(i) - τLast,POD(i)

vii. ELSE IF R(1) = 1 AND E.Arrive(i-1) + Crew.Rest + τLast,POD(i) ≥ EAD(i), THEN
   A. E.Depart(i−1) = max {E.Depart(i-1), E.Arrive(i-1) + Crew.Rest}
   B. E.Arrive(i) = E.Depart(i-1) + τLast,POD(i)
C. IF $E.\text{Arrive}(i) + \text{Ground} \geq LAD(i)$, THEN return $\text{FALSE}$

D. $E.\text{Depart}(i) = E.\text{Arrive}(i) + \text{Ground}$

E. $Duty.\text{Time} = E.\text{Depart}(i) - E.\text{Depart}(i - 1)$

viii. ELSE IF $R(1) = 1$, THEN

A. $E.\text{Depart}(i - 1) = EAD(i) - \tau_{\text{Last,POD}(i)}$

B. $E.\text{Arrive}(i) = EAD(i)$

C. $E.\text{Depart}(i) = E.\text{Arrive}(i) + \text{Ground}$

D. $Duty.\text{Time} = E.\text{Depart}(i) - E.\text{Depart}(i - 1)$

ix. ELSE

A. IF $Duty.\text{Time} + R(2) \geq FDP$, THEN

- $E.\text{Depart}(i - 1) = \max \{E.\text{Depart}(i - 1), E.\text{Arrive}(i - 1) + \text{Crew.Rest}\}$

- $Duty.\text{Time} = R(2) + \text{Ground.Fuel}$

B. ELSE

- $Duty.\text{Time} = Duty.\text{Time} + R(2) + \text{Ground.Fuel}$

C. $\text{Rest} = 0$

D. for $i = 3 : \text{length}(R)$

- IF $Duty.\text{Time} + R(i) < FDP$, THEN

- $Duty.\text{Time} = Duty.\text{Time} + R(i) + \text{Ground.Fuel}$

- ELSE

- $\text{Rest} = \text{Rest} + 1$

- $Duty.\text{Time} = R(i) + \text{Ground.Fuel}$

E. $E.\text{Arrive}(i) = E.\text{Depart}(i - 1) + \tau_{\text{Last,POE}(i)} + \text{Crew.Rest*Rest + Ground.Fuel*}(R(1) - \text{Rest} - 1)$

F. IF $E.\text{Arrive}(i) + \text{Ground} \geq LAD(i)$, THEN return $\text{FALSE}$

G. IF $E.\text{Arrive}(i) \geq EAD(i)$, THEN

- $E.\text{Depart}(i) = E.\text{Arrive}(i) + \text{Ground}$

- $Duty.\text{Time} = Duty.\text{Time} + E.\text{Depart}(i) - E.\text{Arrive}(i) - \text{Ground.Fuel}$

H. ELSE IF $EAD(i) - E.\text{Arrive}(i) \geq \text{Crew.Rest}$, THEN

- $E.\text{Arrive}(i) = EAD(i)$

- $E.\text{Depart}(i) = E.\text{Arrive}(i) + \text{Ground}$

- $Duty.\text{Time} = R(\text{length}(R)) + \text{Ground}$

I. ELSE
\[ E.\text{Arrive}(i) = EAD(i) \]

\[ E.\text{Depart}(i) = E.\text{Arrive}(i) + \text{Ground} \]

\[ \text{Duty.Time} = \text{Duty.Time} + E.\text{Depart}(i) - E.\text{Arrive}(i) - \text{Ground.Fuel} \]

\[ \text{Last} = \text{POD}(i) \]

7. \( \text{On} = L(1) \)

8. \( L.\text{Depart}(1) = \max_{i \in O(L)} \{ LAD(i) \} \)

9. \( L.\text{Arrive}(1) = L.\text{Depart}(1) - \text{Ground} \)

10. \( \text{Last} = \text{POE}(1) \)

11. \( \text{Duty.Time} = 0 \)

12. For each \( i = 2 : 2 \times |O(L)| \)

(a) IF \( L(i) \not\subset \text{On}, \) THEN

i. \( \text{On} = \text{On} \cup L(i) \)

ii. \( R = \text{ROUTING(\text{Last, POE}(i))} \)

iii. IF \( \text{Duty.Time} + \tau_{\text{Last, POE}(i)} < FDP, \) THEN

A. \( L.\text{Arrive}(i) = L.\text{Depart}(i - 1) + \tau_{\text{Last, POE}(i)} \)

B. \( L.\text{Depart}(i) = L.\text{Arrive}(i) + \text{Ground} \)

C. \( \text{Duty.Time} = \text{Duty.Time} + L.\text{Depart}(i) - L.\text{Depart}(i - 1) \)

iv. ELSE IF \( R(1) = 1, \) THEN

A. \( L.\text{Depart}(i - 1) = L.\text{Arrive}(i - 1) + \text{Crew.Rest} \)

B. \( L.\text{Arrive}(i) = L.\text{Depart}(i - 1) + \tau_{\text{Last, POE}(i)} \)

C. \( L.\text{Depart}(i) = L.\text{Arrive}(i) + \text{Ground} \)

D. \( \text{Duty.Time} = L.\text{Depart}(i) - L.\text{Depart}(i - 1) \)

v. ELSE

A. IF \( \text{Duty.Time} + R(2) \geq FDP, \) THEN

\[ \bullet L.\text{Depart}(i - 1) = L.\text{Arrive}(i - 1) + \text{Crew.Rest} \]

\[ \bullet \text{Duty.Time} = R(2) + \text{Ground.Fuel} \]

B. ELSE

\[ \bullet \text{Duty.Time} = \text{Duty.Time} + R(2) + \text{Ground.Fuel} \]

C. \( \text{Rest} = 0 \)

D. for \( i = 3 : \text{length}(R) \)
• IF \( \text{Duty.Time} + R(i) < FDP \), THEN
  - \( \text{Duty.Time} = \text{Duty.Time} + R(i) + \text{Ground.Fuel} \)

• ELSE
  - \( \text{Rest} = \text{Rest} + 1 \)
  - \( \text{Duty.Time} = R(i) + \text{Ground.Fuel} \)

E. \( L.\text{Arrive}(i) = L.\text{Depart}(i-1) + \tau_{\text{Last.POE}(i)} + \text{Crew.Rest} \cdot \text{Rest} + \text{Ground.Fuel} \cdot (R(1) - \text{Rest} - 1) \)

F. \( L.\text{Depart}(i) = L.\text{Arrive}(i) + \text{Ground} \)

G. \( \text{Duty.Time} = \text{Duty.Time} + L.\text{Depart}(i) - L.\text{Depart}(i-1) - \text{Ground.Fuel} \)

vi. \( \text{Last} = \text{POE}(i) \)

(b) ELSE

i. \( \text{On} = \text{On} \setminus L(i) \)

ii. \( R = \text{ROUTING(Last, POD}(i)) \)

iii. IF \( \text{Duty.Time} + \tau_{\text{Last.POE}(i)} < FDP \), THEN
    A. \( L.\text{Arrive}(i) = L.\text{Depart}(i-1) + \tau_{\text{Last.\text{POD}(i)}} \)
    B. \( L.\text{Depart}(i) = L.\text{Arrive}(i) + \text{Ground} \)
    C. \( \text{Duty.Time} = \text{Duty.Time} + L.\text{Depart}(i) - L.\text{Depart}(i-1) \)

iv. ELSE IF \( R(1) = 1 \), THEN
    A. \( L.\text{Depart}(i-1) = L.\text{Arrive}(i-1) + \text{Crew.Rest} \)
    B. \( L.\text{Arrive}(i) = L.\text{Depart}(i-1) + \tau_{\text{Last.\text{POD}(i)}} \)
    C. \( L.\text{Depart}(i) = L.\text{Arrive}(i) + \text{Ground} \)
    D. \( \text{Duty.Time} = L.\text{Depart}(i) - L.\text{Depart}(i-1) \)

v. ELSE
    A. IF \( \text{Duty.Time} + R(2) \geq FDP \), THEN
        • \( L.\text{Depart}(i-1) = L.\text{Arrive}(i-1) + \text{Crew.Rest} \)
        • \( \text{Duty.Time} = R(2) + \text{Ground.Fuel} \)
    B. ELSE
        • \( \text{Duty.Time} = \text{Duty.Time} + R(2) + \text{Ground.Fuel} \)
    C. \( \text{Rest} = 0 \)
    D. for \( i = 3 \) : \( \text{length}(R) \)
        • IF \( \text{Duty.Time} + R(i) < FDP \), THEN
            - \( \text{Duty.Time} = \text{Duty.Time} + R(i) + \text{Ground.Fuel} \)
• ELSE

  - \( Rest = Rest + 1 \)
  - \( Duty.Time = R(i) + Ground.Fuel \)

E. \( L.Arrive(i) = L.Depart(i-1) + \tau_{Last,POD(i)} + Crew.Rest \times Rest + Ground.Fuel \times (R(1) - Rest - 1) \)

F. \( L.Depart(i) = L.Arrive(i) + Ground \)

G. \( Duty.Time = Duty.Time + L.Depart(i) - L.Depart(i-1) - Ground.Fuel \)

vi. \( Last = POD(i) \)

13. Calculate minimum travel time

14. For each \( i = 1 : length(L) \)

   (a) \( L.Arrive(i) = L.Arrive(i) - L.Arrive(length(L)) + LAD(length(L)) - Ground \)

   (b) \( L.Depart(i) = L.Depart(i) - L.Arrive(length(L)) + LAD(length(L)) - Ground \)

15. \( On = L(length(L)) \)

16. For each \( i = length(L) - 1 : 1 \)

   (a) IF \( L(i) \not\in On \) AND \( L.Arrive(i) + Ground > LAD(i) \), THEN

      i. For each \( j = 1 : i - 1 \)

         A. \( L.Arrive(j) = L.Arrive(j) - L.Arrive(i) + LAD(i) - Ground \)

         B. \( L.Depart(j) = L.Depart(j) - L.Arrive(i) + LAD(i) - Ground \)

   ii. \( L.Arrive(j) = L.Arrive(j) - L.Arrive(i) + LAD(i) - Ground \)

17. return \( TRUE \)

Example

Let

\[
\begin{align*}
Ground &= 3 \\
FDP &= 15 \\
Crew.Rest &= 12 \\
Max.Tour &= 120
\end{align*}
\]

We will use the same data from the example describing the overall logic. We will determine if the sequence \( \{1^+,3^+,3^-,1^-\} \) is feasible.
Figure C-1: Sequence of a possible path with travel times

Table C.1: Requirements data

<table>
<thead>
<tr>
<th>Requirement</th>
<th>POE</th>
<th>POD</th>
<th>ALD</th>
<th>EAD</th>
<th>LAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>X</td>
<td>0</td>
<td>24</td>
<td>72</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>Y</td>
<td>24</td>
<td>48</td>
<td>72</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>Z</td>
<td>24</td>
<td>48</td>
<td>96</td>
</tr>
</tbody>
</table>

The earliest an aircraft can depart the first stop is the available to load date plus the minimum time to load the aircraft. The earliest arrival time at the first stop does not matter, so we set it to the available to load date at the first stop. The vectors of interest at this point are:

\[ E.Arrive = [0] \]

\[ E.Depart = [3] \]

\[ Duty.Time = 0 \]

Next, we must calculate the earliest arrival and departure times for every subsequent stop. The second stop is a pickup, so we ensure that requirement 2 is compatible with requirement 1. Because the travel time from location A to location C is 4 units which is less than \( FDP \), then the earliest arrival time at the second stop is the earliest departure time at the previous stop plus travel time. The earliest departure time at stop 2 is the maximum of the earliest departure time plus the minimum time to load the aircraft and the ALD plus the minimum time to load the aircraft. In this case, the earliest departure time is 27. Because the time between the earliest arrival and departure is greater than a crew rest, the duty time gets reset. As a result, our vectors of interest are:

\[ E.Arrive = [0, 7] \]

\[ E.Depart = [3, 27] \]

\[ Duty.Time = 0 \]
The third stop is a delivery. The travel time from location $C$ to location $Z$ is 6 units, so the earliest the aircraft can possibly arrive is 33. However, the EAD for this requirement is 48. Because there is only one flight segment from location $C$ to location $Z$, we must adjust the earliest departure time to ensure it does not arrive before EAD. There is enough time to unload the aircraft before LAD. The vectors after the third stop are:

\[
E.\text{Arrive} = [0, 7, 48]
\]

\[
E.\text{Depart} = [3, 42, 51]
\]

\[
\text{Duty.Time} = 9
\]

The last stop is a delivery. The travel time from location $Z$ to location $X$ is 6 units, so the earliest the aircraft can possibly arrive is 58. This is after requirement 1’s EAD and the additional duty time for travel time is less than $FDP$, so we just need to calculate the earliest departure time. This is the earliest arrival time plus the minimum time to unload the aircraft. The vectors of interest are:

\[
E.\text{Arrive} = [0, 7, 48, 54]
\]

\[
E.\text{Depart} = [3, 43, 51, 57]
\]

Next, we calculate inflated latest departure and arrival times at each stop by not taking into account requirement time restrictions. The maximum LAD between requirement 1 and 3 is 96, so the vectors of interest are:

\[
L.\text{Arrive} = [93]
\]

\[
L.\text{Depart} = [96]
\]

\[
\text{Duty.Time} = 0
\]

The travel time for the first flight segment is 4 time units which is less than $FDP$, so the vectors become:

\[
L.\text{Arrive} = [93, 100]
\]

\[
L.\text{Depart} = [96, 103]
\]

\[
\text{Duty.Time} = 7
\]

The travel time for the second flight segment is 6 time units, which means we can land without exceeding $FDP$. Including the third stop, the vectors of interest are:
\[ L.\text{Arrive} = [93, 100, 109] \]

\[ L.\text{Depart} = [96, 103, 112] \]

\[ \text{Duty.Time} = 16 \]

After including the ground time to unload the aircraft results in a duty time exceeding \( FDP \), so we crew rest no matter the length of the next flight segment. As a result, we adjust the latest departure time at the third stop and calculate the latest arrival time and latest departure time at the fourth stop based on the travel time.

\[ L.\text{Arrive} = [93, 100, 109, 127] \]

\[ L.\text{Depart} = [96, 103, 121, 130] \]

We are able to calculate the minimum travel time at this point. For this sequence, the minimum travel time is 31 time units. Next, we must adjust the latest arrival and departure times to ensure we arrive by LAD at all delivery locations. We do this in reverse order because the latest arrival and departure times depend on the LAD of stops in the future. We adjust the latest arrival and departure times at all locations by the amount the inflated latest arrival time exceeded the time necessary to unload the cargo by LAD (69). As a result, we adjust all times by 58:

\[ L.\text{Arrive} = [35, 42, 51, 69] \]

\[ L.\text{Depart} = [38, 45, 63, 72] \]

Next, we check each subsequent delivery location in reverse order. The third stop is the only other delivery location. In this case, the latest arrival time provides enough time to unload the cargo by LAD. The table below shows the earliest and latest arrival and departure times for each location.

<table>
<thead>
<tr>
<th></th>
<th>( 1^+ )</th>
<th>( 3^+ )</th>
<th>( 3^- )</th>
<th>( 1^- )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Earliest arrival time</td>
<td>0</td>
<td>7</td>
<td>48</td>
<td>54</td>
</tr>
<tr>
<td>Earliest departure time</td>
<td>3</td>
<td>43</td>
<td>51</td>
<td>57</td>
</tr>
<tr>
<td>Latest arrival time</td>
<td>35</td>
<td>42</td>
<td>51</td>
<td>69</td>
</tr>
<tr>
<td>Latest departure time</td>
<td>38</td>
<td>45</td>
<td>63</td>
<td>72</td>
</tr>
</tbody>
</table>

Next, we are able to calculate the minimum delay allowable over all stops. As you can see from the table, the minimum delay allowable is 2 time units.
Appendix D

Logic for DOMINANT

This function will determine which sequence is dominant among all those that are feasible. We define dominance as the sequence with the greatest minimum delay allowable over all the stops. In other words, it is the sequence that is most resilient to delays or changes. In case of ties, we first choose the sequence with the shortest minimum travel time. We break ties further by choosing the sequence with the longest scheduling horizon which allows the best chance of merging more requirements in future iterations. We define scheduling horizon as follows:

**Definition** The *scheduling horizon* is the difference between the latest departure time at the last stop and the earliest departure time at the first stop

If ties remain, we choose a sequence arbitrarily.

1. For each sequence
   
   (a) Determine the minimum delay over all stops

2. Choose the sequence with the maximum minimum delay over all stops

3. Break ties by choosing the sequence with the minimum travel time

4. Break ties by choosing the sequence with the longest scheduling horizon
Appendix E

Port Locations

<table>
<thead>
<tr>
<th>Code</th>
<th>City</th>
<th>Country</th>
<th>Latitude</th>
<th>Longitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMS</td>
<td>Amsterdam</td>
<td>Netherlands</td>
<td>52.30861</td>
<td>4.763889</td>
</tr>
<tr>
<td>ATL</td>
<td>Atlanta</td>
<td>USA</td>
<td>33.63672</td>
<td>-84.4281</td>
</tr>
<tr>
<td>BKK</td>
<td>Bangkok</td>
<td>Thailand</td>
<td>13.68111</td>
<td>100.7473</td>
</tr>
<tr>
<td>CAN</td>
<td>Guangzhou</td>
<td>China</td>
<td>23.39244</td>
<td>113.2988</td>
</tr>
<tr>
<td>CDG</td>
<td>Paris</td>
<td>France</td>
<td>49.01278</td>
<td>2.55</td>
</tr>
<tr>
<td>CGK</td>
<td>Jakarta</td>
<td>Indonesia</td>
<td>-6.12557</td>
<td>106.6559</td>
</tr>
<tr>
<td>DEN</td>
<td>Denver</td>
<td>USA</td>
<td>39.86166</td>
<td>-104.673</td>
</tr>
<tr>
<td>DFW</td>
<td>Dallas</td>
<td>USA</td>
<td>32.89683</td>
<td>-97.038</td>
</tr>
<tr>
<td>DMK</td>
<td>Bangkok</td>
<td>Thailand</td>
<td>13.91258</td>
<td>100.6068</td>
</tr>
<tr>
<td>DTW</td>
<td>Detroit</td>
<td>USA</td>
<td>42.21244</td>
<td>-83.3534</td>
</tr>
<tr>
<td>DXB</td>
<td>Dubai</td>
<td>United Arab Emirates</td>
<td>25.25278</td>
<td>55.36444</td>
</tr>
<tr>
<td>EWR</td>
<td>New York</td>
<td>USA</td>
<td>40.6925</td>
<td>-74.1687</td>
</tr>
<tr>
<td>FCO</td>
<td>Rome</td>
<td>Italy</td>
<td>41.80448</td>
<td>12.2508</td>
</tr>
<tr>
<td>FRA</td>
<td>Frankfurt</td>
<td>Germany</td>
<td>50.02642</td>
<td>8.543125</td>
</tr>
<tr>
<td>HKG</td>
<td>Hong Kong</td>
<td>Hong Kong</td>
<td>22.30892</td>
<td>113.9146</td>
</tr>
<tr>
<td>HND</td>
<td>Tokyo</td>
<td>Japan</td>
<td>35.55226</td>
<td>139.7797</td>
</tr>
<tr>
<td>IAH</td>
<td>Houston</td>
<td>USA</td>
<td>29.98443</td>
<td>-95.3414</td>
</tr>
<tr>
<td>IST</td>
<td>Istanbul</td>
<td>Turkey</td>
<td>40.97692</td>
<td>28.81461</td>
</tr>
<tr>
<td>JFK</td>
<td>New York</td>
<td>USA</td>
<td>40.63975</td>
<td>-73.7789</td>
</tr>
<tr>
<td>KUL</td>
<td>Kuala Lumpur</td>
<td>Malaysia</td>
<td>2.745578</td>
<td>101.7099</td>
</tr>
<tr>
<td>LAS</td>
<td>Las Vegas</td>
<td>USA</td>
<td>36.08006</td>
<td>-115.152</td>
</tr>
<tr>
<td>LAX</td>
<td>Los Angeles</td>
<td>USA</td>
<td>33.94254</td>
<td>-118.408</td>
</tr>
<tr>
<td>Code</td>
<td>City</td>
<td>Country</td>
<td>Latitude</td>
<td>Longitude</td>
</tr>
<tr>
<td>------</td>
<td>----------</td>
<td>---------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>LGW</td>
<td>London</td>
<td>United Kingdom</td>
<td>51.14806</td>
<td>-0.19028</td>
</tr>
<tr>
<td>LHR</td>
<td>London</td>
<td>United Kingdom</td>
<td>51.4775</td>
<td>-0.46139</td>
</tr>
<tr>
<td>MAD</td>
<td>Madrid</td>
<td>Spain</td>
<td>40.49356</td>
<td>-3.56676</td>
</tr>
<tr>
<td>MCO</td>
<td>Orlando</td>
<td>USA</td>
<td>28.42939</td>
<td>-81.309</td>
</tr>
<tr>
<td>MIA</td>
<td>Miami</td>
<td>USA</td>
<td>25.79325</td>
<td>-80.2906</td>
</tr>
<tr>
<td>MSP</td>
<td>Orlando</td>
<td>USA</td>
<td>44.88196</td>
<td>-93.215</td>
</tr>
<tr>
<td>MUC</td>
<td>Munich</td>
<td>Germany</td>
<td>48.35378</td>
<td>11.78609</td>
</tr>
<tr>
<td>NRT</td>
<td>Narita</td>
<td>Japan</td>
<td>35.76472</td>
<td>140.3864</td>
</tr>
<tr>
<td>ORD</td>
<td>Chicago</td>
<td>USA</td>
<td>41.9786</td>
<td>-87.9048</td>
</tr>
<tr>
<td>PEK</td>
<td>Beijing</td>
<td>China</td>
<td>40.08011</td>
<td>116.5846</td>
</tr>
<tr>
<td>PHX</td>
<td>Phoenix</td>
<td>USA</td>
<td>33.43428</td>
<td>-112.309</td>
</tr>
<tr>
<td>PVC</td>
<td>Shanghai</td>
<td>China</td>
<td>31.14338</td>
<td>121.8052</td>
</tr>
<tr>
<td>SEA</td>
<td>Seattle</td>
<td>USA</td>
<td>47.449</td>
<td>-122.309</td>
</tr>
<tr>
<td>SFO</td>
<td>San Francisco</td>
<td>USA</td>
<td>37.61897</td>
<td>-122.375</td>
</tr>
<tr>
<td>SIN</td>
<td>Singapore</td>
<td>Singapore</td>
<td>1.350189</td>
<td>103.9944</td>
</tr>
<tr>
<td>STL</td>
<td>St. Louis</td>
<td>USA</td>
<td>38.7487</td>
<td>-90.37</td>
</tr>
<tr>
<td>SYD</td>
<td>Sydney</td>
<td>Australia</td>
<td>-33.9461</td>
<td>151.1772</td>
</tr>
<tr>
<td>YYZ</td>
<td>Toronto</td>
<td>Canada</td>
<td>43.67722</td>
<td>-79.6306</td>
</tr>
</tbody>
</table>
Bibliography


