APPROACHES TO SOLVING THE EXPRESS SHIPMENT SERVICE NETWORK DESIGN PROBLEM

A Thesis
Presented to
The Academic Faculty

by

Leah Josefine Ruckle

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Aerospace Engineering

Georgia Institute of Technology
August 2018

Copyright © 2018 by Leah Josefine Ruckle

APPROACHES TO SOLVING THE EXPRESS SHIPMENT SERVICE NETWORK DESIGN PROBLEM

Approved by:

Prof. Dimitri Mavris, Advisor
School of Aerospace Engineering
Georgia Institute of Technology

Prof. Brian German
School of Aerospace Engineering
Georgia Institute of Technology

Prof. Alejandro Toriello
School of Industrial & Systems Engineering
Georgia Institute of Technology

Prof. Graeme Kennedy
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Elena Garcia
School of Aerospace Engineering
Georgia Institute of Technology

Date Approved: July 9, 2018

To Rohini Ralby, my Guru.

ACKNOWLEDGEMENTS

Thank you to all the people who have helped me on this journey and made this dissertation possible. First and foremost, thank you to my advisor, Prof. Dimitri Mavris who believed in me before I even set foot on campus. Thank you for all of the things that you have taught, shown, and done for me. A special thank you as well to the members of my doctoral committee, Professors Brian German, Alejandro Toriello and Graeme Kennedy, and Dr. Elena Garcia. I am grateful for your insightful questions, feedback and guidance which undoubtedly made this dissertation richer.

Thank you as well to my wonderful boyfriend, Matt Daskilewicz. Finishing this dissertation these past few months would not have been possible without your selfless support. Thank you for being my sounding board, cooking me dinner every night when I worked late, giving me ample hugs, and most of all, being my best friend. I love you, Matthew. I look forward to our future adventures together.

Thank you as well to my Guru, Rohini Ralby. You have been with me every step of the way since before I can remember. I so greatly appreciate your steadfast guidance and Love. Thank you so much for your support and daily phone calls these past several months, and all that I have learned from you over the years. It is no exaggeration to say that I would not be where I am today without you.

Thank you as well to my parents, Melanie and Keith for encouraging my curiosity about the world and my interest in building things, no matter how many legos you stepped on or pried out of the vacuum cleaner.

Finally, thank you to all my friends who have supported and cheered for me along the way, including (but not limited to) Adam Cox, Mallory Rose, David Pate, Xiaofan Fei, Kelly Collett, Erica Feldscher, David Soud, Ian Ralby, and Aaron Ralby.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Express shipment package delivery is a multi-billion-dollar industry specializing in the rapid transportation of parcels, documents and freight items between customers around the world. It is challenged and characterized by extremely tight service guarantees, the huge volume of material that must be moved daily, and a large geographical area of service. Despite the scope of these companies and the challenges they face, relatively little work has focused on developing computational methods to study and improve their airline operations. Therefore, the primary objective of this dissertation is to address these challenges and develop an algorithm that quickly and efficiently routes one day's worth of domestic express shipment packages and schedules the best possible vehicles to move them.

To this end, the express shipment problem is modeled as a mixed integer program called the Express Shipment Service Network Design problem (ESSND), and three different solution methods, a monolithic, an exact decomposition, and a heuristic decomposition, are considered. The fundamental hypothesis of this work is that of these three solution methods, either a heuristic decomposition or exact decomposition approach will outperform the baseline monolithic approach.

The monolithic approach is implemented using a state-of-the-art branch-and-cut-based optimization software (Gurobi 7.5) and serves as the baseline solution method. Solving the problem instances with the full set of options is found to be intractable so a heuristic is used to remove options that are unlikely to be present in the optimal solution. With this modest problem-size reduction, the monolithic approach is capable of producing good solutions for the baseline, single-day problem instance encompassing same-day, overnight and two-day domestic demand. However, the runtime to generate high-quality solutions is slower than desired, and the optimum is elusive, even after two days of runtime.

Benders decomposition, an exact decomposition approach is also implemented. Benders decomposition is chosen because it is designed to handle mixed integer programs containing "complicating variables," which are variables whose omission would make the problem markedly easier to solve. Benders decomposition reformulates the original problem into a master problem containing the complicating variables and one or more subproblems containing the other variables. It then attempts to exploit the fact that in an optimal solution only a subset of the problem constraints will be active. It iteratively generates constraints in the form of Benders "cuts" by solving the subproblems and adding those cutting planes to the master problem.

Since its original development in 1962, a number of modifications and improvements to Benders decomposition have been developed. Several of those modifications from the literature are implemented in this work. First, a "single search tree" Benders is implemented, which capitalizes on advancements in modern branch-and-bound solvers to avoid repeatedly rebuilding and resolving the branch-and-bound tree, thus potentially avoiding significant rework. Second, valid inequalities based on the cut-set inequalities used by Kim (1997)[129] are added to the master problem to help shape the otherwise poor master problem relaxation. Finally, two improved Benders cuts are implemented. The first of which is Pareto-optimal cuts generated using the modifications by Papadakos (2008). The second of which is the minimal infeasible subsystem (MIS) cuts developed by Fischetti et al. (2010)[82].

Unfortunately, despite these modifications, all implementations of Benders decomposition, including a commercially-developed one, were found to be inferior to the monolithic approach. All Benders decomposition implementations are tested on two problem instances, a small ten-airport case and the full-sized baseline case. All

Benders variants were capable of generating optimal solutions for the small problem instance, but for all except the commercially-developed implementation, generating these solutions took considerably longer than the monolithic solution approach. For the full-sized baseline case, none of the Benders decomposition implementations are capable of finding feasible solutions before surpassing the available computational resources. Benders decomposition, particularly for problems like the ESSND, suffers from the handicap that feasible solutions will not be found until the optimal solution is found. As a result, the algorithm cannot be terminated before convergence and produce a feasible solution, unlike the monolithic approach.

Finally, a heuristic decomposition approach is developed and implemented. This heuristic capitalizes on the observation that the monolithic approach, despite having only modest performance for full-sized problem instances, quickly solves smaller problem instances. Therefore the ESSND problem is decomposed into a Package Routing Problem (PRP) and an Aircraft Movement Problem (AMP), which can be solved to optimality in minutes even for the full-scale baseline problem instance. Since these two problems are coupled, a fixed point iteration approach is initially explored but is found to be underwhelming.

Instead, a sequential approach is found to be successful. This sequential approach first solves the Package Routing Problem that includes integral aircraft movement variables. The fundamental idea is that better package routing solutions are those that more efficiently load the aircraft, therefore aircraft considerations should be included in the Package Routing Problem. The basic implementation of this sequential heuristic only includes the feeder limit constraint and aircraft variable integrality constraint as additions to the Package Routing Problem (which is re-named the Package Routing with Aircraft Problem). This sequential heuristic, which is called the *Basic Sequential* Algorithm, is found to produce high-quality solutions more quickly than the monolithic approach for the first hour of runtime.

Two potential improvements to the Basic Full-Set Sequential algorithm are explored. First, the *Collapsed Product Set Sequential* Algorithm is developed. The Collapsed Product Set Sequential Algorithm exploits the fact that many commodities in the express shipment problem share origin-destination pairs and only differ in their product type or origination day. Using the concept of "analogous commodities," a smaller problem encompassing only the overnight box products, is solved and the routings chosen for the priority box products are used as a template to assign the routings of all analogous commodities. This algorithm is also capable of producing good-quality solutions more quickly than the monolithic approach for the first hour of runtime, but its performance is generally worse than the Basic Sequential Algorithm. However, the advantage of the Collapsed Product Set Sequential Algorithm is that it is relatively insensitive to problem size, and it is capable of solving the full-week problem, a previously intractable problem that is more valuable to express shipment companies than the single-day problem, in only a few hours.

The second improvement to the Basic Sequential Algorithm is the *Improved Sequential* algorithm. This algorithm improves the solutions produced by the Package Routing with Aircraft Problem by including a phase-based hull count constraint. This addition significantly improves the quality of the solutions found by the sequential algorithm. In an hour, the Improved Sequential Algorithm finds solutions of the same quality as the monolithic approach found after *two days* of runtime. This definitively supports the fundamental hypothesis that a heuristic approach (or exact decomposition approach) would be able to find high-quality solutions more quickly than a monolithic. In sensitivity studies, the Improved Sequential Algorithm consistently out-performed the baseline monolithic approach, showing that it performs well in a variety of problem instances, and it was capable of finding good-quality solutions to problem instances with a longer temporal scope, including a week-long problem instance.

The Improved Sequential Algorithm is the primary contribution of this work. It is substantially faster than the monolithic approach, the baseline, and for the baseline problem instance, it found a better solution in one hour than the monolithic approach found in two-days. The Template-Based Sequential algorithm is also a contribution, particularly in its insensitivity to problem size. It is capable of generating good-quality solutions for previously intractable problem instances. Finally, although Benders decomposition was found to be inferior to the baseline monolithic approach for the express shipment problem, several enhancements from the literature were implemented and tested, resulting in some important lessons for future researchers looking to follow a similar approach. Together, the heuristic algorithms are significant improvements over the baseline approach and open the door to more informed strategic and tactical-level planning for express shipment companies, potentially having implications ranging in the millions of dollars.

CHAPTER I

INTRODUCTION AND MOTIVATION

Express shipment package delivery is a multi-billion dollar industry in the United States and abroad. It has become an essential part of business operations and, with the rise of e-commerce, an expectation of the everyday consumer. Express shipment package delivery is challenged by extremely tight service guarantees, the huge volume of material that must be moved in a multimodal network, and the large geographical area of service. These challenges necessitate the use of large jet aircraft fleets, thus prompting the desire of package delivery companies to use these aircraft as efficiently as possible.

Every day, express shipment package delivery companies pick up millions of parcels, documents, and freight items from customers around the world and transport them using extensive vehicle fleets through an organized network of intermediate warehouses and sorting facilities to their final destination within a couple days. This extensive transportation network can be broken into two categories, 1) last mile service, and 2) line-haul service. *Last mile service* encompasses the parts of the service that directly interact with the customer, the initial pick up and the final delivery legs. After being picked up, express packages[1] are consolidated at a local warehouse and moved via truck to a nearby airport, where it enters the line-haul portion of the service network. *Line haul service* is the movement of similar packages en masse from a spoke airport to a sorting facility located at a network hub where they are reorganized according to their destination, and then finally flown to that destination airport.

[1]For brevity, the term *package* will refer to anything carried by an express shipment company, whether it be parcels, documents or freight items.

This line haul service portion constitutes the fundamental aspect of express shipment operations which sets it apart from other similar logistics problems such as less-than-truckload (LTL), railroad and oceanic shipping. Therefore, this work will concentrate solely on the line haul service portion of express shipment operations and call this problem the *express shipment problem*. To formalize the definition for this purposes of this dissertation,

The ***express shipment problem*** *is the problem of determining the lowest cost way to use a heterogeneous vehicle fleet composed of aircraft and trucks to move a set of express shipment packages from their origin airports to their destination airports within a domestic multi-hub-and-spoke network and within the guaranteed service periods.*

Even without considering the last mile service, the express shipment problem is a large, complex and interconnected problem that has major real-world ramifications for express shipment companies ranging the hundreds of millions of dollars annually. Following the classification scheme for logistical problems outlined by Crainic and Laporte (1997)[61], the express shipment problem as it is considered in this work is a *tactical*-level problem with some operational-level considerations. Since even relatively small changes to daily operations can result in significant cost savings for these companies, ideally a solution method to the express shipment problem would not only cover tactical needs but also enable the exploration of strategic-level questions. For an express shipment carrier, those questions may include, “Should the company invest in more aircraft, and if so, what kind?”, or “Which sorting facilities should be upgraded or re-timed?” or “Should more sorting facilities be opened, and if so, where?”

In order to enable the study of these kinds of questions, a solution method would need to be fast, realistic and produce good solutions in order to allow many possible scenarios to be considered without requiring excessing resources or time. Currently, this problem is primarily solved by experienced human planners with the aid of computational analysis tools, but this approach does not allow for the rapid generation of high-quality solutions that could allow for detailed strategic planning. A computer-based method is better-suited to match those needs.

In order to leverage the advantages of computer-based optimization techniques, the express shipment problem is modeled as an extension of the service network design problem (SNDP) called the *Express Shipment Service Network Design problem (ESSND)*. This problem has been tackled a few times previously in the literature and goes by a number of names, though each problem formulation is an extension of the SNDP. Some names from the notable works are the Express Service Design problem (ESD) [24], Express Package Service Network Design problem (EPSND) [132] and Express Shipment Delivery Problem (ESDP) [201]. The name Express Shipment Service Network Design problem (ESSND) is the name given to the problem in the work by Armacost et al. (2000)[12] and is the name used in this work, in large part because it best reflects the problem lineage.

However, even with the enhanced speed and power of a digital computer compared to human-based methods, solving the ESSND problem to optimality in its original form is extremely challenging due to the large number of variables and their combinatorial nature. Adding to the difficulty, the variables corresponding the movement of aircraft must be integer-valued in order to represent feasible solutions. This precludes the possibility of using the more efficient continuous variable optimization methods. In the face of these challenges and in light of the large potential benefits to express shipment companies, the primary goal of this work is to find a quick and effective method for solving the express shipment problem. More specifically,

*The **primary objective** of this work is to develop a quick and effective algorithm that generates low-cost vehicle schedules to completely route one days worth of domestic express shipment packages.*

The Express Shipment Service Network Design problem, as it is formulated in this work, is a mixed integer program (MIP). There a number of solution methods that have been developed to solve MIPs and they can be broadly categorized into three groups, 1) monolithic approaches, 2) exact decomposition approaches, and 3) heuristic approaches. The monolithic approaches do not require a reformulation of the original problem and, if run to completion, produce the optimal solution. The most prominent algorithms are branch-and-bound, cutting plane algorithms, and the combination of the two, which is called branch-and-cut. The exact decomposition approaches also produce a provably optimal solution if run to completion but solve only a subset or a reformulation of the original problem at any given time. Well-known approaches include Dantzig-Wolfe decomposition and Benders decomposition. These approaches rely on the fact that in the optimal solution, typically only a small number of the variables will be non-zero and only a small number of the constraints will be active. Finally, heuristic approaches are problem-specific solution methods that give up the pursuit of the provable optimum in exchange for the ability to produce a good-quality feasible solution in a short amount of time. The methods that heuristic approaches employ to find or improve solutions are typically tailored specifically to the problem at hand, and in some applications, they may be strategically guided through the use of a metaheuristic.

This work studies the effectiveness of a monolithic, an exact decomposition and a heuristic approach for solving the Express Shipment Service Network Design problem in an effort to fulfill the primary research objective. Of the three approaches, I hypothesize that a heuristic approach will be capable of producing high-quality solutions more quickly than either of the exact approaches. This is primarily because due to important characteristics in the ESSND problem itself and how the two exact solution methods function.

For the monolithic approach, the poorness of the ESSND linear relaxation hamstrings the ability of the underlying branch-and-bound tree to prune options, and as a result, the algorithm devolves into something more akin to an inefficient explicit enumeration approach. For the Benders decomposition, a poor ESSND master problem relaxation means that a very large number of iterations will be needed in order to find a feasible and optimal solution. My hypothesis is that a heuristic approach will be able to sidestep the pitfalls of these exact methods by exploiting problem-specific features rather than falling prey to them. To test this hypothesis, all three methods are implemented and compared.

The monolithic approach was developed as part of a collaboration, of which I was a member, between FedEx Services and Georgia Tech. A state-of-the-art optimization software, Gurobi 7.5, was used as the branch-and-cut implementation. The single-day baseline problem instance was found to be intractable if the full option set was considered so heuristics were developed to remove options unlikely to be present in the optimal solution. This monolithic algorithm was taken as the baseline approach against which the exact decomposition and heuristic approaches were compared. These other two approaches were developed entirely by the author.

The first of the two approaches was a heuristic approach. Of the three approaches, developing a heuristic approach requires the most creativity since it is tailored to the problem at hand, but this additional design effort can pay dividends in runtime, particularly if finding the exact optimum is not a priority. Since no standard heuristic algorithm exists for the express shipment problem, one was developed through a series of hypotheses and experiments with influence from the *Multidisciplinary Design Optimization (MDO)* field. The cornerstone of this heuristic approach is the fact that the monolithic approach can efficiently solve smaller problems or relaxed problems, but struggles as the number of options increases. Therefore, the heuristic algorithm takes a decomposition-based approach in which subsets of the full problem are solved and then frozen as other parts of the problem are then solved. Initially, an iterative, fixed point iteration-based approach was posed and tested, but ultimately, a sequential approach was found to be more successful.

Finally, as for the exact decomposition approach, Benders decomposition was chosen due to the presence of *complicating variables*[2] in the Express Shipment Service Network Design problem, and its recent popularity and success. Several improvements from the literature were included in the Benders decomposition implementation in this work. Specifically, the algorithm was implemented as a "single search tree" Benders decomposition, valid inequalities were added to the master problem, and improved cut-generation methods were used. The same method used in the monolithic approach to generate the reduced option space was used to generate the options for the Benders decomposition. Otherwise, the Benders decomposition implementation was completely developed and implemented by the author.

After implementing and testing these three solution methods, a heuristic approach was found to be the most successful. Benders decomposition, while successful on small problem instances, was incapable of solving full-sized problem instances encompassing a single day's worth of domestic demand. In regards to the heuristic approach, several algorithms were developed, but the most successful was the *Improved Sequential Algorithm*. That algorithm is capable of producing better solutions to the express shipment problem in one hour than the monolithic approach is capable of producing in two days of runtime for the baseline single-day problem instance. The different heuristic decomposition algorithm, called the *Collapsed Product Set Sequential Algorithm*, is less sensitive to problem size than the other approaches explored in this work, and is capable of producing good-quality solutions to problems encompassing a full week's worth of domestic demand, a more useful problem for express shipment

[2]Complicating variables can be defined simply as those variables whose exclusion would make the problem significantly easier to solve. In mixed integer programs, the integer variables are considered "complicating."

companies, in fewer than sixteen hours.

The primary contribution of this work is the Improved Sequential Algorithm. The ability of this heuristic algorithms to produce high-quality solutions quickly will enable express shipment companies to conduct detailed strategic and tactical-level studies that could have implications in the range of millions of dollars and beyond. Additionally, the ability to provide good-quality feasible solutions for the previously intractable full-week problem opens the door to analyses encompassing more useful time frames for these companies. The results of the Benders decomposition are also a contribution, since many important lessons were learned in the process of implementing it. The results of the Benders decomposition implementation provide guidance for any future forays into its implementation for the Express Shipment Service Network Design problem.

This has been a brief introduction to the primary challenges of the express shipment problem, the methods that were implemented and studied to solve this problem and the contributions of this work. The remainder of this dissertation provides more detail and is organized as follows,

- Chapter 2 fully characterizes the express shipment problem and describes, in detail, the real-world data set that will be used in this work.

- Chapter 3 reviews how problems like the express shipment problem are typically modeled in the literature, and presents the Express Shipment Service Network Design problem formulation.

- Chapter 4 provides an introduction to solution methods for mixed integer programming methods like the ESSND problem, and summarizes the established literature concerning previous work on the ESSND problem and similar logistics problems.

- Chapter 5 details the research questions, hypotheses, and experiments of this work.

- Chapter 6 presents the monolithic approach and results.
- Chapter 7 presents the development of the heuristic decomposition algorithm and results.
- Chapter 8 presents the Benders decomposition implementation and results
- Chapter 9 summarizes the conclusions and contributions of this work and offers some suggestions for future work.

CHAPTER II

PROBLEM CHARACTERIZATION

Since the first Federal Express flight in 1973, the express shipment industry has grown into a multi-billion dollar industry in the United States and abroad. Today, overnight and two-day package delivery is a critical part of business operations, and an expectation of the everyday consumer. According to recent reports, the express shipment industry is expected to continue to thrive and experience moderate growth thanks in large part to e-commerce and globalization.[58, 124]

Over time, smaller, regional shippers have consolidated into larger companies, and currently, the US package delivery market is dominated FedEx and UPS (United Parcel Service). In the 2016 fiscal year, FedEx reported \$50.4 billion in annual revenue [79], and UPS reported \$58.4 billion in 2015 [186]. Outside of the United States, Deutsche Post DHL and TNT (acquired by FedEx in 2016) are also important companies, particularly in Europe. Within the United States, the United States Postal Service (USPS) provides some competition in ground shipping, but its express products are all moved by FedEx as a result of a \$10.5 billion, seven-year agreement made in 2013 and extended through 2024 in 2017 [79, 80].

For many package delivery companies, express shipment is just one part of their business. These companies also offer other “delivery solutions” such as ground (parcel) shipping, less-than-truckload (LTL) freight shipping and supply-chain logistics. Within FedEx, each of these delivery solutions is handled by a different company under the FedEx corporate umbrella with FedEx Express handling all of the express (same-day, overnight and two-day products) shipments. UPS is organized differently and all delivery options are organized under the single UPS corporation. These additional delivery services share some similarities with the express shipment problem, but will not be considered in this work.

2.1 The Express Shipment Problem

Express shipment companies are large, intricate companies that work on a global scale. Considering every aspect of these companies all at once is a task that is insurmountable and, realistically, unnecessary. For the purposes of this dissertation, *express shipment problem* specifically refers to,

> *The problem of determining the lowest cost way to use a heterogeneous, multi-modal vehicle fleet to move a set of express shipment packages from their origin airports to their destination airports within a domestic multi-hub-and-spoke network and within the guaranteed service periods.*

Despite being only a part of the overall express shipment "picture," the line-haul portion is the *crux* of what makes express shipment different from other logistical problems. It tackles the issue of using a diverse fleet of, primarily, expensive jet aircraft to move a very large number of packages in a short period of time subject to network limitations. Therefore, only a part of the express shipment problem, namely the domestic line-haul portion of operations, will be considered in this work. According to the classification scheme laid out by Crainic and Laporte, the express shipment problem is a *tactical*-level problem (though it contains some operational-level elements as well).[61] For realistic problem instances, the express shipment problem is extremely large and difficult to solve well. More details on this problem definition, its scope, and its challenges will be presented in the following sections of this chapter.

2.2 Scope of Express Shipment Carriers Today

Express shipment companies move a massive number of parcels and documents daily. Within the U.S. express shipment market, FedEx and UPS are nearly evenly matched.

During similar time frames, FedEx Express and UPS moved nearly identical average domestic package volumes. In FY2017 FedEx reported moving an average of 2.726 million domestic packages daily and in FY2016 UPS reported moving an average of 2.730 million daily packages domestically[1].[79, 186] Both companies deliver to essentially every point around the globe, with service to over 220 countries and territories. This translates to massive and complex networks of packages, aircraft, ground crews, pilots, airports, truck drivers and more that must work together seamlessly in order to consistently satisfy the level of service guarantees.

Each of these companies offers a wide variety of shipment services including freight, ground and express transportation. Freight shipment focuses on Less-Than-Truckload (LTL) shipping in addition to ship and rail transport. Ground shipping moves packages and documents via trucks and other low-cost transportation methods. Express shipping moves packages and documents within extremely tight time constraints, typically via aircraft. Express service, while expensive to operate, also generates a large amount of revenue. FedEx reported that FedEx Express generated \$27.4 billion in FY2017, while FedEx Ground and FedEx Freight generated \$18.1 billion and \$6.4 billion, respectively [79]. UPS does not differentiate between express service and ground service revenues in their reports but reported \$50.7 billion for package operations (express and ground service) compared to \$10.2 billion in revenue for supply chain and freight service [186].

Express package delivery is focused on speed and provides same-day, next-day, and two-day service. These service guarantees are extremely challenging, especially considering the large volumes that must be picked up, sorted and delivered. These tight service guarantees necessitate the use of jet aircraft, even for domestic service. Some areas are close enough to sorting facilities to allow for package transportation

[1]FedEx's fiscal year ends on May 31 of the given year, whereas UPS's fiscal year ends on December 31 of the given year.

via truck, but the bulk of the volume must be flown. The Airline Deregulation Act of 1978 allowed for package delivery companies to own and operate their own cargo airlines, thus enabling them to become air integrators. According to public SEC (U.S. Securities and Exchange Commission) filings, FedEx Express currently operates a fleet of 657 jet and feeder[2] aircraft (598 are FedEx-owned, 59 are leased), and UPS operates a fleet of 657 jet and feeder aircraft (237 are UPS-owned, 420 are leased or chartered) [79, 186]. Overall, in order to meet the large demand for overnight and two-day delivery service, express shipment companies operate vast and complex multi-modal networks involving hundreds of jet and feeder aircraft, thousands of trucks and several hundred thousand employees.

The massive amount of volume that must be moved, the extremely tight service guarantees and the operation of a large jet aircraft fleet results in a logistical problem that is large, complex and critical with a large financial impact. Given the scope and complexity of these networks and operations, improvements to the operations of air integrators could save millions of dollars annually.

2.2.1 Problem Scope

Geographic Scope This dissertation will focus specifically on the express shipment problem within the continental United States. Domestic express shipment operations within the United States provide an approachable, though by no means easy, problem instance that encompasses the same critical questions and essential challenges encountered when considering the global problem. The size of the United States, even when excluding Alaska, Hawai'i and the territories, necessitates the use of large airline fleets, captures some of the complexities introduced by time zones, and the extensive American use of express shipment results in a large amount of daily demand.

[2]Feeder aircraft are small propeller-driven aircraft.

Additionally, considering a fully domestic problem removes distractions such as customs inspections, large time zone challenges, regulatory differences and aircraft range limitations which make the problem more difficult, but do not encompass the heart of the problem which is to route packages and schedule the best possible vehicles to move them. In essence, the continental United States defines a "Goldilocks" problem instance for the express shipment problem in which it is large and challenging enough to be interesting and worthy of study without adding complications that distract from the overall purpose of the work.

Temporal Scope For the express shipment problem, the most appropriate period of time to schedule is a week. The relatively small amount of weekend volume provides a better opportunity to reposition aircraft, if necessary, and, because of the two extra days afforded by the weekend, much more late-week volume can truck rather than fly. The weekend creates a natural break at the end of the time period. By contrast, if only one day's worth of operations is considered, the overnight and same-day volume can be handled without an issue, but the two-day volume is problematic because it "rolls over" to the next day and still requires aircraft in order to arrive on time. The issue of handling roll-over volume still exists when considering a week, but it is much more limited in scope. All that being said, the majority of the work in this dissertation will focus on solving a single day's worth of operations because solving problem instances that encompass a week using exact methods is intractable, as will be shown in Chapter 6. Later, in Chapter 7, the concept of solving a one-week problem instance will be revisited and the MDAO-inspired heuristic approach will be shown to be effective on that more useful and previously intractable problem instance.

2.3 Movement of Packages Through the Express Shipment Network

With few exceptions, all packages in the custody of an express shipment company follow the same functional path from their origin to their destination as shown in Figure 1. Each stop along the path is called an activity. A package first enters the

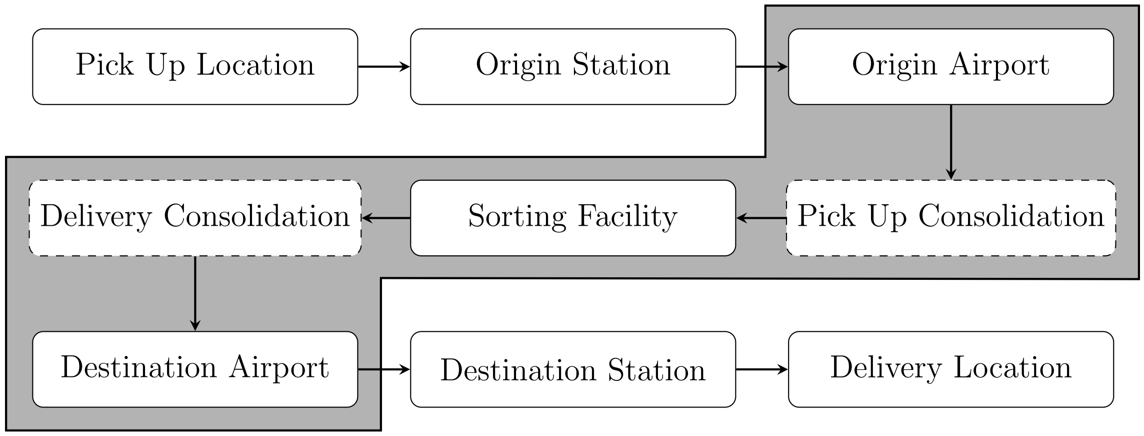


Figure 1: Package Activities from Origin to Destination

custody of the package delivery company at a pick up location such as a business, private residence or drop box. The package is then transported via small ground vehicle to an origin station[3]. At the origin station, local packages are sorted by service type (express, ground, etc.). The express packages are then transported via truck to a designated nearby airport. This airport, called a *gateway*, is the collection point for all of the stations in a given region. The packages are then loaded onto aircraft and flown to a large sorting facility, which also serves as the network hub for the airline operations. If the origin gateway does not produce a large amount of volume, it may be *consolidated* with a nearby larger market via truck, feeder or two-legged flight. Truck and feeder consolidation operates on a point-to-point basis between the two spoke airports. A two-legged flight is when a trunk aircraft makes

[3]This terminology is not strictly universal. Origin stations may also be called "ground centers" depending on the company.

an intermediate stop en route from a spoke to a hub in order to pick up more volume. Once at the sorting facility, all of the aircraft are unloaded, and packages bound for the same destination are grouped together. After the sorting is completed, packages are loaded onto aircraft and the process essentially reverses itself. Aircraft deliver the packages to the destination gateway. From the destination gateway, the packages are moved via truck to a destination station where they are loaded onto a small ground vehicle for final delivery.

2.3.1 Last Mile Service and Line Haul Service

The sequence of activities followed by a package can be divided into two categories, 1) last mile delivery, and 2) line-haul service. Last mile service is the strictly ground-based service consisting of the transportation from the origin location to the origin ground center and the origin airport, and the transportation from the destination airport to the destination ground center and final delivery location. The air-based service that encompasses the transport from the origin airport to sorting facility and on to destination airport is called "line haul" service. Although "line haul" is a term that commonly refers to less-than-truckload service between terminals, in this context, the volume is transported via aircraft. In Figure 1, the line haul portion is highlighted in gray, and the last mile delivery parts are on either side.

This work focuses specifically on the line-haul portion of express shipment operations. Last mile service is a challenging logistical problem in its own right with strong ties to the vehicle routing problem (VRP). The vehicle routing problem and its permutations are well-studied, but it is still an active area of research. As a close relative of the traveling salesman problem, it is quite difficult to generate optimal solutions for the VRP in a reasonable time frame. Fortunately, the last mile service and line haul service problems can be solved in isolation from each other through the use of time-based constraints. Given the difficulty of each problem even in isolation, solving the two separately is, by far, the most common approach both in academia and in industry.

The feasibility of the line haul problem can be guaranteed through the use of time-based constraints. Since last mile service does not share equipment with line-haul service, time is the is the only resource that must be linked between the two services. The two problems can be solved independently by enforcing “due times.” A last mile service problem solution can be joined with an air transportation problem solution by making sure that the packages from the origin ground center are delivered to the origin airport no later than a due time that is known to allow enough travel time to the sort, and likewise, packages must be delivered to a destination airport no later than the pre-set due time. Some absolute optimality may be sacrificed through this approach, but any gained improvement would almost certainly not be worth the enormous problem size increase.

When speaking of the “express shipment problem,” and the “Express Shipment Service Network Design problem,” only the line-haul service portion of the overall process is considered. Last mile service is a separate problem and last mile operations are considered fixed when solving the express shipment delivery problem.

2.3.2 Activities

Each package stop, represented by a block in Figure 1, is called an *activity*. An activity is an event at a particular location in which a something needs to be done with the package. Activities can be thought as the points along a routing when a package needs to be touched while in the custody of the express shipment company. For line-haul, there are four *types* of activities: origin, destination, consolidation, and sort. The activity type, coupled with a location, constitutes an activity.

At an airport, often an activity will occur multiple times per day. For example, packages may originate at a spoke airport once in the evening and once in the morning.

Both of these events would be an origin activity but pertain to separate *activity instances*. An activity instance is a unique combination of an activity and the time at which that activity occurs.

2.3.3 Important Activity Times, Cycles and Phases

In order to ensure that each activity has enough time to process all of its packages, each activity has several important times associated with it as shown in Figure 2. The most important times are the due time and the available time. The due time is the latest time that a package can be delivered to an activity and still have enough time to be processed. The due time is critical to ensure that everything occurs on time. Any delays in the system can have a cascading effect and affect the entire process. The available time is the time that a package is available to leave that activity and be transported to the next activity. Thus, the time elapsed between the due time and the available time is the minimum processing time. The open time is the time when an activity will start accepting packages.

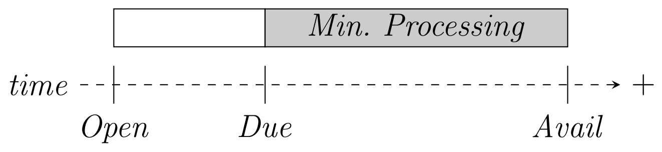


Figure 2: Activity Times

These times create time windows which can be very restrictive. Origin available times are late in the day, usually after business hours, in order to allow customers the most time to drop off their packages. For next day service, this means that all operations, including the sort, must occur overnight in order to deliver the packages the following morning. Two-day packages, which have more relaxed time constraints, are transported by truck if there is time to do so. Otherwise, the two-day volume can be loaded on the nighttime aircraft if there is excess capacity, but in general two-day volume is transported by aircraft during the day.

Same-day packages, of course, must fly during the day. For the purposes of this work, same-day volume is volume that must be moved as a result of a contract with a third party. Many express shipment companies offer same-day shipping of their own products, at least locally, but this volume is typically handled differently from the overnight and two-day volume. Same-day volume as the result of a contract is volume that is delivered in bulk to the air integrator in the morning and is delivered back to that customer (at a different location) in the evening. The customer handles the last mile service.

The movement of some products during the day and some products overnight results in periods of the day called *cycles*. These cycles are oriented according to sorting times at the hubs. There are night operations which move the next-day volume through a nighttime sort, and there are day operations which move the same-day and two-day volume through a daytime sort. Of the two cycles, nighttime operations are more difficult to satisfy because there is a shorter amount of time in which to move all of the volume. This results in the counterintuitive fact that overnight packages (i.e. one-day) have less time to move than same-day (i.e. zero-day) packages. Cycles can be further broken into *phases* based on whether an activity occurs before the sorting activity instance (i.e. a pick up phase) or after the sorting activity instance (i.e. a delivery phase).

2.4 Packages Aggregated as Commodity Flows

Due to the magnitude of the package volume, modeling package flows on an individual parcel-level is an unnecessary level of detail. Instead, similar packages can be aggregated as commodities, and then a decision can be made on how to route and transport each commodity. This approach assumes that the packages are numerous enough that the commodities can be assumed to be continuous. In general, this is a safe assumption.

A commodity is defined as the unique combination of origin, destination, product and origin day. The origin and destination are the airports at which the commodity enters and leaves the air network. The product type describes the level of service (same-day, next-day or two-day) and the sort classification (box, document or freight). The origin day is the day that the package enters the network. Although this aggregation of packages into commodity flows simplifies the problem and transforms it from hopelessly intractable to possibly solvable, the number of commodities is still large, even for a single day of operations.

The demand of each commodity is represented in terms of its spatial volume (measured in cubic feet or “cube”). Spatial volume is more representative of the real-world constraints because express shipment aircraft tend to “cube out” before they “weigh out” since parcels tend to have a low weight density. For the purposes of this work, commodity demand is assumed to be deterministic, accurate and predefined. Robust scheduling would add an additional layer of complexity on an already challenging problem, and demand forecasting is, like last mile service, a field in and of itself.

2.5 Air Fleet Considerations

In order to satisfy their level of service guarantees, express shipment companies operate large, heterogeneous aircraft fleets. These fleets encompass aircraft of a range of sizes from wide-body jets to propeller-driven feeder aircraft.

Although air transportation is faster than ground transportation, it is much more expensive. Aircraft are more expensive to own, operate and maintain compared to any other long-distance transportation mode. Additionally, they require a larger and more expensive workforce per vehicle, a specialized infrastructure and incur a large purchasing cost. As a consequence, reducing the total number of flight hours can have an enormous impact on the total transportation cost. Reducing the total number of flight hours by even a small amount for every operating day can result in millions of dollars in annual savings.

In addition to using their own aircraft, air integrators can purchase cargo space in the belly of passenger aircraft. This option, called “interline,” is even more expensive and is typically not extensively used by major air integrators in the United States which have extensive domestic airlines of their own. As a result, this option is usually reserved for international markets or last-minute contingency planning when confronted with larger than expected volume.

Air integrator fleets also tend to be very diverse as a reflection of the diverse markets they serve. This fleet diversity enables a more efficient use of aircraft because it allows the capacity to be more closely matched to the cargo volume. However, this fleet diversity also makes the problem more difficult to solve. If there were only a single aircraft type, then only multiples of that vehicle would need to be assigned to a potential flight. Instead, if there is more than one type, then a variable for each feasible aircraft type is necessary and values must be assigned to each of those variables. Ultimately, although reducing the fleet diversity would simplify the problem, it would also reduce the realism of the solution.

2.6 Air Network Topology

In order to minimize transportation costs, express shipment airlines operate in multi-hub-and-spoke networks. Hub-and-spoke networks generally offer the lowest cost network topology for covering a given geographic area, because it requires the fewest flights to connect every location to every other. Due to the large geographic areas and the physical limitations of the network hubs, air integrators operate multi-hub-and-spoke networks which employ several strategically placed hubs rather than a single hub.

Originally, air cargo networks were single hub-and-spoke networks, but as volume and geographical service area increased, more hubs were needed in order to handle the demand. Large carriers found that it was not possible to land all of the aircraft, sort the volume and have all the aircraft takeoff again within the necessary time frame when using a single sorting location. This has produced a hierarchic multi-hub-and-spoke network in which there is one primary hub and several secondary or tertiary regional hubs. For FedEx, the hub in Memphis (MEM) is the "Super Hub" with lower-level hubs in Indianapolis (IND), Fort Worth (AFW), Newark (EWR), Oakland (OAK) and Greensboro, NC (GSO). For UPS, the hub in Louisville is the "World Hub" with lower-level hubs in Philadelphia (PHL), Dallas (DFW), Ontario, CA (ONT), Rockford, IL (RFD) and Columbia, SC (CAE).[79, 186]

In order to ensure that every package that can enter the network can be delivered, every spoke airport is required to have "network connectivity" to the primary hub. This means that every origin and destination location must be able to route through that hub. This allows for last-minute contingency planning, particularly for larger-than-forecast volumes or regional weather problems. The secondary hubs do not connect to all other spokes, only a subset. Overall, the hierarchical hub-and-spoke network preserves some of the advantages of the pure hub-and-spoke network while diffusing some of the volume to more regional sorting hubs. The multi-hub topology does, however, make the problem more difficult to solve compared to a pure hub-and-spoke network because it has a multiplicative effect on the number of potential package routings, thus greatly increasing the problem size.

In this work, the sorting hub locations, as well as the spoke airports (gateways), are assumed to be fixed. Considering changes to the network topology is out of the scope of a tactical-level model and problem, and is typically handled with a hub network design problem or a facility location problem.

2.7 Data Set

Thanks to a collaboration between FedEx Services and a team of researchers, of which I am a member, at the Georgia Institute of Technology, I have access to a complete planning data set from FedEx. The data spans one week of forecast domestic demand in a recently passed year. Additionally, this dataset includes facility locations, activity times and instances, sort capacities, vehicle properties and cost information. This dataset will be used for all of the experiments and studies of this work. Although the time frame of this data set has, at this point, passed, some information, particularly commodity demand and cost, is still sensitive and cannot be shared in its full form. As appropriate, characteristics of the data set are described in this section so that the reader has a thorough understanding of the scope and details of the problem instance.

2.7.1 Facility Locations

As previously mentioned, this work focuses on the express shipment problem within the continental United States. That is reflected in the facility set, which is shown geographically in Figure 3. In total, there are 115 facilities and they are all located at airports. In reality, FedEx Express operates out of some additional "secondary" airports in small, remote areas which always consolidate to nearby markets. To simplify the problem slightly, this volume has been integrated into the consolidation market and those secondary airports are excluded from the problem.

In Figure 3, the hubs are highlighted for easy identification. In this data set, the FedEx hub at Greensboro, NC (GSO) is not considered part of the hub set because it was not a major hub during the time frame of the data set. The others are, as mentioned previously, MEM (primary), IND (national), EWR (regional), AFW (regional), and OAK (regional).

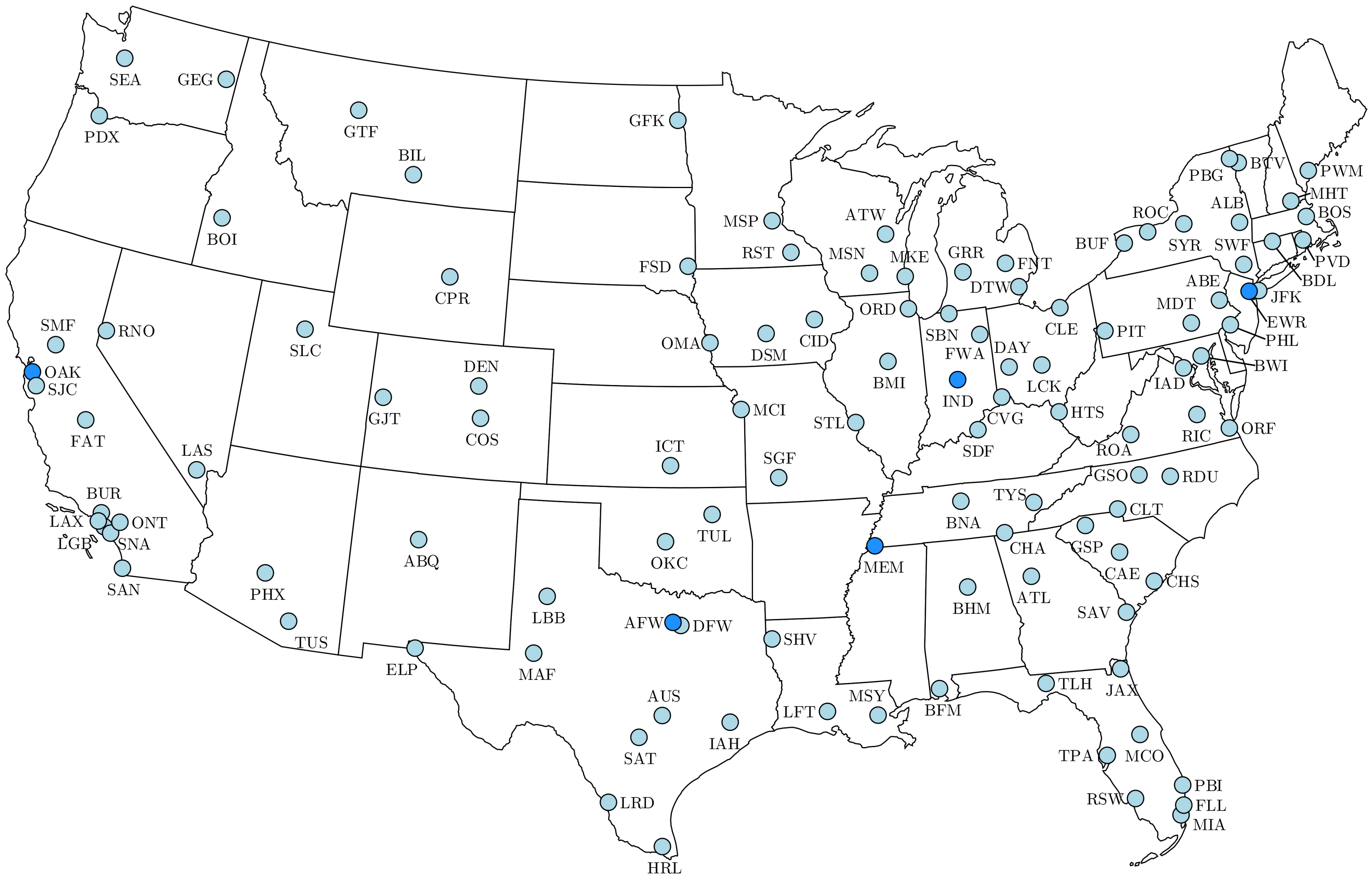


Figure 3: FedEx Air Facilities in the Data Set

2.7.2 Sort Capacity

Sorting facilities have limited capacity due to physical limitations on the number of pieces that can be correctly sorted in a given period despite high levels of automation and technological sophistication. The actual sort capacity values are proprietary but the capacities that FedEx makes public in their investor reports, see Table 1, are representative.[79]

Table 1: Public FedEx Sort Capacities [79]

Facility	Region	Cap. (pcs/hr)
MEM	Primary	475,000
IND	National	214,000
EWR	Northeast	156,000
AFW	Southwest	76,000
OAK	West Coast	63,000

The capacities shown in Table 1 are given in terms of packages *and* documents. In reality, these products are sorted on different machines and are therefore tracked separately as different product types. Additionally, each sort has a maximum amount of freight products (i.e. parcels over 150 lbs) that it can handle. This differentiation of products and their individual sort capacities is captured in this work, but sorting capacities for freight items are not made public by FedEx and cannot be shared.

2.7.3 Product Types and Commit Days

In addition to the sort-based product classifications described in Section 2.7.2 (box, document, and freight), products are classified by the level of service that the company has promised the customer. The main classifications are *priority*, *economy*, and *postal* (U.S. Postal Service) volume. For the most part, this translates to overnight, two-day and same-day service, respectively. Additionally, FedEx offers overnight and two-day products that deliver on Saturdays, and the USPS volume similarly has distinctions for weekend delivery. Finally, priority box volume is divided into two categories: 1) regular priority boxes, and 2) First Overnight boxes. First Overnight boxes are delivered earlier in the morning than the priority boxes. The product sort classifications and level of service categories combine to define unique product types, which are shown in Table 2.

Table 2: Product Types of the Data Set

Level of Service	Base Commit	Box	Document	Freight
Priority	Overnight	✓×2	✓	✓
Economy	Two-Day	✓	✓	✓
Priority Saturday	Overnight	✓	✓	✓
Economy Saturday	Two-Day	✓	✗	✓
USPS 0-Day	Same-Day	✓	✗	✗
USPS 1-Day	Overnight	✓	✗	✗
USPS Friday	Overnight	✓	✗	✗

In total, there are fifteen different product types in the dataset. Priority boxes are shown as having two types because the priority box products and First Overnight packages are handled similarly for the line-haul portion of their transport. In reality, FedEx Express carries and tracks even more products, such as those carrying live products or perishables, but those require special handling and the products in Table 2 constitute the vast majority of the volume.

The total number of calendar days that the company has promised to deliver the package to the final destination is called the number of *commit days*. Every level of service has a *base* number of commit days which corresponds to the number of *business* days the company has promised the customer. Commit days are always counted in terms of the number of "midnights" crossed from pick up to delivery. Early in the week, the total number of commit days will equal the base number of commit days, but later in the week, a product will have more commit days because it is delivered after the weekend. For example, an economy package mailed on Thursday will have 2 base commit days, but a total of 4 commit days because it does not need to be delivered until Monday. As a result, late-week products, particularly economy products, tend to have many more routing options than their early-week counterparts.

2.7.4 Commodity Demand

As mentioned in Section 2.4, a commodity is defined as a unique combination of the origin, destination, product type and origin day. Due to its sensitive nature, specific demand information cannot be shared. However, some characteristics of the commodity demands can be used to understand data in general.

First, Figure 4 shows the total originated demand for each day in the week of the data set. As would be expected, the volume is greatest on the weekday and

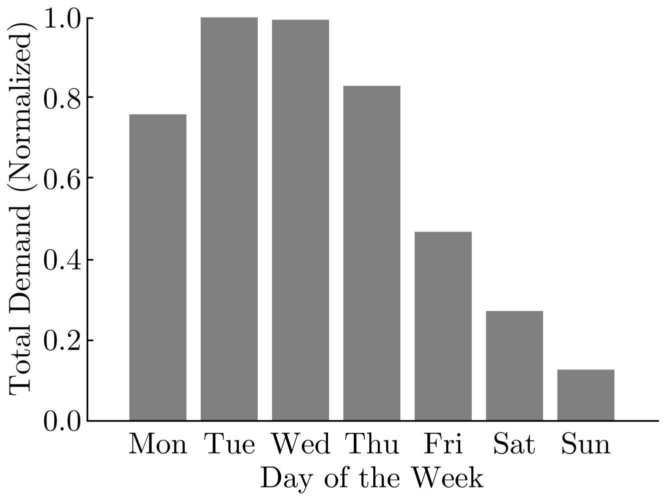


Figure 4: Total Originated Volume For Each Day of Data Set

significantly smaller on the weekends. As would be expected, the demand volume is lowest on the weekend and highest on Monday through Thursday, the busiest days for business. Monday's volume is small primarily because there is no postal volume (i.e. no mail pickup on Sundays). Friday's volume is the lowest weekday in large part due to the preprocessing performed on the data set (by FedEx) to remove any commodities that could be completely moved via truck from origin to destination. Friday products (and Thursday economy) products have two extra days to move, which makes many more markets available to truck their volume, and those commodities are therefore excluded from the dataset.

Another illuminating way to view the commodity demand is by how much volume is originated at in each market. Figure 5 shows the distribution of total originated volume on an airport-wise basis. The majority of markets (identified by airports) in

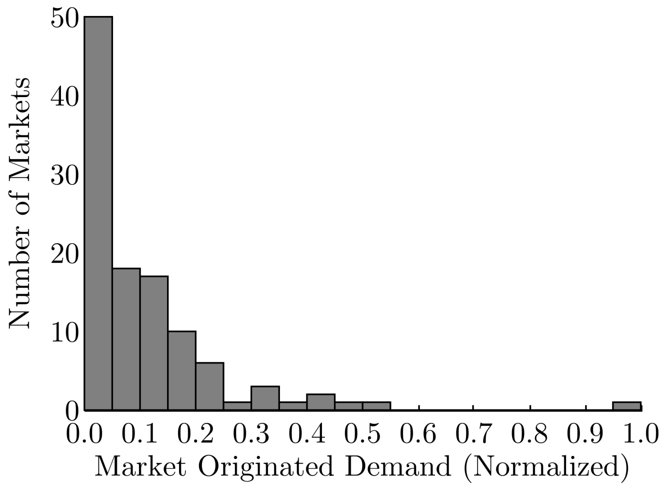


Figure 5: Market-Wise Distribution of Total Volume Originated on Tuesday

the data set are small compared to the largest market. Close to half (50 out of 115) of the airports originate five or less percent of the volume of the largest market. Overall, the shape of the distribution makes sense, particularly when considered alongside the map of facilities in Figure 3. Overall, FedEx flies out of a lot of airports, and these airports have been deliberately chosen to follow the population density of the country. The densely-populated regions of the country have many facilities, while even sparsely-populated regions like the Mountain West have some facilities. Additionally, major metropolitan areas, such as New York City, Los Angeles, and Dallas have multiple facilities. In total, the dataset demand features a large set of small markets, some medium-sized markets, a few large markets and one exceptionally large market.

2.7.5 Vehicle Fleet and Hull Count

FedEx Express operates a large, international fleet of aircraft and as a result, the precise size and composition of the fleet that is within the continental United States is fluid. Aircraft may fly in from overseas, make a few stops carrying domestic volume, and then fly back overseas. As a result, it is not possible to specify an *exact* fleet hull count and composition at any given time, but it is still meaningful to use a reasonable estimate. The hull counts used in this dataset can be found in Table 3. Trucks are assumed to be unlimited, and they can carry more volume than a feeder, but less than a jet aircraft.

Table 3: Data Set Fleet Composition

Name	Hull Count	Class	Max. Payload (lbs)
B777F	7	Super Wide	233,300
MD-11	43	Wide	192,600
MD-10-10	37	Wide	137,500
B767-300F	30	Wide	127,100
A-300	49	Wide	106,600
A-310	8	Wide	83,170
B757-200	80	Narrow	63,000
ATR-72	10	Feeder	17,970
ATR-42	10	Feeder	12,070
C-208	50	Feeder	2,830
Truck	∞	Truck	*unspecified*

As mentioned in Section 2.4, express shipment aircraft tend to “cube out” before they “weigh out” and therefore the spatial capacity of each vehicle type is more important than its weight capacity. However, the spatial capacities of these vehicles is an important planning consideration and the exact specifications cannot be detailed here. To provide a basis for comparison, the maximum gross structural payload for each vehicle is provided, as specified in the 2017 FedEx investor report.[79] When using the dataset, the spatial capacities are used, not the weight capacities.

2.7.6 Cost

In order to protect proprietary information, exact cost information cannot be shared, but the data can be characterized. First, only vehicle-based costs are considered. Costs related to the magnitude of the package flows are negligible compared to the base cost of flying an aircraft. Therefore, direct package costs are assumed to be zero. Second, there are two categories of aircraft-related cost, 1) cycle costs, and 2) flight hour costs. Cycle costs are those that are automatically incurred anytime the aircraft takes off. They can include landing fees, depreciation costs, ground crew wages, etc. Flight hour costs are those that scale with the length of the flight. They can include fuel cost, maintenance and repair costs, pilot salaries, etc. Truck costs only include a "flight" hour cost. The total cost of a flight is the sum of the cycle cost and the total flight hour cost. Third, costs scale with vehicle capacity although newer vehicles tend to be cheaper per unit capacity than older aircraft. Finally, trucks are, by far, the cheapest vehicle per unit capacity, and feeder aircraft are the cheapest aircraft type.

The dataset described in this section will be used extensively throughout this work to conduct experiments on the studied algorithmic approaches to solving the express shipment problem. The baseline problem instance for these studies will encompass one full day's worth of express shipment volume for the domestic problem instance described here. Additionally, experiments will be conducted on problem instances of larger scope, up to a week worth of demand volume, to study how different solution methods' performance scales with problem size.

2.8 Summary

Express shipment companies are large, global companies that operate large aircraft fleets and scores of facilities with the overall mission to deliver large numbers of packages across wide geographical distances in very little time. The problem of optimizing or at least improving their operations is a correspondingly large, challenging and complex. That problem, which here is referred to is as the "express shipment problem," can be summarized as,

> *The problem of determining the lowest cost way to use a heterogeneous, multi-modal vehicle fleet to move a set of express shipment packages from their origin airports to their destination airports within a domestic multi-hub-and-spoke network and within the guaranteed service periods.*

This dissertation will focus on solving the express shipment problem for a single day's worth of domestic demand using a data set provided by FedEx and, where meaningful, will extend that scope to a full week of demand. The next chapter, Chapter 3, briefly reviews how similar problems are captured mathematically and presents the optimization formulation for the Express Shipment Service Network Design problem.

CHAPTER III

EXPRESS SHIPMENT SERVICE NETWORK DESIGN: MODELING THE EXPRESS SHIPMENT PROBLEM

As shown in Chapter 2, the airline operations of an express shipment carrier are extensive, complex and highly interconnected. The scope, complexity, and difficulty of the problem preclude the possibility of an efficient and high-quality human-based optimization method. It is certainly possible for a team of human planners to generate feasible solutions for these extensive problems, but this kind of hand-made approach is highly time-intensive, typically built heavily on historical schedules and lacks a meaningful way of optimizing or improving the solution.

However, this human-based method is largely how planning is done today.[1] While it is not true that computers outperform humans in all venues, computers tend to outperform humans in arenas in which the option space is massive and the elements of the system are highly interconnected, such as in the express shipment problem.

In order to take advantage of the strengths of digital computing, the problem must first be captured in a model that is accessible to computers. This chapter presents a mixed integer programming formulation that captures the relevant aspects of the express shipment problem.

3.1 Modeling Logistics Problems as Mixed Integer Programs

The choice to model logistics problems as mixed integer programs has roots in the literature that stretch back to the early days of optimization. Many of the most famous logistics-related problems, such as the Traveling Salesman Problem (TSP),

[1]This is mentioned in Armacost (2000)[11] and corroborated by personal discussions with individuals in the industry.

Vehicle Routing Problem (VRP), Crew Scheduling Problem and Facility Location Problem are modeled as mixed integer programs (MIP).

A mixed integer program (MIP) is an optimization problem that can be expressed in the form,[2]

$$\max\{cx + hy \mid Ax + Gy \leq b, x \in \mathbb{R}^p_+, y \in \mathbb{Z}^n_+\}$$

where $\mathbb{R}^p_+$ is the set of nonnegative real p-dimensional vectors and $\mathbb{Z}^n_+$ is the set of nonnegative integral n-dimensional vectors, and $x = (x_1, \ldots, x_p)$ and $y = (y_1, \ldots, y_n)$ are the variables ("unknowns"). The data (c,h,A,G,b) specifies a problem *instance* where c and h are appropriately-sized vectors and A and G are appropriately-sized matrices.

The *feasible region* (also called the feasible set or design space) is the set $S = \{Ax + Gy \leq b, x \in \mathbb{R}^p_+ \, y \in \mathbb{Z}^n_+\}$ and an instance is *feasible* if $S \neq \varnothing$. The feasible point (x^0, y^0) for which the objective function, $z = cx + hy$, is as large as possible is called the *optimal solution* and $z = cx^0 + hy^0$ is the *optimal value.*

The mixed integer program is called "mixed" because it contains a mix of both continuous and integer variables. There are several special cases of this optimization problem. In the case in which there are no integer variables,

$$\max\{cx \mid Ax \leq b, x \in \mathbb{R}^p_+\}$$

the problem is a *linear program* (LP). In the special case in which there are no continuous variables,

$$\max\{hy \mid Gy \leq b, y \in \mathbb{Z}^n_+\}$$

the problem is called an *(pure) integer program* (IP). Finally, in the special case where the integer variables represent logical decisions and are constrained to $y \in \mathbb{B}^n$, the

[2]This brief introduction to mixed integer programs closely follows the introduction presented in Nemhauser and Wolsey (1989) §I.1. [160]

problem is called a 0-1 mixed integer program (0-1 MIP). Finally, if $y \in \mathbb{B}^n$ and $x = \varnothing$, the problem is called a *binary integer program* (BIP).

Mixed integer *nonlinear* programs are a class of closely-related optimization problems in which the constraints and/or the objective function, is nonlinear. Some approaches for solving mixed integer linear programs have been successfully adapted to their nonlinear cousins, but in general, mixed integer nonlinear problems are much more challenging to solve.[112] In this work, as is the general norm, the term "mixed integer program" will only refer to the fully linear case unless otherwise specified.

In general, formulating an optimization problem as a mixed integer program is preferable to other classes of optimization problems because their linear mathematical structure often makes them easier to solve. Linear programs, which are fully linear and fully continuous, can often be solved very quickly and have been shown to be solvable in polynomial time. The presence of integral variables in an optimization problem greatly increases the difficulty of finding the optimal solution compared to a fully continuous problem, but this may be unavoidable in order to accurately capture the real world system, such as in the express shipment problem. Integer programming has been shown to be $\mathcal{NP}$-complete, and the special case 0-1 integer programming problem is one of Karp's 21 problems.[126]

The branch-and-bound algorithm, which is a systematic enumeration algorithm (explained further in Chapter 4) is a popular approach for solving problems with integer variables and it depends on the maintenance of upper and lower bounds. When solving a linear mixed integer program, an LP relaxation of the original problem is used in the maintenance of the bounds as well as subproblem evaluations along the search tree. As a result, although IPs are still difficult problems to solve, formulating a problem such that it has an LP relaxation is still advantageous when implementing common solution approaches.

3.2 Network Design Problems

As stated previously, many well-known logistical and combinatorial optimization problems can be formulated as mixed integer programs, but more specifically, many of those problems, including the express shipment problem, can be formulated as a specific mixed integer program called a Network Design problem (NDP). Briefly stated, the goal of the NDP is to find an optimal subgraph $F = (N', A')$ of a graph $G = (N, A)$ subject to flow capacity and balance side constraints. Many applications include additional side constraints specific to the problem needs. Some of the best-known specializations and variations of the NDP include the traveling salesman problem (TSP), vehicle routing problem (VRP), minimum spanning tree problem (MSTP), and the facility location problem (FLP). As described by Magnanti and Wong (1984), "many of the most well-known and intensively studied problems encountered in transportation planning can be viewed as specializations and variations of this generic design model."[146]

The most general form of the Network Design problem is to find the optimal subgraph F of a graph G subject to forcing, flow balance, and side constraints. There are typically two types of variables: 1) binary variables capturing the decision to include arc (i, j) in the subgraph F, and 2) flow variables (which are continuous) capturing the amount of flow on an arc. The forcing constraints require the amount of flow on an arc to be less than the capacity on the arc. The flow balance constraints require flow to be conserved throughout the network. The side constraints are additional problem-specific constraints.

Depending on the application, the objective function may be linear or nonlinear. Nonlinear objective functions may be the result of congestion effects or other coupled phenomena. This is not the case for the express shipment problem and the problem can be comfortably modeled as a mixed integer program. Table 4, adapted from Magnanti and Wong (1984), provides an overview of well-known NDPs and NDP

Table 4: Specializations and Variations of the Network Design Problem (Adapted from Magnanti and Wong[146])

Problem Name	Demand Structure	Objective Function	Capacities	Side Constraints
Minimal Spanning Tree	Complete (undirected network)	Linear in design variables, no flow costs	Uncapacitated	None
Shortest Path	Arbitrary	Linear in flows, no design costs	Uncapacitated	None
Steiner Tree Problem	Complete on a subset of nodes (undirected network)	Linear in design variables, no flow costs	Uncapacitated	None
Multicommodity Flow Problem	Single source	Linear in design variables, no flow costs	Uncapacitated	None
Traveling Salesman Problem	Complete	Linear in design and flow variables, large constant fixed costs	Uncapacitated	None
Vehicle Routing Problem	Single Source	Linear design variables, no flow costs	Fixed capacity on all arcs	Assignment constraints on design variables
Facility Location Problem	Arbitrary	Linear in flow variables, fixed costs on facility nodes	Capacities on facility nodes	Node
Fixed Charge Network Design Problem	Arbitrary	Linear in flows and design variables	Uncapacitated	None
Service Network Design Problem	Arbitrary	Linear in flows and design variables	Fixed capacity on all arcs	Assignment and balance constraints on design variables
Network Loading Problem	Arbitrary	Linear in design variables, no flow costs	Fixed capacity on all arcs	Assignment and integral design variables

variations that are applicable to the express shipment problem.

In the special case in which the network is capacitated and there is more than one commodity being moved through the network, the Network Design problem becomes the multicommodity flow problem.[4] For example, many telecommunications problems can be modeled as an MCFP in which calls, defined by an origin and destination, must be routed through the telecommunications infrastructure in which each link (arc) has a limited bandwidth. The Express Shipment Service Network Design problem (ESSND), the extension of the NDP presented in Section 3.3 is a special case of the MCFP since each origin-destination contains commodities (defined as the combination of origin, destination, product and origination day) that have to be routed through the network. The ESSND problem adds additional constraints regarding the nature of the arc capacity – specifically that it must be applied in integral units in the form of vehicles which must be balanced and conserved through the network.

The network loading problem (NLP) is another special case of the NDP, and is one in which capacity is assigned to arcs in the form of *"facilities"* in integer quantities.[11, 148] For a transportation network, a "facility" would be a vehicle. Facility (vehicle) costs are assumed to dominate flow costs, and as a consequence flow costs are not included in the objective function. Therefore, the total cost for the network is the sum of the costs incurred by installing facilities on the network. The NLP model can be used for either single or multicommodity problems. The ESSND problem adds additional constraints to enforce facility (vehicle) balance and fleet size.

Of the general Network Design problems found in the literature, the Service Network Design problem (SNDP) is the most similar problem to the ESSND problem. It is the special case of the NLP in which the installed facilities must be balanced, that is, facilities (vehicles), as well as flow, must be conserved through the network.

3.2.1 Network Design Problem Formulation

The objective of the Network Design problem is to find the optimal subgraph F in a graph G subject to forcing, flow balance and side constraints. In order to express the problem in a more detailed form, the following notation is first defined,

Sets

- $i \in N$: node in the set of nodes
- $(i, j) \in A$: arc in the set of arcs
- $k \in K$: commodity in the set of commodities
- S: set of side constraints

Cost Parameters

- c_{ij}^k: cost of sending one unit of flow of $k \in K$ along arc $(i, j) \in A$
- d_{ij}: fixed cost of including arc $(i, j) \in A$

Additional Parameters

- b_k: total quantity of commodity k
- u_{ij}: capacity of arc (i, j)

Decision Variables

- x_{ij}^k: fraction of commodity $k \in K$ on arc $(i, j) \in A$
- $y_{ij} = \begin{cases} 1 & \text{if arc } (i, j) \text{ is included in the subgraph } F \\ 0 & \text{otherwise} \end{cases}$

When the objective function, $\phi(x_{ij}^k, y_{ij})$, is linear, the Network Design Problem can be mathematically formulated in the following way,

$$\begin{aligned}
\min \quad & \phi(x_{ij}^k, y_{ij}) = \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^k b^k x_{ij}^k + \sum_{(i,j) \in A} d_{ij} y_{ij} && && \text{(3.1a)} \\
\text{s.t.} \quad & \sum_{k \in K} b^k x_{ij}^k \leq u_{ij} y_{ij} && \forall (i,j) \in A && \text{(3.1b)} \\
& \sum_{j:(i,j) \in A} x_{ij}^k - \sum_{j:(j,i) \in A} x_{ji}^k = \begin{cases} 1 & \text{if } i = O(k), \\ -1 & \text{if } i = D(k), \\ 0 & \text{otherwise} \end{cases} && \forall i \in N, k \in K && \text{(3.1c)} \\
& (x_{ij}^k, y_{ij}) \in S && && \text{(3.1d)} \\
& 0 \geq x_{ij}^k \leq 1 && \forall (i,j) \in A, k \in K && \text{(3.1e)} \\
& y_{ij} \in \{0,1\} && \forall (i,j) \in A && \text{(3.1f)}
\end{aligned}$$

The objective, Equation 3.1a, is to minimize the total cost of the network, $\phi(x_{ij}^k, y_{ij})$, which is the sum of the linear cost of each commodity and the fixed cost of each included arc. Equation 3.1b represents the "forcing" constraints, which ensure that the flow on any arc does not exceed the capacity assigned to that arc. Next is the flow balance, also called the flow conservation, constraint, Eq. 3.1c, which ensures that commodities only enter or leave the network at their source or sink nodes, respectively. The side constraints, Eq. 3.1d, serve as a catch-all for constraints that may be added to tailor the NDP to specific applications. Finally, Eq. 3.1e and Eq. 3.1f are the range constraints on the flow and decision variables. They ensure that the flows are nonnegative, and the decision variables are binary.

There are some constraints that are common additions to the Network Design problem. One of them is a budget constraint,

$$\sum_{(i,j) \in A} e_{ij} y_{ij} \leq B$$

where e_{ij} represents the cost of using the arc (i, j) and B represents the total budget. A budget constraint may be used in a situation in which infrastructure must be built in order to use an arc (i, j), such as installing power lines or telecommunication cables.

There can also be topological constraints. In a network improvement model, there is a pre-existing network which must be automatically included in the model. A pre-existing arc, (i, j), can be represented in the model with the constraint $y_{ij} = 1$, which states that the decision to include the arc (i, j) in the solution network must be true. Additionally, there can be conditional topological constraints, such as $y_{rs} \leq y_{ij}$, which states that arc (i, j) can be chosen if and only if arc (r, s) is also chosen.

3.2.2 Service Network Design Problem Formulations

The Service Network Design Problem (SNDP) is a variant of the Network Loading Problem (NLP) which is, in turn, a variant of the Network Design Problem. The SNDP differs from the NDP in the fact that arc capacities must be assigned in the form of *facilities*, which must balance through the network. Therefore, the SNDP formulation includes a design balance constraint, and the design variable domain is changed from binary to nonnegative integer. The balance requirement and the retainment of the flow costs in the objective function differentiates a Service Network Design problem from a network loading problem.

In the context of express shipment, the "facilities" in regards to the problem formulation terminology are *vehicles* and should not be confused with "facilities" in the problem-application-sense, which are physical locations owned by a company which serve as origins, destinations and sorts. To reduce the conflicting use of the word while respecting its established use in the respective contexts, vehicles in mathematical formulations will be represented as $f \in F$, which provides consistency with formulations in the literature, but otherwise, outside of this subsection, "facilities" will refer only to origin, destination and sorting locations owned by an express shipment company.

Kim (1997), in his Ph.D. thesis on Service Network Design problems and their application to the express shipment problem, presents three formulations for the SNDP, 1) node-arc, 2) path-based, and 3) tree-based.[129] In the node-arc formulation, the flow and decision variables are directly tied to specific nodes and arcs in the graph, just like the NDP formulation in Formulation 3.1. In the path formulation, the flow and decision variables correspond to paths (i.e. ordered subsets of arcs) through the network rather than individual arcs. Finally, in the tree formulation, which was originally presented by Jones (1993), commodities with the same origin, or, alternatively, the same destination, are aggregated together in a single *super commodity* and flow options are represented as trees rooted at the origin (destination) of the super commodity.[121] In the tree formulation, design variables are still represented as paths.

Kim highlights several advantages that path-based and tree-based formulations have over node-arc formulations. First, non-linear costs can be easily captured in path-based and tree-based formulations without introducing non-linearity in the problem formulation. Second, some constraints, such as a limit on the number of arcs in a path are difficult to write effectively in a node-arc formulation but are trivial in a path-based or tree-based formulation. Finally, the number of constraints, particularly flow balance constraints, is significantly lower in path-based and tree-based formulations compared node-arc formulations. However, the reduced number of constraints comes at the cost of significantly more variables to achieve equivalent formulations.

To best illustrate the SNDP's relationship to the NDP, and later, the Express Shipment Service Network Design problem, the node-arc, and path-based formulations are presented here. First, using the notation defined for the NDP in Formulation 3.1 and additionally defining $f \in F$ to be a facility in the set of facilities (i.e. vehicle in the set of vehicles), and the design variables to be

$$y_{ij}^f = \begin{cases} 1 & \text{if arc } (i,j) \text{ is served with the facility (vehicle) } f \\ 0 & \text{otherwise} \end{cases}$$

the node-arc formulation for the Service Network Design problem can be written,

$$\begin{aligned}
\min \quad & \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^k b^k x_{ij}^k + \sum_{(i,j) \in A} d_{ij}^f y_{ij}^f && && \text{(3.2a)} \\
\text{s.t.} \quad & \sum_{k \in K} b^k x_{ij}^k \leq \sum_{f \in F} u^f y_{ij}^f && \forall (i,j) \in A && \text{(3.2b)} \\
& \sum_{j:(i,j) \in A} x_{ij}^k - \sum_{j:(j,i) \in A} x_{ji}^k = \begin{cases} 1 & \text{if } i = O(k), \\ -1 & \text{if } i = D(k), \\ 0 & \text{otherwise} \end{cases} && \forall i \in N, k \in K && \text{(3.2c)} \\
& \sum_{j:(i,j) \in A} y_{ij}^f - \sum_{j:(j,i) \in A} y_{ji}^f = 0 && \forall i \in N, f \in F && \text{(3.2d)} \\
& 0 \leq x_{ij}^k \leq 1 && \forall (i,j) \in A, k \in K && \text{(3.2e)} \\
& y_{ij}^f \in \mathbb{Z}_+ && \forall (i,j) \in A, f \in F && \text{(3.2f)}
\end{aligned}$$

The objective function (Eq 3.2a), forcing constraint (Eq 3.2b) and flow balance constraint (Eq 3.2c) are nearly identical to those in the NDP formulation (Formulation 3.1). The only difference is that capacities, u^f, are based on the assignment of facilities rather than a property of the arc itself, and the fixed arc cost, d_{ij}^f, is based on the assigned facility as well as the arc. Constraint 3.2d, which ensures the balance of facilities throughout the network, is new. Finally, the domain of the flow variables, Eq. 3.2e, is unchanged relative to the NDP formulation, but the decision variables, Eq. 3.2f, correspond to a facility choice for arc (i,j) and are nonnegative integral-valued, rather than binary-valued.

Alternatively, the SNDP can be represented in a path-based formulation. To represent this formulation, the following additional notation must be defined. To differentiate flow-based paths from facility-based paths, flow-based paths are called “paths” while facility-based paths are called “routes.”

Sets

- $p \in P^k$: path in the set of paths for commodity k
- $r \in R^f$: route in the set of design routes for facility f
- $f \in F$: facility in the set of facilities

Cost Parameters

- c_p^f: cost of sending one unit of flow of $k \in K$ along path $p \in P^k$
- d_r^f: fixed cost of providing one unit of facility $f \in F$ along route $r \in R^f$

Additional Parameters

- u^f: capacity of facility f
- b_k: total quantity of commodity k

Indicators

- $\beta_i^r = \begin{cases} 1, & \text{if route } r \text{ starts at } i \\ -1, & \text{if route } r \text{ ends at } i \\ 0, & \text{otherwise} \end{cases}$
- $\delta_{ij}^r = \begin{cases} 1, & \text{if route } r \text{ includes arc } (i,j) \\ 0, & \text{otherwise} \end{cases}$
- $\delta_{ij}^p = \begin{cases} 1, & \text{if path } p \text{ includes arc } (i,j) \\ 0, & \text{otherwise} \end{cases}$

Decision Variables

- x_p^k: fraction of b^k on path $p \in P^k$ for $k \in K$
- y_r^f: number of facilities, $f \in F$, installed on $r \in R^f$

Using this notation, the path-based SNDP formulation can be expressed,

$$
\begin{aligned}
\min \quad & \sum_{k \in K} \sum_{p \in P^k} c_p^k b^k x_p^k + \sum_{f \in F} \sum_{r \in R^f} d_r^f y_r^f && && \text{(3.3a)} \\
\text{s.t.} \quad & \sum_{k \in K} \sum_{p \in P^k} \delta_{ij}^p b^k x_p^k \leq \sum_{f \in F} \sum_{r \in R^f} \delta_{ij}^r u^f y_r^f && \forall (i,j) \in A && \text{(3.3b)} \\
& \sum_{p \in P^k} x_p^k = 1 && \forall k \in K && \text{(3.3c)} \\
& \sum_{r \in R^f} \beta_i^r y_r^f = 0 && \forall i \in N, f \in F && \text{(3.3d)} \\
& 0 \leq x_p^k \leq 1 && \forall p \in P^k, k \in K && \text{(3.3e)} \\
& y_r^f \in \mathbb{Z}_+ && \forall r \in R^f, f \in F && \text{(3.3f)}
\end{aligned}
$$

As with the node-arc formulation, the objective function, Equation 3.3a, is the sum of the flow costs and facility costs. The first constraint, Equation 3.3b, is the forcing constraint, which uses indicator variables to enforce the constraint on every arc. Next, the flow balance constraint from the node-arc formulation, Eq. 3.2c, has been replaced with a flow *cover* constraint which ensures that all of the flow for each commodity is accounted for within the network. Flow balance is automatically ensured through the use of the path variables so it is sufficient to ensure that the full commodity demand is satisfied without tracking it throughout the network. Constraint 3.3d enforces facility balance through the network. Finally, the flow and design variable domains, Equations 3.3e and 3.3f are nearly identical to the node-arc formulation, except that the variables are path-based and route-based rather than arc-based.

3.3 Express Shipment Service Network Design Problem

While the Service Network Design problem captures most of the essence of express shipment, i.e. finding the lowest cost assignment of vehicles to move flow from origin to destination within a network while ensuring the balance of both flow and vehicles, there are additional constraints that must be added to the SNDP formulation to truly capture the express shipment problem. Based on the problem description from Chapter 2, a complete model of the express shipment problem must be capable of encapsulating and handling its essential parts,

- Find the minimum cost way to move all of the packages
- Cost is a function of vehicle use
- Commodity volume is large, diverse and fully-connected
- A commodity can only enter the network and its origin and leave the network at it destination (i.e. commodity flow must be conserved)
- Diverse and multi-modal vehicle fleet
- Diverse vehicle capacities
- Limited number of each vehicle type
- Vehicles must be assigned in integral quantities
- In order for a vehicle to travel from one location to another, it must first be present at the origin location (i.e. vehicles must be balanced/conserved throughout the network)
- Multi-hub-and-spoke network structure
- Sorting facilities have limits on the number of packages they can process during a given period
- Level of service (i.e. time) requirements must be satisfied

Of the requirements listed above, many, such as the calculation and minimization of cost, handling a large set of diverse commodities and the use of diversely-sized vehicles to provide capacities to commodity flows are handled seamlessly by modeling the problem as a Service Network Design problem. Others, particularly the limited fleet size and volume capacities at sorting facilities, are not part of the typical Service Network Design problem formulation but can be easily included as linear side constraints. The final consideration, level of service/time requirements, has not been explicitly discussed in the context of any of the formulations thus far but can be handled with an intelligent choice of the graph, G, upon which the SNDP is based. Namely, a time-space graph can be used to encapsulate temporal as well as spatial considerations of the problem.

3.3.1 Time-Space Graphs

In a time-space graph (also known as a time-space network or time-expanded network), nodes represent points in both space *and* time. The concept was first introduced by Ford and Fulkerson in 1958 who modeled dynamic flows on a static "expanded" network.[86, 87] Since that point, time-space networks have been, along with connection networks, the most common way to model time-dependent aspects of many problems, including transportation planning problems.

The primary idea behind time-space graphs is that a spatial graph can be expanded to capture time-dependent problem features by adding a time property to each node. The original spatial graph is modified such that a node exists for each point in time (according to a pre-determined time step) and space, and arcs represent physically feasible connections in time and space. Figure 6 displays a notional time-space graph involving three locations and ten time periods for a total of thirty nodes.

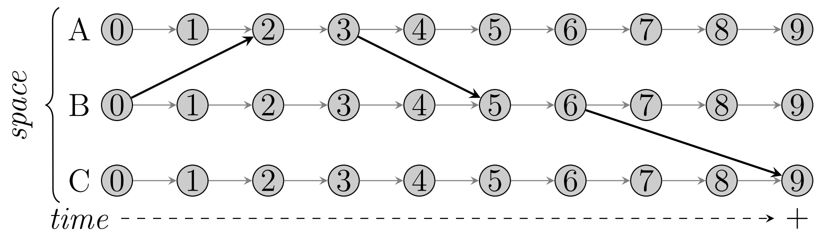


Figure 6: Notional Time-Space Graph

The basic time-space graph contains nodes for each place in space and each step in time. There are two types of connections represented by the arcs: 1) connections *between* locations, and 2) connections *at* locations. For transportation problems like the ESSND problem, arcs between locations (shown in black) represent feasible flights between locations. Connections at a location, sometimes called time-line arcs, capture the passing of time at a location (gray arcs).

For more information and background on time-space networks, the introduction by Skutella (2009)[180] provides a thorough summary, as well as Chapter 19.6 of the text by Ahuja, Magnanti and Orlin (1995)[5], and the original text by Ford and Fulkerson (1962)[87]. Although they can be an effective tool in handling time-dependent aspects of networks, time-space graphs have some drawbacks, particularly the time discretization and the corresponding explosion of problem size. Many applications of time-space networks employ novel approaches to reduce the size of the time-space graph without sacrificing the effectiveness of the time-space graph. The network reduction approach undertaken for this work is detailed in Chapter 6, but for this chapter, it is sufficient to mention that a time-space graph is used to handle the time-dependent nature of the problem.

3.3.2 Express Shipment Service Network Design Problem Formulation

In the literature, there is no universally-used problem definition for the express shipment problem. Many applications add carrier-specific or application-specific side constraints or assumptions. However, there is a "core" set of constraints that are found, either explicitly or implicitly, in the majority of express shipment formulations. The problem definition used in this work includes those core constraints as well as a "wrap-around" constraint and a feeder-use constraint.

The core formulation of the Express Shipment Service Network Design problem inherits the constraints of the SNDP, as implied in the name, and includes additional constraints for hull count and sort capacity. These two additional constraints capture the critical aspects of the problem that elevate it beyond a Service Network Design problem. The hull count constraint is critical since express shipment airlines own a limited number of vehicles and the composition of the aircraft fleet can have a large impact on how those vehicles are used. The sort capacity constraint encompasses another critical limitation faced by express shipment carriers. A key reason that express shipment carriers operate in a multi-hub-and-spoke network is because the primary sorting locations (such as FedEx's Super Hub and UPS's Worldport) have neared the current practical, physical and technological limits of processing within their limited time window. The secondary hubs provide additional sorting capacity within the network and help to lower some transportation costs by sorting some intraregional volume.

In addition to these core constraints, the formulation of the Express Shipment Service Network Design problem used in this work includes a feeder-use constraint. The feeder use limit restricts the use of feeders to one per arc. This reflects the carrier-specific preference to only use feeder aircraft to serve small markets. Feeder aircraft are considerably cheaper than trunk aircraft and the larger feeders have about half of the capacity of the smallest trunk aircraft. As a result, if left unrestricted, an optimizer will often assign multiple feeder aircraft to a serve a medium-sized market rather than assign a single small trunk aircraft, contrary to the preferences of the carrier.

As a form of the SNDP, the formulation considerations raised by Kim (1997) also apply to the Express Shipment Service Network Design problem. In some regards, particularly the number and ease of writing constraints, a path-based or tree-based formulation is superior. On the other hand, node-arc formulations tend to have many fewer variables for realistic problem sizes, which is crucial in many MIP solution methods.

Although Kim (1997) [129] decided that the path-based and tree-based formulations were superior to the node-arc formulation for his application and problem instances, I found that there were appealing aspects to *both* the path-based and node-arc formulations. On the one hand, using a path-based formulation is appealing when faced with the massive number of flow balance constraints that would be required if the node-arc formulation were used with such a large commodity set. On the other hand, the highly diverse vehicle fleet and the massive number of commodities translates to a huge number of design and flow variables in the path-based and tree-based formulations.

In an effort to capture the best of both worlds, an *arc-path* formulation, which is a hybrid of the two formulations, is used in this work. Arc-path formulations, in which the flows are expressed in terms of paths and the design decisions are expressed in terms of arcs, are not as common as node-arc or path-based formulations but they do have precedent in the literature for other Service Network Design problems and multicommodity network flow problems.[10, 128, 162] The arc-path formulation takes advantage of the elimination of flow balance constraints from the path-based formulation while also keeping the number of design (vehicle) variables relatively low. This formulation does still suffer from a very large set of path-based flow variables, but this problem can be mitigated by preprocessing the set of flow variables and removing those that are unlikely to be present in the optimal solution. The impact that preprocessing has on the final solution value is based on how aggressive that remove scheme is. The scheme used in this work is presented in Chapter 6.

In order to mathematically capture the Express Shipment Service Network Design problem, additional notation is needed. That notation, including the relevant notation previously defined for the NDP and SNDP formulations, is,

Sets

- $i \in N$: node in the set of nodes
- $i^+ \in N^+$: source node in the set of source nodes
- $i^- \in N^-$: sink node in the set of sink nodes
- $(i,j) \in A$: arc in the set of arcs
- $k \in K$: commodity in the set of commodities
- $p \in P^k$: path in the set of paths for commodity k
- $h \in H$: hub in the set of hubs
- $\theta \in \Theta$: airport in the set of airports
- $f \in F$: vehicle type in the set of vehicle types
- $\tilde{F}$: set of truck vehicle types

Cost Parameters

- d_{ij}^f: fixed cost of providing one unit of facility (vehicle) $f \in F$ along route $r \in R^f$

Additional Parameters

- u^f: capacity of vehicle f
- b_k: total quantity of commodity k
- n_k: number of boxes in commodity k
- e_h: sorting capacity of hub h

Indicators

- $\delta_{ij}^p = \begin{cases} 1, & \text{if path } p \text{ includes arc } (i,j) \\ 0, & \text{otherwise} \end{cases}$
- $\delta_h^p = \begin{cases} 1, & \text{if path } p \text{ passes through hub } h \\ 0, & \text{otherwise} \end{cases}$
- $\delta_i^\theta = \begin{cases} 1, & \text{if node } i \text{ represents a point in time at airport } \theta \\ 0, & \text{otherwise} \end{cases}$

Decision Variables

- x_p^k: fraction of b^k on path $p \in P^k$ for $k \in K$
- y_{ij}^f: number of facilities, $f \in F$, installed on $(i,j) \in A$

Using this notation, the arc-path formulation for the Express Shipment Service Network Design problem as defined in this work can be written,

$$\min \quad \sum_{(i,j)\in A} d_{ij}^f y_{ij}^f \tag{3.4a}$$

$$\text{s.t.} \quad \sum_{k\in K}\sum_{p\in P^k} \delta_{ij}^p b^k x_p^k \leq \sum_{f\in F} u^f y_{ij}^f \qquad \forall (i,j) \in A \tag{3.4b}$$

$$\sum_{p\in P^k} x_p^k = 1 \qquad \forall k \in K \tag{3.4c}$$

$$\sum_{j:(i,j)\in A} y_{ij}^f = \sum_{j:(j,i)\in A} y_{ji}^f \qquad \forall i \in N, f \in F \setminus \tilde{F} \tag{3.4d}$$

$$\sum_{(i^+,j)\in A} y_{i^+j}^f \leq n_f \qquad \forall f \in F \setminus \tilde{F} \tag{3.4e}$$

$$\sum_{f\in \hat{F}} y_{ij}^f \leq 1 \qquad \forall (i,j) \in A \tag{3.4f}$$

$$\sum_{k\in K}\sum_{p\in P^k} \delta_h^p n_k x_p^k \leq e_h \qquad \forall h \in H \tag{3.4g}$$

$$0 \leq x_p^k \leq 1 \qquad \forall p \in P^k, k \in K \tag{3.4h}$$

$$y_{ij}^f \in \mathbb{Z}_+ \qquad \forall (i,j) \in A, f \in F \tag{3.4i}$$

As in the network loading problem, the objective, Eq. 3.4a, is to minimize the total fixed cost of installing vehicles on arcs. On its own, commodity flow cost is inconsequential compared to the cost of flying an aircraft and so in the model only installing vehicles to *move* commodities incurs a cost. The forcing constraint, Eq. 3.4b, is a combination of the forcing constraints from the node-arc and path-based SNDP formulations. The cover constraint, Eq. 3.4c, is unchanged from the SNDP path-based formulation. The vehicle balance constraint, Eq. 3.4d, is unchanged from the SNDP node-arc formulation. Equation 3.4e is the hull count constraint. Note that for the vehicle balance, wrap around and hull count constraints, only aircraft are constrained while trucks are not. Trucks are assumed to be so numerous and cheap that the are not subjected to the same constraints necessary to monitor aircraft movements throughout the network. The next equation, Eq. 3.4g, is the sort capacity constraint. Finally, Equations 3.4h and 3.4i are the bounds on the flow and design variables.

3.4 Summary

This chapter presented the mathematical formulation of the express shipment problem as a mixed integer program called the Express Shipment Service Network Design problem (ESSND). The ESSND is an extension of the Service Network Design problem, which is an extension of the network design problem. The connections of the ESSND to these more general problems was illustrated as well as the necessity of adding additional problem-specific constraints.

One of the primary motivations behind formulating the express shipment problem mathematically was so that computational solution methods could be employed in solving this large, complex optimization problem. In this chapter, it was briefly mentioned that formulating a problem as a mixed integer program was desirable compared to other problem structures involving integral variables because its linear constraints and objective function makes it generally easier to solve (there are, of course, exceptions). In the next chapter, Ch. 4, solution methods for MIPs in general as well as for MIPs related to the express shipment problem will be discussed.

CHAPTER IV

METHODS FOR SOLVING MIXED INTEGER PROGRAMS, THE ESSND AND RELATED PROBLEMS

This chapter discusses methods for solving mixed integer programs and provides a review of the literature focused on solution approaches for the ESSND and related problems. In order to provide context for the review of the literature, and set the stage for the solution methods considered for the express shipment problem, the common solution methods for solving MIPs is discussed first. Solution methods are categorized according to whether they are a monolithic approach, an exact decomposition approach or a heuristic approach. The methods in each category are discussed in brief in order to facilitate future discussions within this work.

The second half of this chapter focuses specifically on how these MIP solution methods have been applied to other ESSND problems and ESSND-related problems. The body of work in the literature that focuses directly on the express shipment problem is small compared to some other transportation fields, but that body of work is covered extensively in Section 4.2. Related problems including the letter mail/postal package delivery problem, LTL shipping problems, and passenger airline problems are covered in lighter detail in the following section.

4.1 A Brief Overview of Solution Approaches for Mixed Integer Programs

Since the introduction of linear programming during WWII and mixed integer programming shortly thereafter, the use of mathematical programming to solve real-world problems has become wide-spread and popular in both academia and industry.[67] The body of research devoted to efficiently formulating and solving mathematical programs is immense and can be found in countless books, journals, and conference proceedings today. As such, this work will only provide a *brief* overview of the most common methods for solving mixed integer programs in an effort to provide context for a) the problem-specific solution approaches and techniques found in the literature and summarized in this chapter, and b) the solution approaches mentioned and used in this work.

4.1.1 Linear Programming Solution Methods

Before a discussion of solution methods for mixed integer programs can meaningfully begin, solution methods for linear programs, the fully-continuous version of MIPs, must be presented. Linear programming was first introduced and solved by Leonid Kantorovich in 1939 in the USSR but it did not initially catch on. It was independently developed and solved in 1947 by George Dantzig who invented the simplex method to solve linear programs.[67, 177] In the same year, John von Neumann proposed the concept of duality, and together these two concepts revolutionized mathematical optimization. Since that time, the simplex method, with just a few modifications and variations, has been a cornerstone of solving LPs. In practice, it performs very well in the vast majority of applications but unfortunately has poor worst-case performance. For many years, it was not known if LPs were in $\mathcal{P}$, but in 1979 Leonid Khachiyan showed that the problem was indeed solvable in polynomial time. Unfortunately, although it was a theoretical breakthrough, in practice Khachiyan's algorithm, the ellipsoid method, performed poorly.[31] Not long after, in 1984, Narendra Karmarkar presented an interior-point method for solving LPs in polynomial time, which was both a practical as well as theoretical breakthrough.[30, 125]

Today, simplex and interior-point (also called barrier) methods are the two primary approaches for solving linear programs. They form the backbone of modern commercial solvers and approaches. Not only are they used to solve linear programs, but they are used to solve crucial subproblem linear relaxations within mixed integer program solution methods.

At the risk of oversimplifying both algorithmic approaches, the simplex algorithm operates by systematically moving from vertex to vertex of the convex polyhedron defined by the LP linear inequality constraints. Interior-point methods, by contrast, move within the interior of the feasible region to find optimal solutions at the vertices. A further dive into LP solution methods is beyond the scope of this brief summary, but the most important takeaway should be the fact that linear programs can be solved extremely quickly in practice using simplex and interior-point methods, and they are provably solvable in polynomial time. There are a multitude of resources for the reader interested in studying this topic further, including, but not limited to, the texts by Bertsimas and Tsitsiklis (1997)[29], Chvátal (1983)[45], Schrijver (1998)[177], Vanderbei (2014)[193], and for a more historical look, Dantzig (1963)[66].

4.1.2 Monolithic Solution Methods for Mixed Integer Programming

When it comes to solving integer programs or mixed-integer programs, two fundamental algorithms have been developed: 1) the cutting plane method, and 2) the branch-and-bound algorithm. Of the two, the general cutting plane algorithm was developed first, by Gomory in 1958 (a TSP-specific cutting plane was first developed by Dantzig, Fulkerson and Johnson in 1954).[55, 65, 99, 100, 101] Branch-and-bound (B&B) was first introduced by Land and Doig in 1960 and named by Little et al. in their work on the traveling salesman problem in 1963.[136, 141] Both of these algorithms seek to address the same difficulty in integer and mixed-integer programming. That difficulty is that the constraints of these mathematical programs rarely define the convex hull of the integer space, and therefore efficient algorithms developed for linear programs such as simplex and interior-point methods will not produce integral solutions and therefore cannot be used in isolation to solve IPs and MIPs to integral optimality. Since these methods do not require a user to reformulate the MIP, they will be called *monolithic* solution methods in this work.

4.1.2.1 Cutting Plane Methods

Cutting plane methods are based on the observation that *if* a linear programming solution method were given the convex hull of the set of feasible integer points, then it would find the optimal integer point. Cutting plane methods successively add additional linear inequalities (cutting planes) to the original formulation in order to eventually define the convex hull of the integer feasible set, at least in the region of the space relevant to finding the optimal solution. As long as none of the feasible integer solutions are incorrectly cut off from the feasible space, an optimal integral solution found by an LP solution method will be optimal for the original IP or MIP. Cutting plane methods can be either problem-specific or general based on the type of cutting planes used.

Figure 7 displays a notional application of cutting planes to find the optimal solution to a two-dimensional IP. In the figure, the gray-shaded space represents the polytope created by the linear constraints. The black dots within the polytope are the feasible integer points within the space. Their convex hull is shown in gold. The objective points towards the top right corner. If the integrality constraints were relaxed in the IP, yielding a formulation called the *LP relaxation*, the solution would be the right-hand red circle, but this solution is infeasible in the original IP formulation because the variable values are fractional. If the two shown cutting planes are added to the original formulation, they trim down the polytope such that the optimal integer point becomes a vertex of the polytope and can be found using LP solution methods.

Unfortunately, cutting plane methods often suffer from two major drawbacks. One drawback is that the problem of finding valid and useful cutting planes, called the

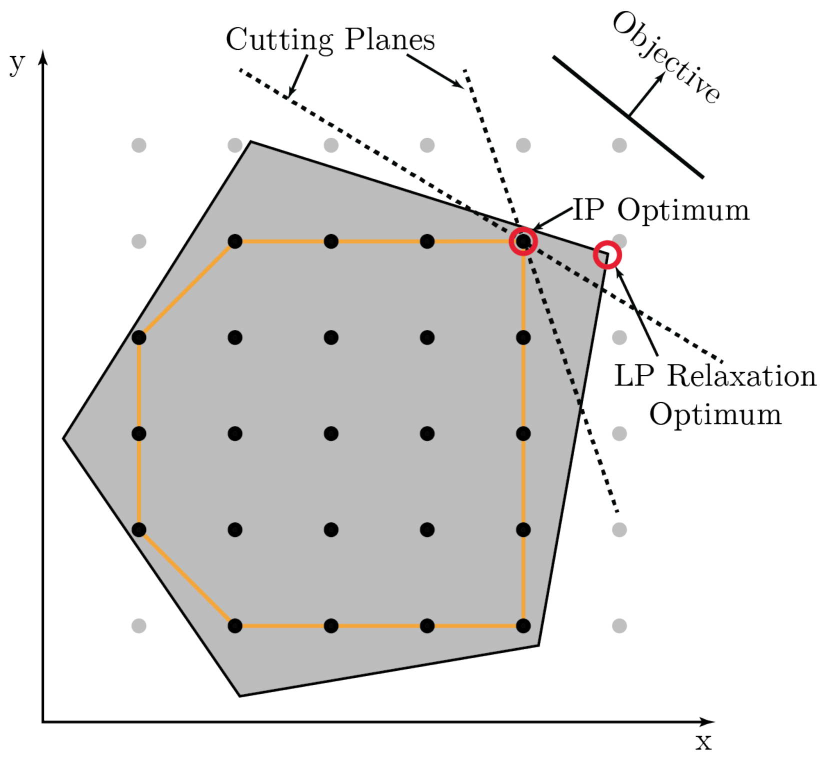


Figure 7: Using Cutting Planes to Find an Optimal IP Solution

separation problem, can be a challenging problem in itself. The other major drawback is that "pure" cutting plane methods often suffer from numerical instability and convergence issues. Cutting planes added later in the algorithmic progression tend to cut off less of the space relative to the earlier cutting planes and as a result, a large, potentially prohibitive, number of cutting planes is needed to achieve convergence. In general, these issues, particularly the slow convergence make "pure" cutting plane methods unappealing and seldom used.

4.1.2.2 Branch-and-Bound Methods

The other general approach to solving MIPs and IPs is branch-and-bound. Branch-and-bound methods are an *implicit* enumeration approach wherein a tree of possible solution values is successively created, explored and pruned until the optimal value is found and proven to be optimal. B&B differentiates itself from a "brute-force" enumeration technique by intelligently exploring branches and pruning provably sub-optimal and infeasible branches from the tree through the maintenance of solution bounds.

A notional example of a branch-and-bound algorithm for a four-dimensional BIP minimization problem as shown in Figure 8. Black nodes are those that are infeasible, gray nodes are those that have been pruned and the gold node is the optimal solution. The numbers inside the nodes are simply identifiers and not necessarily the progression of the algorithm. The numbers next to nodes 7-14 represent the upper and lower bounds of the node.

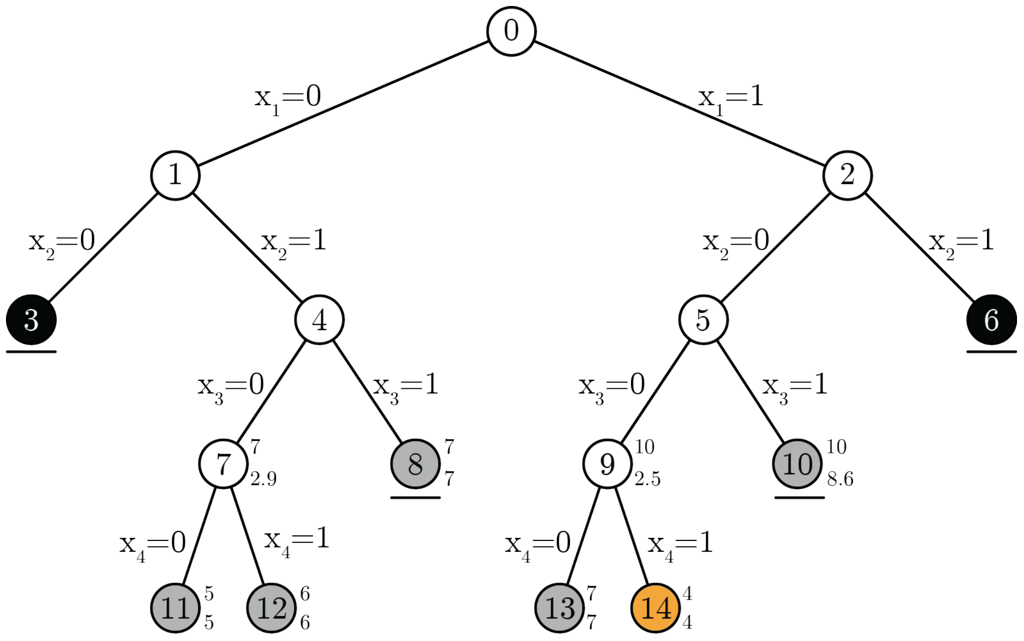


Figure 8: Using Branch-and-Bound to Find an Optimal Solution to a BIP Minimization Problem

The basic structure of a branch-and-bound method is to first, after initialization, select a subproblem and solve its relaxation. This produces a lower bound in the case of a minimization problem like the notional example in Fig. 8. Second, pruning is attempted using what is known about the problem thus far, specifically the best-found integer solution (if one exists) and the subproblem lower bound. Finally, termination criterion is checked. If it is not satisfied, the problem is divided into subproblems and the algorithm returns to the first step until the termination criterion has been satisfied.

B&B prunes branches through three criteria, 1) pruning by optimality, 2) pruning by bound, 3) pruning by infeasibility.[202] The first criteria, pruning by optimality, occurs when the lower bound and upper bound on a node in the tree are equal. This means that the objective value of all solutions of that branch will have the same value so it not worth the effort to evaluate them. This quality of having several solutions that produce the same objective value is called *degeneracy*. Node 8 of Fig. 8 has been pruned by optimality, and the solutions $x = \{0, 1, 1, 0\}$ and $x = \{0, 1, 1, 1\}$ are degenerate. The second criteria, pruning by bound, occurs when, in the case of a minimization problem, the lower bound is *higher* than the upper bound of another node. This indicates that it is impossible for the optimal solution to exist on that branch. Nodes 10 in Fig. 8 has been pruned by bound. The third criteria, pruning by infeasibility occurs when a branch is found to be infeasible, which means that every other node on that branch is also infeasible. Nodes 3 and 6 are pruned by infeasibility in Fig. 8. Finally, nodes 11, 12, 13 and 14 are evaluated exactly and node 14 is found to be optimal.

The major implementation decisions in the implementation of a branch-and-bound algorithm are 1) the strategy for dividing the problem into subproblems, 2) the strategy for selecting the next subproblem, and 3) how to relax the subproblem. For a detailed discussion of the first two points, Nemhauser and Wolsey (1989)[160] and Wolsey (1998)[202] are good resources. For the third implementation decision, the most common decision, and the one used in commercial solvers, is to solve the LP relaxation.

When the LP relaxation is used for the subproblem relaxations, division is performed by adding linear constraints to the subproblem to define that *particular* subproblem. In the BIP example in Fig. 8, since the variables were binary, this was done with equality constraints. However, if the domain of $x \in \mathbb{Z}_+^4$ and the relaxation of the root node produced a fractional solution with $x_1 = 3.5$, the algorithm could create two branches, $x_1 \leq 3$ and $x_1 \geq 4$, by adding those constraints to the respective subproblems. Finally, it should be mentioned that although this discussion has been focused on solving integer programs, branch-and-bound is easily extended to mixed integer programs.

One of the advantages of branch-and-bound from an implementation standpoint is that, because an upper and lower bound is being maintained on the solution throughout the entire progression of the algorithm, the user knows how far away the current solution is from the absolute best-possible solution. Progress is tracked in terms of the *MIP optimality gap*, which for a minimization problem, is defined as,

$$\text{optimality gap} = \frac{|z^*_{IP} - \underline{z}|}{\max\{z^*_{IP}, \varepsilon\}} \tag{4.1}$$

where z^*_{IP} is the current best found integral solution, $\underline{z}$ is the lower bound and ε is a small number, perhaps 10^{-4}, to avoid division by very small numbers. For maximization problems, the gap is calculated exactly the same except that the upper bound, $\bar{z}$, is used rather than the lower bound.

The optimality gap is additionally useful as a termination criterion, particularly for very large problems in which the B&B algorithm may never find and prove the absolute optimal solution, the user may choose to terminate the algorithm at a pre-determined optimality gap. In fact, the default termination criterion of Gurobi and Cplex, two of the most popular commercial solvers, is set to be gap $\leq 10^{-4}$.

Although the optimality gap is a useful tool, it is important to remember that it is the comparison of the best-found integer solution to the *relaxation* bound. As a result, if a problem has a poor relaxation, the optimization bound may be poor regardless of the actual quality of the integer solution. The optimality gap is an important and useful tool for understanding the progress of the branch-and-bound algorithm, and it will be used and referenced extensively in this work, but it is not without limitations.

4.1.2.3 Branch-and-Cut Methods

The shortcomings of general cutting plane methods were well-known from the start and cutting plane methods were widely considered to be impractical, even by Gomory himself.[102] As a result, for several decades, the branch-and-bound algorithm was the accepted method of solving mixed integer programs. In particular, an extension of branch-and-bound, called branch-and-cut (B&C), was considered to be one of the most effective ways to solve mathematical programs containing integer variables. Branch-and-cut combines the implicit enumeration of branch-and-bound with the bound strengthening of cutting planes. However, these early branch-and-cut schemes had to be specifically tailored to individual problems so, while they were effective, they were also time-intensive and challenging to implement.[54]

A major turning point occurred in the mid-1990s when Balas, Ceria, Cornuéjols and Natraj (1996) revisited Gormory's original general cutting planes to improve the progress of the branch-and-bound algorithm.[18, 55] From that work, interest in Gormory cuts and other general cuts was revitalized and the *general* branch-and-cut algorithm was developed. By adding cutting planes to the problem, the linear relaxation is improved which, in turn, allows for more pruning and overall better performance. The general B&C algorithm is now the foundation of modern integer and mixed-integer programming solvers.

In addition to the techniques discussed here for the basic implementations of these algorithms, commercial solvers employ sophisticated preprocessing algorithms that serve to improve the formulation before it enters the branch-and-cut algorithm. Additionally, heuristics are used to find integral solutions within the branch-and-bound tree. Finally, the branch-and-bound methods discussed here are the *sequential* implementation of branch-and-bound. Commercial solvers, when provided with multiple processors, will typically default to using *parallel* branch-and-bound, which solves subproblems simultaneously on different processors. For large problems with many subproblems, this can result in a significant performance increase. Across the board, commercial solvers such as CPLEX and Gurobi have far surpassed the basic implementation of the algorithms described here and in many cases have started to border on "black box" solvers thanks to highly sophisticated proprietary additions and modifications that are not present in the literature. They are exceptionally good at using these general algorithms to solve a wide array of MIP problem instances and employing them to solve MIPs, where appropriate, is an academic and industry standard practice.

The literature on these algorithms stretches back nearly as long as integer and mixed-integer programming has been a field of study, and the explanation here is intended only to give the reader a sense of how the commercial solvers operate. There is a plethora of sources on this subject, but some notable introductions and explanations can be found in Clausen (1999) [47], Jünger et al. (2009)[122], Klotz and Newman (2013)[134], Nemhauser and Wolsey (1989)[160], Papadimitriou and Steiglitz (1998)[165], Schrijver (1998)[177], and Wolsey (1998)[202].

4.1.3 Decomposition Solution Methods for Mixed Integer Programming

Decomposition is an approach in which a problem is split (reformulated) into smaller problems which are solved separately, either sequentially or in parallel.[37] The primary motivation behind decomposition is to take advantage of subproblems within the original problem which are easier to solve, reduce the non-linear effects of problem scaling, and exploit parallel computing advantages where possible. In this work, methods which require a deliberate reformulation of the problem will be considered *decomposition* methods.

Decomposition in mixed integer programming typically falls into two primary categories: 1) exact, and 2) approximate. Exact methods are those which utilize a decomposition reformulation but are guaranteed to produce an optimal solution if solved to completion. Approximate methods are heuristic approaches that do not guarantee a globally optimal solution but may be valuable in transforming an otherwise intractable problem into a tractable problem. For example, sequential optimization approaches are common in passenger airline scheduling problems in which flight scheduling, crew scheduling, and maintenance scheduling need to be optimized. Heuristic approaches will be discussed in more detail in Subsection 4.1.4.

The two primary exact decomposition methods in mixed integer programming are Dantzig-Wolfe decomposition and Benders decomposition. Dantzig-Wolfe decomposition was first formulated in 1960 for linear programs with "complicating constraints" and has since been extended to mixed integer and integer programs.[51, 68, 191, 192] It is a delayed column generation approach in which only a subset of the original variables is included in the master problem at any given time. The master problem is iteratively solved with subproblems, called pricing problems, which determine which variables should be added and which should be removed from the master problem. The primary idea behind Dantzig-Wolfe decomposition, and all column generation techniques, is that in the optimal solution many of the variable values will be zero so only the "necessary" variables need to be in the master problem.[72]

Closely related to Dantzig-Wolfe decomposition is another column generation approach called "branch-and-price" which combines column generation with the branch-and-bound algorithm.[22, 21, 72, 174, 191] This can be taken a step further and combined with a cutting plane algorithm to produce an algorithm called "branch-and-price-and-cut." Unlike branch-and-cut, there is no general branch-and-price algorithm, and all implementations are problem-specific. As a result, branch-and-price has not been adopted into general commercial solvers.

Whereas Dantzig-Wolfe decomposition is a column generation approach, Benders decomposition can be thought of as a *row* generation approach. Benders decomposition is appropriate in cases where there are “complicating variables,” which are variables that make the problem much more difficult to solve. The natural application of this approach is mixed integer programming because the integrality constraints on a subset of the variables makes the problem significantly more difficult to solve. It was originally formulated by J.F. Benders in 1962 and has recently become a popular decomposition approach.[27, 168] In Benders decomposition for mixed integer programming, the master problem contains only the integer variables and the constraints containing integer variables, thus constituting a relaxation of the original problem. The subproblems serve to successively add cutting planes (i.e. rows) to the master problem to drive it to a feasible optimal solution.

For linear programs, Benders decomposition is closely related to Dantzig-Wolfe decomposition and Lagrangian optimization.[139] When the problem is purely linear and continuous, solving a problem via Dantzig-Wolfe decomposition is equivalent to solving the dual of the problem with Benders decomposition. Lim (2010) additionally shows that for LPs, solving a problem with Benders decomposition is closely related to solving the problem using Lagrangian optimization. These relationships do not hold true for problems with integral variables due to duality relationships.

Lagrangian relaxation is another solution method well-suited to problems with complicating constraints, though it is not a decomposition method in the same style as Dantzig-Wolfe decomposition and Benders decomposition. Lagrangian relaxation seeks to make difficult problems easier to solve by moving complicating constraints to the objective function and using a penalty term to enforce the constraint. For more information specific to Lagrangian relaxation, Fisher (1981)[83], Fisher (1985)[84], Frangioni (2005)[88], and Nemhauser and Wolsey (1989)[160] are valuable resources among others.

Of these decomposition approaches, Benders decomposition is the most appropriate for the Express Shipment Service Network Design problem according to the literature because of the presence of complicating variables. Benders decomposition is used as an exact decomposition benchmark in this work and a more extensive explanation can be found in Chapter 8. For more information on the other decomposition approaches, Conejo et al. (2006)[51] focuses exclusively on mathematical programming decompositions, additionally mathematical programming texts such as Bertsimas and Tsitsiklis (1997)[29], Nemhauser and Wolsey (1988)[160], Schrjiver (1998)[177] and Wolsey (1998)[202] are valuable resources in addition to the approach-specific resources mentioned above.

4.1.4 Heuristic and Metaheuristic Solution Methods for Mixed Integer Programming

The solution methods described thus far have been exact (also called "complete") methods which are guaranteed to produce the optimal solution if they are run to convergence. However, these solution approaches for mixed integer programs can be very slow to converge.[1] Since combinatorial optimization problems and integer programs are $\mathcal{NP}$-hard, in the worst case solving these problems may require exponential time to solve exactly, assuming $\mathcal{P} \neq \mathcal{NP}$.[90]

In some situations, the long computational runtime may be acceptable, but for many practical real-world problems, faster solution times are desired, even at the expense of exact optimality. In response to this need, many heuristics (approximate) and metaheuristic methods have been developed to produce high-quality solutions in a fraction of the time needed by exact approaches. Heuristics are "rules of thumb" used to obtain a "*good enough*" solution. By their nature, implementations of heuristic

[1]For example, the current best-found solution to the Mona Lisa TSP Challenge, a 100,000 city TSP instance created by Robert Bosch, took 11.5 CPU years to obtain a 0.0019% gap.[34, 187]

approaches tend to be highly-problem specific, but many can be categorized as either 1) constructive, or 2) local (neighborhood) search.

Constructive heuristics are those that successively build a solution from nothing until a complete solution instance is formed. Problem-specific rules are used in order to decide how to proceed throughout the construction. For example, a feasible TSP tour could be constructed by always selecting the nearest unvisited city until all cities have been visited, then returning to the tour origin.[26] A local search heuristic starts with a feasible solution and explores its "neighborhood" to find the best local solution. For example, in the 2-opt local search heuristic for the TSP, a complete tour is first created by any method, then the algorithm swaps any nodes where the route crosses over itself.[64] Heuristics can be extremely fast and, if well-constructed, produce high-quality solutions. However, they can be difficult to adapt to other problems, tend to get stuck in local optima, and may exhibit very poor worst-case behavior.[173]

Metaheuristics are more generalized approaches that typically apply a heuristic approach with a formalized higher-level guidance that seeks to alleviate the shortcomings of basic heuristics. As explained by Voß et al. (2012), "A metaheuristic is an iterative master process that guides and modifies the operations of subordinate heuristics to efficiently produce high-quality solutions. It may manipulate a complete (or incomplete) single solution or a collection of solutions at each iteration. The subordinate heuristics may be high (or low) level procedures, or a simple local search, or just a construction method."[196]

To date, many metaheuristics have been developed and range from simplistic to complex. In light of the number and variety of metaheuristic approaches, it is more illuminating, at least in this brief overview, to discuss classifications for metaheuristics rather than attempt to enumerate even a subset of the algorithms. Classifications illuminate common schemes undertaken by metaheuristics without needing to explain the particulars of the approach. Blum and Roli (2003) present an excellent survey of metaheuristics and their classification scheme is presented here.[32] They identify the following ways to classify metaheuristics,

- Population-based vs. single point search
- Dynamic vs. static objective function
- One vs. various neighborhood structures
- Memory usage vs. memory-less methods
- Nature-inspired vs. non-nature-inspired

First, algorithms may either keep track of a population of solutions or a single solution at a time. Algorithms that work only on a single point are called “trajectory methods” because they follow the “trajectory” of that point through the space. Genetic Algorithms [40, 113, 114, 183] and Particle Swarm Optimization [48, 127, 166, 195] are examples of population-based methods. Examples of trajectory-based methods are Variable Neighborhood Search [109, 110, 158], Tabu Search [94, 95, 96, 97] and Simulated Annealing [133, 189], though it should be mentioned that population-based variants exist for many trajectory-based approaches. Second, some metaheuristics, like Guided Local Search [197, 198, 199], modify the objective function throughout the progression of the algorithm in an effort to escape local optima, while other methods keep the objective function as it was originally formulated. Third, some methods, like Variable Neighborhood Search, diversify the search space by changing the means of improving the solution throughout the algorithm. Most, however, retain the same “fitness landscape topology” throughout the progression. Additionally, methods differ in their use of memory. Some methods, like Greedy Randomized Adaptive Search Procedure (GRASP) [81, 169] make decisions solely on the information available in the current solution. Other methods, like Tabu Search “recall” results from past solutions and use that information to make decisions. Finally, many metaheuristics are inspired by natural processes (e.g. Genetic Algorithms, Ant Colony Algorithms), but aside from explaining the inspiration behind the approach, it is not a helpful classification in understanding the underlying structure of the algorithm.

A comprehensive survey of metaheuristics is beyond the intention of this brief introduction, particularly since metaheuristics are not a focus of this work and a multitude of books and survey papers have already been written on metaheuristics. In addition to the previously mentioned survey by Blum and Roli (2003)[32], for more information on metaheuristics the interested reader may wish to consult Boussaïd et al. (2013)[35], Gendreau and Potvin (2010)[91], Glover and Kochenberger (2006)[98], Luke (2009)[143], Talbi (2009)[184] as well as many others.

4.2 Express Shipment Service Network Design Problem Analysis and Solution Approaches

Compared to other transportation planning problems and applications such as less-than-truckload (LTL) shipping, passenger airline operations, commercial and cargo rail operations and oceanic shipping, express shipment has received relatively little focus in the literature. The largest body of work produced by one group of researchers was the product of a collaboration between MIT and UPS in the late 1990s and early 2000s. That collaboration focused on solving the express shipment problem for UPS's Next-Day Air (NDA) network (only overnight packages). The products of that collaboration are detailed in Subsection 4.2.1. Next, solution approaches for the express shipment problem from other research groups and collaborations are summarized in Subsection 4.2.2. Finally, research studying the express shipment problem, airlines and companies is summarized in Subsection 4.2.3.

4.2.1 Collaboration Between MIT and UPS

A number of papers and Ph.D. theses resulted from the collaboration between MIT and UPS, and most of the solution approaches fall into three major categories, 1) column generation, 2) composite variable formulations, and 3) heuristic. Unsurprisingly, these categories echo those presented earlier in this chapter in Section 4.1.

The first work in the collaboration was that of Barnhart and Schneur (1996).[24] They solve the air network design problem for express shipment using a column generation approach. They only solve the problem for next-day delivery, which they model as an integer program. Notably, aircraft capacities are not explicitly enforced; instead, the cost of a route that exceeds the aircraft capacity includes the cost of using belly space in passenger aircraft. Then, they solve the LP relaxation and use explicit column generation to find the final solution.

Kim (1997)[129], in his Ph.D. thesis, provides an approach to solving large-scale service network design problem with a case study focus on express service network design problems. Kim presents three formulations for the SNDP 1) a node-arc formulation, 2) a path-based formulation, and 3) a tree-based formulation. He decides that the node-arc formulation is inferior, and exact and approximate solution methods are developed for the path-based and tree-based formulations. Network reduction heuristics are also presented to remove unnecessary options in the underlying time-space graph. Kim additionally presents formulations for the express shipment application of the service network design problem formulations, and cutset inequalities for the ESSND problem to improve the lower bounds of the problem. He finds that the ESSND problem is too large to solve for real-world applications so he presents an approximate model and a heuristic solution approach. The work is summarized in the journal publication by Kim et al. (1999). [132]

Another paper based on the work in Kim's Ph.D. work is that of Kim and Barnhart (1997).[130]. In that paper, they discuss large-scale service network problems and, again, use the ESSND problem as an example. Problem size, loose bounds and the fact that problem is NP-hard are all cited as challenges to solving the problem. They use a path-based model and employ a branch-and-price-and-cut solution approach to solve it. In another publication, Kim and Barnhart [131] discuss challenges of the SNDP and present an algorithm from Kim's Ph.D. work that uses synchronized column generation and row generation.

Barnhart et al. (2002)[23] provide an alternative heuristic solution technique to the models presented in Kim's Ph.D. work (and related papers). They decompose the problem into the aircraft route generation and the package movement subproblems. The subproblems are then solved iteratively. Aircraft routes are generated first, then shipment flows are solved on the network (a multicommodity flow problem) and "promising route variables" are fixed. If there is enough capacity on the network to move all of the commodities, the algorithm ends. If not, commodities that can be flown on the fixed routes are removed from the model. If the model is then small enough to be solved exactly, it is solved exactly using the approach from Kim et al. (1999)[132], and otherwise new routes are generated and the process begins again.

In his Ph.D. thesis, Armacost (2000)[11] presents an alternate formulation for the ESSND problem that uses "composite variables." These variables represent aircraft route combinations that also capture package flows. This eliminates the need for the flow decisions. Additionally, Armacost shows his composite variable formulation to have tighter bounds than the conventional formulation and proves the formulation to be at least as strong as the network design-based formulation. Thanks to this tighter formulation, realistic problem instances can be solved in acceptable time frames. Armacost demonstrates the performance of his approach using a real-world problem instance from UPS and reports significant potential cost savings. However, his approach only covers overnight delivery volume and assumes a pre-set sort assignment for each origin-destination pair. The work of his Ph.D. thesis is summarized in the journal publication, Armacost et al. (2002)[12], and a summary of its implementation is presented in Armacost et al. (2004)[13]. The applicability of composite variable formulations to other transportation planning and logistics problems is studied in the Ph.D. thesis by Cohn (2002)[50].

In his Ph.D. thesis, Shen (2004)[178], expands the work of Armacost (2000)[11] to two-day volume and flexible hub assignment. The addition of these options in the design space greatly expands the size of the problem. In response, he presents three column generation approaches, 1) "naïve" column generation, 2) "aggregate information-enhanced" column generation, and 3) "disaggregate information-enhanced" column generation. Shen finds that the disaggregate information-enhanced column generation approach is the most appropriate. Additionally, Shen introduces another formulation for the ESSNDP entitled the "gateway cover and flow formulation." The work is summarized in the work by Barnhart and Shen (2005)[25].

One fundamental theme through this previous work is the modeling of the problem as a mixed integer linear program. No fewer than six different mathematical formulations are presented, but all are either mixed integer programs or integer programs. This modeling choice is natural given the strong precedent of solving other transportation problems using mathematical programming, and the strong similarities of the Service Network Design problem (SNDP) to the express shipment problem as shown extensively in Kim (1997).[130]

4.2.2 ESSND Problem Solution Methods

The collaboration between MIT and UPS is not the only work that has focused on the express shipment problem. Others have also tackled the problem and the closely related air cargo problem. Their work is summarized here.

In perhaps the first academic handling of this problem, Marsten and Muller (1980)[153] optimize the service network and aircraft routing of the Flying Tiger Line, an all-cargo airline. They build a mixed integer program based on "spider graphs" which capture the hub-and-spoke network structure of the airline but do not account for time. Then, they schedule flights to and from the hub with intermediate stops, if feasible. They consider both the night network and the day network but solve the two problems sequentially. The hull balance necessity between the two networks is handled by forcing the day network to use whichever aircraft was left at a spoke by the night network. They additionally consider prime (night) cargo, day cargo, and second-day cargo. The networks were smaller then they are today, but Marsten and Muller are able to solve real-world problems and "[determine] the design of Flying Tiger's service network for the mid-80's," which at that time, was still to come. The model was also used to perform strategic studies for the company.

Derigs et al. (2009)[71] present a model and solution method for incremental planning and a model and solution for integrated planning for air cargo. The goal of their approach is to choose the best combination of flights, given a list of mandatory flights and optional flights. The incremental planning approach decomposes the ESSND problem into flight selection, route plan generation and schedule evaluation subproblems. The modules are modified iteratively until a final solution is found. The integrated planning approach uses column generation and a shortest path algorithm to define routes.

Derigs and Frederichs (2013)[70] build on their previous work and expand the scope of their problem instance. As before, they solve the ESSND problem for the case in which a pre-existing schedule must be optimally modified. They identify, with the support of the literature, a decomposition into flight selection, fleet assignment, rotation planning and cargo routing subproblems. They provide subproblem formulations and present a branch-and-price-and-cut approach to solve the subproblems.

Li et al. (2006)[137] solve the integrated cargo routing and fleet assignment problems for a combination cargo-passenger carrier using Benders decomposition. They first identify the cargo routing and fleet assignment subproblems then present an integrated formulation which maximizes profits. They use Benders decomposition to solve the overall problem, which they argue decomposes back into a cargo routing subproblem and a fleet assignment feasibility problem. They report significant improvements in runtime through the use of Benders decomposition though they are not clear on the scope of their test problems.

Zäpfel and Wasner (2002)[206] study the optimality of hub and spoke networks for parcel service providers. They determine that a pure hub and spoke topology is not always appropriate and instead, including shortcuts provides a better solution. They use an iterative algorithm to perform the line haul network analysis that successively reduces the design space by eliminating poor options. They note that for large problems the design space is not reduced sufficiently to produce an optimal solution so a local search algorithm is used. They also study the joint line haul and pickup/delivery (i.e. last mile) problem.

In a follow-on to their 2002 paper, Wasner and Zäpfel (2004)[200] tackle an expanded scope parcel service problem which includes hub location, last mile service, and line haul vehicle routing. This results in a very large, complex and interrelated model. In response, they develop an iterative heuristic algorithm that decomposes the problem into subproblems and features several feedback loops to converge on a final solution. The algorithm is applied to an Austrian postal network and the potential for very large savings is found.

Louwerse et al. (2014)[142] solve a form of the ESSND problem using column generation. They also find that the problem is too large to solve as a single MIP. In their formulation, they divide the problem into movements between depots and hubs, and movements between hubs. They use the composite variable formulation from Armacost et al. (2000,2002,2004)[11, 12, 13] to strengthen their formulation. Their solution approach uses column generation and a local search algorithm then iterates until convergence. They find large cost improvements in their test networks over networks used by a real-life carrier.

Cohn et al. (2007)[49] and Root and Cohn (2008)[172] develop an alternative formulation for expressing the line haul problem for ground-based express package carriers. In their alternative formulation, variables represent combinations of loads and their corresponding routings rather than explicitly enforcing these qualities through the constraints. This alternative formulation strengthens the LP relaxation, reduces the number of constraints and allows the inclusion of operational considerations that would otherwise be difficult or impossible to include. However, their approach is entirely focused on the loading and routing of trailers. It is unclear whether this could be scaled to an airline application in which there are diverse vehicle capacities.

Grünert and Sebastian (2000)[105] study optimization models for postal and express shipment companies. They identify important planning tasks within these companies and discuss how they relate to optimization models. Those models are then used for decision support. Analyses are performed on a German case study provided by Deutsche Post AG.

Büdenbender, Grünert and Sebastian (2000)[38] present a network design problem for routing letter mail using direct flights rather than a hub-and-spoke. Their argument is that the time gained by avoiding hub-and-spoke flight enables more trucking. They model the problem as a capacitated warehouse location problem and present a hybrid tabu search/branch-and-bound algorithm to solve it. They show results for a German letter mail network and are capable of solving real-world instances in acceptable time. Grünert, Sebastian and Th arigen (1999)[104] present a decision support system that incorporates the group's previous developments into a decision support system for Deutsche Post AG.

Finally, there are a few papers that study tangential parts of the express shipment operations that I will mention briefly. Schenk and Klabjan (2008)[175], and Schenk and Klabjan (2010)[176] focus on the sorting of packages at stations and ramps and optimize the problem using approximate dynamic programming. Yaman, Karasan and Kara (2012)[204] study the *time* at which demand should be released from hubs in order to enable next-day delivery in a ground-based express shipment network. Li, Miao and Wang (2013)[138] study the challenge of USPS to intelligently allocate airmail to commercial (i.e. FedEx) flights. Smilowitz, Atamtürk and Daganzo (2003)[181] formulate a model and solve the problem of moving deferred volume in an express shipment network via aircraft with empty space. That is, they look to reduce overall cost by moving truckable volume via already-scheduled aircraft (whose cost is already committed). Limbourg, Schyn and Laporte (2012)[140] study the problem of loading pallets and containers into cargo aircraft subject to aircraft control and safety constraints with the goal of optimally positioning the center of gravity of the cargo.

4.2.3 ESSND Problem Analysis

Chan and Ponder (1979)[44] study the then-new Federal Express Corporation (FEC) a few years after its start in 1973. They discuss the advantages of FEC's hub-and-spoke topology, as well as the need for smaller regional "mini-hubs." They also discuss management practices at the company, including the use of the Clarke and Wright Algorithm [46] to develop the monthly airline route structure and schedule.

Hall (1989)[107] builds off a previous work, Hall (1985)[106], and analyzes the effect of time zones and overnight restrictions on overnight package air networks. Hall calculates the minimum time between customer drop-off at the end of the day and the latest delivery time the next day for U.S. markets (he calls these "time windows"). He also calculates the minimum sort time for U.S. markets, which is defined as the "time required to sort all shipments if all shipments arrive at once." Hall finds that for a longer sort time, it is beneficial to place the hub east of geographic center in order to take advantage of the time zones. Hall also studies routing strategies for multi-hub topologies in which a package has several sorting location options.

O'Kelly (1998)[161] also addresses hub placement, particularly in the context of "user-attracting" systems, such as passenger airlines, versus "delivery systems" such as cargo airlines. He makes the point that since the customers of cargo airline are not concerned by the aircraft routings in the network (unlike passenger airlines), cargo airline companies are free to place their hub facilities in locations that produce the best routings. He cites this as the reason for the concentration of air hubs in the Ohio Valley.

Continuing on the topic of network topology studies, Bowen (2012)[36] studies the FedEx and UPS domestic and international air networks, particularly in the context of the theoretical modeling work done previously by Hall (1989), O'Kelly (1998), and others. He finds that predictions made in the academic literature have been substantiated in the years since their publications. Additionally, he notes, "[t]he network configurations adopted by FedEx and UPS strike a balance between sorting costs and transportation costs. Their networks also display the importance of time, not only in moving goods from place to place but also in affording the integrator's customers the flexibility to stretch the work-day." Additionally, he finds that despite the addition of secondary hubs, air integrator networks are highly centralized at a major hub; surpassing the centralization of passenger airlines.

Kuby and Gray (1993)[135] introduce a network design problem called the "hub network design problem with stopovers and feeders" and analyze the performance of the network in the context of the, then-named, Federal Express Corporation. They formulate a mixed integer programming (MIP) model for the problem and test it on the western portion of the Federal Express network. Kuby and Gray compare the hub-and-spoke network with stopovers and feeders to the pure hub-and-spoke network. They find significant cost savings for the hub-and-spoke network with stopovers and feeders.

Yang and Kornfeld (2003)[205] study the hub and spoke network in the context of next-day air delivery. They find that the pure hub and spoke network is not always the optimal network structure and sometimes inclusion of direct flights is necessary.

However, their model is simplistic. It only considers three different aircraft types, and there is no limit on the fleet size. They find that solving the whole model is intractable for realistic problem sizes using Cplex so they reduce their analysis to a network of 7 cities (from 26 cities) and solve for the exact solution.

Mason et al. (1997)[155] provides an overview of the use of operations research in the start of FedEx, explaining how operations research was used to make critical decisions in the early days of the company. They tell the story of the analysis behind determining the initially served cities, engine maintenance scheduling, choosing new hubs and network structure, human resources planning, and package tracking. The paper is not mathematically intensive but illustrates the importance of OR analysis in the field.

Similarly, Fleuren et al.(2013) [85] provides an overview of the importance of OR at TNT Express, which at that time was an independent express shipment carrier predominant in Europe. It has since been acquired by FedEx. Fleuren et al. describe the development of the routing and network scheduling analysis, pickup and delivery (i.e. last mile) operations analysis, and supply chain model.

Heinitz and Meincke (2014)[111] present work on an economic problem related to the ESSNDP. In their work, they seek to ensure that scheduled itineraries for air cargo demand will be feasible based on expected cargo flows. A multi-step iterative algorithm is used for the network assignment analysis which determines a load plan for each aircraft flight in the network.

Mahapatra (2005)[152], in his Masters Thesis, develops a MIP model for express shipment companies and performs sensitivity studies for different operational strategies (whether or not to allow intermediate stops) and network configurations (single-hub vs. multi-hub). They find that allowing a single intermediate stop and using regional hubs produces the lowest cost combination for current express shipment company (UPS).

4.3 Solution Approaches for Related Problems

Although the literature focused on the express shipment problem is not extensive, the body of work that focuses on other transportation problems is large, as a whole. Operations research and optimization has been applied to a wide variety of transportation planning and logistics problems, some of which are very similar to the express shipment problem. Some of those problems are presented in this section. Postal delivery, passenger airline operations and less-than-truckload shipping are the most similar to express shipment and receive a more in-depth treatment while others are mentioned in brief.

4.3.1 Relevant Survey Papers

Crainic and Laporte (1997)[61] provide a structured organizational framework for classifying operations research models and problems in terms of the recognized decision-making levels: strategic, tactical and operational. Crainic and Laporte categorize common freight transportation-related OR models such as location planning, network design, multimodal planning, service network design, vehicle routing and crew scheduling, amongst others. They discuss qualities and solution methods for these OR models and problems.

Crainic (2000)[59] provides a literature survey of service network design problem models and related mathematical programming developments, particularly focused on the application to LTL service networks. In a discussion of common solution methods, Crainic particularly highlights branch-and-bound and column generation. Additionally, he presents a classification method for SNDPs and formulations.

Wieberneit (2008)[201] provides a survey of service network design in the context of freight transportation for tactical-level planning. The survey includes sections on the Express Shipment Delivery Problem and the Lettermail Problem; as well as covering work done one LTL operations in North America, multimodal LTL operations in Europe and road LTL operations in Europe.

Gorman et al. (2014)[103] also review the use of OR in freight transportation. They review network design, asset planning, terminal management and load management for oceanic freight, rail intermodal freight, less-than-truckload freight and air freight. Within their coverage of air freight, they focus on network design and management, and revenue management. They point out that while more work has been done on the operations and revenue management of passenger airlines, the problems are distinct.

4.3.2 Lettermail/Postal Package Delivery Problems

Of the real-world applications of network design, the letter mail/postal package delivery problem is the closest to the express shipment problem. The two problems are conceptually nearly identical: pickup and deliver parcels in a quick, efficient manner while respecting time constraints. However, the postal package delivery problem differs from the express shipment problem in the fact that it is typically entirely ground-based, includes last mile service and the selection of sorting facilities may be part of the problem definition. The fact that the operations are ground-based usually simplifies the vehicle decisions because the vehicles are typically all the same type of truck, but the fact that last mile service is included as well as sort facility placement greatly complicates the problem.

Çetiner (2003)[41] and Çetiner, Sepil and Süral (2010)[42] develop an iterative two-stage heuristic to solve the hubbing and routing problem for postal delivery. In the first stage, hubs are determined in the network and in the second stage, routes through these hubs are generated. The two stages iterate in order to produce a feasible good solution. They apply this algorithm to a case study from the Turkish postal delivery system.

Ernst and Krishnamoorthy study the p-hub median problem and apply it to a postal delivery network in a series of papers. Ernst and Krishnamoorthy (1996)[77] present a new LP formulation for the single-allocation p-hub median problem which is smaller than others in the literature. They additionally developed a simulated-annealing-based heuristic which produces good solutions and is used to provide and upper bound for a branch-and-bound method. Ernst and Krishnamoorthy (1998)[78] expands the scope of the problem to the multiple-allocation p-hub median problem and presents a heuristic based on shortest paths to solve it as well as new MILP formulations for the problem. Ernst and Krishnamoorthy (1999) return to their earlier single-allocation p-hub median problem and their approaches for solving it.

Ji and Chen (2007)[120] study the vehicle routing problem in the context of postal operations. They model the problem as an integer program and solve a problem instance from the Hong Kong Post using CPLEX. The model is intended to generate mail routes on a daily basis, a task that is currently performed by hand.

Droździel et al. (2017)[73], in a short study, analyze an existing postal network, identify inefficiencies and, using a MIP, optimize the hub placement to reduce costs. They find that if the Polish Post SA in Lublin province were to move the provincial sort, cost savings could be realized.

A survey paper by Alumur and Kara (2008)[9] focuses on network hub location problems in general but touches on the contributions to the general problem from postal delivery applications. Some of the notable contributions to the network hub location problem are thanks to postal delivery applications, particularly the work of Ernst and Krishnamoorthy.

4.3.3 Passenger Airline Problems

Planning problems for passenger airlines share some similarities to those in express shipment. The most obvious, of course, is the goal to efficiently utilize an often large and expensive aircraft fleet with diverse capacities in a large network. Additionally, passenger airline networks tend to have hub-and-spoke topologies, though they tend to include more point-to-point flights than express shipment. The two problems differ, primarily, in the customer demand and the role that takes in the decision making. In passenger airlines, passenger preference has a huge impact on the scheduling of flights since passenger prefer point-to-point flights at convenient times. If a passenger airline fails to provide these flights, they risk losing those passengers to a competitor. As a result, passenger preference plays a major part in the fleet allocation and scheduling problems in passenger airlines, but this problem is non-existent for cargo airlines. However, passenger airlines do not need to “sort” their passengers as cargo airlines need to do with packages. Passenger airlines only need to make sure that there exists a feasible connection for passengers. Finally, express shipment operates in two distinct periods through the day, night network and day network, with distinct points in time where all the aircraft may be on the ground or flying, whereas passenger airlines operate flights continuously throughout the day.

The literature concerning the application of operations research techniques to passenger airlines is rich, deep and continuously growing. As such, a full survey of passenger airline optimization is out of the scope of this work and only a few works will be mentioned here to give the reader a brief understanding of the type of challenges that face passenger airlines and what is done to confront those challenges. Within the scope of passenger airline operations, fleet allocation, flight scheduling, crew scheduling, and maintenance scheduling are all common considerations (crew and maintenance scheduling have yet to be addressed within the express shipment literature).

Ball et al. (2007)[20] provide an overview of the numerous challenges of passenger airline operations and discuss the ongoing work in the field. They comment that current airlines are needing to shift from a planning outlook that focuses solely on reducing costs to one that considers the robustness of a schedule, particularly in light of the increasing complexity of the air transportation system and difficulty in reacting to unplanned disruptions such as severe weather.

Sherali et al. [179] provide a survey of work on the *fleet assignment problem (FAP)*, which focuses on the assignment of aircraft for passenger airlines. The FAP is, given a flight schedule and a fleet of aircraft, decide which aircraft should fly which flight leg. The important tradeoff in that problem is the desire to fly a full aircraft without missing out on potential customers by choosing too small of an aircraft. Thus the objective is to maximize profits rather than to just minimize cost. Sherali et al. discuss the basic problem as well more detailed problems which involve maintenance scheduling, crew scheduling, ground time windows, through-fleets and re-fleeting flexibility.

Abara (1989) [1] presents and solves an integer programming formulation for the fleet assignment problem faced by American Airlines. The formulation allows for flexibility and customization, and it is used by decision makers to perform a number of strategic studies. The problem is solved using a commercial solver.

Hane et al. [108] provide a solution method for solving a large fleet assignment problem. They explore solving the problem using an interior-point algorithm, dual steepest edge simplex, cost perturbation, model aggregation, set-partitioning constraints and modified branching prioritization. They find that their method finds a solution two orders of magnitude faster than a standard LP-based branch-and-bound.

Barnhart et al (1998)[22] solve the combined fleet assignment problem and aircraft routing problem. The fleet assignment problem is to determine the lowest cost assignment of aircraft types (fleets) to flights. The aircraft routing problem is to assign individual aircraft (according to tail numbers) to sequences of flights. They present a model and solution approach to simultaneously solve the fleet assignment and aircraft routing problem, including maintenance requirements. Since the model is exceptionally large, their solution method is a combination column-generation and constraint-generation approach.

Barnhart, Kniker and Lohatepanont (2002)[23] propose a new, itinerary-based, formulation for the airline fleet assignment problem to address suboptimality introduced by more common problem formulations. Their new formulation is more capable of capturing network effects and by considering those effect, produces better solutions. They solve this formulation using a column generation approach for a real-world problem instance.

Rexing et al. [170] study the optimization of takeoff times in the fleet assignment problem. In order to do this, they discretize the time windows available for takeoff which allows for flexibility in the takeoff time. In order to solve the resulting large MIP, they present two solution approaches. The first is a direct solution technique in which the problem is solved directly after the problem size is reduced using heuristics. The second is an iterative solution technique in which arc copies are iteratively added to the model if they are expected to have a positive impact. The resulting smaller model is then solved.

Rios and Ross [171] optimize the air traffic over the whole Air Traffic System using a highly-parallel implementation of Dantzig-Wolfe decomposition. They find that the greater the number of subproblems, the higher the solution quality, convergence and runtime metrics. Heuristics are used to provide integral solutions, but they note that employing a highly decomposed (one fight per decomposition) improves the integrality of the final solution.

Ahuja et al. (2007)[3] study the combined through-fleet-assignment model for passenger airlines. The through-fleet-assignment model (ctFAM) combines the fleet assignment model with the through assignment model. Through connections are those that have the same fleet type for the incoming and outgoing flight through an intermediate stop. This allows for the same aircraft to be used for both flights, which is preferable for customers. Typically the through assignment problem is solved *after* the fleet assignment problem which means that fleeting assignments cannot be changed to produce better through connections. To solve the ctFAM, a MIP, they employ a very large-scale neighborhood search algorithm and use integer programming to identify neighborhoods. They find the combined approach to produce better solutions compared to the sequential approach.

4.3.4 Less-Than-Truckload Problems

Transportation planning for less-than-truckload (LTL) applications is primarily concerned with load planning and backhauling. Load planning is the routing of a shipment through the network, and backhauling is the desirable situation of carrying freight on both the outbound and inbound legs of a delivery. It is contrasted with repositioning or deadheading which is the movement of empty trailers through the network. Both the load planning and backhauling considerations are important to express shipment airline applications, though under different names. The term “load planning” is typically used in reference to the positioning of containers on an aircraft when applied to express shipment, which is an unnecessary level of detail in this work. However, the question of how to assign packages to routes, and thus to vehicles, still applies. Backhauling is typically the norm in express shipment because of the cyclic nature of the operations, however, because of demand asymmetry, vehicle repositionings are still necessary and an important question.

Although LTL shipping and express shipment share similarities, they differ in several significant ways. First, LTL shipping tends to only be concerned with one type of vehicle, whereas express shipment must deal with a diverse set of capacities and vehicle costs. Second, in general trucks are smaller, more numerous and less expensive (per vehicle) than aircraft so scheduling an LTL fleet requires the scheduling of many small vehicles rather than a few large vehicles. Finally, as with passenger airlines, LTL shipping does not have typically have the strong period nature seen in express shipment operations. As a result, many more legs can be strung together in one trip.

A few papers concerning issues within LTL shipping are outlined below, though this is only a small part of the larger set of literature.

In an early paper on the load planning problem, Powell and Sheffi (1983)[167] study the load planning problem for less-than-truckload (LTL) carriers. They discuss challenges and tradeoffs associated with the problem, develop a model formulation, and present a local improvement heuristic for solving a large real-world problem instance.

Lumsden et al. [144] study the hub and spoke network for European truck freight. They first discuss the proliferation of hub and spoke networks for express shipment carriers with analogies and impacts to trucking. They acknowledge the transportation consolidation benefits of the hub and spoke network, but also identify the advantages of direct delivery in a point to point network. They propose a redesign of a trucking network to represent a hub and spoke network with shortcuts, thus gaining the advantages of both networks.

Erera et al. (2013)[75] study the service network design problem faced by less-than-truckload freight transportation carriers, and present integer programming models and a local search-based solution method. These models route both loaded and empty trailer so they can efficiently move backhaul lanes. Additionally, the solution method can produce flexible load plans as well as traditional repeating load plans. The solution method is an iterative improvement scheme that searches a large neighborhood using integer programs.

Erera et al. (2013)[76], in a different work, additionally tackle the scheduling problem for LTL load plans and introduce a better cost calculation than traditional approaches. The two problems are inter-related since a more precise schedule allows for a better estimate of costs. For a given load plan, their approach creates loaded truck dispatches and driver schedules. They find that their results very closely models real-world operational costs for a large US LTL carrier.

Erera et al. (2009)[74] study the assignment of truck drivers to home locations or domiciles in the LTL network. Domiciles are These decisions are complex, are subject to union rules and can have a major impact on the cost-effectiveness of an LTL carrier. They use the iterative solution method presented in Erera et al. (2013)[75] to make domiciling decisions for thousands of drivers over several days within a few minutes.

4.3.5 Other Transportation Planning & Logistics Problems

There are a plethora of transportation planning applications in the literature. Of those, postal delivery, passenger airlines and LTL load planning are those most applicable to the express shipment problem. However, there are other notable transportation planning problems worth mentioning, though not in such great detail. Those problems are mentioned briefly here.

Railroad Network Design Barnhart, Jin and Vance (2000) study the railroad blocking problem as a network design problem. Railroad blocking is the task of grouping incoming cars with outgoing trains at rail yards. Cars are classified according to their origin-destination pair, but this classification and the reassignment of cars to new trains is a major source of delay in rail operations. The authors propose a network design problem model for the railroad blocking problem to group cars with similar O-D pairs together, subject to rail yard constraints, in order to make the process faster and more reliable. They solve their NDP model with Lagrangian relaxation.

Also on the subject of railroads, Jeong et al. (2007)[119] explores the feasibility and optimality of employing a hub-and-spoke network for railroad freight, a transportation method that does not usually operate using the hub and spoke topology. To represent the additional time freight would spend as a result of the change from the usual point-to-point network, a time-based cost term is also included. They solve the resulting MIP using a heuristic algorithm applied to a large-scale European railroad network. They do not pre-define hubs and, instead, obtain the natural hubs from the optimization output.

Liner Shipping Agarwal and Ergun (2008)[2] solve a scheduling and cargo routing network design problem for liner shipping. They develop and present a mixed integer programming model that simultaneously optimizes the ship-scheduling and cargo-routing problems simultaneously. They explore solving the problem with a number of algorithms, specifically a greedy heuristic, a column generation-based algorithm and a two-phase Benders decomposition-based algorithm. Additionally, an iterative search algorithm is presented to generate ship schedules.

Rental Fleet Sizing Wu and Hartman (2010)[203] solve a large time-space network for a rental fleet sizing problem (RFS). The problem, which features uncertain and period demand, becomes intractable to solve exactly for realistic problem instances. In response, Wu and Hartman decide to use an aggregation technique to reduce the problem size. They aggregate the initial fleet, empty travel time, loaded travel time and demand. This results in a smaller problem which can be solved, at the cost of a lower granularity of information in the final solution.

Repositioning Empty Vehicles Dejax and Crainic (1987)[69] provide a survey paper on the handling of empty capacity flows. The repositioning of empty shipping containers, trucks, railroad cars, etc. is particularly important in environments with strongly asymmetric demand. However, repositioning does not directly generate revenue so it must be implemented intelligently. The problem gets more complex if, as in the railroad industry, there is the option to rent or rent out cars. Additionally, complications can arise from legal and human resource restrictions such as union agreements.

Vehicle Scheduling in Public Transit Bunte and Kliewer (2009)[39] provide an overview of vehicle scheduling models for public transportation networks. They present models for the single-depot and multi-depot vehicle routing problems. Additionally, they compare LP bounds of the different modeling approaches. Finally, they present extensions to the problem in the form of discrete and continuous time windows, and route constraints.

4.3.6 Other Network Design Problems

Magnanti and Wong (1984)[146] provide an extensive review of network design problems in transportation planning. In addition to covering specific instances of the NDP, such as the TSP, VRP, and Steiner trees, Magnanti and Wong also cover optimization techniques for NDPs. First, they note that many solution methods, whether stated or unstated, depend on Benders cuts, and report that many NDPs have been successfully solved using Benders decomposition. Second, they enumerate the many successful uses of branch-and-bound for network design problems along with proposed improvements including the use of Benders cuts. The third and fourth optimization methods they discuss is the use of Lagrangian relaxation and dual ascent procedures. They note, once more, that these methods are closely related to Benders cuts. Magnanti and Wong also cover the common heuristic approaches: add, delete and interchange. These heuristics are used to locally change networks and are particularly useful for large networks in which obtaining an exact optimum is difficult or intractable. Finally, they cover the optimization of network design problems with non-linear routing costs.

Andersen, Crainic and Christiansen (2009)[10] study the service network design problem with asset management (SNDPAM). The SNDPAM arises in problems in which design-balance constraints are included in the SNDP formulation (such as the ESSNDP). They make the case that the impact of these constraints had not been previously intensely studied so the propose four SNDAM formulations and study the effect of them in service network design.

Crainic et al. (2014)[60] study the service network design problem with resource constraints (SNDRC). This is the situation in which resource management (e.g. truck drivers, train operators, power units, specific vehicles) at terminals becomes an important and constraining part of the problem. They find high-quality solutions using an approach combining column generation, slope scaling and the intensification and diversification metaheuristic. They benchmark against an MIP solver and column generation-based heuristic. They report good results for the algorithm in terms of runtime and deviation from the optimum.

Barnhart, Hane and Vance (2000)[21] present two formulations for the origin-destination integer multicommodity flow problem (ODIMCF), an extension of the multicommodity flow problem that is encountered in applications such as package flow problems and bandwidth packing problems. In the ODIMCF, commodities are defined by their origin-destination pair and can only travel along one route from their origin to their destination. They present a column generation model and a branch-and-bound solution approach involving both column and row generation to solve the ODIMCF.

Balakrishnan et al. (1994)[14] and Balakrishnan et al. (1994)[15] study the Two-level Network Design (TLND) problem in the context of a two-level hierarchical (containing primary and secondary nodes and edges) network. They compare the undirected and directed TLNDP problem and present formulations for both. Additionally, they propose two classes of heuristic solution methods, “forward” and “reverse,” for the TLND problem. Finally, they analyze the worst-case performance of several published heuristic methods for the TLND problem and propose a dual ascent heuristic of their own.

In a different paper, Balakrishnan et al. [16] study a similar class of models called overlay optimization problems. In an overlay optimization problem, there are "two sets of decisions: the choice of activity levels to provide a basic level of service to all customers, and decisions regarding which of these activities to enhance to meet more stringent service requirements for subsets of key customers." They analyze the worst-case performance of heuristic strategies for solving overlay optimization problems as well as the gap between the optimal value and linear programming relaxations. Finally, they present worst-case bounds and LP relaxations for the uncapacitated, fixed-charge network design problem.

Magnanti et al. [149] completely characterize the convex hull of the single-facility, multi-commodity network loading problem (NLP) and the three node NLP. Both problems are important for solving large NLPs. For the single-facility, multi-commodity NLP, they use this characterization to develop an LP formulation with a bound at least as strong as the bound given by a Lagrangian relaxation.

Magnanti and Mirchandani [147] study the one, two and three-facility (i.e. vehicle type), single commodity network loading problem. They find that the NLP is strongly NP-hard, but the shortest-path heuristic provides a good solution. They characterize the optimal solution for these problems and identify two new families of facets. Finally, they generalize their results for NLPs with more than three facilities, and the multi-commodity NLP.

Chabrier et al. [43] present their experimental findings on the performance robustness of five common algorithms in solving the NDP. Those algorithms are mixed integer programming (MIP), column generation (CG), a simple constraint programming (CP) algorithm, a more complex constraint programming algorithm, and a genetic algorithm. They analyze the performance of these algorithms using a large industrial data set from a telecommunications company and test the performance robustness by adding and removing constraints to form different specific problems. Each algorithm is tested in its most basic form as well as with common improvement approaches.

They find that no single algorithm outperforms all of the others.

4.4 Summary

The first half of this chapter focused on an overview of how mixed integer programs are solved. Three primary categories were identified. First, non-decomposed exact methods such as cutting plane methods, branch-and-bound and their combination, branch-and-cut. This class of methods will be called *monolithic* approaches for the purposes of this work. Second, decomposed exact methods such as Dantzig-Wolfe decomposition and Benders decomposition. Third, heuristic and metaheuristic methods, which are approximate methods that are typically tailored to the problem at hand.

The second half of this chapter focused on what formulations and solution methods have been previously proposed and implemented for the express shipment problem. The methods and analyses focused on the express shipment problem were discussed comprehensively while related problems were only touched on to show how they related to the focal problem in this work. The next chapter, Ch. 5, will lay out the research plan for this work in detail, outlining the research objective, questions, and hypotheses.

CHAPTER V

RESEARCH PLAN

At this point, the express shipment problem has been detailed and characterized, it has been modeled as a mixed integer program, and approaches to solving MIPs have been discussed. Now that the background and foundation have been set, solution methods for this application of the express shipment problem can be discussed.

5.1 Solution Methods

Recall that in Chapter 2 the express shipment problem was defined as,

> *The problem of determining the lowest cost way to use a heterogeneous, multi-modal vehicle fleet to move a set of express shipment packages from their origin airports to their destination airports within a domestic multi-hub-and-spoke network and within the guaranteed service periods.*

The goal of this dissertation is to develop an algorithm to address that problem computationally. Express shipment is a multi-billion dollar industry today, but the current planning methods are primarily human-based and therefore slow and vulnerable to sub-optimality. Due to these companies' large-scale operations and extensive airlines, even small changes on the tactical level could translate to millions of dollars in cost saving. Additionally, if there existed a computational planning method that was capable of solving the express shipment problem in a short time frame, the strategic-level studies that could be performed could translate to an even greater business impact. Considering the potential impact of a quick and effective computational method for solving the express shipment problem, *The **primary objective** of this work is to develop a quick and effective algorithm that generates low-cost vehicle schedules to completely route one days worth of domestic express shipment packages.*

To this end, in Chapter 3, the express shipment problem was modeled as an extension of the Service Network Design problem called the Express Shipment Service Network Design problem. The ESSND problem captures the express shipment problem mathematically so that computational optimization methods may be applied to it. As outlined in Chapter 4, there are three general classes of solution approaches for solving mixed integer programs like the Express Shipment Service Network Design problem, 1) monolithic, 2) exact decomposition, and 3) heuristic. The primary objective can be restated as a fundamental research question, which is

Research Question 0. *What is an appropriate algorithm to quickly and effectively generate a low-cost vehicle schedule to completely routine route one day's worth of express shipment packages?*

This fundamental research question was explored by implementing, testing and comparing an algorithm from each of the three solution generation approaches laid out in Ch. 4. The first option, a monolithic approach, is often the easiest to implement, particularly if a commercial solver such as Gurobi or Cplex is used. These solvers are highly-sophisticated and are often successful in solving many MIPs despite lacking a problem-specific implementation. However, because these solvers rely on a branch-and-cut algorithm as their foundation, they are susceptible to poor scaling with problem size, particularly if the LP relaxation bounds are weak.

The second option, an exact decomposition approach, attempts to mitigate some of the issues of a monolithic branch-and-bound-based approach by solving only a subset of the problem at any given time. Approaches like Dantzig-Wolfe and Benders decomposition attempt to exploit the fact that only a subset of the variables will have non-zero values in the final solution and only a subset of the constraints will be active.

If these subsets of variables and/or constraints can be efficiently identified, a problem instance, which otherwise may have been difficult to solve or even intractable, can be solved much more quickly. Unfortunately, it is not always easy to find the necessary subset of variables or constraints and many iterations between the master problem and subproblem may be required, thus resulting in long solve times.

Finally, the third option, a heuristic or metaheuristic approach, sacrifices the guarantee of an optimal solution in the interest of a faster runtime. In general, it is difficult to make sweeping statements about heuristic and metaheuristic approaches because the field is broad and implementations are often problem-specific. The main takeaway, however, is that they are often capable of producing very good solutions in a reasonable amount of time by sacrificing the optimality guarantee. For many problems, particularly real-world problems which tend to be large, complex and include problem-specific side constraints, a heuristic approach may be the best way to produce high-quality (or simply feasible) solutions.

When considering these three solution methods, a monolithic approach had already been implemented and tested through a collaboration, of which I was a member, between Georgia Tech and FedEx Services. This monolithic approach uses Gurobi as its branch-and-cut solver along with some custom pre-processing heuristics to make the single-day problem tractable. For realistic problem instances, solving one day's worth of volume for one day (wall time) produces a solution with a 10.8% optimality gap and it requires an additional eleven hours to produce a solution with less than a 10% gap. Problems encompassing more than a single day's worth of volume are hopelessly intractable. This approach will serve as the baseline approach for this dissertation.

The two other solution methods available for consideration are an exact decomposition approach and a heuristic approach. Both of these approaches are commonly employed in the literature when a monolithic approach is found to be insufficient.

Therefore, I hypothesize,

Hypothesis 0. *Based on the evidence in the literature that heuristic and exact decomposition approaches can out-perform monolithic approaches, I hypothesize that a heuristic approach or exact decomposition approach will able to produce solutions for the Express Shipment Service Network Design problem of comparable quality to the baseline monolithic approach in significantly less time.*

This hypothesis is tested by implementing an approach from all three classes of solution methods. The monolithic approach was developed through the aforementioned collaboration with FedEx, but the Benders decomposition and heuristic method were implemented solely by the author. The details of the monolithic approach and sensitivity studies are presented in Chapter 6. As mentioned before, the monolithic approach produces high-quality results, but the runtime to generate those high-quality results is longer than desired, sometimes on the order of *days* of runtime.

The heuristic decomposition and exact decomposition, since they are implemented as part of this dissertation, inspire several more research questions and hypotheses to address those research questions. Of the two approaches, the heuristic decomposition approach is presented first, in Chapter 7, and the exact decomposition approach, Benders decomposition, is presented next, in Chapter 8. Therefore, the first set of research questions will pertain to the development and implementation of the heuristic approach, and the second set of research questions will pertain to the implementation of the Benders decomposition approach. There are a total of six research questions, four which are focused on the heuristic algorithm and two that are focused on the Benders decomposition algorithm. Section 5.2 summarizes the implementation-based research questions, corresponding hypotheses, and important experiments for the heuristic decomposition approach, and Section 5.3 does likewise for the Benders decomposition implementation.

5.2 Heuristic Decomposition-Based Research Questions

The first research question, which is the same for any researcher looking to implement a heuristic approach, is,

Research Question 1. *What should the heuristic algorithm be?*

To help in answering this question, let us make two observations. First, heuristic approaches are successful because they can respond directly to specific problem difficulties and exploit problem-specific characteristics. Second, the monolithic approach is adept at solving smaller problems but struggles to solve realistic problem instances. These observations, particularly the second, suggests that a heuristic *decomposition* approach may be successful. This observation leads to the hypothesis,

Hypothesis 1.1. *Based on the observation that the sensitivity studies performed on the monolithic approach show that smaller problem instances can be much more quickly solved than the full-sized problem instance, I hypothesize that a decomposition-based heuristic will be capable of quickly producing high-quality solutions to the express shipment problem.*

This observation gets us one step closer but still leaves many implementation questions unanswered. For guidance on answering these implementation questions, I turned to my personal background in aerospace engineering. Within aerospace engineering, a common way to tackle large, complex optimization problems, particularly in academic studies, is a technique called *Multidisciplinary Design and Optimization (MDO)*. The primary objective of MDO is to leverage optimization techniques when designing engineering systems with many subsystems in an effort to produce a better overall result than if each of the subsystems were designed independently.

In many regards, MDO is a heuristic decomposition approach. Complex engineering systems, such as aircraft, are made up of subsystems (i.e. aerodynamics, structures, propulsion, controls), which, traditionally, were designed independently.

MDO seeks to intelligently define the disciplinary boundaries, and then optimize those disciplines in conjunction with each other to yield an overall superior aircraft, automobile, spacecraft, etc. By defining disciplinary subproblems, solving them individually (or semi-independently), and recombining them into a feasible solution, MDO fits the format of a decomposition technique. MDO differs from the exact decomposition approaches for mathematical programming in that its architectures do not typically guarantee a globally optimal solution and its architectures are typically tailored toward solving problems that may be non-linear and/or non-convex, feature "black-box" applications or present other challenges.

Since MDO architectures are typically used on problems that behave mathematically different from MIPs, a direct use of an MDO architecture would be misguided and likely ill-fated. However, the high-level strategies that MDO uses to craft solution methods for engineering system design can help to answer research question 1 for the express shipment problem. In particular, guidance can be taken from MDO regarding the definitions of disciplines, the arrangement of those disciplines relative to each other, and the production of a high-quality feasible solution from those disciplines. This leads me to refine my previous hypothesis to be,

Hypothesis 1.2. *An MDO-inspired decomposition-based heuristic will be capable of quickly producing high-quality solutions to the express shipment problem based on the observation that the sensitivity studies performed on the monolithic approach show that smaller problem instances can be much more quickly solved than the full-sized problem instance, and Multidisciplinary Design Optimization is an optimization field focused on developing heuristic decomposition algorithms.*

This hypothesis sets the context for the heuristic decomposition approach. Two major implementation questions still remain. Those questions are, first,

Research Question 2. *How should the disciplines be defined in the heuristic decomposition approach?*

And, second,

Research Question 3. *How should the disciplines be arranged relative to each other in the heuristic decomposition approach?*

The next two subsections consider these research questions in more depth, present hypotheses to address these questions, and summarize the results of notable experiments.

5.2.1 Discipline Definition

First, the question of how to define the disciplines. Ideally, the disciplines would be defined in such that they would represent logical subdivisions within the design space, they could be solved easily and there would few coupling variables between disciplines. In MDO approaches, disciplines tend to be based on physical or functional boundaries of the vehicle, such as aerodynamics, structures, propulsion, etc. and problems are solved in the primal space.

From this observation from the MDO literature, I make the following hypothesis in response to research question 2,

Hypothesis 2.1. *Based on the successful use of functional and physical discipline definitions in the MDO literature, I hypothesize that either a functional or physical discipline definition can be used in a heuristic decomposition approach for the express shipment problem.*

Using MDO practices as guidance, let us define the decomposition subproblems by either physical or functional boundaries. If considering functional boundaries, the natural choice would originate from the two fundamental express shipment questions, "How should the packages be routed?" and "How should the aircraft be scheduled to best move the packages?" These two questions are reflected in the formulation as the package flow and aircraft movement (design) variables, and constitute the heart of the express shipment problem. This discipline definition would result in decomposing the problem into a Package Routing Problem (PRP) and an Aircraft Movement Problem (AMP), and since the disciplines are reflected in the variables, this option will be called the *variable-based decomposition*. Exploratory experiments showed that the PRP and the AMP could be solved very quickly for the baseline problem instance. The primary drawback of this discipline definition is that, although only the forcing constraint is the only constraint that is involved in both of the subproblems, *every* variable would be coupled between the two subproblems.

There are two other candidate options, 1) a *time-based decomposition*, and 2) a *space-based decomposition*. These disciplinary definitions are based on physical properties of the problem, and they are based on breaking up the time-space graph. The time-based (temporal) decomposition exploits the periodic nature of express shipment wherein all the aircraft fly in from the spokes to the hubs then back out again repetitively, all at about the same time throughout the day. Therefore, the problem can be divided according to its phases. The primary drawback of the temporal decomposition is the limit on the smallest temporal "resolution" and the handling of multi-day products. The smallest possible time unit is a half-phase (e.g. night pickup, day delivery, etc.), but this poses serious flow and vehicle balance challenges. Increasing the time unit to a full phase alleviates the flow balance issue partially, but the vehicle balance and some flow balance challenges remain. Unfortunately, exploratory experiments found that even single-cycle problem instances were too large to be solved in a reasonable period of time.

The space-based (spatial) decomposition divides the geographic area of service (which is the continental United States in this dissertation) into regions and then solves each region independently. The most obvious way to define the regions would be according to the current arrangement of sorting hubs. This produces regional-based subproblems that were found to solve very quickly in exploratory experiments.

The primary drawback of the spatial decomposition is that the demand graph is fully connected. Every location connects to every other location, and as a result, routing intra-regional volume is a major hurdle in the spatial decomposition. Additionally, a regional network does not reflect the current practices of express shipment airlines. Express airlines operate multi-hub-and-spoke networks, but the majority of the volume flows through the primary hub, while the regional hubs provide ancillary support.

Of the three potential discipline definitions, the variable-based decomposition is the most appealing because it splits the problem according to its two most fundamental questions, it features subproblems that are easy to solve, and it avoids the vehicle balance, flow balance and hull count challenges of the physical decompositions. Therefore, I hypothesize,

Hypothesis 2.2. *Based on the results of the exploratory experiments conducted on the temporal, spatial, and functional decomposition definitions, and the success of physical and functional discipline definitions in the MDO literature, I hypothesize that a functional decomposition, which features a Package Routing Problem and an Aircraft Movement Problem, will be an appropriate discipline definition for a heuristic decomposition approach for the express shipment problem.*

The next subsection discusses how the Package Routing Problem and Aircraft Movement Problem are arranged relative to each other to produce high-quality feasible solutions to the express shipment problem.

5.2.2 Discipline Arrangement

The next question is how the disciplines should be arranged relative to each other. In hypothesis 2.2, the disciplines were defined as the Package Routing Problem and the Aircraft Movement Problem. Both of these problems depend on the outputs of the other and are therefore coupled. The Package Routing Problem requires the aircraft movements in order to know which arcs have capacity, and the Aircraft Movement Problem needs to know the package routings in order to know where capacity is needed. In MDO, when two disciplines require the outputs of each other, a common approach is to produce a consistent solution using a feedback loop. This feedback loop is often implemented through fixed point iteration. Given that this is a common way to handle coupled disciplines, I hypothesize,

Hypothesis 3.1. *Since the Package Routing Problem and the Aircraft Movement Problem are coupled, I hypothesize that feedback will be required to produce a mutually feasible solution, and, given that fixed point iteration is a common way to handle coupled disciplines in the MDO literature, the Package Routing Problem and Aircraft Movement Problem should be arranged in a fixed point iteration scheme in the heuristic decomposition approach for the express shipment problem.*

However, when the fixed point iteration (FPI) scheme was implemented and tested, the solutions it converged to were not very good. As a consequence, hypothesis 3.1 was rejected. The reason why the FPI-based scheme was unsuccessful is because the decision of how to use the aircraft was completely removed from the decision of how route packages. As a result, trade-offs such as assigning a longer package routing in order to group more packages on a larger aircraft were never considered. In order to intelligently use the aircraft, package routings *must* take into account the aircraft that will be required to move them. This finding led to the alteration of the Package Routing Problem to include the aircraft movement variables, the constraint requiring them to be positive integers, and the feeder limit constraint. The resulting problem formulation was called the Package Routing with Aircraft Problem (PRAP).

The Package Routing with Aircraft Problem, though more difficult to solve, incentivized the assignment of package routings such that *whole* real-world aircraft would be considered and packed as efficiently as possible. With this alteration in the Package Routing Problem, the role of the aircraft movement feedback in the FPI scheme was redundant with the aircraft movement variables in the PRAP. In response to this change, I hypothesize,

Hypothesis 3.2. *Since the addition of the aircraft movement variables to the Package Routing Problem adds aircraft-based considerations, I hypothesize that feedback is* not *necessary between the Package Routing with Aircraft Problem and the Aircraft Movement Problem, and a* sequential *algorithm that first solves the PRAP and then the AMP will provide high-quality solutions.*

The development of the PRAP and the hypothesis that a sequential heuristic algorithm would be sufficient led to the development of the *Basic Sequential Algorithm.* The Basic Sequential Algorithm was found to out-perform the monolithic approach for the baseline single-day problem instance for the first hour of runtime and perform on-par with the monolithic approach for runtimes beyond one hour. This performance lent experimental support to hypothesis 1.2, 2.2, and 3.2. It also led to the follow up research question of how to potentially refine or improve the sequential algorithm. That question is handled in the next subsection.

5.2.3 Improvements to the Sequential Algorithm

The success of the Basic Sequential Algorithm raises the question,

Research Question 4. *How can the Basic Sequential Algorithm be improved?*

In response to research question 4, two avenues of improvement were considered for the Basic Sequential Algorithm. Both avenues focus on improving the Package Routing with Aircraft Problem, because, as the first of the two disciplines in the sequential algorithm, it is ultimately responsible for the final solution quality. The first of the two avenues was to reduce the size of the PRAP so that it could be solved more quickly. The second of the two avenues was to improve the approximation that the PRAP makes of the ESSND by including additional constraints without sacrificing runtime.

Starting with the first of the two algorithm improvement avenues, the observation was made that many commodities are very similar to each other and for a given origin-destination pair, there are often many commodities that differ only in the product type or number of commit days (i.e. number of days the company has to move the package). Furthermore, the overnight priority box commodities constitute the most volume of all of the commodities and are the most-tightly time and sort-capacity constrained. Using these two observations, I make the hypothesis:

Hypothesis 4.1. *Based on the observations that 1) in an express shipment problem instance there exist many commodities that may only differ by the product type and/or commit day, and 2) of these similar commodities, the priority box commodities constitute the largest volume, face the tightest sort capacities, and are the most time-constrained, I hypothesize that the Basic Sequential Algorithm can be improved by solving the Package Routing with Aircraft Problem for the problem instance that only includes the overnight box volume, and assigning the routings for all commodities based on the routing assigned for the analogous priority overnight commodity.*

This hypothesis leads to the *Collapsed Product Set Sequential Algorithm*, which uses the idea of "analogous commodities" to shrink the PRAP problem instance. Analogous commodities are those with the same origin and destination but different product types or commit days. Within the ESSND problem, each commodity gets its own variable, but only some commodities are the primary drivers of aircraft movement decisions. Of the product types, box volume, particularly overnight box volume constitutes the greatest demand. The Collapsed Product Set Sequential Algorithm seeks to exploit this quality of the express shipment problem and shrink the PRAP problem instance to only that of the overnight priority box volume. The routings assigned those priority box commodities then serve as a "template" for all of their analogous commodities. To differentiate between the PRAP that is solved for all of the commodities and the PRAP that is solved for the collapsed product set, the latter is called the *Collapsed Product Set Package Routing with Aircraft Problem* (CP-PRAP). To further shrink the PRAP, consolidation is also removed. Consolidation is added to the final solution after identifying suitable opportunities from the solution of the AMP by solving the *Consolidation Assignment Problem*.

When tested experimentally, the Collapsed Product Set Sequential Algorithm performed better than the monolithic approach for runtimes less than one hour, but it was not superior to the Basic Sequential Algorithm for the baseline problem instance. However, in the sensitivity studies, it was found that the Collapsed Product Set Sequential Algorithm does not suffer from the same runtime growth issues and can solve a full week's worth of demand, a previously intractable problem, in under four hours if consolidation is not considered, and in under sixteen hours if it is.

The other avenue for improving the Basic Sequential Algorithm was to improve the PRAP by including more constraints so it would better approximate the ESSND problem formulation. The primary motivating observation for this was the fact that while the PRAP does consider whole aircraft as it assigns package routings, since it does not include hull count or balance constraints, it assigned package routings to use significantly more of certain aircraft than were available in the fleet. The prompted the hypothesis,

Hypothesis 4.2. *Based on the observation that the PRAP assigns package routings with little bearing on the available aircraft fleet, I hypothesize that including a hull count constraint to the PRAP would improve the performance of the sequential algorithm.*

The original hull count constraint could not be used to improve the PRAP because it was dependent on the burdensome vehicle balance constraint, so the phase-based hull count constraint was developed. This new hull count constraint was included in the PRAP to form the *Improved Sequential Algorithm*. This algorithm was found to be significantly superior to not only the Basic Sequential Algorithm but also to the monolithic approach. In one hour, it produced a better solution than the monolithic approach was able to find in two days for the baseline problem instance; a substantial performance improvement.

The performance of the Improved Sequential Algorithm also inspired the hypothesis,

Hypothesis 4.3. *Based on the observation that adding the phase-based hull count constraint to the PRAP greatly improved the sequential algorithm, I hypothesize that including a modified vehicle balance constraint to the PRAP will further improve the performance of the sequential algorithm by adding additional realism to the PRAP.*

A relaxed form of the vehicle balance constraint was developed, called the sort-centric balance constraint, was added to the Package Routing with Aircraft Problem. However, it was found to hinder performance rather than improve it. As a result, the Improved Sequential Algorithm includes the phase-based hull count constraint but not the sort-centric balance constraint. Sensitivity studies show that the Improved Sequential Algorithm out-performs the monolithic approach for many different problem instances, including those that are more difficult and those that are less difficult to solve than the baseline problem instance.

5.3 Benders Decomposition-Based Research Questions

Unlike the heuristic decomposition approach, the many of major details of the Benders decomposition implementation are already established through the work of Benders (1962)[27] and those that expanded on his work. However, two research questions still arise. The first is,

Research Question 5. *Can Benders decomposition be used to solve the Express Shipment Service Network Design problem faster than the baseline monolithic approach?*

My hypothesis in response to research question 5 is,

Hypothesis 5. *Based on the fact that Benders decomposition was specially developed to solved mixed integer programs, and it is commonly and successfully used for that purpose in the literature for similar transportation planning problems, I hypothesize that Benders decomposition will be capable of solving the Express Shipment Service Network Design problem faster than the baseline monolithic approach.*

Although the literature suggests that Benders decomposition may be suitable for the express shipment problem, it also notes that often researchers need to alter the "classical" implementation of Benders decomposition to get the desired performance. Since Benders decomposition is currently a popular approach and several decades have passed since it was first presented, the body of literature encompassing improvements to Benders decomposition is extensive. This leads to the research question,

Research Question 6. *Which, if any, enhancements to Benders decomposition from the literature improve its performance when solving the Express Shipment Service Network Design problem?*

To answer this question, the literature was consulted. By using the classification scheme from the Rahmaniani (2017)[168] survey paper, knowing certain features of the express shipment problem, and identifying popular current approaches, the extensive list of potential enhancements to Benders decomposition was reduced to four items. This led to the hypothesis,

Hypothesis 6. *Based on this literature review of previously-developed methods for improving Benders decomposition, I hypothesize that implementing a single-tree Benders decomposition algorithm (i.e. improving the solution procedure), including the cut-set valid inequalities developed by Kim (1997)[129] (i.e. improving the solution generation), and implementing an approximate version of the Pareto-optimal Benders* *cuts from Papadakos (2008)[163] or the minimal infeasible subsystem (MIS) Benders cuts from Fischetti et al. (2010)[82] (i.e. improving the cut generation) will improve the performance of the Benders decomposition algorithm in solving the express shipment problem.*

When tested, however, Benders decomposition, even with improvements, was found to be inferior to the monolithic approach. Therefore, hypothesis 5 is rejected. It's primary handicap is that it cannot be prematurely terminated and as a result, if it struggles to find an optimal solution as was the case for the ESSND, no solution will be generated. However, some lessons were learned regarding the enhancement approaches for Benders decomposition. Of those implemented, the minimal infeasible subsystem (MIS) cuts by Fischetti et al. (2010)[82] provided the most improvement compared to the "basic" Benders implementation. The valid inequalities and the approximate Pareto-optimal cuts were actually found to hamper the progress of the algorithm. No alternative to the single-tree solution procedure was studied so it's effect is unknown, but it is currently a very popular and well-regarded method for implementing Benders decomposition on modern solvers.

5.4 Experimental Testbed

All of the experiments in this dissertation are performed on the same experimental testbed so that results can be compared across the three different solution generation approaches. In this testbed, the method for generating the package routings and aircraft movements is the same for each solution generation technique. This ensures that for a given problem instance, each solution generation technique is given the same set of options. The technique for generating these options is described in Chapter 6. Within the experimental testbed, experiments can be performed by modifying (filtering) the data set that is used to generate the options and/or altering the solution generation approach. The experimental testbed is summarized in Figure 9.

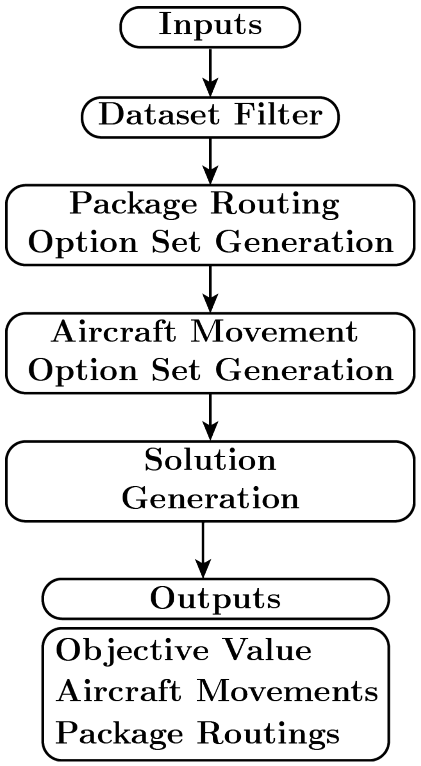


Figure 9: Experimental Testbed

First, the input files, which fully describe the problem as characterized in Chapter 2, are parsed into the testbed. They include important data such as the facility locations, the activity times, fleet composition and properties, and the commodity demand.

Second, the dataset is filtered to define the problem instance for the experiment. Since this dissertation does not consider the added complications of international volume, and unusual products such as live animals and three-day packages, that demand (which constitutes a small fraction of the total volume) is removed in the data filter process. Additionally, since the baseline problem instance only encompasses a twenty-four hour (Monday night to Tuesday night) period, any volume outside of that time period is also removed. If relevant to the experiment, additional filtering may be performed to test a smaller problem instance such as the reduced airport set and reduced product set sensitivity studies performed on the monolithic approach in

Chapter 6, or problems with a larger temporal scope may be considered by expanding the time period to additional days.

Third, the package routings and aircraft movement options are generated as described in sections 6.2 & 6.1 of Chapter 6. For all of the experiments in this dissertation, the package routing and aircraft movement options are always generated in the same fashion. This ensures that each solution strategy, whether it be monolithic, exact decomposition or heuristic, is presented with the exact same option set given a particular problem instance, and the results are purely due to the solution strategy itself.

Fourth, solutions are generated using a monolithic, heuristic decomposition or exact decomposition approach. When a problem instance is solved in a purely monolithic approach, the full problem instance is given to a commercial MIP optimization software (Gurobi 7.5) and solved for a given period of time or until completion. For the heuristic decomposition solution generation approaches, in this dissertation, all of the approaches considered and developed all use mixed integer programming subproblems which are relaxations of the original ESSND problem formulation. These MIP subproblems are also solved using Gurobi 7.5 and different arrangements of these subproblems are studied. Finally, for the exact decomposition solution generation approach, Benders decomposition, which is composed of a master problem IP and a subproblem LP, is implemented. These two problems are solved using CPLEX 12.7, a different commercial MIP solver.

Finally, each solution generation method produces an objective function value, package routings, an aircraft schedule, and an optimality gap. If an optimal solution is found, then the optimality gap will be zero, otherwise, it will be the optimality gap at the time of termination. The optimality gap for the heuristic algorithms are calculated using on the *best-found* lower bound value from the equivalent monolithic problem instance.

The dataset filter and the solution generation are the only procedures that are altered for experiments, and often either one or the other is altered, not both. That is, either the dataset is filtered to study an different-size problem instance or the solution generation procedure is altered. Since the generation of the optimization model itself is considered part of the solution generation procedure for the purposes of the diagram in Figure 9, an alteration to the solution generation procedure could constitute either a change to the optimization model (i.e. a relaxation) or the use of a heuristic or exact decomposition method. The inputs, package routing option generation, aircraft movement option generation, and types of outputs are the same in every experiment.

5.5 Summary

In this chapter an overview of the research questions, hypothesis and experiments were presented. The primary objective of this work is to develop a quick and effective algorithm that generates low-cost vehicle schedules to completely route one days worth of domestic express shipment packages. To that end, approaches within each of the three solution method classes were discussed – monolithic using Gurobi, a heuristic decomposition, and Benders decomposition. The monolithic approach is taken to be the baseline and the fundamental hypothesis is that either a heuristic decomposition or exact decomposition approach will be able to out-perform it.

Cross-disciplinary inspiration is taken from Multidisciplinary Design Optimization, a field mostly-commonly used in aerospace and mechanical engineering, to provide guidance on certain implementation questions for the heuristic approach. Although the results of initial experiments led to hypothesis 3.1 being reposed and the refinement of the PRAP, ultimately a heuristic capable of solving the baseline single-day problem instance significantly faster than the monolithic approach was developed.

The next chapter, Ch. 6, will outline the monolithic approach and the results of some sensitivity studies related to how its performance scales with problem size. Following that, Ch. 7 will describe the development of the heuristic approaches in detail. Finally, Ch. 8 will cover the Benders decomposition implementation and results.

CHAPTER VI

SOLVING THE ESSND PROBLEM USING A MONOLITHIC APPROACH

As shown in Chapter 3, the express shipment problem can be formulated as a variant of the service network design problem (SNDP) called the express shipment service network design problem (ESSND). The primary research objective of this work, as presented in Chapter 5, is to develop an algorithm that quickly and effectively solves the ESSND problem for a single day's worth of volume. In Chapter 4, three classes of methods for solving mixed integer programs like the ESSND were discussed – monolithic, exact decomposition and heuristic. The implementation of a monolithic approach is presented in this chapter as well as sensitivity studies that motivate the consideration of alternative approaches.

First, the creation of the time-space graph is described in Section 6.1. Next, heuristics are described which dramatically reduce the size of the optimization problem, and make the single-day problem computationally tractable for a monolithic approach. This reduced-size problem is solved using the commercial Gurobi optimization software. Finally, the results of solving an ESSND problem with this reduced option space are presented and some sensitivity experiments are performed.

This specific monolithic approach, including the means of creating the time-space graph and the problem-reduction heuristics, was developed as part of a multi-year collaboration between the Georgia Tech and FedEx. I had the privilege of being a member of the Georgia Tech team for several years, but cannot claim this monolithic approach to be my own – it was a team effort. The exact decomposition approach (Bender decomposition) presented in Chapter 8 and the heuristic MDAO-inspired decomposition approach presented in Chapter 7 were implemented solely by the author using the framework (i.e. time-space graph and commodity routing creation algorithms) built for this monolithic approach. This monolithic approach will primarily serve as the non-decomposition benchmark as well as the original motivation for pursuing alternative solution methods.

6.1 Creation of the Time-Space Graph

Time-space graphs were briefly introduced in Section 3.3.1 to handle the time-related aspects of the express shipment service network design problem. In a time-space graph, the spatial graph is "expanded" by discretizing time and representing space *and time* at each node in the graph. Each arc then represents connections between points in space that can be feasibly connected in space *and time.* In the basic implementation, time is evenly discretized according to a uniform time step, and a node at each time step and point in space is created. When the chosen time step is large and the number of locations is small, this approach is manageable, but when the problem necessitates a small time step, this strategy results in a dramatic increase in problem size.

6.1.1 Network Creation in the Literature

In many applications, many of the nodes and arcs are unnecessary and this dramatic increase in problem size is problematic, so more nuanced ways of generating the time-space graph are commonly used when considering a large time-space graph. These methods create a smaller time-space graph by having different time steps throughout the graph, as appropriate. For instance, if it is known that a connection may occur within a 30 minute time window, then inside that time window it may be important to have a small time discretization, but outside of that time window, the discretization can be very coarse because nothing will be happening. To identify the appropriate time step in different parts of the time-space graph, network preprocessing is employed to reduce the node set. The problem of generating the necessary node set has been approached in two ways, 1) network refinement, and 2) network reduction.

Network refinement starts with a coarse time discretization and selectively decreases the time step in parts of the time-space network that require more granularity. Boland et al. (2017) introduce an approach for solving the Continuous-Time Service Network Design Problem (CTSNDP) which uses an iterative refinement algorithm to appropriately split nodes in a time-space network, thus selectively decreasing the time step without requiring the same level of detail throughout the entire network.[33] Network reduction is a more common approach. It starts with a highly granular network and selectively consolidates unnecessary nodes and their related arcs. Some implementations of network reduction have been applied by Berge and Hopperstad (1993)[28], Hane et al. (1995)[108] and Kim and Barnhart (1999)[132].

6.1.2 Network Creation for the Express Shipment Problem

Within the express shipment problem, it is important to know, to a fine degree of detail, when aircraft take off and when they land, so a small time step is necessary during those times. However, at many locations there may be hours during which no aircraft will be taking off or landing so it is not necessary to provide fine time granularity during those times. Therefore, a reduced time-space network is necessary.

Thanks to the specific qualities of the express shipment problem, neither the network refinement nor network reduction techniques need to be undertaken as they are outlined in their respective sources. Instead, the time-space network can be generated by building it essentially *backwards*. First, the package routings[1] are created, then the vehicle arcs are created based on the needs of the package routings, and finally, nodes are created based on the times required by the arcs. This is possible because of some features of the express shipment problem,

[1]A package routing is synonymous with a commodity routing.

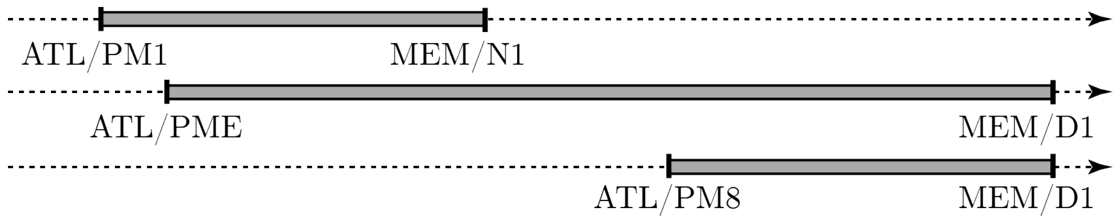
1. Operations are strongly periodic, which means that similar activities happen at similar times.
2. Commodities have fixed times when they are available and when they are due.
3. It is not important to know *precisely* when a commodity arrives to or departs from a location as long as the movement respects the level-of-service restrictions.
4. It is important that aircraft movements be precisely timed so that potential connections are not missed.

These qualities, particularly number 3, mean that it is possible to know *approximately* when greater resolution is needed in the time-space network and when a coarse discretization is permissible. To make it more clear how this works, let us go into more detail.

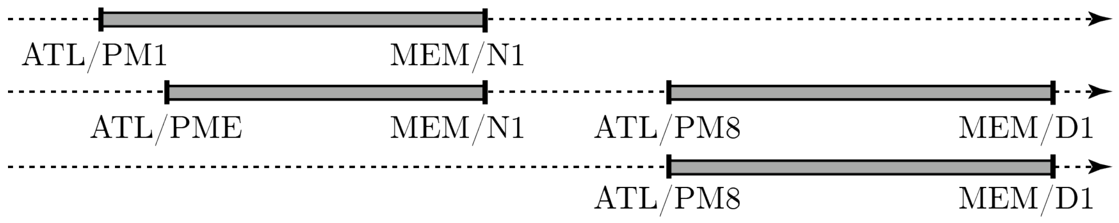
First, the package routings are created. This process is detailed in Section 6.2, but for now it is sufficient to know that a package routing is defined as the unique pairing of a commodity and an ordered list of *lanes*. A *lane* is an ordered pair of activity instances. For example, a priority package originating in Atlanta and terminating in Seattle on the fifteenth day of the month would have the routing: {(*ATL/PM1[15]*, *MEM/N1[15]*), (*MEM/N1[15]*, *SEA/AM1[16]*)} where *PM1* is the evening priority pick up activity, *N1* is the night sort activity and *AM1* is the morning priority delivery activity. Each commodity can have several routings. Additional routings could send the commodity through a different sorting hub or make an intermediate stop.

The concept of a lane is important because some lanes can be grouped together if it is possible to move all of their packages at the same time without delivering any of them late. This situation arises when handling the economy volume. For example, both priority and economy volume originate in the evening, but priority volume is available earlier. If the priority volume can wait to depart until the economy volume is available without being late for its next activity, the two lanes can be grouped.

A group of one or more lanes that can feasibly be moved together is called a



(a) Lanes for Overnight, Deferred, and Same-Day Products Prior to Splitting



(b) Lanes After Splitting Lane for Deferred Products

Figure 10: An Example of Lane Splitting for Deferred Products

package arc. For pick up phase package arcs, the destination activity instances for all of the lanes are the same, and for delivery phase package arcs, the origin activity instances for all of the arcs are the same. This means that for economy products, which have more time to move than priority and postal products, ground lanes are added to their routings so that the appropriate activity instances will match with their faster-moving counterparts.

Extending the previous example, the economy volume will be available at ATL slightly later than the priority volume and is intended to sort during the day. If time allows, it could move with the priority volume to the sort and wait there for the day sort to occur, or it could wait on the ground at ATL to move with the day-time postal volume. Therefore, as illustrated in Figure 10, the lane (*ATL/PME[15]*, *MEM/D1[15]*) is split into the routing segments {(*ATL/PME[15]*, *MEM/N1[15]*), (*MEM/N1[15]*, *MEM/D1[16]*)} and {(*ATL/PME[15]*, *ATL/PM8[16]*), (*ATL/PM8[16]*, *MEM/D1[16]*)} to represent these two options, where *D1* is the day sort activity and *PM8* is the *morning* postal volume pick up activity. Using the ground lanes, the flight lanes can be grouped with priority and postal volume into package arcs.

Once the package arcs have been created, vehicle arcs associated with the package arcs are created in order to provide options for moving the volume. The vehicle arcs are created by determining the appropriate takeoff time for each feasible vehicle type and each package arc. There are various ways this takeoff slot could be determined, but in this implementation, the takeoff time is chosen such that it splits the difference between the earliest possible time and the latest possible time. This means that there will be an equal amount of time between the latest package arc available time and the vehicle arc departure time as there is between the vehicle arc arrival time and the earliest package arc due time. An example is shown in Figure 11 for a pick up phase package arc with two lanes.

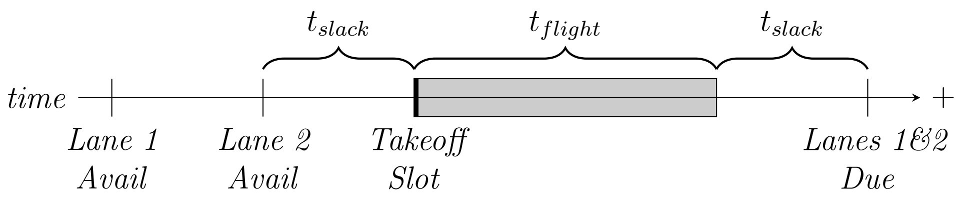


Figure 11: Example Takeoff Slot Calculation

Once the vehicle's takeoff time is set, the vehicle arc's arrival time is calculated using the sum of the vehicle's transit time and ground time[2]. Since the ground time is included in the vehicle "arrival" time calculation, the arrival time actually represents when the vehicle can first leave the destination location. Additionally, takeoff slots for repositioning arcs are added to each location at six-hour intervals that align with the times when package-carrying aircraft will be flying.

Now that all of the arrival and departure times for a location are known, the nodes are created by splitting the location timeline according to the arrival and departure times required by the vehicle arcs. Once these nodes have been created, they are associated with the appropriate vehicle arcs. The only stipulation on the node size is

[2]A vehicle's ground time is the minimum amount of time that an aircraft has to spend on the ground for refueling, unloading and loading, crew change, etc. before it can take off again. Trucks do not have a ground time.

that they be at least one minute long. There is no maximum length. Finally, after all of the nodes are created and associated with the vehicle arcs, ground arcs are created between the nodes, and the time-space graph has been fully created. This results in a time-space graph like the notional example shown in Figure 12 where locations A and B are spoke airport and location C is a hub airport.

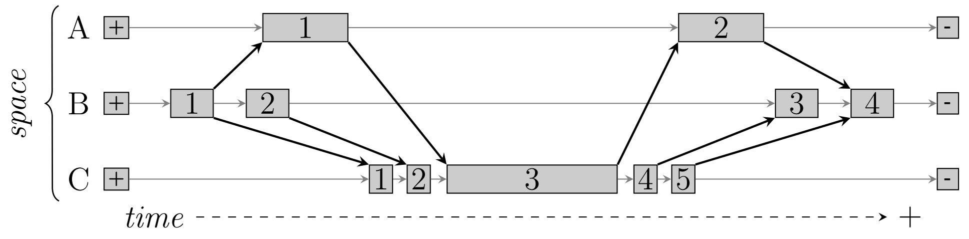


Figure 12: Notional Time Space Graph for ESSND Problem

This discussion of package arcs and vehicle arcs may raise the question, "which are the arcs of the time-space graph according to the mathematical formulation?" The answer is *both*, depending on the application. The exact definition of an arc in the ESSND problem was intentionally left a little loose in Chapter 3 so parallels to the problem heritage could be seen. This murkiness arises from the fact that different vehicles have different transit times between the same two locations.

In the forcing constraint, the arcs are best thought of as package arcs, because what is important is that all of the packages can be feasibly carried by the vehicles. If all vehicle types are feasible, small variations in time are inconsequential and so the vehicle arcs should be handled on an aggregate level and the constraint should exist for all *package* arcs. However, for all of the other constraints involving arcs, such as the hull balance, hull count and feeder limit constraints, vehicles are being tracked through the network so it does need to be applied on a *vehicle* arc level. Of the two arc types, the package arcs most closely align with the intent of the time-space arcs but the vehicle arcs are necessary to accommodate vehicular variations. This creates a small notation issue, but the intention is typically obvious from context.

6.2 Defining Commodity Routing Options

In the *arc-path* formulation, such as the ESSND problem formulation, commodity routings are defined explicitly before the optimization. In a pure *node-arc* formulation, any feasible routing is an option, but the burden of ensuring the feasibility of a routing falls to the optimizer. This increased burden can increase runtime and may result in modeling difficulties if feasibility rules are non-linear. The arc-path formulation ensures that all package routings are time- and equipment-feasible, but the disadvantage of the arc-path formulation is that enumerating every possible routing is computationally unappealing and results in many more variables. Fortunately, enumerating the full set of feasible options is rarely necessary, and only a subset of the options needs to be created. Failing to enumerate all possible commodity options does run the risk of missing the true optimal solution, but the magnitude of this risk is largely determined by how aggressive and sophisticated the approach for cutting down the option space is. Ultimately, it comes down to a trade-off between an aversion to excluding the optimal solution and runtime or tractability. In the express shipment problem, which is otherwise intractable when given the full option space for even a day, the risk omitting the true optimum is acceptable in order to produce a solution.

As mentioned in Section 6.1, a package routing is an ordered sequence of lanes, which are, themselves, ordered pairs of activity instances. Section 6.1 also introduced the concept of a package arc, which is a grouping of one or more similarly-timed lanes. For this implementation, rules for generating routings were based on a few observations of the system and some corresponding assumptions,

1. All origin and destination activities are timed such that there exists a feasible routing through a sorting activity at the primary hub.
2. Only the primary sort connects to all markets, therefore all commodities must have at least one routing through the primary sort.

3. Only the primary sort connects to all markets, therefore small markets, particularly those which could not fill more than one small trunk aircraft, should only route through the main sort.
4. Same-day commodities, which are picked up in the morning and delivered in the evening of the same calendar day, only have the option to sort during the day.
5. Overnight commodities, which are picked up in the evening and delivered on the morning of the next calendar day, only have the option to sort during the night.
6. The night cycle is more heavily constrained in terms of time and sorting capacity than the day cycle, therefore two-day commodities, which could otherwise feasibly be sorted during the night or day sorting activity, must sort during the day.

These observations and corresponding rules provide the basis for how routings are created. First, it is necessary that all commodities have at least one routing. Since only the primary hub is the only hub that connects to all spoke airports, appropriate routings are generated through the primary hub. To do this, first, all commodities that can sort during the day (i.e. postal and economy volume) are given a routing through the daytime sort activity instance. The remaining commodities (i.e. priority) are given a routing through the primary nighttime sorting activity instance. This ensures that all commodities have at least one routing.

Next, all of the commodities from large markets are given routings through all secondary sorting facilities that are time-feasible with respect to pick up and delivery times. It is assumed that the best use of secondary sorts will be from markets that require (or could require) two aircraft to serve it. Large airports are defined as those that originate or terminate 10,000 cubic feet or more of volume on either the night or day cycle for one day.

Finally, after secondary sort-based routings are generated for large markets, consolidation routings are generated for small markets. Small markets are defined as those that originate less than 20,000 cubic feet of volume in a phase. Consolidation is subject to several logical rules,

1. Both the origin and the intermediate location must connect to the same sort.
2. Spokes that can truck to a sort cannot consolidate at an intermediate location.
3. Consolidation cannot occur at locations that can truck to a sort because those locations will not fly an aircraft.
4. Consolidation cannot occur between two locations that would require a feeder flight longer than two hours.
5. Consolidation cannot occur at an intermediate location that originates less volume than the spoke.
6. Consolidation cannot occur at a hub, else it would be equivalent to arriving early to a sort activity and therefore useless.

These rules are used to create a set of consolidation options for each small market. In an effort to keep the option space small, only one of the consolidation options is retained for each consolidation-eligible spoke, the spatially closest, and these routings are added to the set of package routings.

After all of the commodity routings have been generated, they are assigned a flow variable, x_p^k. In the mathematical formulation, the flow variables are continuous on the range $[0, 1]$, but based on experimental results, Gurobi solves the fully integral problem, where $x_p^k \in \mathbb{B}$, faster than the mixed integer program. Therefore, the flow variables are implemented as a binary variable. This choice ultimately makes sense for practical reasons as well since it is preferable to express shipment carriers to move all of a commodity together rather than splitting it up.

6.3 Generating Vehicle Movement Options

Each vehicle arc (i, j) represents a vehicle movement decision variable, y_{ij}^{f}, in the ESSND formulation. In the mathematical formulation, those decision variables are defined as non-negative integers, with no upper bound. That variable definition is sufficient for the mathematical formulation, but in practical implementations it is beneficial to have tighter bounds on the variables to help contain the branch-and-bound tree. Since vehicle arcs are associated with package arcs, the maximum possible flow on the arc is known. This maximum flow is used to determine the bounds on each design variable, which are

$$0 \leq y_{ij}^{f} \leq \left\lceil \frac{b_{ij}^{max}}{u^{f}} \right\rceil \tag{6.1}$$

where b_{ij}^{max} is the maximum possible volume that could flow on the arc (i, j) and u_f is the capacity of vehicle type f. Even if the ratio b_{ij}^{max}/u^{f} is small, the design variable is included in the model because it may be necessary to use a larger aircraft to serve an arc due to the hull count and hull balance constraints. Repositioning arcs, which do not have any associated volume are restricted to a single aircraft, and ground arcs which need to be able to accommodate many aircraft on the ground at one time (particularly at hub facilities during the sort activities) are given a higher upper bound.

Additionally, there are a couple details in the vehicle arc creation that were omitted in Section 6.1 to avoid distractions in the time-space network creation discussion. First, a non-physical, rule-breaking vehicle type is included in the equipment set to aid the optimizer in finding feasible solutions. This vehicle is the size and speed of a 757 aircraft, it does not obey hull balance or hull count constraints, and it is very expensive. Since it is a rule breaker, it will be called a “pirate ship” for some fun. Pirate ships are treated the same as any other vehicles for the purposes of vehicle arc creation. In practical terms, it is most akin to buying belly space on passenger aircraft, which is referred to as “interline.” These vehicles are so expensive that they are soon removed after a solution is found, but they help the optimizer start the B&B tree which results in a faster overall runtime.

Second, if any package arc can be feasibly moved via truck, then only a truck vehicle arc is created. This forces any lane that can truck to truck. This is done because trucks are much less expensive and much more numerous than aircraft. It is assumed that any trucking option will be cheaper than any flying option, and since FedEx has so many trucks, there will be a truck wherever one is needed.

6.4 Results

Once the commodity routings and time-space graph, and their associated variables, are created the resulting integer program is solved using the state-of-the-art commercial optimization solver Gurobi 7.5. This section outlines the computational testbed used for all experiments in this work and outlines the baseline problem instance. Solutions for the baseline problem instance are generated using the option-reduction heuristics and without them. In comparing the performance of these two cases, the use of these heuristics are justified and some summarizing metrics are tabulated for the baseline solution so that it may be compared to other, similar problems and solution methods.

6.4.1 Computational Testbed

Ample computational power, in terms of the current state of the art, was used to run the experiments of this dissertation. A summary of the hardware and software properties of the computational testbed are found in Table 5. These properties are used for all experiments in this work unless otherwise noted.

Table 5: Hardware and Software Properties of Computational Testbed

Property	Specification
Processor	Dual CPU Intel Xeon E5-2680 V3
Number of Cores	12 cores per CPU die (24 cores total)
Processor Speed	2.5 GHz
RAM Size	240 GB
RAM Speed	2133 MHz
Operating System	Red Hat Enterprise Linux 6.7
Optimization Software	Gurobi 7.5.1
Optimization Settings	48 hr time limit, default otherwise
Implementation Language	Java 8

6.4.2 Baseline Problem Instance

The baseline problem instance used in this work is summarized in Table 6. This

Table 6: Baseline Problem Instance

Property	Value
Number of Airports	115 (all)
Number of Product Types	15 (all)
Number of Commodities	74,660
Temporal Scope	1 day (Mon to Tues)
Consolidation Options	1 per eligible spoke
Secondary Sort Threshold	10,000 lbs

problem instance includes all of the domestic airports and products described in Section 2.7. It encompasses one day's worth of volume but spans two calendar days. Specifically, it includes all of the volume that is picked up on Monday evening and Tuesday morning. With the operations of the carrier, this is a more natural day definition than a calendar day. The settings for the problem-reduction heuristics are the same as described in Section 6.2. It is important to note that this baseline problem instance is the baseline in large part because it is the largest useful problem that can be solved with the monolithic approach. Realistically, a full-week problem instance would be more useful for an express shipment company because the weekend provides a more natural break than a day, but comparing one intractable result to another does not lend itself well to analysis. Therefore, a single day will be taken

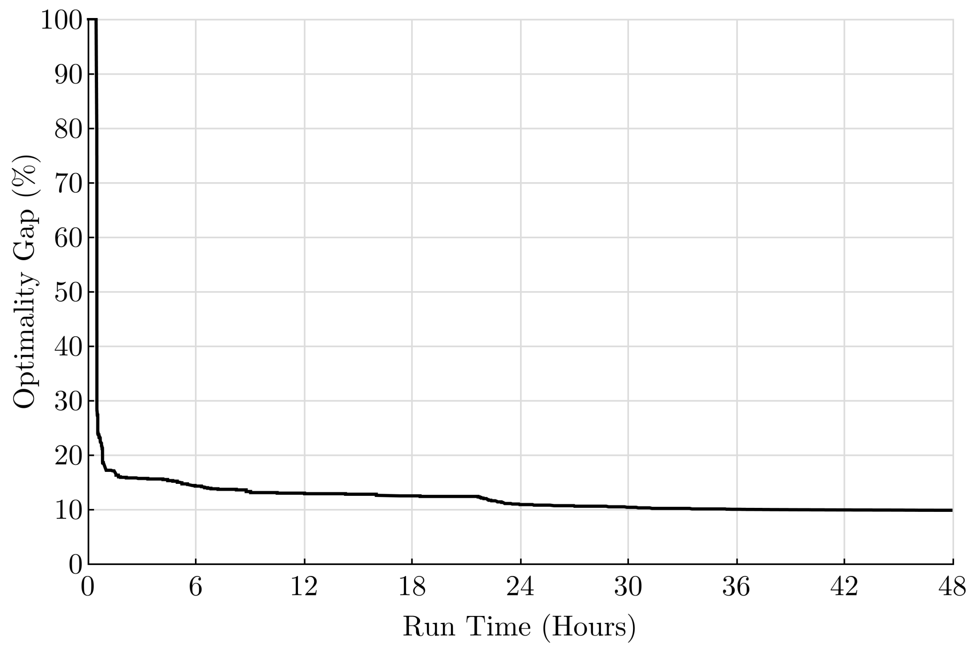


Figure 13: Convergence in Baseline Problem Instance

as the baseline problem instance with the understanding that a full week would be preferable but not possible (at least for most approaches).

The convergence behavior of the baseline problem instance is shown in Figure 13. After an initial dramatic decrease in the optimality gap, the progress is relatively flat. Gurobi finds a solution with a lower than 20% gap within the first hour but spends the remaining forty-seven hours reducing the gap an additional ten percent.

This convergence behavior is primarily driven by the problem size and the poor quality of the LP relaxation. The LP relaxation is a fundamental part of the underlying branch-and-bound algorithm, and its quality has a large impact on the efficiency of the algorithm. Its effect is best explained in the following proposition from Nemhauser and Wolsey[160] for maximization problems,

Proposition. *If node t of the enumeration tree with constraint set S^t is such that* $\max\{cx|x \in S^t_R\} < z_{IP}$, *then node t cannot be pruned.*

They go on to comment, “[this proposition] indicates that, regardless of how we develop the tree, the bounds (quality of the relaxations) are the primary factor in the efficiency of a branch-and-bound algorithm." The same property holds for the ESSND with just a change of signs. The bounds of the ESSND problem are poor and as a result, after a brief period of success in finding high-quality solutions, the algorithm has difficulty improving on that solution or proving that it is optimal. If branches cannot be pruned from the B&B tree, the algorithm becomes more akin to an explicit enumeration technique, and its progress dramatically slows down.

The ESSND problem bound is poor because one of the fundamental challenges of the problem, to load aircraft as efficiently as possible, is not captured in the LP relaxation. Instead, the optimal solution of the LP solution will assign a fractional number of the most cost-efficient (best cost-to-capacity ratio) aircraft to move the volume. This assignment typically has little bearing on the real-world solution and results in a poor bound.

Despite the fact that the monolithic approach does not terminate within the 48-hour time limit, it produces a good-quality solution. Some problem and result metrics are summarized in Table 7. The best found solution had an optimality gap of 9.76%

Table 7: Baseline Problem Instance Results

Metric	Value
Number of Rows	136,379
Number of Columns	249,586
Final Optimality Gap	9.76%
Total Trunk Hulls	207 aircraft (of 252)
Trunk Flight Hours	945.6 hours
Total Feeder Hulls	51 aircraft (of 70)
Feeder Flight Hours	167.7 hours
Total Used Trucks	330 trucks

and used fewer hulls than it had available. In regards to optimality gap, recall that this metric indicates that this solution is *no more than* 9.76% worse than the true optimum. It is *possible* that the solution is *closer* than 9.76%, particularly since the optimality gap is based on the lower bound set by the LP relaxation.

It is also notable that the solution does not include all of the aircraft available in

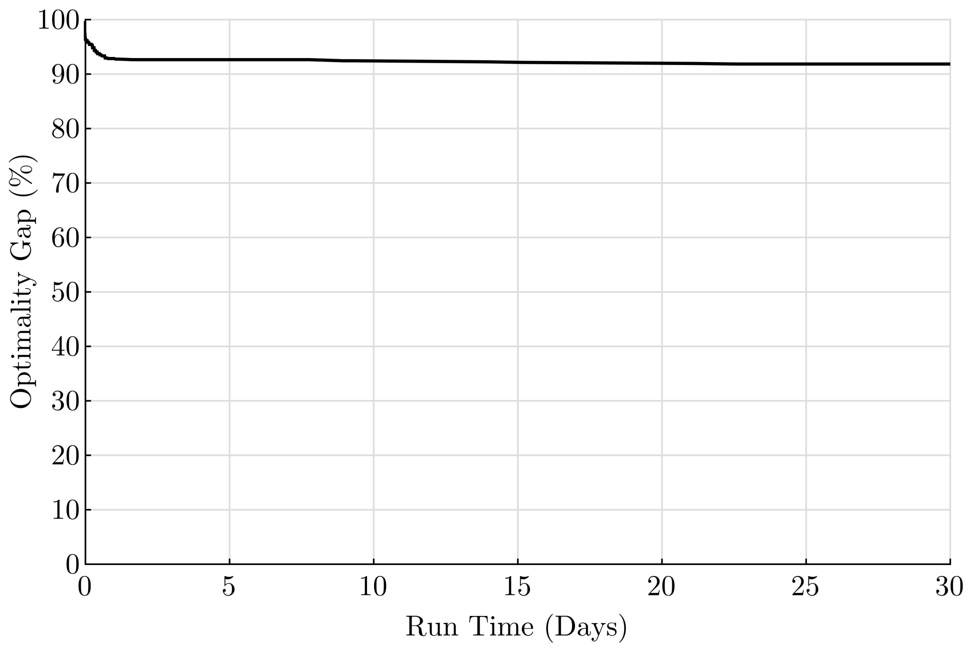


Figure 14: Convergence in Unreduced Problem Instance

the fleet. A lower fleet usage is appealing to the carrier since aircraft are expensive to buy, own and repair. These costs are not included in the objective function so they are not explicitly considered, but it is attractive that the whole fleet does not need to be used.

6.4.3 Unreduced Problem Instance

To underscore the necessity of reducing the routing options and the associated time-space graph, an experiment was run that included all of the possible routing options for the baseline single-day problem instance. It's convergence behavior, or rather lack thereof, is shown in Figure 14.

This case was run for a month and it made no progress. At the end of its run period, the solution still included 161 pirate ships (rule-breaker vehicles). By contrast, the baseline problem instance, with option-reducing heuristics took only about thirty minutes to find a solution without pirate ships. The convergence behavior is extremely flat. Some initial, very modest progress was made during the first day, but after that point, no appreciable gains were made. This is likely a product of the same pruning problem that plagued the baseline case, though on a much greater scale. Table 8 summarizes some problem size and result metrics. The number of rows is similar, but the unreduced case has nearly three times as many variables. This translates to a massive size increase in the B&B tree, and since the poor LP bound means that its pruning is limited, the algorithm is very ineffective.

Table 8: Comparison of the Baseline and Unreduced Problem Results

Case	Rows	Columns	LB Diff.	Obj. Diff.	Optimality Gap	Run Time
Unreduced	169,271	732,417	-20.24%	837.82%	91.6%	30 days
Baseline	136,379	249,586	0%	0%	9.76%	48 hrs

This ineffectiveness is highlighted most strongly in the objective value difference. The unreduced case's objective value is over eight times higher than the baseline. The lower bound difference between the two cases shows that theoretically, if the routings were not heuristically reduced, a better optimum *could* be found, but the problem is completely intractable.

6.5 Sensitivity Studies

To fully explore how the problem size, heuristic thresholds, and constraints affect the algorithm performance, a number of sensitivity studies were performed. Those studies are summarized in Table 9. The baseline values are in bold.

Table 9: Monolithic Approach Sensitivity Studies

Focus	Studies
Number of Airports	10, 25, 50, 75, 100, **115**
Product Set	Boxes, Overnight, **All**
Temporal Scope	**1 day**, 2 days, 3, days, 4 days, full week
Consolidation Options	0/spoke, **1/spoke**, 2/spoke, 3/spoke
Secondary Sort Threshold	7,500 lbs, **10,000 lbs**, 12,500 lbs
Constraint Relaxation	Hull Balance, Hull Count, Feeder Limit, Sort Capacity

Each experiment is a deviation from the baseline in one metric. Given how computationally intensive each run is, studying the cross effects of more than one consideration was not possible. However, valuable insight can still be generated from these sensitivity studies.

In order to protect some sensitive parts of the data, particularly cost-related values, some metrics are represented as a percent difference from the baseline, which is calculated,

$$\text{difference} = \left(\frac{\text{value} - \text{baseline value}}{\text{baseline value}}\right) \times 100$$

Therefore, negative differences indicate that the value is smaller than the baseline and likewise with positive differences. This also means that the baseline case is easily identifiable as the one with the 0% difference.

6.5.1 Number of Airports

Experiments with a subset of the airport set were performed to explore how the monolithic approach performs with increased problem size. The aircraft subsets were selected randomly. Each airport set is a subset of all of the larger sets to ensure that each larger set is truly larger (and more difficult) than the smaller sets. Additionally, the primary hub airport, MEM, was included in the smallest subset (and therefore all others) since most of the volume would be routing through MEM anyways. Only the commodities that have origin and destinations within the airport subset are included, so there are dramatic increases in problem size as the airport subset size increases. As appropriate, routings involving secondary sorts are created regardless of whether the sort airport is included in the airport subset. The results of the sensitivity experiments are shown in Table 10.

The most dramatic result from Table 10 is the sharp jump in runtime between the 25 airport and 50 airport cases. Doubling the number of airports makes the run time increase from a couple minutes to a couple days, and, although it was close, the

Table 10: Results of Number of Airports Sensitivity Studies

No. Airports	Rows	Columns	LB Diff.	Optimality Gap	Run Time
10 airports	9,402	16,335	-95.82%	0%	9 sec
25 airports	21,734	35,472	-89.17%	0%	135 sec
50 airports	49,625	85,856	-67.51%	0.44%	48 hrs
75 airports	77,613	138,506	-46.76%	6.03%	48 hrs
100 airports	113,251	205,883	-14.05%	7.00%	48 hrs
115 airports	136,379	249,586	0%	9.76%	48 hrs

50 airport case did not even finish. Otherwise, the problem scales as expected. There are dramatic increases in the problem size as the number of airports in the airport subset is increased as seen in the number of rows and columns, and, implicitly, in the lower bound difference. As expected, the increase in problem size makes it more difficult for the optimizer to finish, as seen with the in the increase in the optimality gap as the problem size increases.

6.5.2 Product Set

Two alternatives from the baseline were studied in the product set sensitivity test – a case with just box products and a case with just overnight volume. Since the baseline case is a weekday, this means that the box product case only includes priority, first overnight, two-day and postal boxes, for a total of four product types. The overnight product set is first overnight boxes and priority boxes, documents and freight, for a total of four product types. The results are shown in Table 11.

Table 11: Results of Product Set Sensitivity Studies

Product Set	Rows	Columns	LB Diff.	Optimality Gap	Run Time
All Products	136,379	249,586	0%	9.76%	48 hrs
Boxes	105,462	176,865	-3.68%	7.78%	48 hrs
Overnight	81,324	138,535	-45.79%	5.16%	48 hrs

As would be expected, since both of these product sets are subsets of the baseline, both the lower bound and optimality gap are smaller than the baseline. Of the two, the overnight product set is smaller and that is reflected in the problem size and optimality gap. Initially, this large difference in problem size is somewhat unexpected since both cases had four product types. However, the discrepancy arises primarily from the fact that not all markets, particularly small markets, have document or freight volume. Every market does send volume to every other market in some form, but not all markets send volume in all forms to each other market. Between box, document, and freight product types, box volume is, by far, the largest. There is also the secondary effect that day volume, particularly two-day volume has much more time to consolidate and so there may be some markets that have a consolidation routing during the day but not during the night. This would further explain the differences in the problem sizes.

It is notable that the overnight product set case did not terminate before the time limit. This was the problem that MIT and UPS studied two decades ago, and despite the impressive advances in both computational power and optimization solvers since that time, it has not been enough on its own to overcome the challenges of express shipment problem. This does come with the caveat that it is not clear exactly which product types MIT and UPS included. First Overnight[3] products are too new to have been part of the product set, but whether their approach explicitly tracked document and freight products as well as boxes is not clear.

6.5.3 Temporal Scope

Several expanded-scope experiments were run to study how the run time scaled with increased temporal scope. The results of these studies are found in Table 12. The problem instances encompassed two, three, four, and seven days worth of volume. The two, three and four-day cases are all weekday cases, and the seven-day case is the only one that includes weekend volume. The weekend volume, including Friday, is small

[3]First Overnight products are similar to priority box products for the purposes of line-haul transportation, but are delivered earlier in the morning.

compared to the weekday volume (see Figure 4 on page 26, §2.7) so solving individual five and six-day problem instances does not lend much to the discussion. The primary

Table 12: Results of Temporal Scope Sensitivity Studies

Days	Rows	Columns	LB Diff.	Optimality Gap	Run Time
1 day	136,379	249,586	0%	9.76%	48 hrs
2 days	266,739	489,735	89.10%	80.60%	48 hrs
3 days	395,619	723,532	176.88%	99.1%	48 hrs
4 days	521,043	923,803	267.36%	99.1%	48 hrs
7 days	686,102	1,153,267	364.70%	*No Sol.*	48 hrs

takeaway from these experiments is that a problem instance encompassing more than a single day's worth of volume is not solvable with a monolithic approach. The solutions for the two, three and four-day cases are extremely poor and the full-week problem instance did not even find a single solution after running for two days.

Although solving a single day's worth of volume, both the night and day operations, is of great value and more than most studies in the past, ideally, a solution approach would be able to produce a good solution for a week's worth of volume. A week is a more natural unit of time in express shipment operations because the slow weekend period allows for more of a break and "reset" period for the network. The results of the temporal scope sensitivity study show that this monolithic approach is ill-equipped to deal with any time period greater than a single day, let alone a full week.

6.5.4 Consolidation Opportunities

The effect of including consolidation was studied through experiments that allowed a maximum of zero, one, two or three consolidation options per eligible spoke location. In the case where zero options were permitted, only flights to and from the network hubs were allowed. In cases where at least one consolidation option was allowed, the consolidation options for a spoke were those that were spatially closest to it. The results for this set of experiments are shown in Table 13.

Table 13: Results of Consolidation Opportunity Sensitivity Studies

Opportunities Per Spoke	Rows	Columns	LB Diff.	Optimality Gap	Run Time
0 Consol.	132,891	211,677	9.56%	5.22%	48 hrs
1 Consol.	136,379	249,586	0%	9.76%	48 hrs
2 Consol.	137,128	255,377	-0.05%	8.11%	48 hrs
3 Consol.	138,370	280,102	-0.76%	7.63%	48 hrs

Unsurprisingly, the number of rows and columns scale with the number of consolidation opportunities per spoke and including more consolidation improves the lower bound. What is a little surprising is the fact that of the three experimental and one baseline cases, the baseline case had the largest optimality gap at the end of the time limit. Given the increased size of the the two and three option cases, it is expected that those cases would have had a more difficult time finding good solutions and therefore a higher optimality gap.

Table 14: Solution Details of the Consolidation Opportunity Sensitivity Studies

Opportunities Per Spoke	Trunk Hours	Trunk Hulls	Feeder Hours	Feeder Hulls	Trucks	Objective Diff.
0 Consol.	994.2	237	186.7	70	223	4.32%
1 Consol.	945.6	207	167.7	51	330	0%
2 Consol.	915.2	203	183.6	54	341	-1.86%
3 Consol.	905.7	248	177.3	54	366	-3.05%

Table 14 shows that the quality of the solutions does improve as more consolidation options are added, which is expected, and the increase in problem size does not prevent the optimizer from finding good solutions, which is less expected. One interesting result is that the number of trucks used increases while the number of truck flight hours decreases with an increased number of consolidations. This does indicate that expensive trunk aircraft use is being replaced by cheaper intra-spoke ground transportation and shows that good solutions do take advantage of consolidation opportunities.

6.5.5 Secondary Sort Threshold

Two sensitivity experiments were performed to study the effect of the secondary sort threshold on the monolithic approach's performance. The secondary sort threshold is the dividing line between those markets whose commodities are given routing options to multiple sorts and those whose commodities are only given routing options through the primary sort. Any commodities originating from a market that originates and terminates *less* volume than the threshold amount for all phases of the day is only given routings through the primary sort. This means that a problem instance with a *higher* secondary sort threshold will have *fewer* commodity routings. The results are shown in Table 15.

Table 15: Results of Secondary Sort Threshold Sensitivity Studies

Secondary Sort Thresh.	Rows	Columns	LB Diff.	Optimality Gap	Run Time
7,500 lbs	138,210	265,158	-1.17%	12.1%	48 hrs
10,000 lbs	136,379	249,586	0%	9.76%	48 hrs
12,500 lbs	134,624	238,155	0.03%	7.90%	48 hrs

As expected, allowing more markets to sort products at secondary hubs results in a larger problem and a *lower* lower bound. The lower bound difference between the 7,500 lbs threshold and the baseline threshold is larger than the difference between the 12,500 lbs threshold and the baseline. This suggests that a better solution could be found if the sort threshold were lowered, but this comes at the cost of increased runtime. On other hand, the secondary sort threshold could be reasonably increased without much detriment to solution quality with respect to the baseline while also reducing the problem size. The problem size does have an impact on the solution quality for this study as seen in Table 16.

The solution details in Table 16 shows the disadvantage of including more options in the problem. Although the 7,500 lbs case has the lowest lower bound of the three cases, and therefore *potentially* the best optimal solution, in a given time period it

Table 16: Solution Details of the Secondary Sort Threshold Sensitivity Studies

Secondary Sort Thresh.	Trunk Hours	Trunk Hulls	Feeder Hours	Feeder Hulls	Trucks	Objective Diff.
7,500 lbs	948.5	214	194.3	51	336	1.48%
10,000 lbs	945.6	207	167.7	51	330	0%
12,500 lbs	916.8	208	170.3	52	330	-1.99%

produces the worst solution. On the other hand, the 12,500 lbs case has the highest lower bound but produces the lowest cost solution after two days of runtime. This is because of the difficulty of finding good solutions within the branch-and-bound tree. As long as good solutions are not removed from the branch-and-bound tree when the problem size is reduced, they will be more easily found if the tree is smaller. The options of the 12,500 lbs case are a subset of those in the baseline case, which are a subset of those in the 7,500 lbs case, so the good solution found by the 12,500 lbs case exists in both the baseline and the 7,500 lbs cases but because of the size of the branch-and-bound tree it had not been found within the allowed runtime.

However, the problem size is not the only factor in play. When compared to the problem sizes of the consolidation sensitivity experiments, the secondary sort threshold cases are similarly sized. The consolidation sensitivity experiments were able to find better solutions when given more options; unlike the secondary sort threshold experiments. This is a result of the *difficulty* of the decisions that were added. Both added (or removed) commodity routings, but those commodity routings had different impacts on the network as a whole. With consolidation, the impact of choosing to consolidate or not is relatively localized. It will affect the spoke and the intermediate stop directly, and there may be some secondary hull balance effects, but that is about the extent of the effect.

The secondary sort threshold affects the sort *membership* of the network, which can have major ripple effects. This is because the inclusion or exclusion of a spoke from a secondary sort's membership will affect the volume coming in to and out of every other spoke in that sort's membership set. For example, say that CLT[4] volume is allowed to sort at secondary sorts. Sending volume to all of the sorts is out of the question so a subset must be chosen. Let's imagine that CLT connects to MEM and IND, and that BWI[5] also connects to MEM and IND. Let's say, for the sake of this example, that with this configuration, BWI can send one exactly full aircraft to MEM and one to IND, and all of the CLT-bound volume is routed through IND. If this is changed, perhaps CLT *only* sorts at MEM, then BWI will have to either upgauge its MEM-bound flight to carry the CLT volume or route all of its volume through MEM, thus having cascading effects on all of the other markets that connect to IND. In this implementation, the burden of deciding the secondary sort membership falls to the optimizer.

Getting the sort membership right has the potential to realize great efficiencies in the network, but finding these efficiencies is very difficult thanks to the network effects. Therefore, when more markets are forced to sort at exclusively at MEM (i.e. higher secondary sort threshold), the number of these difficult decisions is reduced and it is easier to find good solutions, but at the risk of missing the true optimal solution.

6.5.6 Formulation Relaxations

The final sensitivity study what was performed focused on the effect of relaxing constraints within the formulation. The forcing and cover constraints cannot be relaxed, but the sort capacity, hull balance, hull count and feeder limit constraints were relaxed. The results are shown in Table 17.

Of the relaxations, the hull balance constraint has the largest impact. The number of constraints is drastically smaller when hull balance is not enforced, and the lower bound and the optimality gap are the lowest of all the relaxations. The next

[4]Charlotte Douglass International airport

[5]i.e. Baltimore-Washington International Thurgood Marshall airport

Table 17: Results of Secondary Sort Threshold Sensitivity Studies

Relaxed Constraint	Rows	Columns	LB Diff.	Optimality Gap	Run Time
Hull Balance	76,129	249,586	-6.74%	7.09%	48 hrs
Hull Count	136,369	249,586	-4.42%	7.79%	48 hrs
Feeder Limit	136,034	249,586	-0.11%	8.60%	48 hrs
Sort Capacity	136,369	249,586	-0.17%	8.26%	48 hrs
Baseline	136,379	249,586	0%	9.76%	48 hrs

most significant relaxation is the hull count relaxation, despite the fact that it only reduces the number of rows by ten from the baseline (there are ten constrained vehicle types). The hull count has such a large impact because it allows new vehicles to be used wherever there needed without penalty, thus essentially bypassing the balance constraint. The hull balance and hull count constraint work together to enforce the realistic use of the vehicle fleet and when one is missing the other is left toothless. The other two relaxations, the sort capacity, and feeder limit constraints, have significantly less impact. This is not surprising since the feeder limit only applies to a small subset of the arcs, and FedEx has put significant effort into their network to make sure that the sorts are appropriately sized and located.

Since these relaxations will produce a solution that is, in some way, infeasible with respect to the original formulation, it is interesting to quantify the degree to which these relaxations deviate from the baseline. For the hull balance and hull count constraints, this can be done by looking at the number of hulls used in each solution compared to the baseline and the real-world hull limit. Those hull counts are found in Table 18. The aircraft are listed from largest to smallest.

Interestingly, the total number of trunk hulls is very similar across all three cases, but the real deviation from the baseline is in the composition of the fleet. The hull balance relaxation uses exclusively the newest, and most cost-effective, aircraft. The hull count relaxation fleet is more diverse, but it also prefers newer aircraft to older aircraft.

Table 18: Hull Usage in the Hull Balance and Hull Count Constraint Relaxations

Aircraft	Number in Fleet	Baseline	Balance Relaxation	Count Relaxation
B777F	5	5	47	20
MD-11	43	30	0	5
MD-10-10	37	7	0	1
B767-300F	30	30	100	85
A-300	49	49	0	37
A-310	8	8	0	0
B757-200	80	75	62	60
ATR-72	10	10	48	38
ATR-42	10	10	22	20
C-208	50	31	21	21
Trunk Total	252	207	209	208
Feeder Total	70	51	91	79

Violations of the feeder count constraint can be tabulated by counting the number of times that more than one feeder is used on an arc. Table 19 tallies the usage of multiple feeders on a single arc. In total, there were thirty-five instances in which multiple feeders were used on an arc. The C-208 was the only feeder to be used more than twice on a single arc, because it was the most numerous feeder. The ATR-72 and ATR-42 were both limited to ten hulls so there were not many available to double or triple up on an arc.

Table 19: Feeder Usage in the Feeder Limit Constraint Relaxation

Aircraft	2/arc	3/arc	4/arc	5/arc
ATR-72	6	0	0	0
ATR-42	3	0	0	0
C-208	16	8	1	1

As for the sort capacity constraint relaxation, only two sort activity instances were over capacity – the MEM night sort and the IND day sort. They were assigned 110% and 106% of their capacity, respectively. These numbers are not egregious from a relative sense, but given the huge capacities of these two sorts, it does equate to a large amount of volume. Handling the additional volume would be particularly difficult for the MEM night sort since it is so time constrained.

6.6 Summary

This chapter focused on the monolithic approach for solving the express shipment service network design problem. Attempting to solve the ESSND problem without any reduction in the problem size is shown to be intractable, and a heuristic approach for creating a subset of good commodity routings is explained. When this heuristic approach is used, Gurobi is capable of producing good solutions for the single-day problem instance involving all of the product and commodities. This problem instance is taken as the baseline problem instance for this chapter and the remainder of this work. Finally, a number of sensitivity studies were performed to explore the effect of the problem size, thresholds on the heuristics and formulation constraints. Of particular note was the fact that the monolithic approach was incapable of solving any problem instances encompassing more than a day worth of volume. In the next chapter, Chapter 7, the heuristic approach will be developed, implemented and the results will be compared to those of the monolithic approach.

CHAPTER VII

HEURISTIC DECOMPOSITION

In the previous chapter, the monolithic approach, which was developed through a collaboration between researchers at Georgia Tech and FedEx, was described and sensitivity studies were performed. That approach will serve as the baseline approach in this dissertation. After using a heuristic method to generate package routing and aircraft movement options, the monolithic approach uses a commercial solver, Gurobi 7.5, to solve the full-sized problem instance as one, single optimization problem. It is capable of producing high-quality solutions, but it requires a longer-than-preferred runtime to do so. *If* it were run to convergence (i.e. the integer solution value equals the lower bound), it would produce an optimal solution, but given the convergence behavior, reaching convergence would likely require a very long runtime.

Recall that the primary objective of this work is to,

Research Objective 1. *Develop a quick and effective algorithm that generates low-cost vehicle schedules to completely route one day's worth of domestic express shipment packages.*

and my fundamental hypothesis in light of that research objective was,

Hypothesis 0. *Based on the evidence in the literature that heuristic and exact decomposition approaches can out-perform monolithic approaches, I hypothesize that a heuristic approach or exact decomposition approach will able to produce solutions for the Express Shipment Service Network Design problem of comparable quality to the baseline monolithic approach in significantly less time.*

This chapter will focus on the development of a heuristic approach to solve the express shipment problem. The sensitivity studies performed on the monolithic approach in Chapter 6 showed that smaller-sized problem instances could be solved considerably faster than larger-sized instances. Based on this observation, a heuristic *decomposition* algorithm is developed to solve the express shipment problem. First, in Section 7.1, a set of implementation-based research questions are posed. Next, in Sections 7.2 & 7.3, those research questions are studied through exploratory experiments, hypotheses are generated based on the observations of those experiments, those hypotheses are then tested in additional experiments, and finally re-posed when necessary based on the results of those experiments. This results in the development of an effective, though basic, heuristic decomposition algorithm. In Sections 7.4 & 7.5, options for improving the basic heuristic algorithm are posed and evaluated. Ultimately, through these improvement explorations, a heuristic algorithm is developed that is capable of producing comparable and *better* solutions to the express shipment problem then the baseline monolithic approach was capable of finding after forty-eight hours of runtime. Finally, scalability and sensitivity studies are performed on using this new, improved heuristic algorithm to show its applicability to problems beyond the baseline problem instance used during its development.

All of the experiments performed in this chapter use the experimental testbed described in Section 5.4, the computational testbed summarized in Table 5, and use Gurobi 7.5 as the commercial MIP optimization software.

7.1 Research Questions for the Development of the Heuristic Decomposition Approach

For any researcher wishing to develop or implement a heuristic algorithm, the fundamental question is,

Research Question 1. *What should the heuristic algorithm be?*

This question is not always easy to answer, particularly for applications like the express shipment problem for which there is comparatively little other work. One of the advantages of exact general approaches like branch-and-bound or Benders decomposition is that they are well-established algorithms that provide at least a firm foundation for development if not a complete algorithm in themselves. Heuristic approaches, however, are not general and must be designed for the specific problem at hand. This does allow for greater levels of customization and exploitation of problem-specific characteristics, but also a potentially more difficult development process since much, if not all, of the algorithm may need to be developed "from scratch."

To develop a heuristic approach for the express shipment problem, let us first recall the results of the sensitivity studies performed in section 6.5 for the monolithic approach. Of particular note are results of the studies focused on the number of airports (§6.5.1), temporal scope (§6.5.3), and formulation relaxations (§6.5.6). These studies show that the monolithic approach is very sensitive to problem size. Very small problems, such as the ten-airport and twenty-five-airport cases could be solved on the order of seconds and minutes, but even moderately larger problems such as the fifty-airport case did not converge to the optimal solution even after two days. Problem instances larger than the baseline single-day problem instance were found to be completely intractable using a monolithic approach. Additionally, relaxing the problem formulation by removing constraints, particularly the vehicle balance constraint, was also found to result in lower optimality gaps after a two-day runtime compared to the baseline approach.

The fact that Gurobi can solve smaller problem instances considerably faster than it can solve the full-sized problem instance suggests that a decomposition approach in which several smaller problems are solved individually and then intelligently recombined into a consistent, feasible solution may likely produce a good heuristic algorithm. Therefore, I hypothesize,

Hypothesis 1.1. *Based on the observation that the sensitivity studies performed* *on the monolithic approach show that smaller problem instances can be much more quickly solved than the full-sized problem instance, I hypothesize that a decomposition-based heuristic will be capable of quickly producing high-quality solutions to the express shipment problem.*

Hypothesizing that a heuristic decomposition approach will be successful in solving the express shipment problem gets us one step closer to the development of a heuristic algorithm, but observing that a decomposition approach may be successful is not enough to start implementing an approach. At the very least, questions remain concerning *how* the problem should be decomposed, and how the subproblems should be arranged and recombined to produce a feasible final solution to the original problem instance. For guidance on the design questions, I drew some inspiration from my personal background in aerospace engineering.

Within aerospace and mechanical engineering, there exists a field of optimization techniques called *Multidisciplinary Design Optimization (MDO)* which has arisen to help engineers grapple with the large, complex optimization problems encountered when designing physical vehicles, machines, and components. The primary objective of MDO is to leverage numerical optimization techniques in the design of complex engineering systems which feature interrelated subsystems so as to produce a better final design than if the subsystems were designed in isolation. The motivating application for MDO methods is the optimization of systems for which, "the sum of optimum components is not an optimum system."[194]

MDO is essentially a field of study for designing heuristic decomposition approaches in the context of engineering design, though it is not typically described in that exact terminology. To understand how it is typically described, let us consult several definitions from the literature. Sobieszczanski-Sobieski and Haftka (1997)[182] define MDO as, "methodology for the design of systems in which strong interaction between disciplines motivates designers to simultaneously manipulate variables in several disciplines." Alexandrov (2005)[6] defines MDO as, "an area of research concerned with developing systematic approaches to the design of complex engineering artifacts and systems governed by interacting physical phenomena" Finally, Martins and Lambe (2013)[154] define MDO as "a field of engineering that focuses on the use of numerical optimization for the design of systems that involve a number of disciplines or subsystems."

From these definitions, it is clear that MDO is intended for *engineering design* problems that include several interacting disciplines. In MDO terminology, *disciplines* are the individual subproblems to be solved. Although it is not explicitly mentioned in the Sobieszczanski-Sobieski and Haftka (1997) and Alexandrov (2005) definitions, numerical optimization typically plays an important role in almost all MDO applications. Due to the nature of engineering design problems, constrained non-linear optimization techniques are commonly used. Finally, it is important to note that a fundamental part of MDO is the development of "systematic approaches to the design of complex engineering artifacts and systems." These systematic approaches are typically formalized as *architectures* and are, in a way, "meta-algorithms." They describe how the numerical optimization algorithms and other computational tools should be used together to produce high-quality designs.

Given that the purpose of Multidisciplinary Design Optimization is to address optimization problems with multiple disciplines (i.e. decomposed problems), I hypothesize that taking some inspiration from MDO will be useful in designing a decomposition-based heuristic for the express shipment problem. However, an approach that completely implements an MDO architecture is unlikely to be successful because MDO is intended for engineering design problems, and therefore typically employs techniques designed to handle the highly non-linear functions, the wide variety of shared and local variables, and the black-box analyses common to design problems. Those techniques would be out of place for solving the ESSND problem which is a MIP and therefore has a linear objective function and constraints which are linear inequalities, only two types of variables, all of which are shared (i.e. no "local" variables), and no black-box analyses.

Employing an MDO architecture as it was originally conceived for an engineering design problem would bring with it unnecessary algorithmic infrastructure and complexity when a simple, fast, and effective algorithm is what is desired for this application. Although the established architectures of MDO are not directly applicable to the ESSND problem, the *mindset* that is employed in developing an MDO architecture is valuable and applicable to this application. In particular, MDO is concerned with how disciplines are defined and how disciplines interact with each other.

Since MDO will be used more for its mindset and inspiration in techniques, and not for the implementation of a pre-existing architecture, a full survey of MDO would be out of place here. Several extensive surveys of MDO have been performed, and there are countless journal and conference papers detailing architectures, techniques, and applications in the literature today. It is recommended that the reader interested in learning about MDO in more depth than the cursory treatment here peruse the works by Martins and Lambe (2013)[154], Balesdent et al. (2012)[19], Allison (2004)[8], Tosserams, Etman and Rooda (2009)[185], Alexandrov and Hussaini (eds.) (1997)[7], and Sobieszczanski-Sobieski and Haftka (1997)[182].

Despite the fact that MDO is typically used to solve problems that mathematically look very different from the ESSND problem, I believe that some inspiration can still be gained from approaches that MDO has found useful in generating good-quality solutions in problems with multiple interacting disciplines. Therefore, I refine hypothesis 1.1 to be,

Hypothesis 1.2. *An MDO-inspired decomposition-based heuristic will be capable of quickly producing high-quality solutions to the express shipment problem based on the* *observation that the sensitivity studies performed on the monolithic approach show that smaller problem instances can be much more quickly solved than the full-sized problem instance, and Multidisciplinary Design Optimization is an optimization field focused on developing heuristic decomposition algorithms.*

This hypothesis sets the context for the heuristic decomposition approach. Two major implementation questions still remain. Those questions are, first,

Research Question 2. *How should the disciplines be defined in the heuristic decomposition approach?*

And, second,

Research Question 3. *How should the disciplines be arranged relative to each other in the heuristic decomposition approach?*

More simply, the questions are "how should the problem be broken up?" and, "how should it be put back together?". The next two sections will explore these two questions in more depth.

7.2 Discipline Definition

The first implementation question is how the disciplines should be defined. In many MDO applications, this question may already be addressed, at least partially, thanks to historical precedent, physical limitations, or the availability of analysis software and experimental testbeds. Since MDO has been developed to improve the design of engineering systems, MDO discipline definitions are often either physical or functional in nature. For example, if designing an aircraft, a natural discipline definition would follow the major traditional design areas such as aerodynamics, structures, propulsion, controls, etc., or specific parts of the aircraft such as the tail, fuselage, wings, etc.

In the express shipment problem, a historical precedent does not exist for defining the disciplines so, one must be formulated by considering the properties of the problem. Using the MDO practices as guidance, let us propose three discipline definitions along physical and functional boundaries. A physical decomposition of the express shipment problem is one which breaks up the time-space graph. Therefore, it is one which is either spatially or temporally-based. As a result, each subproblem would solve the full ESSND problem formulation for each subproblem problem *instance* which is *smaller* than the whole, un-decomposed problem instance. A functional decomposition is one in which the ESSND problem *formulation* is broken up and the full-sized problem instance is solved for each subproblem formulation.

From this observation from the MDO literature, I make the following hypothesis in response to research question 2,

Hypothesis 2.1. *Based on the successful use of functional and physical discipline definitions in the MDO literature, I hypothesize that either a functional or physical discipline definition can be used in a heuristic decomposition approach for the express shipment problem.*

These different discipline definitions need to be evaluated in order to further refine this hypothesis. When choosing an appropriate decomposition definition, there are three important considerations. They are,

1. The time it takes to solve each subproblem.
2. Whether the subproblems can be solved in parallel.
3. The nature of the coupling.

First, the time it takes to solve each subproblem is an important consideration, particularly in this application, because the primary goal of implementing a heuristic algorithm is to generate good solutions more quickly. Second, long subproblem runtimes may be more acceptable if the problems can be easily solved in parallel, thus reducing the total wall time to generate a solution. Some problems lend themselves more naturally to parallelization than others.

Finally, coupling is a natural byproduct of almost any non-trivial discipline definition and it must be taken into account to ensure that consistent, feasible final solutions can be found in a reasonable amount of time. Coupling is the shared information that exists between subproblems. For example, when designing a wing, the aerodynamics and structures are coupled since the aerodynamics affects the loads on the wing, and the stiffness and weight of the wing affect the aerodynamic performance and requirements. If coupling were not present, each subproblem could be solved completely independently and there is no reason to have solved the combined problem in the first place. For most problems, however, coupling exists and, both the *amount* and *type*, are important because each can have a large impact on how difficult it may be to reconcile the individual subproblem solutions into a final, consistent, feasible solution. If there is a lot of coupling between the subproblems, it may be very difficult to find a consistent solution. However, some types of coupling may be harder to satisfy than others, even if there is less coupling overall. These three considerations listed above will be the primary criteria for evaluating the three discipline definitions posed in the following subsections.

7.2.1 Physical Decomposition: Temporal

A temporal decomposition definition is a physical decomposition which breaks up the time-space graph into periods of time which can be solved independently and then recombined into a cohesive solution. The appeal of this approach is that it can exploit the periodic nature of the express shipment problem. Unlike many other airline or cargo problems, the express shipment problem is strongly periodic.

These periods are centered around the sorting activity instances and divide the day into four distinct *phases* – night pick up, night delivery, day pick up, and day delivery. During the periods between the phases, particularly during the sort activities, the domestic airline fleet sits on the ground, waiting to be reloaded. These periods of inactivity provide good opportunities for discipline boundaries, whether they be during the sorting activities or between the nighttime and daytime operations. Figure 15 shows a notional temporal decomposition defined according to cycles.

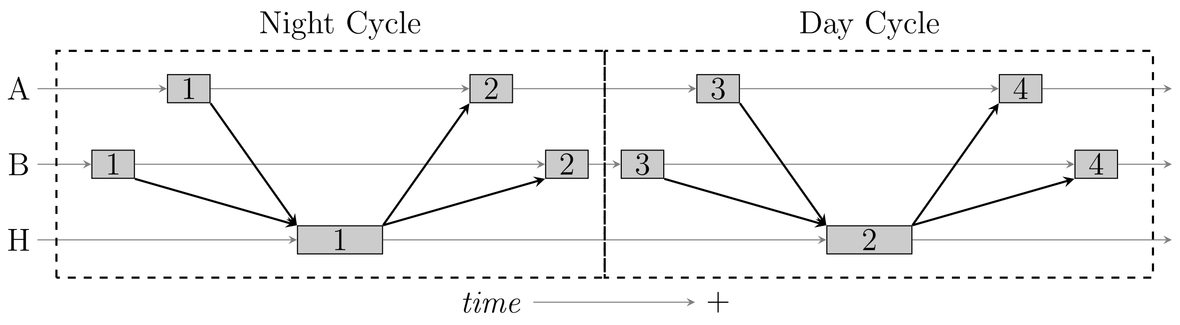


Figure 15: Temporal Decomposition

The advantages and disadvantages of a temporal approach can be weighed in terms of the three considerations introduced in section 7.2. Those considerations were the runtime of the subproblems, whether the subproblems could be solved in parallel, and nature of the coupling between the subproblems. The runtime of the subproblems can be determined experimentally, but let us first define appropriately-sized subproblems, and consider the nature of the coupling.

Type and Degree of Coupling The absolute smallest possible time unit that could be considered for a temporal decomposition is a single phase. If the time unit were any smaller, it would no longer be capitalizing on the natural breaks in operations provided by the periodic nature of express shipment operations. However, a phase-based temporal decomposition would result in extremely heavy coupling between the subproblems since package and aircraft flow balance would need to be enforced between them. In a phase-based require an extremely large number of balance constraints for both the packages and the aircraft. In fact, *every* package routing variable would require a balance constraint between subproblems since a phase only defines, at most, *half* of a routing, and nearly every aircraft movement would also require a balance constraint between subproblems.

The amount of coupling between subproblems can be greatly reduced by using a cycle-based temporal decomposition definition instead of a phase-based temporal decomposition. With a cycle-based temporal decomposition, flow balance constraints are not needed for the same-day and overnight package routing variables since those products are moved entirely within a single cycle. Coupling would still arise, however, from the package routings of the two-day products and the aircraft movements between cycles, but the coupling would be considerably lower than the coupling in the phase-based temporal decomposition.

Potentially, a "wrap-around" constraint, which is an extended balance constraint from the end of the day to start of the day, could be used for each subproblem to reduce the conflicts between the subproblems. The wrap-around constraint ensures that each airport has the same number and type of planes at the end of the cycle as it had at the beginning of the cycle. Assuming that each cycle is similar to its predecessor and successor, this would reduce the number of aircraft balance constraint violations between the subproblems and potentially make it easier to reconcile the differences between the subproblems.

Parallelization Fortunately, the temporal decomposition approach does generally lend itself to parallelization. Since each cycle-based subproblem is still modeled using the full ESSND problem formulation, a solution can be generated for each subproblem independently. The individual subproblems would then need to be recombined and reconciled to produce a final solution. The difficulty of this subproblem reconciliation would be largely affected by how flexible each subproblem would be to changes, which is closely related to the type and degree of coupling present in the subproblems.

Subproblem Runtime The runtime of the subproblems can be tested with an exploratory experiment. There are two cycles per day, 1) a night cycle, during which the overnight commodities are moved, and 2) a day cycle, during which the same-day and two-day commodities are moved[1]. That means, that for the baseline problem instance (summarized in Table 6), there are two subproblems to test, a night-cycle subproblem and a day-cycle subproblem.

These two experiments are tested in the experimental testbed described in Section 5.4, where the problem instance is filtered to include only overnight products (night-cycle) or two-day and same-day products (day-cycle). These reduced-size problem instances are solved using Gurobi using a monolithic solution generation approach and run for 24 hours. The results of this experiment are summarized in Table 20.

Table 20: Temporal Decomposition Subproblem Runtimes

Problem	Rows	Columns	Optimality Gap	Total Runtime
Night-Cycle	81,324	138,535	5.20%	24 hrs
Day-Cycle	56,947	117,828	4.98%	24 hrs

After 24 hours, the optimal solution for neither problem had been found. Considering that these subproblems would still need to be reconciled, potentially iteratively, this long runtime is a major disadvantage. One of the key shortcomings of a temporal decomposition is that it is limited in it's "resolution." Problem instances smaller than a cycle would be burdened by prohibitively large and difficult coupling between the subproblems, but cycle-sized and larger problem instances cannot be solved in reasonable periods of time.

7.2.2 Physical Decomposition: Spatial

The express shipment problem could also be decomposed spatially. In a spatial decomposition, the time-space graph is divided along geographical boundaries into regions. The most straight-forward regional definition would be to divide the United States geographically according to the areas local to the five sorting hubs - an airport would

[1]Strictly speaking, two-day commodities *could* be moved during the night cycle but in practice two-day volume is predominantly moved during the day cycle with the same-day volume.

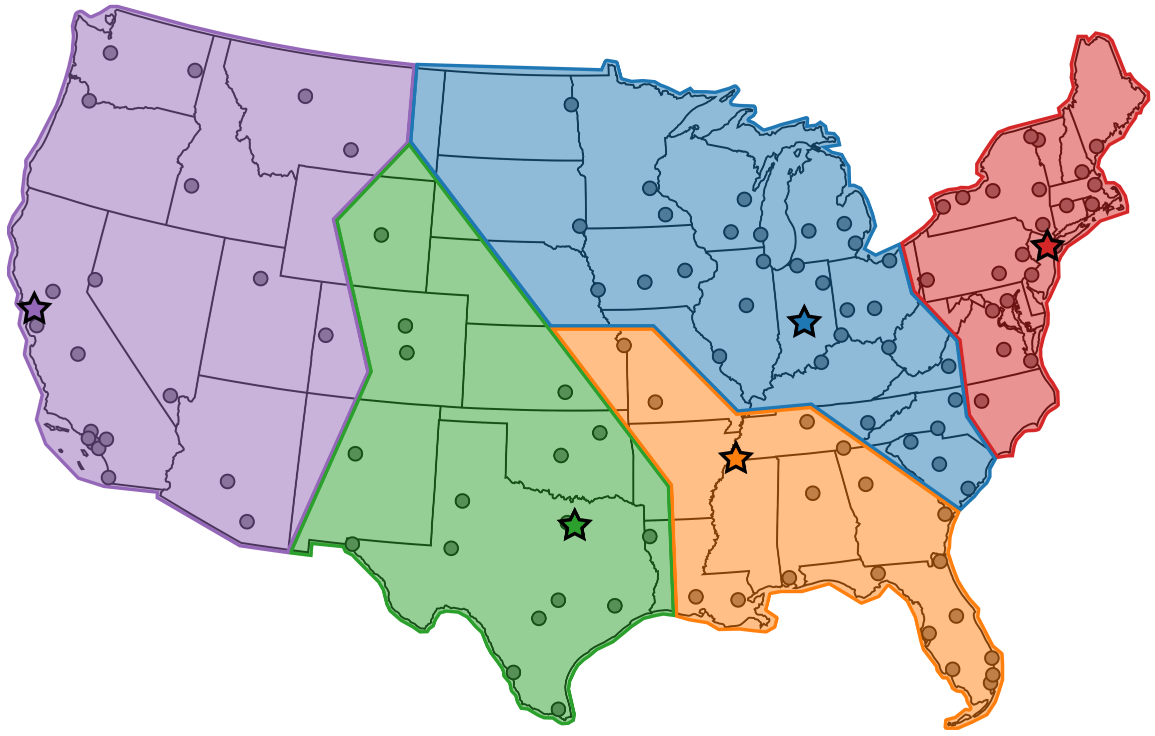


Figure 16: Sort-Based Regional Decomposition

"belong" to whichever sorting hub was closest. Airports in those regions would then sort all of their volume at their regional hub. This sort-based spatial decomposition is shown for the baseline problem instance in Figure 16.

Since there are five sorting hubs in the problem instance, there are five regions. Thanks to the geographical distribution of these hubs, this translates to regions located roughly in the Northeast (EWR), Southeast (MEM), Midwest (IND), Southwest and Mountain West (AFW), and West Coast (OAK).

Parallelization Like the temporal decomposition, the spatial decomposition lends itself to parallelization. Each of the regional subproblems can be solved independently without knowledge of the other subproblems since each subproblem is modeled with the whole ESSND problem formulation.

Subproblem Runtime Each of these regions only encompasses between eighteen and thirty-two airports. The airport-set sensitivity studies performed in Section 6.5 showed that problems of this size could be solved extremely quickly, even when the spokes can connect to multiple hubs. The runtime of each regional subproblem can be tested experimentally where each subproblem is defined as one of the regions shown in Figure 16.

These five experiments are tested in the experimental testbed described in Section 5.4, where each problem instance is filtered to include only those products whose origin *and* destination are both in the region. Since this is an exploratory experiment, the routing option generation method was *not* modified to force the use of the regional hub. Each problem instance is solved using Gurobi using a monolithic solution generation approach. The results of this experiment are summarized in Table 21.

Table 21: Spatial Decomposition Subproblem Runtimes Without Interregional Volume

Region	Number of Airports	Runtime
AFW	18	10 sec
EWR	23	120 sec
IND	32	367 sec
MEM	19	17 sec
OAK	23	9 sec

Consistent with the results of the airport-set sensitivity studies in Section 6.5, these regional subproblems can be solved very quickly. The longest case, the IND region case, took only about six minutes to solve to optimality. However, and this is critical, none of these subproblems consider the interregional volume, which is substantial and extremely difficult to reconcile. Ultimately, although the runtimes of the strictly-regional subproblems are encouraging, the interregional volume cannot be ignored and presents major challenges.

Type and Degree of Coupling The question of how to handle the interregional volume is a significant hurdle that does not have an obvious answer. Since every airport sends volume to every other airport, there is a massive amount of interregional volume. This interregional volume gives rise to a large number of vehicle and package flow balance issues that must be reconciled between the subproblems.

To illustrate this point, imagine four locations divided into two regions with two locations in each region as seen in Figure 17. To keep the example simple, hubs are omitted. The full ESSND problem is solved within both of these regions for the packages that originate and terminate *within* that region.

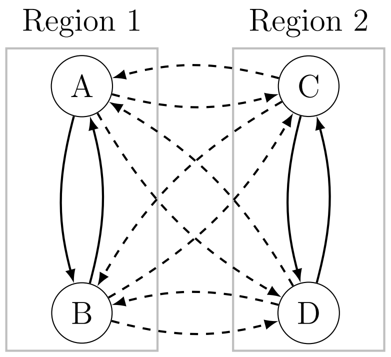


Figure 17: Coupling Between Regions in a Spatial Decomposition

The issue arises in what to do with the commodities that bridge regions 1 and 2. Commodities that originate within the one region but are delivered in the other region must be treated as coupled variables between the two subproblems. As a result, region 1 has only two local arcs, but eight shared arcs, which require coupled variables, and similarly for region 2. As a result, a spatial decomposition as a massive number of coupled packaged variables would exist between the subproblems, and because the aircraft fleet is shared across the United States, aircraft balance and hull count would also be coupled between the subproblems.

The use of hubs as consolidation points would reduce the number of interregional arcs, but this is not a complete fix either. One option is that each region consolidates all of its originated volume at its regional hub, and then it sends several large flights to each other regional hub where it is distributed to delivery flights. Not only is this "solution" is impossible with the current time constraints, it would require each interregional package to visit *two* sorting activities, which would dramatically increase costs and risk of package delays.

A different option would be to again consolidate all of the volume originated at the regional hubs but, rather than fly to a second hub, fly the volume to its destination. While it might be time-feasible (though not universally), it would be impossible from an operational and fleet-size perspective. This arrangement would require each hub to have full connectivity within the network, each spoke would have to accommodate at least five aircraft (one from each sort), and many of those flights would likely be underloaded. It would fail to take advantage of the economies of scale that express shipment companies depend on.

A third option would be to send all of the interregional volume through the primary hub at MEM since it is the only hub to have connectivity with all of the spokes. Unfortunately, the interregional volume is so large that this approach would overwhelm the sort and greatly exceed its capacity.

There are, of course, other possible schemes for dealing with the interregional volume, but the most compelling argument against solving the express shipment problem according to regional subproblems is that it does not faithfully capture the current operations and strategy of express shipment airlines today. In reality, regional hubs exist to take some of the burden off the primary hub by providing a local sorting capability for the large markets in a particular area. Small markets only route through the primary sort. This strategy is a reflection of the advantages of a pure hub-and-spoke network. In terms of the number of flights needed to fully connect a network, a pure hub-and-spoke network is the most efficient network topology. Keeping the number of flights low is important because it allows for a smaller fleet and it means that those aircraft are more fully loaded. Physical limitations in the sorting capacity, as well as transportation costs (e.g. SEA to LAX via OAK instead of MEM), requires a multi-hub-and-spoke approach, but express shipment companies prefer to operate in a pure-hub-and-spoke-like strategy for small markets in order to fill aircraft. The markets that can fill multiple aircraft are those that take advantage of the secondary hubs. This strikes a compromise between the pure-hub-and-spoke topology and the multi-hub-and-spoke topology where every hub is fully-connected.

Thought of another way, a spatial decomposition scheme reduces to a sort-assignment problem. That is, it is the problem of assigning commodities to the sorting facility that they will route through. The difficulty of this problem was previously discussed in the secondary sort threshold sensitivity study in Section 6.5. To summarize that previous discussion, the sort assignments for every market impacts every other market, and these "network effects" makes sort assignment an extremely challenging problem to solve, despite appearing superficially simple.

In summary, the spatial and temporal decompositions are based on breaking up the time-space network into smaller subgraphs and solving several smaller problem instances using the full ESSND problem formulation, then recombining them. The problem formulation itself remains unchanged, it is only the subproblem scope that changes. However, these decomposition approaches, at best, both face the significant challenge of reconciling the vehicle and package balance constraints between the subproblems, not to mention issues with resolution limitations and interregional volume challenges. An alternative approach would be to decompose the problem formulation itself, which is a functional decomposition strategy and is discussed in the next section, Section 7.2.3.

7.2.3 Functional Decomposition

Within the express shipment problem, there are two fundamental questions, "how should packages be routed?" and "how should aircraft be scheduled in order to move these packages?" These two questions capture the two interconnected fundamental functions within package delivery – the routing of packages and the flying of aircraft to move them. The functional distinction is so ingrained in the problem that it is captured in the mathematical formulation as the flow (package routing) and design (aircraft movement) variables.

Therefore, a functional decomposition of the express shipment problem would be to define two problems – one that captures the package routing considerations, and 2) one that captures the aircraft movement considerations. Since this discipline definition is captured in the two variable types of the ESSND problem formulation, it will be called the *variable-based decomposition*. Since the discipline definition splits the ESSND problem according to variable type, the two resulting optimization problems look very similar to those of the ESSND problem implementation of Benders decomposition. The problem formulated to answer the "how should packages be routed?" question, which will be called the *Package Routing Problem* (PRP), is

$$\begin{aligned}
\min \quad & \phi(x_p^k) && && \text{(7.1a)}\\
\text{s.t.} \quad & \sum_{k\in K}\sum_{p\in P^k} \delta_{ij}^p b^k x_p^k \le \sum_{f\in F} u^f \bar{y}_{ij}^f && \forall (i,j)\in A && \text{(7.1b)}\\
& \sum_{p\in P^k} x_p^k = 1 && \forall k\in K && \text{(7.1c)}\\
& \sum_{k\in K}\sum_{p\in P^k} \delta_h^p n_k x_p^k \le e_h && \forall h\in H && \text{(7.1d)}\\
& 0 \le x_p^k \le 1 && \forall p\in P^k, k\in K && \text{(7.1e)}
\end{aligned}$$

where $\phi(x_p^k)$ is a stand-in objective function. The problem formulated to answer the "how should aircraft be scheduled to move these packages?" question, which will be called the *Aircraft Movement Problem* (AMP), is

$$\min \quad \sum_{(i,j)\in A} d_{ij}^f y_{ij}^f \tag{7.2a}$$

$$\text{s.t.} \quad \sum_{k\in K}\sum_{p\in P^k} \delta_{ij}^p b^k \bar{x}_p^k \leq \sum_{f\in F} u^f y_{ij}^f \qquad \forall (i,j)\in A \tag{7.2b}$$

$$\sum_{j:(i,j)\in A} y_{ij}^f = \sum_{j:(j,i)\in A} y_{ji}^f \qquad \forall i\in N, f\in F\setminus\tilde{F} \tag{7.2c}$$

$$\sum_{(i^+,j)\in A} y_{i^+j}^f \leq n_f \qquad \forall f\in F\setminus\tilde{F} \tag{7.2d}$$

$$\sum_{f\in\hat{F}} y_{ij}^f \leq 1 \qquad \forall (i,j)\in A \tag{7.2e}$$

$$y_{ij}^f \in \mathbb{Z}_+ \qquad \forall (i,j)\in A, f\in F \tag{7.2f}$$

In Chapter 8, we will see that the PRP is very similar to the Benders decomposition subproblem, and the AMP is very similar to the Benders decomposition master problem. One notable exception, however, is that the forcing constraint is contained in both the master and subproblems in the functional decomposition whereas in Benders decomposition it is only present in the Benders subproblem and enforced in the master problem through the iterative addition of Benders cuts, a specific type of cutting plane.

Type and Degree of Coupling Whereas in the temporal and spatial decompositions, coupling was primarily a result of the balance and hull count constraints, in the functional decomposition, coupling is a result of the forcing (capacity) constraint. Since the forcing constraint contains both aircraft movement and package routing variables, it is present in both subproblems and gives rise to coupling. The Aircraft Movement Problem must know the package routings in order to assign aircraft and the Package Routing Problem must know the aircraft schedule and capacities in order to route packages. Since every variable in the optimization problem is involved in the forcing constraint, the *amount* of coupling is very high. However, the coupling is a different *type* than that of the physical decompositions. Of particular note is that, if either the package routings or aircraft capacities are known, then the other problem is significantly easier to solve. The physical decomposition approaches did not have that relationship.

Parallelization As a result of the type of coupling between the subproblems, the functional decomposition is not amenable to parallelization. The Aircraft Movement Problem, if it is not provided with package routings, will not fly any aircraft, and the Package Routing Problem, if it is not provided aircraft movements, will be infeasible. As a result, the two problems depend on each other too directly to be solved in parallel.

Subproblem Runtimes The runtimes of the Package Routing Problem and the Aircraft Movement Problem can be tested experimentally, but before these two problems can be reasonably tested, values for $\bar{x}_p^k$ and $\bar{y}_{ij}^f$ must be determined. If they are set to zero, then the PRP will be infeasible and the AMP will be trivial, which does not meaningfully test whether or not a monolithic solution method is appropriate for solving the PRP and AMP. Therefore, for the purposes of this experiment, $\bar{x}_p^k$ was set to be,

$$\bar{x}_p^k = \begin{cases} 1 & \text{if } p \text{ is the shortest path in } P^k, and \\ 0 & otherwise \end{cases} \tag{7.3}$$

for the Aircraft Movement Problem.

In the original Package Routing Problem formulation (7.1), if there is insufficient capacity in the aircraft network, the problem will be infeasible and no packages will be routed. It would be more useful if, instead, packages were routed such that they *minimize the infeasiblility* if there is insufficient aircraft capacity assigned. To do this, the fictional pirate ship vehicles are used to provide fictional capacity, and the PRP objective is to minimize their use, which is a function of both distance and volume. Real-world aircraft that are added as fixed variables y_{ij}^f to the PRP do not incur a cost. Trucks may also be assigned in the PRP since some commodities only have a trucking option, but since they are real-world vehicles, those trucks do not incur a cost. This resulting modified Package Routing Problem is,

$$\min \quad \sum_{(i,j)\in A} \tilde{d}_{ij}^f y_{ij}^f \tag{7.4}$$

$$\text{s.t.} \quad \sum_{k\in K}\sum_{p\in P^k} \delta_{ij}^p b^k x_p^k \le \sum_{f\in F} u^f y_{ij}^f \qquad \forall (i,j)\in A \tag{7.5}$$

$$\sum_{p\in P^k} x_p^k = 1 \qquad \forall k \in K \tag{7.6}$$

$$\sum_{k\in K}\sum_{p\in P^k} \delta_h^p n_k x_p^k \le e_h \qquad \forall h \in H \tag{7.7}$$

$$0 \le x_p^k \le 1 \qquad \forall p \in P^k, k \in K \tag{7.8}$$

$$y_{ij}^f \in \mathbb{R}_+ \qquad \forall (i,j)\in A, f \in F^\circ \tag{7.9}$$

$$\tag{7.10}$$

where, $F^\circ = \{\text{pirate ship, truck}\}$, and

$$\tilde{d}_{ij}^f = \begin{cases} \text{(i,j) distance} & \text{if } f \text{ is a pirate ship, and} \\ 0 & \text{otherwise.} \end{cases}$$

The objective function is set to be a penalty function on the number of pirate-ship vehicles used in the solution. For this purpose, the pirate ship variables are set to be continuous variables and their cost is only a function of distance and volume (i.e. no cycle cost). Since pirate ships are fictional vehicles, any volume "moved" by pirate ships can be considered un-routed. Since the goal is to move *all* packages, the PRP objective is to minimize the use of pirate ships. Trucks are included in the vehicle set F° because the lanes that can truck are forced to truck, but since trucks are real vehicles, their use is not penalized.

Although the objective function is not actually a function of x_{ij}^{k}, it still scales directly with the total volume-miles traveled on non-physical vehicles, which is directly linked to the package routing choices. The objective function was written this way in order to avoid introducing quadratic terms in the objective function. By keeping the pirate ship variables continuous, their magnitude is proportional to the amount of volume assigned to them.

If no aircraft are added to the PRP, as in the experiment performed in this section, the problem reduces to selecting the shortest path option for each commodity that must fly. Since trucks do not incur a cost in the PRP objective function, the commodities that can truck will do so. In the following section (§7.3.1) where the PRP is used in a fixed-point iteration scheme, this objective function will be slightly more meaningful and incentivize the use of the company aircraft that were assigned in the previous AMP iteration. The assignment of those real vehicles is fixed and using them is penalized.

Experiments exploring the runtime of the PRP and AMP were then run on the experimental testbed described in Section 5.4. The full-sized baseline problem instance was used (i.e. no additional dataset filtering was performed), which can be found in Table 6 of Ch. 6. Since the PRP and AMP are relaxations of the original ESSND problem formulation, the solution generation procedure was slightly adjusted. Instead of generating the full set of constraints, only the appropriate subset was generated for the two problems. That is, for the Package Routing Problem, only the forcing, routing cover, sort capacity, and variable bounding constraints added to the optimization model, along with the penalty function objective function mentioned above. For the Aircraft Movement Problem, only the forcing, vehicle balance, hull count, feeder limit, and variable integrality constraints were added to the optimization model, along with the original ESSND objective. Both of the resulting optimization models were then solved using Gurobi.

Several experiments on the PRP and AMP were run, and these experiments were run on a smaller computational testbed since they were not expected to need the full computational resources of the monolithic testbed (Table 5). The properties of the smaller computational testbed are tabulated in Table 22. The most notable differences are the significant reductions in cores and RAM, an increase in processor speed, and the change of operating system.

Table 22: Hardware and Software Properties of Smaller Computational Testbed

Property	Specification
Processor	Intel i7-2600
Number of Cores	4 (of 8 total physical and virtual cores)
Processor Speed	3.4 GHz
RAM Size	24 GB
Operating System	Windows 7 Enterprise
Optimization Software	Gurobi 7.5.1
Optimization Settings	48 hr time limit, default otherwise
Implementation Language	Java 8

Several different factors were explored in these runtime experiments. Specifically, the effect of including consolidation and the effect of the routing variable domain (continuous vs. binary) are studied. The results of the PRP and AMP experiments are summarized in Table 23. The results are generally as expected. All of the exper-

Table 23: Runtime of the Package Routing Problem and the Aircraft Movement Problem

Problem Name	w/ Consol.	Routing Variable	Runtime
Package Routing Problem	True	Continuous	7.75 sec
Package Routing Problem	True	Binary	22.86 sec
Aircraft Movement Problem	True	Fixed	660.42 sec
Package Routing Problem	False	Continuous	9.66 sec
Package Routing Problem	False	Binary	12.51 sec
Aircraft Movement Problem	False	Fixed	368.18 sec

imental cases were solvable in a reasonable time frame, with the longest run taking eleven minutes. The aircraft routing problem did take significantly longer to solve than the Package Routing Problem, but considering that the full-size problem did not even converge in two-days, a runtime of over ten minutes is not cause for concern at this point. The exclusion or inclusion of consolidation options did have an effect on both problems, though the effect was much more pronounced in the AMP, for which including consolidation nearly doubled the runtime. This is likely a result of the increased size of the time-space graph.

7.2.4 Refinement of the Discipline Definition Hypothesis

In the introduction to this section, three considerations for evaluating discipline definitions were presented. Those were,

1. The time it takes to solve each subproblem.
2. Whether the subproblems can be solved in parallel.
3. The nature of the coupling.

The three discipline definitions, temporal decomposition, spatial decomposition, and functional decomposition, were then evaluated based on those considerations. In an ideal situation, each subproblem would be quick to solve, amenable to parallelization, no coupling would exist between the subproblems. Without coupling the subproblems could be trivially recombined and generating the solution would be simple. Unfortunately, this is not the reality, and the best solution is often the one whose challenges can be most easily overcome.

Of the three discipline definitions considered, I believe that the challenges of the functional decomposition will be the easiest to overcome. Its primary challenge is the large number of coupling variables, since *every* variable is a coupling variable. The subproblems also cannot be run in parallel, but this is not necessarily a deal breaker.

The temporal decomposition was found to have very long subproblem runtimes, thus casting doubt on its suitability as an effective decomposition definition. The spatial decomposition subproblems were considerably faster to solve, but the question of what to do with the large number of interregional commodities is very challenging to answer, and the hull count and vehicle balance constraints must also be reconciled between regions, which is potentially problematic. Based on the observations from the exploratory experiments for the three discipline definitions, I refine hypothesis 2.1,

Hypothesis 2.2. *Based on the results of the exploratory experiments conducted on the temporal, spatial, and functional decomposition definitions, and the success of physical and functional discipline definitions in the MDO literature, I hypothesize that a functional decomposition, which features a Package Routing Problem and an Aircraft Movement Problem, will be an appropriate discipline definition for a heuristic decomposition approach for the express shipment problem.*

This hypothesis addresses research question 2, which focuses on how the problem should be broken up. The next section (§7.3), focuses on how the subproblems should be arranged relative to each other to produce a full solution.

7.3 Discipline Arrangement

Now that the disciplines have been defined, their arrangement relative to each other must be determined. Given that there are two disciplines and they are coupled, they must somehow be "made aware" of the needs of the other in order to produce a consistent, feasible, and, preferably, high-quality, solution. If the Package Routing Problem and Aircraft Movement Problem are considered in isolation, the lack of aircraft capacity in the PRP network will result in an infeasible problem, and the lack of demand in the AMP will result in zero aircraft movements. Therefore, the interaction between the problems must be taken into account. Here is where MDO can lend some inspiration.

The primary purpose of MDO architectures is to describe how disciplines should be arranged relative to each other. These architectures, however, are designed for problems that mathematically look very different from the express shipment problem. In particular, MDO architectures are typically formulated to handled large *non-linear* problems with a variety of variables, some of which are local to the disciplines and some of which are coupled and shared between disciplines. As a result, a lot of the "infrastructure" included in MDO architectures is oriented towards addressing difficulties associated with these types of problems which are irrelevant to the express shipment problem. Implementing one of these architectures directly would be misguided and likely ill-fated. However, some simple techniques from MDO may still be useful.

In their survey of MDO architectures, Martins and Lambe (2013)[154] categorized architectures according to whether a *Multidisciplinary Analysis* (MDA) method was used to enforce consistency between coupling variables or whether consistency was enforced through the use of coupling variable copies and consistency constraints. The former was described as being MDF-like (*Multidisciplinary Feasible*) and the latter was described as IDF-like (*Individual Disciplinary Feasible*). MDF and IDF are single-level MDO architectures in which a single master-level optimizer is used to find the optimum of a problem containing several disciplinary analyses. Martins and Lambe (2013) draw parallels between these single-level architectures and the more complex distributed architectures which typically contain optimization at the discipline-level as well as at the master-level.

Of the two classes of architectures, the MDF-style approaches are more appealing. This is because, with the discipline definition just established in Hy. 2.1, every variable of the ESSND problem would be a coupling variable. This would result in an enormous number of consistency constraints that would need to be handled by a master level optimizer.

There is also another consideration. One of the goals in developing the heuristic decomposition approach is to keep the algorithm as simple as possible, and to only add complexity[2] as it is necessary to provide additional performance. In general, this is a good rule of thumb when designing an algorithm. Complexity should not be added simply for its own sake; it should be added because it is necessary. With this in mind, many of the MDO approaches appear more complex than is necessary for the ESSND problem. Since the disciplines (PRP and AMP) are mixed integer programs, the optimization of those problems is inseparable from the practical means of generating feasible solutions to that problem. This is unlike many MDO applications in which there exists an analysis code that simply calculates the physical property given a set of inputs. The point is somewhat subtle, but the important piece is that this means that if MDO is used as a direct analogy for constructing this heuristic decomposition algorithm, a distributed architecture should be built. That is, one in which there is a master-level optimizer overseeing the progress of discipline-level optimization.

However, from an algorithm complexity view, one must ask whether this master-level optimization is necessary for the ESSND heuristic. Optimization is already being performed on the discipline-level, why introduce another level of optimization if it is not necessary? Furthermore, what would that master-level optimizer act on? What would be the design variables and what would be the local variables, if any? In the interest of only including things to the algorithm as they are found to be necessary, I make the conjecture that a master-level optimizer will *not* be necessary for the ESSND heuristic.

This conjecture means that the MDF-like approaches, or more specifically, the MDA techniques they employ, are more suitable. A common way of producing mutually feasible consistent solutions between two coupled disciplines is through fixed

[2]Algorithmic complexity is only meant in terms of how intricate the algorithm is, *not* the computational complexity, which is a separate though important consideration.

point iteration (FPI). In fixed point iteration, the coupled variables of one discipline are held fixed while a solution is produced by the other discipline. Then, the other discipline is solved, holding the variables of the other discipline fixed to their previously-found values. This iterative process continues until a consistent solution has been found.

Based on these observations, I make the hypothesis,

Hypothesis 3.1. *Since the Package Routing Problem and the Aircraft Movement Problem are coupled, I hypothesize that feedback will be required to produce a mutually feasible solution, and, given that fixed point iteration is a common way to handle coupled disciplines in the MDO literature, the Package Routing Problem and Aircraft Movement Problem should be arranged in a fixed point iteration scheme in the heuristic decomposition approach for the express shipment problem.*

This hypothesis addresses research question 3, and will be tested in the following Section 7.3.1.

7.3.1 Fixed-Point Iteration-Based Solution Generation Approach

In Section 7.2.3, the exploratory experiments showed that the Package Routing Problem (PRP) and the Aircraft Movement Problem (AMP) could be solved relatively quickly. This section tests hypothesis 3.1 that a fixed point iteration scheme is an appropriate discipline arrangement for this discipline definition.

In order to test this hypothesis, an FPI scheme must be implemented using the Package Routing Problem (7.4) and the Aircraft Movement Problem (7.2) that were formulated previously. Thanks to the slight modifications to the PRP introduced in Section 7.2.3, no further modifications to either the PRP or AMP were necessary. The FPI implementation is outlined in algorithm 1.

In the FPI implementation, the Package Routing Problem is solved first, and all volume is moved via pirate ships and trucks as in runtime experiments conducted in

Algorithm 1 Fixed Point Iteration Scheme

```
1: $\mu \leftarrow \infty$
2: while $\mu > 0$ do
3: Package Routing Problem:
4:     Solve Eq. 7.4 with a monolithic approach
5:     $\bar{x}_p^k \leftarrow x_p^k$
6: Aircraft Movement Problem:
7:     Solve Eq. 7.2 with a monolithic approach
8:     $\mu \leftarrow$ number of changed aircraft movements
9:     $\bar{y}_{ij}^f \leftarrow y_{ij}^f$
10: end while
```

Section 7.2.3. Then, using these package routings, which will be the shortest-path routings, the Aircraft Movement Problem is solved. This assigns real aircraft to move the packages along the routings assigned in the PRP. The algorithms iterates, consecutively solving the PRP and the AMP. At each iteration, the scheduled aircraft movements are compared to the previous iteration's aircraft movements. If the movements are identical, then the approach has converged to a consistent solution and is terminated. The number of changed movements was chosen as the termination criterion because the routing problem is degenerate so solutions with identical cost may have different routings.

7.3.1.1 Testing the Fixed Point Iteration Approach

Two FPI experimental cases were run for the baseline problem instance (Table 6), one in which the package routing variables x_p^k were continuous ($0 \leq x_p^k \leq 1$) and one in which they were binary. These cases were run on the experimental testbed described in Section 5.4 using the same computational testbed used for the monolithic approach (Table 5). Since the experiments are run on the baseline problem instance, no additional dataset filtering is performed. The fixed point iteration algorithm described in Algorithm 1 is used in the Solution Generation step, and the individual MIP problems (PRP and AMP) are solved using Gurobi. The convergence behavior of the FPI scheme is shown in Figure 18 for the two experiments.

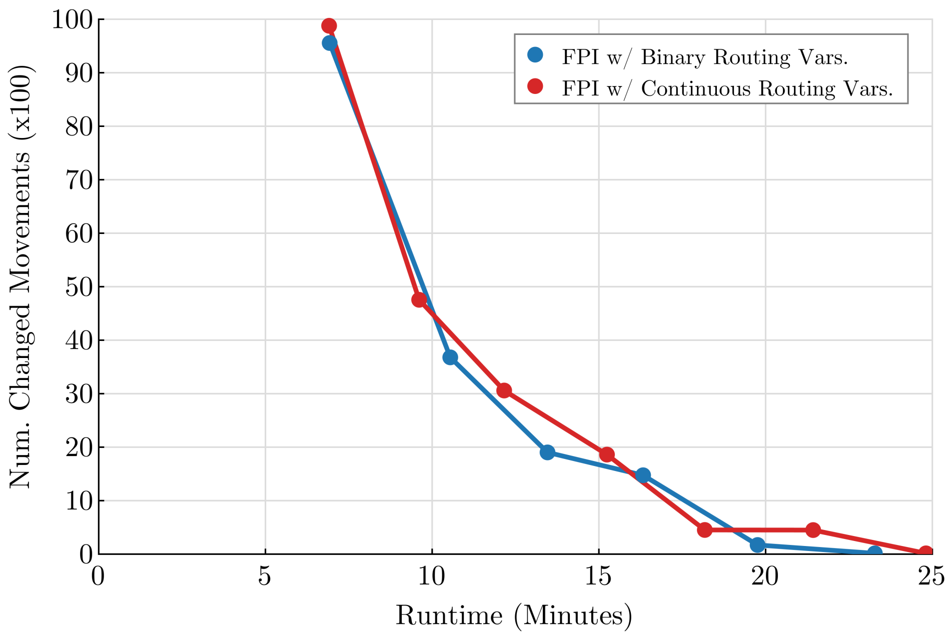


Figure 18: Convergence Behavior of the Fixed Point Iteration Scheme

In Figure 18, the convergence behavior of the FPI algorithm with binary routing variables is shown in blue and the convergence behavior of the FPI algorithm with continuous routing variables is shown in red. The convergence behavior is shown in terms of the hundreds of changed movements from one run to the next, which was the termination criterion. The behavior of the two algorithms is very similar, although the FPI with continuous routing variables required one more iteration and a few more minutes to converge compared to the FPI with binary routing variables. The summary of the two experiments is shown in Table 24.

Table 24: Results of Fixed Point Iteration-Based Solution Generation Approach

Routing Var.	Iter.	Runtime	Opt. Gap	Obj. Diff.
Continuous	7	24.85 min	31.07%	25.78%
Binary	6	23.30 min	30.34%	30.76%

Both cases took nearly the same amount of time, though the continuous case took a little over a minute longer and required an additional iteration to converge. Although it is not explicitly shown in the table, both experiments have similar objective values, but unfortunately, those objective values are not very good compared to the monolithic values. The optimality gap for these cases is calculated relative to the lower bound of the equivalent *monolithic* case. That calculation is,

$$\text{Optimality Gap} = \left(\frac{\text{objective value} - \text{lower bound}}{\text{objective value}}\right) \times 100$$

Of the two, the binary case has the marginally better optimality gap, but the difference is not enough to be significant for these purposes. The similarity is because the objective values are similar to each other and the lower bounds are also similar to each other. There is a more significant difference in the objective function differences. That value was calculated in the same manner as the differences in the monolithic chapter,

$$\text{Difference} = \left(\frac{\text{value} - \text{baseline}}{\text{baseline}}\right) \times 100$$

To calculate the objective value difference, the baseline value was taken as the best-found objective value of the baseline monolithic case. Of the two cases considered in this experiment, the continuous routing variable case had a better objective value difference, but this is primarily a relic of the relatively poor performance of the equivalent monolithic case. The monolithic case with continuous routing variables only achieved a 13.3% gap after 48 hours and despite having a slightly *lower* lower bound than the monolithic case with binary routing variables, it had a higher objective value. Since the continuous and binary routing variable FPI implementations found solutions with similar objective values, the continuous case only looks better because the baseline value was higher.

Ultimately, the fixed point iteration approach is not satisfactorily capable of producing high-quality solutions to the express shipment problem in an acceptable amount of time. A solution with an objective function value that is 25-30% higher than the corresponding monolithic solution translates to a several million dollar difference for a single day of operations. This makes the FPI approach suitable for only the roughest solution estimates. It succeeds in finding a consistent solution, but that solution is not particularly good. This poor performance of the fixed point iteration approach means that hypothesis 3.1 is rejected, and it necessitates a re-evaluation of the hypothesis. This is discussed in the following two sections (§7.3.2 & §7.3.3).

7.3.2 Package Routing with Aircraft Problem

Before deciding how to proceed, the cause of the poor FPI performance must first be discussed. The primary cause of the FPI's lackluster performance is the fact that decision for how to route packages is completely separated from the decision for how to schedule the aircraft, and the FPI is not adequately capable of reconciling that separation. The secondary cause of the FPI algorithm's performance is the Package Routing Problem objective function. When there are no real aircraft scheduled, the PRP becomes a shortest path problem, and when there are real aircraft scheduled in the PRP, those aircraft do not incur a cost to use.

To see how these two factors result in the FPI's poor performance, let us perform a brief thought experiment on how the FPI algorithm would assign package routings and aircraft movements for a single pick up phase. Imagine the first step of the FPI algorithm. No real aircraft are scheduled for the Package Routing Problem so the shortest routing is chosen for each of the commodities. Since every airport sends commodities to every other airport, that means that every airport will be connected to every hub that it is allowed to connect to. Then, in the Aircraft Movement Problem, aircraft movements are assigned to move as many of those packages as possible. These aircraft movements are then held fixed in the PRP. Since routing packages on these real aircraft does not incur a cost, all package routings that can use a real aircraft will use one. These routings are given to the AMP, and the algorithm continues to iterate.

The problem arises in the situation in which it would be more efficient to use a *single large* aircraft rather than *several small* aircraft. Because the starting point of the algorithm results in the connection of each spoke airport to potentially many hubs, the resulting initial AMP solution will be to assign aircraft to move these many small flows. The next iteration of the PRP may assign more packages to fly on the real aircraft, but it will not assign more volume to fly on the aircraft than there is capacity, as a result of the forcing constraint. As a result, on the next iteration of the AMP, there is no reason for the aircraft selection to be changed since the originally-assigned capacity is sufficient for the assigned package routings. Neither problem has the incentive to make the tradeoff to move more packages to a single sort via a larger aircraft and as a result, many aircraft are used and they are used inefficiently.

In short, the problem is that the Package Routing Problem and the Aircraft Movement Problem do not know *enough* about each other. It is not sufficient to provide x_p^k and y_{ij}^f values to the relevant problems, there must be some mechanism for them to recognize the types of trade-offs that the FPI approach misses. To put it colloquially, the algorithm must be able to see the forest *and* the trees. The FPI approach is too myopic, and it misses the forest for the trees.

There are two potential avenues for addressing this issue. One approach would be to use a master-level optimizer that would, somehow, take a "higher-level" view of the optimization problem, and be able to identify these beneficial trade-offs. This would make the heuristic decomposition algorithm more akin to an MDF-like architecture (Multidisciplinary Feasible). The other approach would be to, somehow, reintroduce part of either the Package Routing Problem or the Aircraft Movement Problem in the other problem.

Of the two options, the second option would be more straight-forward to implement. The first option faces a number of implementation-based questions. Primary amongst them is, "What would the master-level optimizer operate on?" If it operated on the package routing and aircraft movement variables directly, it could easily devolve into the monolithic approach, but if new variables are used in the master-level optimizer, what should those variables be and how would you know they were valid additions to the problem? Furthermore, the master-level optimizer would need to be capable of finding very high-quality solutions in a relatively few number of iterations because each FPI analysis takes approximately twenty-five minutes on this computational testbed[3] (assuming this remains constant). Based on the implementation challenges and the goal to not unnecessarily complicate the algorithm, which led to the related conjecture in introduction to this section (§7.3), this first option will not be pursued.

The second option could be implemented by adding aircraft movement variables to the Package Routing Problem. Of the two subproblems, it makes more sense to preserve the arrangement that the Package Routing Problem is solved prior to the Aircraft Movement Problem since demand is needed to assign aircraft movements. In the current arrangement, the primary difficulty is that package routings are being assigned without any regard to how they affect the Aircraft Movement Problem. Since the ESSND problem objective is only a function of the aircraft movements, the best solution to the express shipment problem will be one that most efficiently uses the aircraft. To efficiently use aircraft, package routings must be assigned such that the total volume on a lane is close to but does not exceed the capacity of an available aircraft. This consideration is not taken into account in the current Package Routing Problem.

To introduce the incentive to assign package routings such that they full *whole* aircraft, the integral aircraft movement variables and some of the aircraft-specific constraints can be added to the original Package Routing Problem. Additionally,

[3]On the smaller computational testbed described in Table 22 the FPI approach with integer variables took 64 minutes to find a consistent solution.

since integral aircraft movement variables will be present in the Package Routing Problem, the original ESSND problem objective can be used for the new Package Routing Problem. This solves the question of how to define an objective for the Package Routing Problem. Since this new version of the Package Routing Problem will contain aircraft movement variables, it will be called the *Package Routing with Aircraft Problem* (PRAP).

In order to implement and test the PRAP, its formulation must be established, which means that a decision must be made regarding which AMP-specific constraints to include. First and foremost, the purpose of including aircraft movement variables in the Package Routing Problem was to ensure that package routings were assigned in such a way that whole aircraft would be filled in the final solution. Therefore, it is essential that the aircraft movement variables be integral, which is expressed in the constraint,

$$y_{ij}^f \in \mathbb{Z}_+ \quad \forall (i,j) \in A, f \in F \tag{7.11}$$

For additional guidance on what to include in the PRAP, the sensitivity studies performed on the monolithic approach provide some insight. Most notably, the balance constraints constitute the majority of the purely-aircraft-related constraints and removing these constraints resulted in the greatest runtime improvement of the relaxations. Additionally, the balance and hull count constraints work together, and without one, the other is ineffective. Finally, the feeder limit has only a small effect on the solution and the runtime.

If the most important contribution of the AMP surrogate is to emphasize the advantage of filling integral aircraft, the balance and hull count constraints can be relaxed. The feeder limit constraint can and should remain because without it, feeders could be used in overabundance where a trunk aircraft would be the better choice.

The feeder limit constraint is,

$$\sum_{f \in \hat{F}} y_{ij}^f \leq 1 \quad \forall (i,j) \in A \tag{7.12}$$

Including these two constraints, the objective and the aircraft movement variables in the Package Routing Problem, the Package Routing with Aircraft Problem formulation is,

$$\begin{aligned}
\min \quad & \sum_{(i,j) \in A} d_{ij}^f y_{ij}^f & & \text{(7.13a)}\\
\text{s.t.} \quad & \sum_{k \in K} \sum_{p \in P^k} \delta_{ij}^p b^k x_p^k \leq \sum_{f \in F} u^f y_{ij}^f & \forall (i,j) \in A & \quad \text{(7.13b)}\\
& \sum_{p \in P^k} x_p^k = 1 & \forall k \in K & \quad \text{(7.13c)}\\
& \sum_{k \in K} \sum_{p \in P^k} \delta_h^p n_k x_p^k \leq e_h & \forall h \in H & \quad \text{(7.13d)}\\
& \sum_{f \in \hat{F}} y_{ij}^f \leq 1 & \forall (i,j) \in A & \quad \text{(7.13e)}\\
& x_p^k \in \mathbb{B} & \forall p \in P^k, k \in K & \quad \text{(7.13f)}\\
& y_{ij}^f \in \mathbb{Z}_+ & \forall (i,j) \in A, f \in F & \quad \text{(7.13g)}
\end{aligned}$$

where equations 7.13e&7.13g are the two additional aircraft-related constraints. Of the two, the aircraft integrality constraint (7.13g) is much more significant both in terms of its contributions to runtime and influencing the solution. The feeder limit constraint (7.13e) is included primarily because it is helpful and not expensive to do so.

7.3.2.1 Testing the Package Routing with Aircraft Problem

The convergence behavior of the Package Routing with Aircraft Problem is tested for the baseline problem instance on the experimental testbed described in Section 5.4

using the computational testbed described in Table 5. Since the experiment is performed on the baseline problem instance, no additional dataset filtering is performed, and since the problem formulation has changed, the solution generation approach was adjusted so that only the constraints present in the PRAP were included (i.e. the hull balance and hull count constraints were omitted). Since the PRAP is a MIP, it was solved with Gurobi with a walltime limit of 24 hours. The results are summarized in Table 25 along with the PRP performance for reference.

Table 25: Package Routing with Aircraft Problem Runtime Results

Problem	Optimality Gap	Runtime
PRAP	7.73%	24 hrs
PRP	0%	22.86 sec

The inclusion of the integer aircraft movement variables to the PRP to form the PRAP makes a dramatic difference in the runtime of the problem. The Package Routing Problem can no longer be solved to optimality in a single day, let alone a couple dozen seconds, like the original PRP. However, the integrality of the aircraft movement variables, y_{ij}^{f}, is necessary to ensure that package routings are assigned which fill whole, integral aircraft, and the integrality constraint cannot be relaxed. The long runtime of the PRAP presents a conundrum since any approach that depends on solving the PRAP to optimality will inherit its long runtime.

However, it is important to recall that the PRAP is only, ever, an approximation of the ESSND problem. Therefore, it is only ever going to produce an *approximately* good solution, and reconciling the last remaining optimality gap is just an improvement relative to *that approximation*. True, improving that approximate value may result in an overall better value, but at what expense? At what point is the improvement no longer worth the computational cost? Figure 19, which shows the normalized objective function value and lower bound value as a function of time for the PRAP, can help answer this question. In Figure 19, the black line is the objective function value and the gray line is the lower bound value over time. Both the objective value and lower bound are normalized to protect proprietary information.

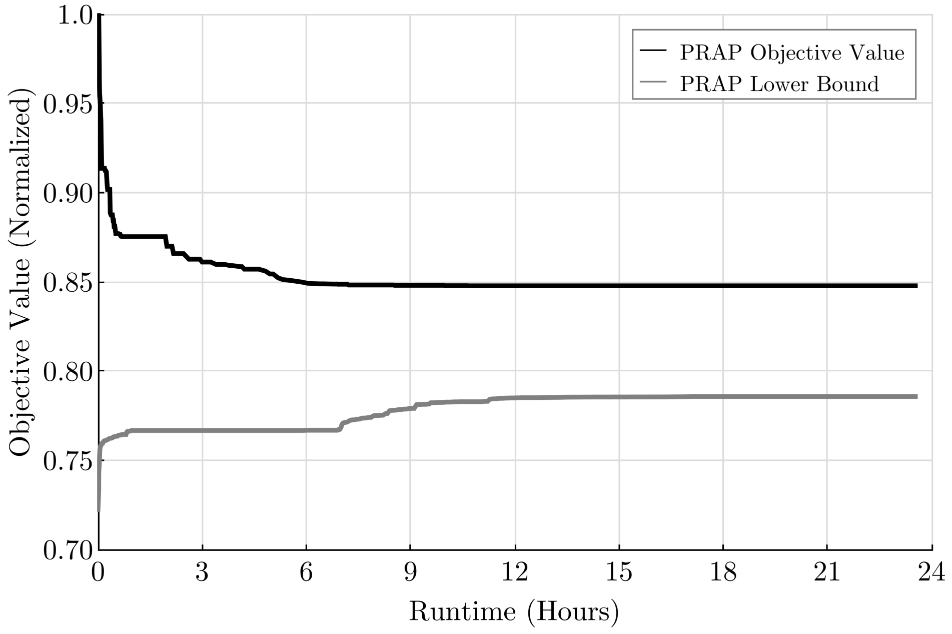


Figure 19: Normalized Objective Value and Lower Bound of the Package Routing with Aircraft Problem

Figure 19 shows that there is an initial sharp drop in the objective function value as the solver removes the high-cost pirate ships. Then, progress briefly flattens out after about an hour before again making moderate progress for the next several hours. After six hours, however, very little progress is made in reducing the objective function value. Some improvement is made to the lower bound, but this is irrelevant for the purposes of finding package routings. Since there is so little improvement in the objective function value after six hours, there is very little incentive to run the PRAP for longer than six hours.

If the PRAP were to be run in an iterative approach, even a six-hour runtime would be potentially problematic. However, since the PRAP includes aircraft movement variables, feedback may likely no longer be needed. The role of the feedback was to introduce and update aircraft-related considerations into the Package Routing

Problem. Since the Package Routing with Aircraft Problem already includes aircraft-related considerations, the feedback is no longer necessary. Furthermore, in the FPI scheme, the information that was being fed back to the Package Routing Problem was *aircraft movements*. Since those are included as variables in the PRAP, it no longer makes sense to feed them back. Therefore, I make the hypothesis,

Hypothesis 3.2. *Since the addition of the aircraft movement variables to the Package Routing Problem adds aircraft-based considerations, I hypothesize that feedback is* not *necessary between the Package Routing with Aircraft Problem and the Aircraft Movement Problem, and a* sequential *algorithm that first solves the PRAP and then the AMP will provide high-quality solutions.*

Since feedback is hypothesized to no longer be necessary, the six-hour runtime for the PRAP is less concerning. The AMP can be solved to optimality relatively quickly, so the total runtime of the algorithm will be primarily sized by the PRAP runtime. Also, the six-hour runtime is a practical *upper limit* on how long the PRAP should be run. It is possible that high-quality solutions could be found with a sequential approach even if the PRAP is terminated with a runtime shorter than six hours. Experiments testing the suitability of a sequential algorithm and the appropriate PRAP runtime are conducted in the next section (§7.3.3).

7.3.3 Basic Sequential Algorithm

As discussed in the previous section, the experiment conducted in Section 7.3.2.1 found that the optimal solution the PRAP could not be found in a reasonable period of time, and therefore an approach that terminated the PRAP prior to find the optimal solution was proposed. This section tests that idea by implementing a sequential algorithm for a set of different of PRAP termination times. The dual purpose of these experiments is to test the sensitivity of the approach to the runtime of the PRAP and to also test whether the sequential approach can produce good quality solutions. This sequential algorithm is called the *Basic Sequential Algorithm* because it will be modified further in following sections.

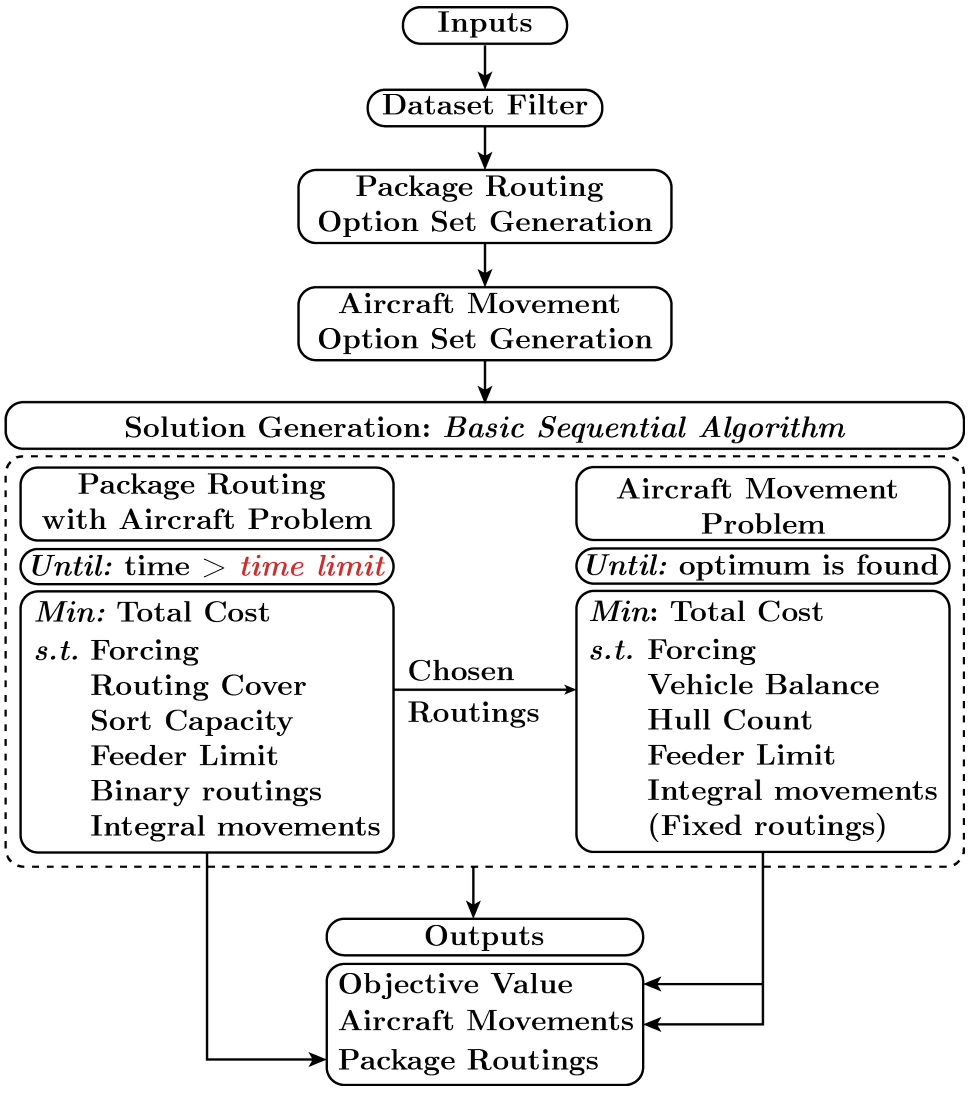


Figure 20: Basic Sequential Algorithm

Figure 20 shows the Basic Sequential Algorithm in the context of the computational testbed. All of the experiments are generated using the baseline problem instance described in Table 6 so the first four steps are not modified for the experiments. The package routing and aircraft movement option set generation processes are the same as those described in Sections 6.1, 6.2 & 6.3 of Chapter 6, which focused on the monolithic approach. The solution generation procedure, which is the Basic

Sequential Algorithm, is what is being tested in the experiments of this section. In the Basic Sequential Algorithm, the Package Routing with Aircraft Problem is solved first for a given amount of time. Then, the Aircraft Movement Problem is solved to optimality using the fixed package routings set in the PRAP. The PRAP provides the package routings in the final solution, and the AMP provides the final objective value (total cost) and the aircraft movements (schedule).

Both the PRAP and the AMP are mixed-integer programs and solved with Gurobi. The entire algorithm is run on the computational testbed summarized in Table 5 for the baseline problem instance described in Table 6. Experiments were run for different PRAP time limits from two minutes to six hours. The results are summarized in Table 26. The second column is the optimality gap of the PRAP when it was

Table 26: Results of the Sequential Algorithm

PRAP Runtime	PRAP Opt. Gap	Objective Difference	Absolute Opt. Gap	AMP Runtime
2 min	24.0%	18.27%	23.70%	320 sec
5 min	17.3%	13.46%	20.46%	362 sec
10 min	14.8%	9.80%	18.62%	230 sec
30 min	13.8%	9.46%	17.55%	283 sec
1 hr	13.0%	9.12%	17.30%	270 sec
3 hrs	12.4%	8.45%	16.79%	216 sec
6 hrs	9.8%	5.77%	14.68%	198 sec

terminated. The third and fourth columns pertain to the quality of the final solution relative to the monolithic approach baseline. The fifth column is the time it took for the AMP to solve to optimality after the PRAP had been terminated. Overall, the algorithm actually performs quite well. The best optimality gap was less than 15%, which was significantly lower than the optimality gap of the FPI-based approach, which had an optimality gap of 30%. As expected, the longer the PRAP was run, the better the final solution was, and the rate of improvement dropped off as the PRAP runtime increased, thus requiring longer runtimes to produce the same amount of improvement. These results and how they compare to the monolithic approach are summarized visually in Figure 21.

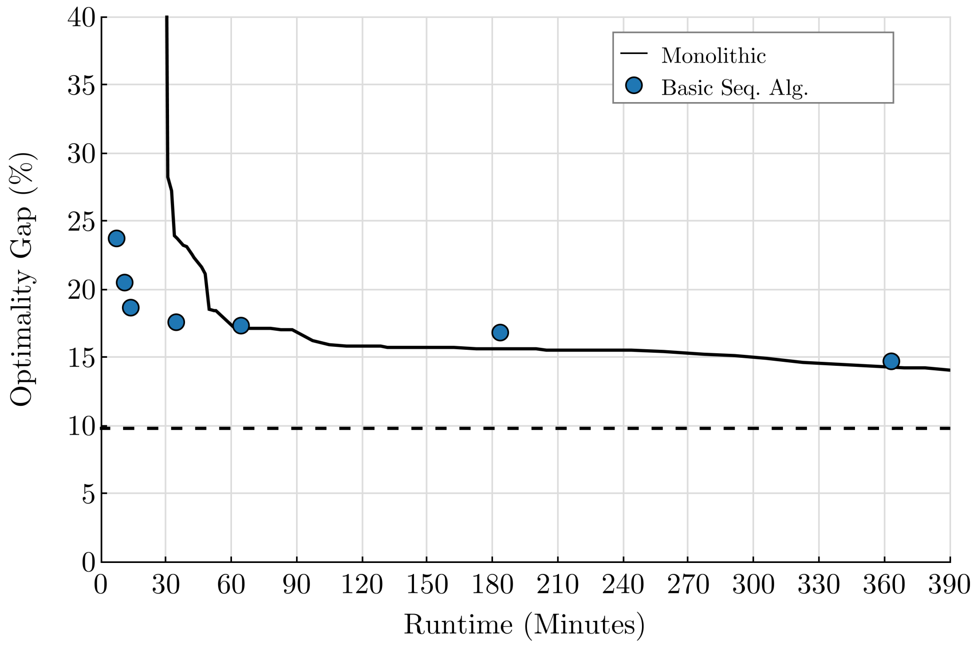


Figure 21: Sequential Algorithm Implementation Compared to Monolithic Approach for Baseline Problem Instance

The solid black line is the convergence behavior of the monolithic approach for the baseline problem instance. The blue dots are the individual Basic Sequential Algorithm runs. The sequential algorithm is most successful compared to the monolithic approach in the first hour of the runtime, and it is able to produce solutions during the initial period when the monolithic approach is incapable of finding a feasible solution. Of these solutions, the case in which the PRAP was run for five minutes out-paces the monolithic approach by the widest margin – it takes the monolithic approach nearly forty additional minutes to find a solution with the same optimality gap. The case in which the PRAP was run for ten minutes is faster than the monolithic by a slightly smaller margin, beating the monolithic approach by thirty-five minutes, and the two-minute and thirty-minute cases both out-pace the monolithic approach by a little less than thirty minutes. After the thirty-minute case, the sequential approach generally keeps pace with the monolithic approach, but it does not find better solutions than the monolithic approach for a given period of time after the thirty-minute mark.

The primary advantage of the sequential algorithm is that it does not struggle to produce good feasible solutions, even when the PRAP is only run for a short period of time. By comparison, the monolithic approach initially struggles to find its first solution, but once it does, the improvement is dramatic for a short period of time, after which the improvement rate is relatively flat. The solutions that the sequential algorithm is able to find in that initial timespan during which the monolithic approach has not found any good feasible solutions is where it has the advantage. And this is to be expected, this niche of finding good solutions in a short time is exactly where heuristics typically excel.

Additionally, since the PRAP does not include all of the aircraft-related considerations (i.e. hull count and balance), there is a limit to the quality of solution that can be produced using it. The monolithic approach surpasses the heuristic approach when the shortcomings of the approximation outweigh its runtime benefits. For the solution method domain of finding good solutions in a short time, the sequential decomposition heuristic algorithm is more successful than the monolithic approach. This means that hypothesis 3.2 is supported and, by extension hypotheses 1.2 and 2.2 are also supported.

The results of the Basic Sequential Algorithm are encouraging. The algorithm is capable of finding good-quality solutions to the express shipment problem faster than the baseline monolithic approach for nearly entire first hour of runtime, and it is capable of keeping pace with the monolithic approach for runtimes longer than an hour. The Basic Sequential Algorithm is, however, just a first prototype of a sequential heuristic decomposition algorithm. This raises the question,

Research Question 4. *How can the Basic Sequential Algorithm be improved?*

The sequential decomposition heuristic has been shown to satisfactorily produce solutions in a satisfyingly simple approach, but perhaps with further refinement, it could be improved. That discussion is handled in the next two sections (§7.4 & 7.5).

7.4 Collapsed Product Set Sequential Algorithm

As the first of the two problems in the sequential heuristic algorithm, the PRAP has the greatest impact on the solution quality. In fact, since the AMP is solved to optimality for the package routings it is given, the solution of the PRAP completely determines the ultimate quality of the final solution. Therefore, the PRAP should be the focus of improving the Basic Sequential Algorithm.

Since the PRAP is an approximation of the full ESSND problem formulation, there are two natural avenues for improvements focused on the PRAP. One avenue would be to further reduce the size of the PRAP problem. Since the problem is a relaxed as it can be without sacrificing solution quality, this would be done by reducing the size of the problem instance that the PRAP solves with the goal of finding good solutions more quickly. The other avenue would be to improve the PRAP approximation by adding additional constraints, but this would need to be done without dramatically increasing the runtime required to obtain good solutions from the PRAP. This section focuses on the first of the two options, and the next section (§7.5) focuses on the second option.

7.4.1 Collapsed Product Set Package Routing Problem

To see how problem instances of the Package Routing with Aircraft Problem might be reduced, let us first consider the fundamental role of the PRAP so that it may be preserved while the extraneous details are identified and omitted. The key is that the essential purpose of the Package Routing with Aircraft Problem is to assign package routings, and at their simplest, a package only needs an assignment to a sort activity instance to be complete since the origin and destination activity instances are known based on the commodity. Some more complex package routings may include consolidation stops, but ultimately, the essential purpose of the PRAP is to solve the underlying *sort assignment* problem.

The difficulty of the sort assignment problem has been previously discussed in Section 6.5 in regards to the secondary sort threshold sensitivity study, and in Section 7.2.1 in regards to the challenges of a spatial decomposition definition. To summarize the discussions of those previous sections, sort membership is difficult because it is subject to extensive network effects. Choosing to route a commodity through a particular sort affects not only that commodity but also all of the other commodities originating at that same origin, all of the commodities terminating at that commodity's destination, *and* all of the markets that also connect to that sort. These extensive ripple effects mean that even small changes may have wide-ranging effects on the solution, and it may be very difficult to find the true optimal configuration. A key element of this challenge is that a good solution is one in which the aircraft are loaded as fully as possible, since aircraft, not packages, incur the cost and empty cargo space is waste. This is what led to the inclusion of aircraft movement variables in the Package Routing Problem.

This means that it is important to preserve the PRAP's role and ability to make sort assignments for commodities, but to identify which parts of the problem instance can be pared away, let us consider several important characteristics of the express shipment problem,

1. Express shipment volume and routing decisions are strongly driven by the box volume. It, by far, constitutes the most volume of the three types (i.e. boxes, documents, and freight).
2. Separate equipment is used to sort boxes, documents, and freight, and of the three types, the box volume is consistently the most sorting-constrained of the three types.

3. Express shipment operations are strongly periodic and can be broken into two daily cycles.
4. The night cycle moves more volume and is more time-constrained than the day cycle.

To summarize, there are many commodities in the express shipment problem, but many of these commodities constitute very little volume despite contributing variables to the optimization problem. Therefore, while they increase the problem size, they are not the primary drivers of the final solution (in particular, the aircraft schedule). Of all the products, the box products constitute the bulk of the volume and are the primary drivers of the aircraft size. Of the box products, priority boxes constitute the most volume.

For all problem instances solved up to this point, commodities that were identical in their origin-destination pair and differed only in their *product type* (i.e. box, document or freight) and/or number of commit days[4] were tracked separately because they were handled differently at the sorting facilities. Since each commodity's routing options are represented as variables in the model, this results in a variable set that may be significantly larger than necessary because many of the variables (i.e. those corresponding to non-box products) will have little overall impact on the aircraft sizing or final solution. Commodities that are similar in their origin-destination pair, but dissimilar in their product type and/or commit days, will be called *analogous commodities.*

Using the above observations of the express shipment problem, I make the hypothesis,

Hypothesis 4.1. *Based on the observations that 1) in an express shipment problem instance there exist many commodities that may only differ by the product type*

[4]i.e. the time to deliver the commodity.

and/or commit day, and 2) of these similar commodities, the priority box commodities constitute the largest volume, face the tightest sort capacities, and are the most time-constrained, I hypothesize that the Basic Sequential Algorithm can be improved by solving the Package Routing with Aircraft Problem for the problem instance that only includes the overnight box volume, and assigning the routings for all commodities based on the routing assigned for the analogous priority overnight commodity.

This strategy attempts improve the package routing assignment step of the sequential heuristic algorithm by reducing the size of the problem that the PRAP needs solve, thereby generating good solutions more quickly. For example, if the solution that the Basic Sequential Algorithm generates in sixty minutes of PRAP runtime could be generated even ten minutes faster, then the heuristic approach would outpace the monolithic approach for the generation of a solution of that quality. By using only the overnight priority box commodities to determine the routings for all other commodities with the same origin-destination pair, the product set is *collapsed* for the purposes of the Package Routing with Aircraft Problem. This is why this version of the sequential algorithm is called the *Collapsed Product Set Sequential Algorithm.*

In order to begin to study hypothesis 4.1, experiments testing the runtime of the Package Routing Problem with the collapsed product set need to be conducted. However, before that can be done, one additional consideration needs to be taken into account. Although priority box volume constitutes the bulk of the overnight box volume, there are actually three different overnight box product types, priority box, First Overnight box[5], and overnight postal boxes.

The First Overnight and overnight postal box volume needs to be included in the PRAP in order to not break the overnight box sort capacities (particularly at MEM), even though their total volume is smaller than the priority volume. Including these

[5]First Overnight boxes are those that behave similarly to priority boxes for the purposes of line haul but are delivered earlier in the morning for last mile service.

products as free variables would defeat some of the advantages of solving a reduced-size problem. Therefore, their routing options are constrained to be equal to their analogous priority box product. That constraint is simply,

$$x_p^{k^\dagger} = x_p^{k'} \quad \forall k' \in K' \tag{7.14}$$

where $k^\dagger$ is the analogous priority box commodity for the First Overnight box or overnight postal box k' and K' is the set of all First Overnight and overnight postal box commodities. This will be called the *analogous commodity* constraint.

The addition of this analogous commodity constraint modifies the PRAP formulation so in order to differentiate the two formulations, and highlight when the package routing problem is being solved over the collapsed product set, this new formulation will be called the *Collapsed Product Package Routing with Aircraft Problem* (CP-PRAP).

Several experiments on the CP-PRAP were conducted on the baseline problem instance. All experiments were conducted on the experimental testbed described in 5.4 and using the computational testbed described in Table 5. In these experiments, both the dataset filter and solution generation processes were altered from the monolithic baseline approach. The dataset filter step, in addition to its normal reductions to remove international, three-day and live-animal products, also removed any commodities that were not overnight box products. The solution generation process was altered to generate the appropriate optimization model (i.e. CP-PRAP). The resulting MIP optimization model was solved using Gurobi.

Four experiments were conducted to test the CP-PRAP and potential variations of it. Specifically, the effect of including consolidation was tested, and the effect of including the vehicle balance and hull count constraints in the optimization problem was tested. The effect of including consolidation opportunities in the package routing option set was tested because it is another potential way to reduce the size of the problem. Consolidation is only applicable for some markets, but it does increase the number of routing options. The effect of re-introducing the vehicle balance and hull count constraints into the CP-PRAP optimization problem was tested to explore whether collapsing the product set sufficiently reduces the runtime of the ESSND without needing to omit the vehicle balance and hull count constraints. The resulting optimization problem is called the *Collapsed Product Set Express Shipment Service Network Design* (CP-ESSND) problem. The results of these experiments are shown in Table 27

Table 27: Collapsed Product Set Package Routing Problem Experimental Results

Formulation	Consolidation	Optimality Gap	Time To 5% Gap	Total Runtime
CP-PRAP	True	4.72%	8.96 hrs	24 hrs
CP-ESSND	True	2.80	11.96 hrs	24 hrs
CP-PRAP	False	0.84%	6.93 min	24 hrs
CP-ESSND	False	0.92%	1.26 hrs	24 hrs

Although none of the problem instances finished after 24 hours of runtime, both of the consolidation-free instances were close. The cases with consolidation, which were larger because they included consolidation, also performed well, but not nearly as well as the cases without consolidation. The CP-PRAP case without consolidation was the fastest of the four. It had found a solution with an optimality gap of less than 5% in under seven minutes. It did not converge to an optimal solution in less than 24 hours, but this is again a result of the difficulty to prune branches from the B&B tree for problems with weak lower bounds. The CP-ESSND without consolidation case had a similar optimality gap after 24 hours but took much longer to find its first solution under a 5% gap, which indicates a slower rate of convergence. The cases with consolidation, by comparison, took several hours to find solutions under a 5% gap and did not achieve as low of an optimality gap as the cases with consolidation.

At this point, it is not clear whether the CP-PRAP or CP-ESSND would produce better package routings in a given time period, but it is clear that both problems can be solved faster when consolidation is not included. This means that consolidation would need to be re-introduced into the final solution at some point in order for the solutions to be comparable to the baseline monolithic approach. In response to this need, the *Consolidation Assignment Problem* (CAP) is developed in the next section as well as the overall structure of the sequential algorithm using the CP-PRAP/CP-ESSND to assign the (non-consolidating) package routings.

7.4.2 Collapsed Product Set Sequential Algorithm

In the Section 7.4.1, the idea of collapsing the product set when assigning package routings was introduced. This section tests whether using that idea improves the performance of the sequential algorithm. The resulting algorithm that is developed is called the *Collapsed Product Set Sequential Algorithm*.

The Collapsed Product Set Sequential Algorithm is similar to the Basic Sequential Algorithm, but with two essential differences. First, the optimization problem to assign the priority box routings only considers the overnight box products. Therefore, an additional step must be performed after the priority box routings are assigned in order to assign routings to all of the other commodities before the Aircraft Movement Problem is run. Second, based on the results of the experiments in Section 7.4.1, consolidation is not included in the optimization problem to assign the package routings so consolidation routings must also be assigned in a separate optimization problem called the *Consolidation Assignment Problem* (CAP).

7.4.2.1 Consolidation Assignment Problem

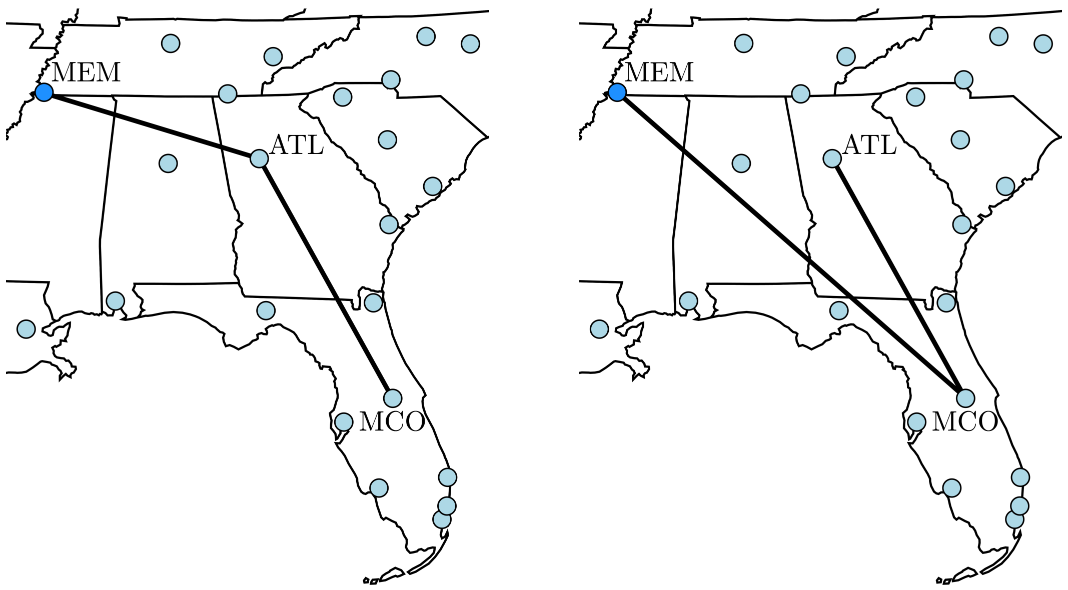
In the baseline approach to including consolidation, options are generated when the package routing options are made. These options are then part of the package routing option set which is given to the solver. Part of the challenge in making these “a priori” consolidation options is that consolidation is appropriate where it helps to more fully load an aircraft, but those aircraft loadings are not known until a solution is generated. Therefore, unless the entire set of consolidation options is included in the package routing option set, which was shown to be too burdensome on the optimizer, the best that can be done is an educated guess on which options may be good options to include. However, if consolidation is added *after* a solution has been generated, it is trivial to find underloaded flights and identify good consolidation opportunities. Since the experiments of Section 7.4.1 motivated me to remove consolidation options from the CP-PRAP and CP-ESSND, there is some flexibility in when consolidation can be performed. Thanks to the comparative ease of an "a posteriori" consolidation approach, the Consolidation Assignment Problem is arranged to be the final step of the Collapsed Product Set Sequential Algorithm, and consolidation-based routings are assigned *after* an aircraft schedule has been generated in the Aircraft Movement Problem.

This Aircraft Movement Problem produces a solution that does not include any consolidation. Previously, when consolidation options were considered in the initial routing option set, good consolidation candidates had to be inferred based on market size. Now that aircraft movements have been determined, it is much easier to identify flights that would be good candidates for consolidation or a two-legged flight[6] because the aircraft loadings are known. The problem of adding consolidation routings will be called the *Consolidation Assignment Problem* (CAP).

To generate the option set for the CAP, first underloaded flights are identified. A flight is considered a candidate for consolidation if it is less than 75% loaded in the AMP solution. Then, the set of all feasible opportunities is generated. Next,

[6]The primary difference between a consolidation and a two-legged flight is the equipment involved. A consolidation uses a truck or feeder aircraft to move volume from a small spoke to a larger spoke for pick up phases (and similarly for delivery). That volume then moves to the hub on a trunk aircraft. A two-legged flight moves all of this volume using a single trunk aircraft. For brevity, the term "consolidation" will be used to refer to both true consolidation and two-legged flights because the method of creating options is the same.

where "chiral" options exist, the longer option is eliminated. For example, two chiral options during a pick up phase might be MCO[7] to ATL to MEM, and ATL to MCO to MEM as shown in Figure 22. Both options may be time-feasible, but the second option is clearly inferior to the first so it would be removed.



(a) Superior Consolidation Option (b) Inferior Consolidation Option

Figure 22: An Example of Chiral Consolidation Options

After the inferior chiral options are removed, the option set is additionally filtered to only consider opportunities that have a symmetric delivery/pick-up option across phases. That is, for the previous MCO to ATL to MEM opportunity pick up, the option would be retained if the MEM to ATL to MCO option also existed on the delivery side of the prior phase. Symmetric options are preferable because they are more likely to be useful. If the pick-up-side consolidation opportunity did not exist, two separate aircraft would be needed. If only one aircraft was used to serve both markets in the delivery phase, where would this second aircraft come from? If it would need to be repositioned, that defeats the benefit of using consolidation. To avoid presenting this complication to the optimizer, asymmetric options are removed.

[7]i.e. Orlando International Airport

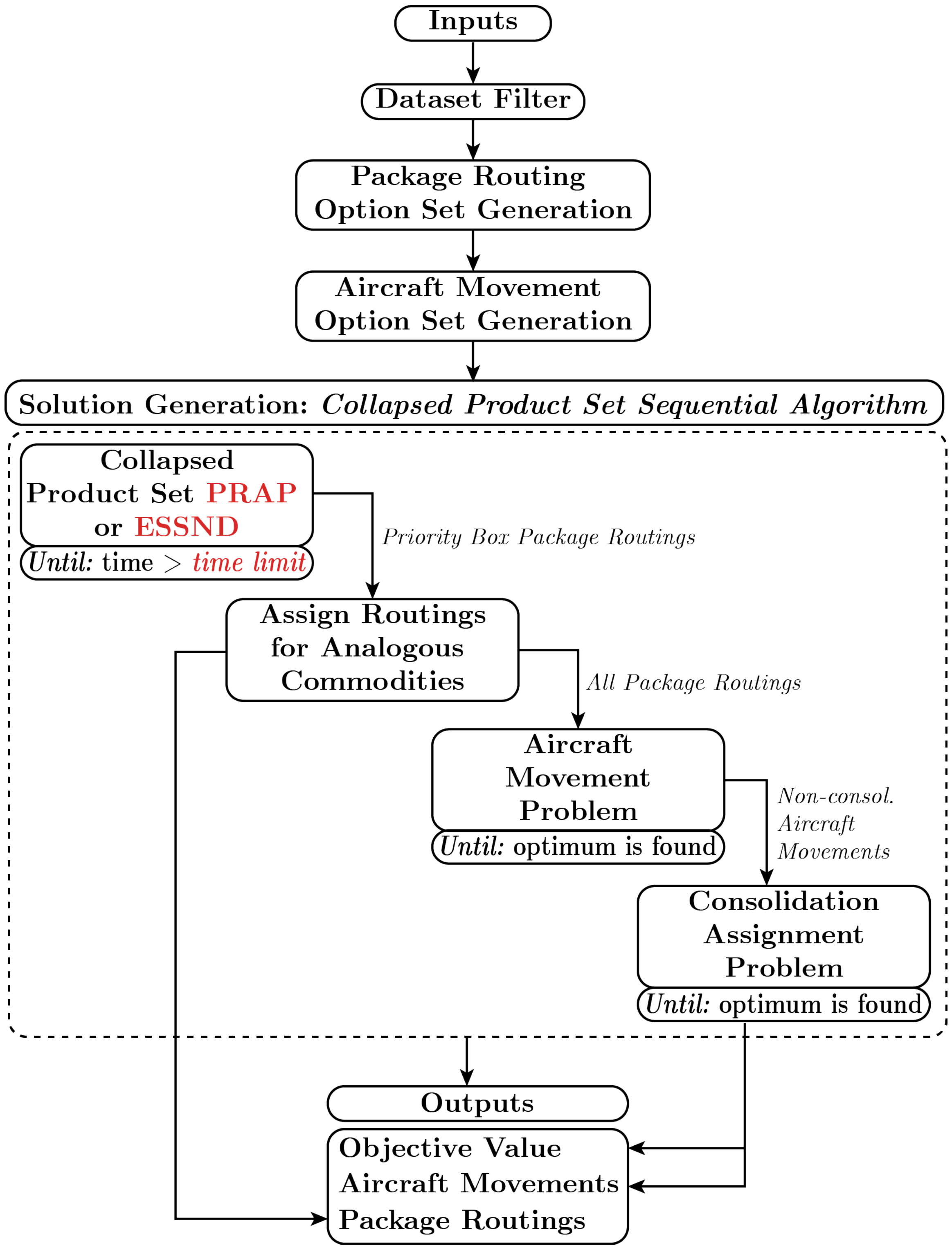
Finally, after finding that a large number of options still existed even for a limited number of consolidation-eligible markets, the shortest option for each consolidation-eligible market was chosen. These new routing options were generated for the applicable commodities and the necessary arcs were added to the time-space graph. Then, a modified version of the AMP was solved in which both the design (vehicle) variables *and* the flow variables of the commodities with new consolidation options were unfrozen. The commodities with no consolidation routings remained frozen. The resulting problem was then solved with Gurobi. Since the CAP is larger than the AMP thanks to additional flow variables, it is important to keep the number of consolidation opportunities low so as to not overwhelm the CAP with options.

7.4.2.2 Structure of the Collapsed Product Set Sequential Algorithm

The Collapsed Product Set Sequential Algorithm is slightly more involved than the Basic Sequential Algorithm. Rather than just two steps (PRAP and AMP), it has four steps. The entire flow of the algorithm within the experimental testbed is shown in Figure 23.

The first several steps of the experimental testbed are unchanged, except that consolidation options are not included in the package routing option set. The methods for generating the package routing and aircraft movement options sets are described in Sections 6.2, 6.1, and 6.3. Then, within the solution generation step, there are four sequential processes. First, the CP-PRAP or ESSND is run for a set period of time. Determining which of the two optimization problems produces better package routings will be one focus of the experiments, and exploring the effect of different runtimes will be the other focus. Then, the priority box package routings from the CP-PRAP or CP-ESSND are used to assign routings to all of the other commodities.

The process for assigning routings based on the analogous priority box routing is straight-forward. If a commodity has the option to route in the same fashion that its



Inputs
Dataset Filter
Package Routing Option Set Generation
Aircraft Movement Option Set Generation
Solution Generation: Collapsed Product Set Sequential Algorithm
Collapsed Product Set PRAP or ESSND
Until: time > time limit
Priority Box Package Routings
Assign Routings for Analogous Commodities
All Package Routings
Aircraft Movement Problem
Until: optimum is found
Non-consol. Aircraft Movements
Consolidation Assignment Problem
Until: optimum is found
Outputs
Objective Value
Aircraft Movements
Package Routings

Figure 23: Collapsed Product Set Sequential Algorithm

analogous priority box product was assigned to route, then it is assigned that option. If a commodity does not have the option to route in the same way, which can happen when a market originates significantly less volume during the day than at night, then that commodity is assigned to route through the primary hub at MEM.

All two-day volume must sort during the day, but since they originate at night, those products have the option of moving either with the overnight volume and waiting at the sort facility, or first waiting at the spoke and then moving with the daytime volume to the daytime sort. This means that a two-day product may have two possible routing options – one that moves early and one that moves late. Since the company prefers that packages do not use the sorts as warehouses (i.e. wait there for the next sort activity), and the total same-day volume is less than the total overnight volume, all two-day commodities are assigned to move with the daytime (i.e. same-day) volume to the sorting facility assigned by the analogous primary box commodity.

After all of the package routings are assigned, then the Aircraft Movement Problem is solved. The AMP is exactly the same as it was for the Basic Sequential Algorithm. Finally, using the aircraft schedule generated by the Aircraft Movement Problem, the consolidation options are generated as described in Section 7.4.2.1 and the Consolidation Assignment Problem is solved.

7.4.2.3 Testing the Collapsed Product Set Sequential Algorithm

The Collapsed Product Set Sequential Algorithm experiments were performed on the testbed described in Section 7.4.2.2. The CP-PRAP, CP-ESSND, AMP, and CAP are all mixed integer programs and solved using Gurobi. The process of assigning the package routings based on the analogous priority box package routings (Step 2 in the Solution Generation) does not require optimization, and is performed as described. Experiments are performed testing the effect of using the CP-PRAP versus the CP-ESSND to assign the priority box package routings, and the effect of the runtime of those optimization problems on the final solution.

Since the monolithic approach surpassed the Basic Sequential Algorithm in performance for cases in which the PRAP was run for an hour or more, the Collapsed Product Set Sequential Algorithm experiments only include cases in which the CP-PRAP or CP-ESSND was run for times between two minutes and one hour. All of the experiments were performed on the computational testbed described in Table 5 for the baseline problem instance described in Table 6. The results of those experiments are shown in Table 28.

Table 28: Solution Quality Results of the Collapsed Product Set Sequential Algorithm

PR Runtime	PR Problem	PR Opt. Gap	Abs. AMP Opt. Gap	Final Opt. Gap	Total Time
2 min	CP-PRAP	9.01%	23.81%	22.81%	7.95 min
5 min	CP-PRAP	8.61%	23.36%	22.76%	18.59 min
10 min	CP-PRAP	4.19%	22.26%	21.48%	21.23 min
30 min	CP-PRAP	2.88%	20.95%	20.25%	38.76 min
60 min	CP-PRAP	2.86%	20.95%	20.25%	71.97 min
2 min	CP-ESSND	28.22%	24.58%	23.79%	9.07 min
5 min	CP-ESSND	13.70%	21.76%	21.05%	16.05 min
10 min	CP-ESSND	10.70%	20.77%	20.08%	14.48 min
30 min	CP-ESSND	8.98%	20.99%	20.34%	43.64 min
60 min	CP-ESSND	4.33%	20.13%	19.49%	68.90 min

In Table 28, the routing optimality gap column indicates the optimality gap of *that* individual CP-PRAP or CP-ESSND problem when it was terminated. The absolute AMP optimality gap and the final optimality gap (after the CAP is solved) are relative to the best (highest) lower bound found through the monolithic approach. Except in one case, running the package routing optimization problem for a longer time period improved the quality of the solution, but the rate of improvement was much flatter than that of the Basic Sequential algorithm. The exception was the case in which the CP-ESSND was run for thirty minutes. The ten minute case actually performed marginally better, but this may just be the result of a "lucky bounce" – it unknowingly found a slightly better solution than the longer case. Additionally, it is interesting that for the CP-PRAP thirty and sixty minute cases, the additional thirty minutes of runtime did not contribute to *any* solution quality improvement. This indicates that a 20.25% gap may be the best solution that algorithm definition can accomplish. Somewhat surprisingly, there is very little difference between the solutions generated by the cases using the CP-ESSND versus the CP-PRAP. At most, they differ by a couple percentage points on the final optimality gap.

Ultimately, however, the Collapsed Product Set Sequential Algorithm is inferior to the Basic Full-Set Sequential algorithm for the baseline problem instance. This can be best seen in Figure 24 where the results of the SAP experiments have been plotted as well as the results of the Basic Full-Set Sequential algorithm and the convergence behavior of the monolithic approach.

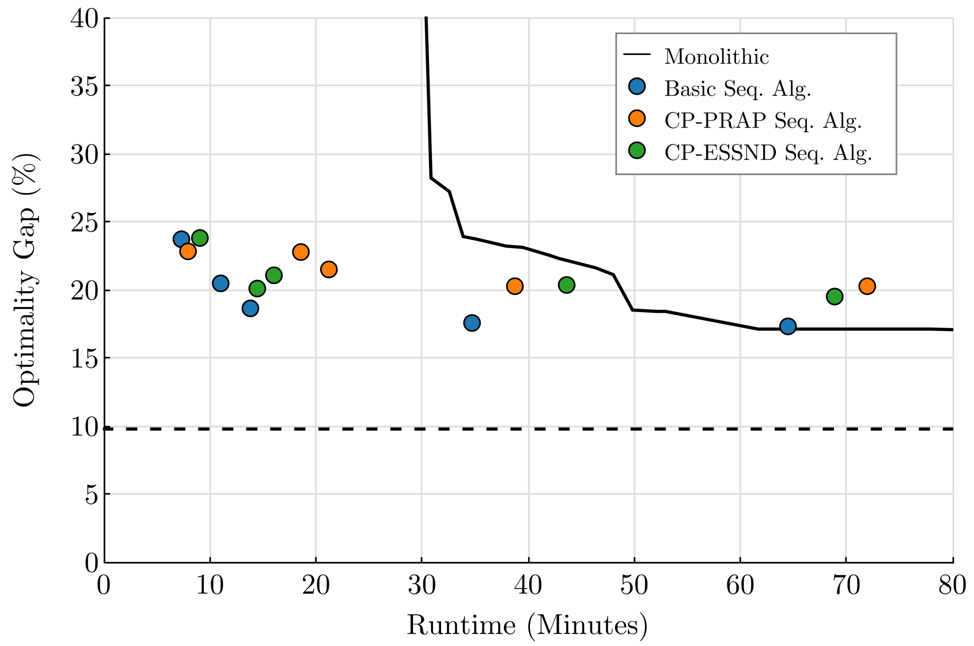


Figure 24: Collapsed Product Set Sequential Algorithm and Basic Sequential Algorithm Compared to Monolithic Approach for Baseline Problem Instance

Except for a marginally better solution found for the CP-PRAP two-minute case, the Basic Full-Set Sequential algorithm produces superior results to either of the two Collapsed Product Set Sequential algorithm variants. Also, whereas the Basic

Sequential Algorithm is on par with the monolithic approach for cases in which the PRAP was run for sixty minutes or longer, the Collapsed Product Set Sequential Algorithm produces solutions that are worse than the monolithic approach by a couple percentage points.

These results are the direct result of forcing the daytime products to move the same as the nighttime products. When the two phases are independent of each other, that additional flexibility allows for routings to be more intelligently assigned to fill aircraft. One of the fundamental assumptions of the SAP is that the daytime volume will be similar, though smaller, than the analogous nighttime volume. That is not necessarily true. Although in the aggregate sense, there is less daytime volume than nighttime volume, some markets actually produce more volume during the day. This disconnect between the nighttime and daytime products leads to sub-optimalities in daytime aircraft loadings and therefore a worse solution. Additionally, all of the two-day volume is required to wait at the spokes and move with the daytime volume despite originating at night. This fails to use empty space on nighttime flights whose cost is already committed. If a larger aircraft is needed to move all of the two-day and same-day volume because the two-day volume is forced to move during the day, this is also a likely source of sub-optimality.

Furthermore, the solution improvement produced by the CAP hardly seems worth the additional computational cost. As seen in Table 29, the CAP takes longer to solve than the AMP in all cases, but as seen in Table 28, it improves the solution by at most a couple percentage points in the optimality gap. The fact that it takes longer to run than the AMP is not a surprise since it is a larger problem, but it provides so little improvement that it is hard to justify its inclusion as it is formulated and implemented here.

In the next section, the Collapsed Product Set Sequential Algorithm will be slightly modified to allow for the two-day volume to choose to move either with

Table 29: Time Breakdown Results of the Collapsed Product Set Sequential Algorithm

PR Runtime	PR Problem	AMP Time	CAP Time	AMP + CAP Time	Total Time
2 min	CP-PRAP	139 sec	218 sec	5.95 min	7.95 min
5 min	CP-PRAP	286 sec	530 sec	13.59 min	18.59 min
10 min	CP-PRAP	238 sec	436 sec	11.23 min	21.23 min
30 min	CP-PRAP	215 sec	311 sec	8.77 min	38.76 min
60 min	CP-PRAP	291 sec	427 sec	11.97 min	71.97 min
2 min	CP-ESSND	122 sec	302 sec	7.07 min	9.07 min
5 min	CP-ESSND	303 sec	360 sec	11.05 min	16.05 min
10 min	CP-ESSND	75 sec	194 sec	4.48 min	14.48 min
30 min	CP-ESSND	288 sec	530 sec	13.65 min	43.64 min
60 min	CP-ESSND	191 sec	343 sec	8.90 min	68.90 min

the night cycle or with the day cycle volume. This will test whether forcing the two-day volume to move exclusively with the daytime volume is a serious source of sub-optimality.

7.4.3 Collapsed Product Set Sequential Algorithm with Flexible Deferred Volume

Since the choice to force all two-day volume to move during the daytime was identified as a potential source of sub-optimality, experiments were run in which two-day volume was given the choice, in the AMP, to move to its assigned sort either during the night or the day. The algorithmic set up was nearly identical to the Collapsed Product Set Sequential algorithm described in the previous section, with the exception that two-day commodities had two routing options in the AMP rather than being completely frozen. All other commodities' routings were still fixed based on the results of the package routing optimization problem.

All of the experiments were run using the experimental testbed described in Section 7.4.2.2. The CP-ESSND was used, rather than the CP-PRAP because it gave marginally better results for more of the cases in the experiments of Section 7.4.2. Like the previous experiments on the Collapsed Product Set Sequential Algorithm, cases were run for different runtimes of the CP-ESSND, from two minutes to one hour. All cases were run on the computational testbed described in Table 5 using the baseline problem instance described in Table 6. The results are presented in Table 30 and the times of the individual parts are presented in Table 31.

Table 30: Solution Quality Results of the Collapsed Product Set Sequential Algorithm with Flexible Two-Day Volume

PR Runtime	PR Opt. Gap	Abs. AMP Opt. Gap	Final Opt. Gap	Total Time
2 min	28.22%	23.90%	23.12%	11.00 min
5 min	13.70%	21.20%	20.45%	16.47 min
10 min	10.70%	20.02%	19.32%	16.87 min
30 min	8.98%	20.38%	19.81%	42.07 min
60 min	4.41%	19.63%	19.21%	66.50 min

Overall, the trends are very similar to the Collapsed Product Set Sequential algorithm without flexible two-day volume. This is hardly surprising since the CP-ESSND solution is the same. Allowing the two-day volume to choose between moving at night and moving during the day does have an effect, but it is low. Solution quality still hovers at around a 20% optimality gap. Allowing the two-day volume to be flexible does increase the AMP runtime, but the increase, while noticeable is not egregious. Those runtimes are found in Table 31.

Table 31: Time Breakdown Results of the Collapsed Product Set Sequential Algorithm with Flexible Two-Day Volume

PR Runtime	AMP Time	CAP Time	AMP + CAP Time	Total Time
2 min	378 sec	162 sec	9.00 min	11.00 min
5 min	433 sec	250 sec	11.47 min	16.47 min
10 min	228 sec	184 sec	6.87 min	16.87 min
30 min	489 sec	235 sec	12.07 min	42.07 min
60 min	240 sec	150 sec	6.50 min	66.50 min

The overall, the trends are as expected. Allowing more flexibility in the AMP does increase that problem's runtime, but it is not prohibitive. In general, the Collapsed Product Set Sequential Algorithm with Flexible Two-Day Volume does produce slightly better solutions than the other Collapsed Product Set Sequential Algorithm, as seen in Figure 25, but those solutions are still not as good as the solutions produced by the Basic Sequential algorithm.

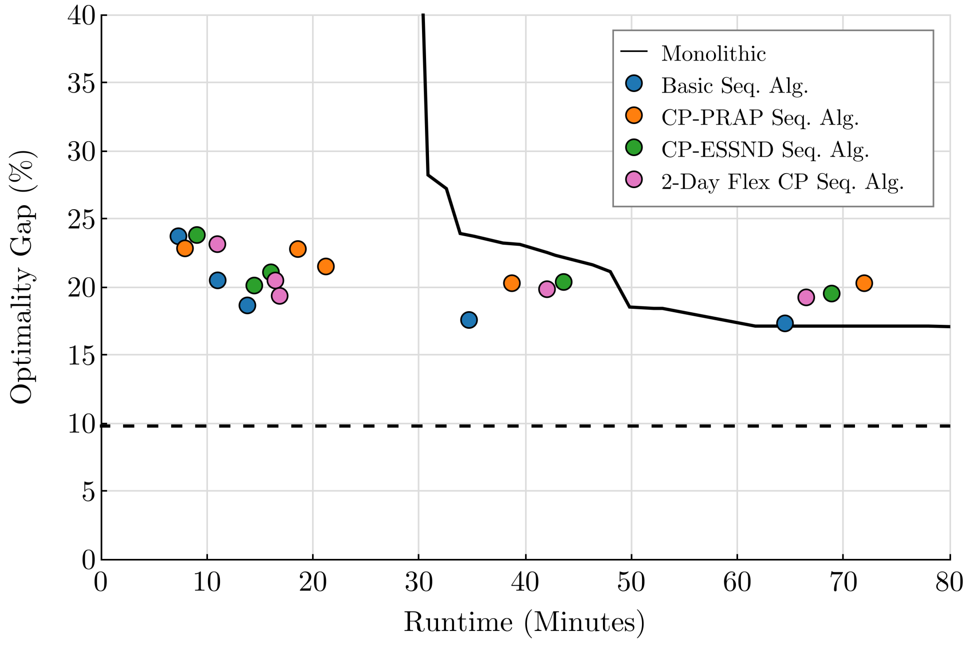


Figure 25: Template-Based Sequential Algorithm with Flexible Two-Day Volume Compared to Other Sequential Algorithms and the Monolithic Approach for Baseline Problem Instance

Ultimately, the idea to reduce the size of the Package Routing with Aircraft Problem which gave rise to the Collapsed Product Set Sequential Algorithm did not improve the Basic Sequential Algorithm. Likely, the Collapsed Product Set Sequential Algorithm was hampered by being too inflexible with the package routings, particularly in the assumption that routings generated for the night cycle would be appropriate for the day cycle. This inflexibility resulting in sub-optimalities and poor overall performance. As a result, hypothesis 4.1 is rejected.

The other idea for improving the Basic Sequential Algorithm was to improve the Basic Sequential Algorithm by adding additional constraints to the PRAP to make it better approximate the original ESSND formulation. This second idea will be studied in the next section (§7.5).

7.5 Improved Sequential Algorithm

The experiments on the Basic Sequential Algorithm (Table 26) showed that it was possible for a sequential decomposition-based heuristic algorithm to perform better than a monolithic approach, at least in the realm where solutions are desired in a short period of time. The resulting algorithm, although a satisfactory proof of concept, still has some areas for improvement. Now that it has been established that a sequential algorithm, developed as the Basic Sequential algorithm has, *can* provide good solutions in a short period of time, explorations will be made into improving that algorithm by improving the solutions generated in the Package Routing with Aircraft Problem by adding additional constraints.

7.5.1 Phase-Based Hull Count Constraint

To improve the Basic Sequential Algorithm by improving the quality of the Package Routing with Aircraft Problem, let us first consider the role and importance of the PRAP. Imagine that, by some method, the PRAP assigned the package routings of the optimal solution. In that case, since the Aircraft Movement Problem is solved to optimality for those assigned routings, the problem optimum would be found. On the other extreme, imagine that the PRAP assigned package routings corresponding to an extremely poor solution. The AMP, since it cannot change package routings, would find the best possible solution for that routing selection, but it would still ultimately be a poor solution. Due to the relatively fast runtime of the AMP and the fact that the AMP can be solved to optimality, the PRAP is *the* ultimate decider of solution quality.

However, finding a high-quality package routing solution is not trivial, particularly in a reasonable time frame. On the one end of the spectrum, *if* the full AMP problem was included in the PRAP, when an optimal solution was found, it would be *the* guaranteed optimal solution because the PRAP would have become the full ESSND, but as has already been established, this would take a very long time. On the other end of the spectrum, if none of the AMP were included in the PRAP, fictional "pirate ship" vehicles or fractional vehicles would be used to move all of the packages and the solution would be completely out of touch with reality, but as was found in the experiments in Section 7.2.3 when exploring different decomposition definitions, this solution would take only seconds to find.

When formulating the Package Routing with Aircraft Problem, a balance was struck between the two extremes of this spectrum. In the Basic Sequential algorithm, the AMP surrogate was composed solely of the design (vehicle) variable integrality constraint and the feeder limit constraint. The hull balance constraints were omitted from the PRAP formulation because they constitute the vast majority of the aircraft-related constraints, and relaxing the constraint was found, in the monolithic sensitivity studies, to have the greatest effect on runtime. The hull count constraint was omitted from the PRAP because it was ineffective without the balance constraint, a result that was also found in the monolithic approach sensitivity studies.

The fundamental idea of the PRAP is to assign package routings for all of the commodities while considering, in as much detail as possible, the needs and behavior of the Aircraft Movement Problem, without severely affecting the runtime. The balance that must be struck is between the quality of the solution and the time it takes to obtain that solution. Additionally, the behavior of the underlying discipline solution method must be taken into account. Due to the poor linear relaxation of the ESSND problem and its subproblems like the PRAP, after an initial period of rapid progress, the monolithic approach struggles to prune branches from the B&C tree and devolves into something more akin to an enumeration approach, which produces solution improvements but very slowly. Ideally, good solutions to the PRAP could be found prior to or in the early stages of this flattening of progress, because once the progress begins to flatten out, longer and longer runtimes are necessary in order to find meaningful improvement.

Up to this point, the sequential approach has been presented in terms of the two disciplines – the Package Routing Problem and the Aircraft Movement Problem. It is a subtle point, but a slightly different way to think of the sequential algorithm is that the PRAP is an approximation of the original ESSND problem which will produce an infeasible solution with respect to the original problem, and the AMP is used to "tidy up" that solution in to something that is actually feasible and flyable. The point is subtle because, with only two disciplines, adding more AMP considerations to the PRAP automatically makes the PRAP a better approximation of the ESSND. However, this slight change of thinking about the problem is useful in improving the PRAP.

The more closely that the PRAP can capture the AMP (and in doing so, approximate the ESSND), the better the final solution of the sequential algorithm will be. In order to do this, let us revisit the two constraints from the ESSND that were omitted in the PRAP – the hull count and hull balance constraints. Of the two, the effect of omitting the consideration of the hull count is more obvious. It is also the more important of the two since the primary aim of the AMP surrogate is to encourage the assignment of package routing such that *whole* real-world aircraft are efficiently loaded. How those whole aircraft are positioned so that they carry those packages (i.e. balanced) is a secondary issue that is important for the implementation of a solution, but not as important for determining good package routings. The hull use of the PRAP when it was given a five-minute runtime (i.e. the case that had the best runtime margin relative to the monolithic approach) is shown in Table 32.

Recall that the linear relaxation of the AMP was considered an ineffective option

Table 32: Hull Usage in the Basic Package Routing with Aircraft Problem

Aircraft	Number in Fleet	Number Used
B777F	5	52
MD-11	43	0
MD-10-10	37	0
B767-300F	30	98
A-300	49	0
A-310	8	0
B757-200	80	83
ATR-72	10	51
ATR-42	10	26
C-208	50	11
Trunk Total	252	233
Feeder Total	70	88

because that approach resulted only in solutions in which fractions of the most cost-efficient aircraft were used to move the volume. As seen in Table 32, the most cost-effective aircraft were also the most popular in the basic PRAP, though in integral increments. Since there is no hull count constraint, the PRAP solution filled the most cost-efficient aircraft, which in this case are the B777F, B767-300F, and B757-200, the youngest aircraft in the fleet.

As a result, the PRAP *is* generating routings based on filling whole aircraft, but it is filling the *wrong* aircraft. It is using more than *ten times* more B777F aircraft than are available and three times more B767-300F aircraft. This results in a more costly schedule because different aircraft will need to be scheduled to fly those packages whose routings were assigned so that they would best fill a B777F, B767-300F or B757-200. In the case of the B777F, this dramatic over-use of aircraft has the additional effect of requiring two or more aircraft to fly those packages because the B777F is the largest aircraft in the fleet. There is also a severe over-use of feeder aircraft despite the feeder limit. The arc-based feeder limit is still effective, but feeder aircraft are very cheap compared to trunk aircraft so they are a popular choice to move small amounts of additional volume. However, they are also smaller than trunk aircraft so this will result in many poorly-loaded trunk flights in the AMP solution.

To improve the PRAP, and, by extension, the sequential algorithm, the hull count issue should be addressed. This leads to the hypothesis,

Hypothesis 4.2. *Based on the observation that the PRAP assigns package routings with little bearing on the available aircraft fleet, I hypothesize that including a hull count constraint to the PRAP would improve the performance of the sequential algorithm.*

To understand why the original hull count constraint was ineffective without the balance constraint, let us revisit it. That constraint, as it is written in the ESSND problem formulation (3.4), is

$$\sum_{(i^+,j)\in A} y^f_{i^+j} \leq n_f \quad \forall f \in F \setminus \tilde{F} \tag{7.15}$$

In the original formulation, the hull count constraint is enforced by totaling the hulls out of a source node i^+. The hull count constraint then depends on the balance constraint to ensure that no additional hulls are added after this point. Without the balance constraint, hulls can be added (or removed) at any point, thus rendering the hull count constraint ineffective.

In order to improve the PRAP, the hull count constraint should be re-introduced, but without the dependence on the balance constraint. To do this, the periodic nature of the problem can be once again be exploited. The hull count constraint can be enforced across any time-based cut in the time-space graph. In the original hull count constraint (7.15), this cut was made over the arcs coming out of the source node. To make the hull count constraint less dependent on the balance constraint, the hull count constraint could be enforced multiple times over different time-based cuts in the time-space graph. The periodic nature of the express shipment problem provides good opportunities for making these cuts. This new hull count constraint will be called the *phase-based hull count* constraint, and written,

$$\sum_{(i,j)\in A} \delta_{ij}^{\psi} y_{ij}^{f} \leq n_f \quad \forall f \in F \setminus \tilde{F},\, \psi \in \Psi \tag{7.16}$$

where $\psi \in \Psi$ is a phase in the set of phases and,

$$\delta_{ij}^{\psi} = \begin{cases} 1 & \text{if } (i,j) \text{ is an arc in phase } \psi \text{ and either } i \in H \text{ or } j \in H, \text{ and} \\ 0 & \text{otherwise} \end{cases}$$

Recall that $\tilde{F}$ is the set of truck vehicle types and $H \subset N$ are the nodes corresponding to sort activity instances at hubs. For a single day, Ψ is composed of the night pick up, night delivery, day pick up and day delivery phases. A hull count constraint is created for each of these phases which stipulates that the total number of aircraft of each type flying from a spoke to a hub (or hub to a spoke for delivery phases) must be less than or equal to the total number of those aircraft in the fleet. Spoke-to-spoke consolidation flights were not counted so as to avoid double counting aircraft. However, this is hardly an issue in reality since the bulk of the consolidation movements are done via trucks which are not hull count-limited.

7.5.1.1 Testing the Impact of the Addition of the Phase-Based Hull Count Constraint to the Package Routing with Aircraft Problem

One concern with adding more constraints to the Package Routing with Aircraft Problem is that it may increase the runtime of the PRAP and make it more difficult to find good solutions in a timely manner. Therefore, the effect of adding the phase-based hull count constraint to the PRAP is tested experimentally.

To test the effect, the PRAP with the phase-based hull count constraint is run using the experimental testbed described in Section 5.4 and the computational testbed described in Table 5. Only the solution generation method was slightly modified so as to reflect the PRAP formulation with the phase-based hull count constraint. Since this problem is a MIP, it was solved with Gurobi. A single experiment was run using the baseline problem instance (described in Table 6). The results of this experiment are found in Table 33.

Table 33: Results of Including the Phase-Based Hull Count Constraint to the Package Routing with Aircraft Problem

Problem	Optimality Gap	Runtime
Extended PRAP	7.17%	24 hrs
Basic PRAP	7.73%	24 hrs
Original PRP	0%	22.86 sec

Compared to the basic PRAP, the extended PRAP (i.e. the PRAP with the phase-based hull count constraint) actually has a slightly *lower* optimality gap after twenty-four hours. This indicates that the addition of the phase-based hull count constraint actually *helps* the solver find good solutions, perhaps by aiding in excluding sub-optimal solutions that would have otherwise been considered. The good progress of the extended PRAP, even compared to the basic PRAP, suggests that using a monolithic approach and terminating it early will be successful, as it was in the Basic Sequential algorithm. The implementation of that algorithm is described and tested in the next section.

7.5.2 Improved Sequential Algorithm with the Phase-Based Hull Count Constraint

Now that the phase-based hull count constraint has been shown to not inhibit the solve time of the PRAP, its effect on the sequential algorithm must be tested. The resulting algorithm will be called the *Improved Sequential Algorithm* and it is very similar to the Basic Sequential Algorithm, except that the phase-based hull count constraint (Eq. 7.16) is included in the Package Routing with Aircraft Problem. Experiments testing the effect of the PRAP runtime on the solution quality, from two minutes to six hours, were run on the experimental testbed described generally in Section 5.4 and described specifically for the Basic Sequential Algorithm in Section 7.3.3. The experiments were performed on the computational testbed summarized in Table 5 using the baseline problem instance described in Table 6. The results of these experiments are shown in Table 34 and Figure 26.

Table 34: Results of the Improved Sequential Algorithm with the Phase-Based Hull Count Constraint

Instance	PRAP Opt. Gap	Objective Difference	Absolute Opt. Gap	AMP Runtime
2 min	28.8%	17.64%	23.29%	150 sec
5 min	24.1%	13.18%	20.26%	115 sec
10 min	14.4%	1.49%	11.08%	60 sec
30 min	12.3%	0.08%	9.82%	90 sec
1 hr	11.8%	-0.27%	9.51%	92 sec
3 hrs	10.5%	-1.24%	8.62%	97 sec
6 hrs	10.3%	-1.27%	8.42%	190 sec

Adding the phase-based hull count constraint to the PRAP *significantly* improves the performance of the sequential algorithm. The objective difference and the absolute optimality gap are calculated relative to the best objective value and lower bound found by the monolithic approach after forty-eight hours. The best-found optimality gap of the monolithic approach was 9.75%. This means that within one hour, the Improved Sequential Algorithm found a *better* solution than the baseline monolithic approach found after *two days*. Whereas the Basic Sequential Algorithm found better solutions than the monolithic baseline for only the first hour of runtime, the Improved Sequential Algorithm significantly outperforms the monolithic approach for all tested time periods. Figure 26 graphically compares the performance of the Improved Sequential Algorithm to the Basic Sequential Algorithm and the monolithic approach.

In Figure 26, the Basic Sequential Algorithm is shown in blue, the Improved

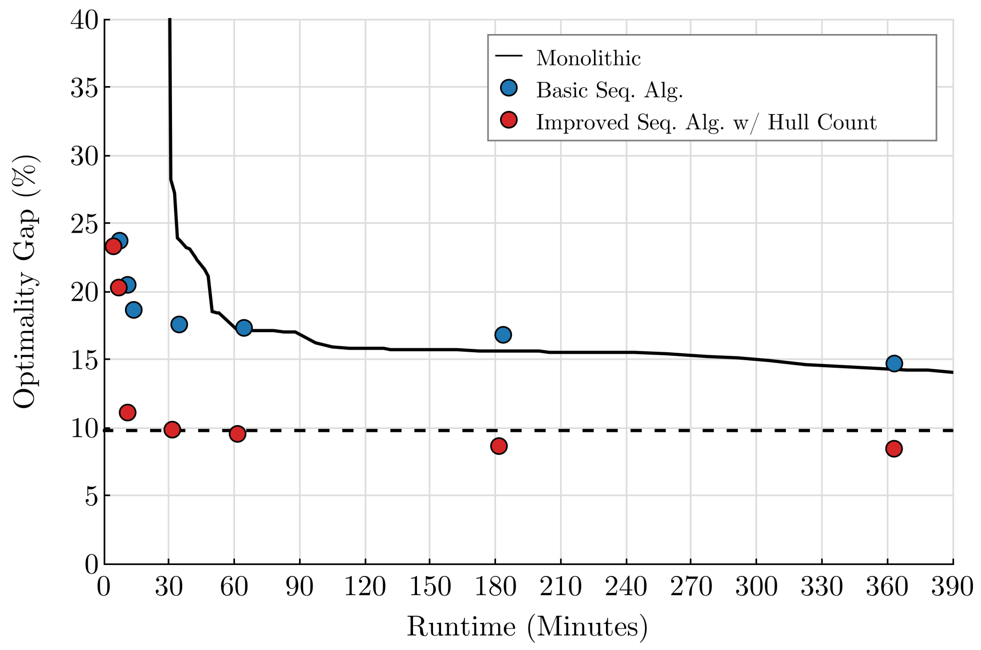


Figure 26: Improved Sequential Algorithm with the Phase-Based Hull Count Constraint Compared to the Basic Sequential Algorithm and Monolithic Approach for the Baseline Problem Instance

Sequential Algorithm is shown in red, the convergence behavior of the monolithic approach is shown with the solid black line, and the best-found optimality gap from the monolithic approach is shown with the dashed black line. For the first couple of points (the two and five-minute cases), the basic and extended sequential algorithms perform similarly, although the Improved Sequential Algorithm is slightly faster with better solutions in both cases. Then, after those first two points, the Improved Sequential Algorithm performs significantly better than any other approach. The case in which the extended PRAP was run for thirty minutes finds a solution only marginally worse than the monolithic approach found after forty-eight hours, and the case in which the extended PRAP was run for an hour finds a solution comfortably better than the monolithic approach. In the case where the extended PRAP was run for six hours, a solution was found that was 1.33 percentage points better in its optimality gap over the forty-eight-hour monolithic solution. Considering that over the second day of runtime for the monolithic approach the optimality gap only improved by only 1.04

percentage points, the gap between the best monolithic approach solution and the best Improved Sequential Algorithm solution is considerable.

The Improved Sequential Algorithm performs better than the Basic Sequential Algorithm because the PRAP better approximates the full ESSND problem with the phase-based hull count constraint. The primary advantage that the approximation has over the original formulation is that it is not burdened with the numerous hull balance constraints. By including the phase-based hull count constraint, package routings are assigned which efficiently load real aircraft and have some bearing on reality, but without the burden of the balance constraints. In a sense, adding the phase-based hull count constraints adds the "right amount" of reality to the approximation – just enough to produce very good solutions but not so much that it becomes burdensome and inhibits the finding of good solutions. The best solution found by the Improved Sequential Algorithm exists in the feasible space of the problem solved by the monolithic approach, but the additional burden of needing to satisfy all of the constraints, including the hull balance constraints, in every solution makes it much more difficult to find those good solutions. Additionally, the poor LP relaxation bound makes it even more difficult to find good solutions because it makes the branch-and-bound approach much less efficient. Based on the results of the Improved Sequential Algorithm with Hull Count, hypothesis 4.2 is supported.

7.5.3 Sort-Centric Balance Constraint

Given the significant performance enhancement found by including the phase-based hull count constraint in the PRAP, the next question is whether the addition of a version of the previously-omitted balance constraint would further improve the performance of the Improved Sequential Algorithm. This leads to the hypothesis,

Hypothesis 4.3. *Based on the observation that adding the phase-based hull count constraint to the PRAP greatly improved the sequential algorithm, I hypothesize that* *including a modified vehicle balance constraint to the PRAP will further improve the performance of the sequential algorithm by adding additional realism to the PRAP.*

The original hull balance constraints, while effective in capturing reality, are also numerous and burdensome to the solver. Therefore, adding the original hull balance constraints would be more detrimental than beneficial, and a relaxed version of the balance constraint would be more appropriate.

The potential addition of a relaxed hull balance constraint is also motivated by the fact that while the phase-based hull count constraint is effective in enforcing the hull count across a phase, it does not fully capture how many hulls would be needed to actually implement that schedule. When *trackings*[8] are created for the extended PRAP, more hulls are needed than are available for some aircraft types. The disparity is nowhere near as severe as that of the basic PRAP, but since reducing the disparity through the phase-based hull count constraint improved the final solution significantly, perhaps reducing it further through a relaxed balance constraint would also have positive effects.

This relaxed balance constraint, called the *sort-centric balance* constraint is centered around each sort activity instance. It stipulates that the aggregate number of aircraft of each type that fly into a sort activity instance must also fly out of that sort activity instance. This precludes the possibility of an aircraft flying into the sort and then sitting on the ground until the next sorting activity. In reality, aircraft do not often sit on the ground at hubs because a certain amount of capacity is needed to fly all of the volume into a sort so a similar amount of capacity should be needed to fly out of a sort.

The sort-centric balance constraint was not similarly implemented around the origin and destination activities at the spokes because it is much more common for aircraft to sit on the ground for an entire cycle (typically the daytime operations) at

[8]Trackings are the assignment of specific tail numbers to a series of flights.

the spokes, and because the timing at spoke airports does not always lend itself well to this kind of accounting (i.e. at some spokes, the daytime origin activity occurs before the nighttime delivery activity). The sort-centric balance constraint is written,

$$\sum_{j:(i,j)\in A'} y_{ij}^f = \sum_{j:(j,i)\in A'} y_{ji}^f \quad \forall i \in H \subset N, f \in F \setminus \tilde{F} \tag{7.17}$$

where H is the set of nodes corresponding to a sort activity instance at a hub, A' is the set of *flight* arcs, and $\tilde{F}$ is the set of truck vehicle types. This constraint is identical to the original hull balance constraint except that it is only enforced over the node subset H, which is much smaller than N.

7.5.3.1 Testing the Impact of the Addition of the Sort-Centric Balance Constraint to the Package Routing with Aircraft Problem

As with the previous modifications to the Package Routing Problem, the effect of the sort-centric balance constraint on the PRAP runtime was tested prior to implementing it in the sequential algorithm. The experiments were performed on the experimental testbed described in Section 5.4 where the only alteration was the inclusion of the sort-centric balance constraint to the PRAP formulation. Two experiments were conducted – one in which only the sort-centric balance constraint was added to the PRAP, and one in which the sort-centric balance *and* phase-based hull count constraints were added to the PRAP. The experiments were run on the computational testbed described in Table 5 using the baseline problem instance described in Table 6. The results are summarized in Table 35.

Adding the sort-centric balance constraint has a small effect on the PRAP optimality gap after 24 hours of runtime compared to the PRAP with only the phase-based hull count constraint. However, the optimality gaps of all of the PRAP variants are all within 0.88 percentage points of each other so there is nothing particularly dramatic in the addition of the sort-centric balance constraint in terms of the solver's

Table 35: Results of Including the Sort-Centric Balance Constraint in the Package Routing with Aircraft Problem

Problem	Optimality Gap	Runtime
PRAP w/ Sort-Centric Balance	7.45%	24 hrs
PRAP w/ Balance & Count	8.05%	24 hrs
PRAP w/ Phase-Based Hull Count	7.17%	24 hrs
Basic PRAP	7.73%	24 hrs
Original PRP	0%	22.86 sec

progress in finding solutions for the problem.

7.5.4 Improved Sequential Algorithm with the Sort-Centric Balance and Phase-Based Hull Count Constraints

The impact of adding the sort-centric balance constraint to the PRAP within the sequential algorithm is studied next. The experiments testing this impact were conducted in the same fashion as the experiments testing the phase-based hull count constraints in Section 7.5.2. They were performed on the experimental testbed that was described generally in Section 5.4 and more specifically for the sequential algorithm in Section 7.3.3 using the computational testbed summarized in Table 5 and performed on the baseline problem instance described in Table 6. The results are shown in Table 36.

Table 36: Results of the Improved Sequential Algorithm with the Sort-Centric Balance and Phase-Based Hull Count Constraints

Instance	PRAP Opt. Gap	Objective Difference	Absolute Opt. Gap	AMP Runtime
2 min	31.3%	20.49%	25.10%	190 sec
5 min	25.8%	15.09%	21.59%	95 sec
10 min	17.8%	7.77%	16.26%	100sec
30 min	14.0%	3.68%	12.96%	117 sec
1 hr	13.6%	3.19%	12.54%	110 sec
3 hrs	13.0%	2.63%	12.06%	223 sec
6 hrs	12.1%	2.57%	12.02%	147 sec

The results are good, but they are not *as* good as the extended PRAP without the sort-centric balance constraint. None of the cases find a solution better than the best-found monolithic approach. The first couple cases actually produce *worse* solutions than even the Basic Sequential Algorithm, but after those first couple cases, the Improved Sequential Algorithm with Hull Count and Balance produces significantly better solutions than the Basic Sequential Algorithm. How the Improved Sequential Algorithm with Hull Count and Balance compares to the other sequential algorithms and the monolithic approach can be best seen in Figure 27.

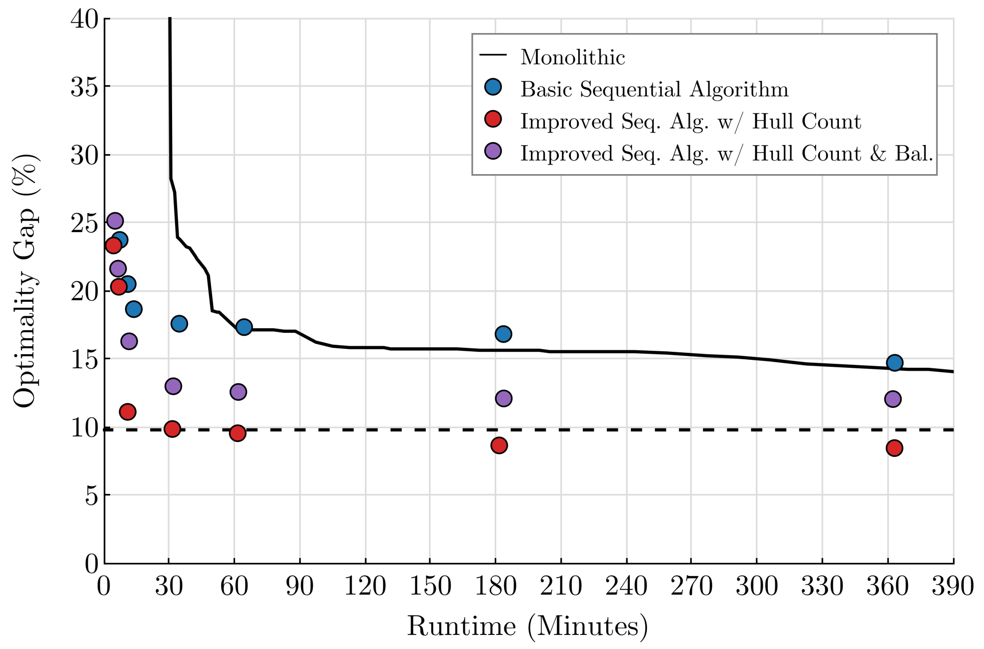


Figure 27: Comparison of the Full-Set Sequential Algorithms and Monolithic Approach for the Baseline Problem Instance

The Improved Sequential Algorithm with Hull Count and Balance cases are shown in purple and the Basic Sequential Algorithm and Improved Sequential with Hull Count cases are in blue and red, respectively. The monolithic convergence is represented by the solid black line and its best-found solution is shown with the dashed black line. Figure 27 shows that the Improved Sequential Algorithm with Hull Count and Balance performs better than both the Basic Sequential Algorithm aside from the first couple of cases, and performs consistently better than the monolithic approach for all cases considered. It does not, however, find a better solution than the best-found monolithic approach and it consistently performs worse than the Improved Sequential Algorithm with (only) Hull Count.

These results show that the addition of the sort-centric balance constraint is not beneficial in finding high-quality solutions. While it does improve the hull usage, the added realism of ensuring hull balance in and out of the sort activity instances proves to be more of a burden than a boon. The constraint makes it more difficult for the solver to find high-quality solutions and the improved quality of those solutions (relative to the original problem) does not make up for the difficulty in finding them. From these results, it can be concluded that while the hull count constraint is necessary to generate high-quality package routings, the hull balance constraint is not necessary and is ultimately detrimental to quickly generating high-quality package routings. In the end, this result makes sense since the hull count constraint has the most impact on the goal to fill whole aircraft within the fleet. Ensuring that these aircraft are balanced throughout the network is a secondary consideration to the actual loading of those aircraft. Furthermore, since aircraft are not given the option to remain on the ground after a sort activity instance, the sort-centric balance constraint also is *over-constraining* the solutions, which also impinges on the Improved Sequential Algorithm's ability to find high-quality solutions. As a result, while the sort-centric balance constraint slightly improves the realism of the PRAP approximation, it also makes the problem more difficult to solve and it does not greatly improve the solution quality.

Therefore, the Improved Sequential Algorithm with Hull Count is found to be the best of the heuristic decomposition approaches for the baseline problem instance encompassing a single day's worth of volume and hypothesis 4.3 is rejected. It best balances the dueling needs of approximation quality and problem difficulty, and is able to quickly produce high-quality solutions much faster than any other method, heuristic or otherwise, studied in this work.

7.6 Sensitivity Studies

In order to test the robustness of the Improved Sequential Algorithm and find it's limits, several sets of sensitivity studies were performed on the algorithm. Studies were performed on variations of the baseline problem instance in which the sort capacity, fleet size or airport subset was reduced, a different day of the week was solved, and for problems with larger temporal scope. The studies focused on the sensitivity to sort capacity, fleet size and temporal scope all aimed to explore how the Improved Sequential Algorithm performed on problem instances that were *harder* or more constrained than the baseline problem instance. The studies focused on the airport sensitivity to the size of the airport subset and different days of the week[9] aimed to explore how the Improved Sequential Algorithm performed on problem instances that were *easier* than the baseline problem instance.

All sensitivity studies were performed on the experimental testbed that was generally described in Section 5.4 and described for the sequential algorithm in Section 7.3.3 with the addition of the phase-based hull count constraint to the Package Routing with Aircraft Problem. The sensitivity studies all focus on altering the *problem instance* that is solved by the Improved Sequential Algorithm. The algorithm itself is not changed. All of the experiments are performed on the computation testbed described in Table 5.

7.6.1 Sort Capacity

The sort capacity sensitivity study explores how the Improved Sequential Algorithm performs when it is more challenging to route packages. As the sort capacity gets

[9]The baseline problem instance was for a Monday, the day with the largest volume.

tighter, it is more challenging to find a feasible set of package routings without surpassing the sort capacity. In the baseline problem instance, the sort capacity is already very tight in part due to the real world challenges of the problem, and in part due to the restrictions I included in the routing generation method (§6.2) in order to make the sort capacity constraint meaningful with a reduced demand set.

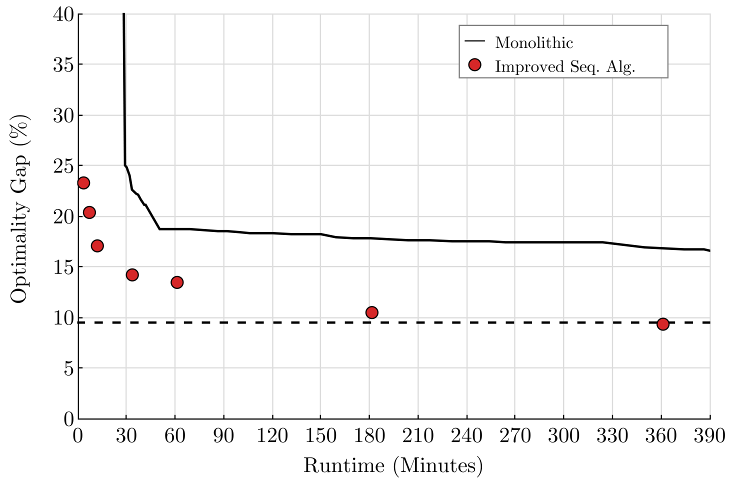
To generate the reduced sort capacity instances, the baseline sort capacities were reduced uniformly by 99 and 98 percent, and rounded to the nearest piece (since sort capacities are in terms of pieces). A sort capacity that was any lower was found to be infeasible. The results of those studies are shown in Figure 28.

For both problem instances, the Improve Sequential Algorithm outperforms the monolithic approach for the runtime period considered. The heuristic algorithm surpasses the best-found monolithic solution after six hours for the problem instance with a 99% sort capacity but it does not surpass the best-found monolithic solution of the problem instance with a 98% sort capacity. The fact that it takes longer to surpass the monolithic approach is likely due to the increased difficulty in finding solutions in the PRAP. The optimality gap results are shown in Table 37.

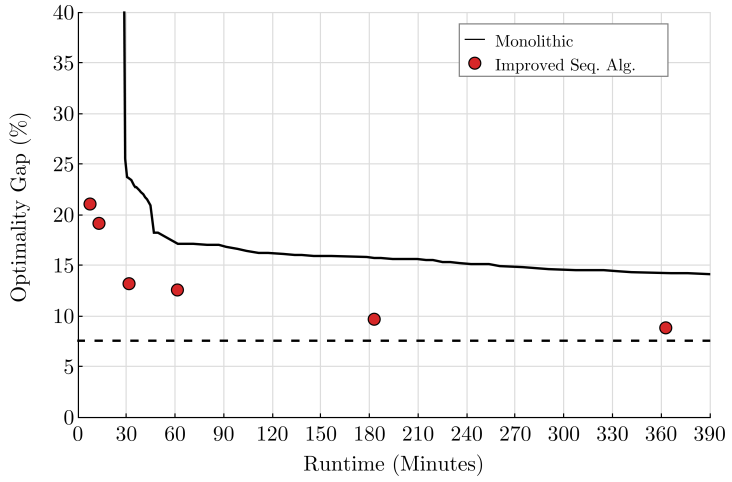
Table 37: Optimality Gap Results of the Improved Sequential Algorithm with Reduced Sort Capacity

PRAP Runtime	99% Capacity	98% Capacity
2 min	23.2%	–
5 min	20.3%	21.0%
10 min	17.0%	19.1%
30 min	14.2%	13.1%
1 hr	13.4%	12.5%
3 hrs	10.4%	9.6%
6 hrs	9.3%	8.8%

Overall, despite the increased challenge to find feasible package routings, the Improved Sequential Algorithm does well. It is still capable of finding high-quality solutions in a reasonable amount of time. The only exception is for the two-minute



(a) 99% Sort Capacity



(b) 98% Sort Capacity

Figure 28: Sort Capacity Sensitivity Study Results for the Improved Sequential Algorithm

case on the 98% capacity problem instance in which the PRAP was not able to find a solution before it was terminated. For both problem instances, solutions with optimality gaps below 10% were found in about six hours. Furthermore, although the heuristic algorithm was not capable of finding a solution *better* than the best-found solution from the monolithic approach in as short of a time period, it was still greatly superior to the monolithic approach in finding high-quality solutions in the studied time period.

7.6.2 Fleet Size

The fleet size sensitivity study explores how the improved sequential algorithm performs when it is more challenging to *move* packages. As the fleet size shrinks, it becomes more important that packages be routed so that the aircraft are used more efficiently, and the balance of the aircraft through the network through time becomes more important. The fleet size in the baseline problem instance was relatively loose. For the most cost-effective aircraft, the hull count constraint was restrictive, but there were many less-cost-effective aircraft for which the hull count constraint was not restrictive which resulted in surplus capacity in the available fleet.

To generate the reduced fleet problem instances, the fleet was reduced uniformly by 5% increments from 95% to 70% and rounded to the nearest whole aircraft. Fleets of 65% and smaller were found to have too little capacity. The results of the reduced fleet studies are shown in Figure 29.

The Improved Sequential Algorithm performs well for all of the reduced fleet size studies. It finds significantly better solutions than the monolithic approach for all of the problem instances, although it does not find better solutions than the best-found monolithic solutions for all of the problem instances. However, for the time period considered, the heuristic algorithm produced superior solutions compared to the monolithic approach. The optimality gaps of each case are tabulated in Table 38. Cases that include pirate ships (i.e. are not flyable schedules) are marked with an asterisk.

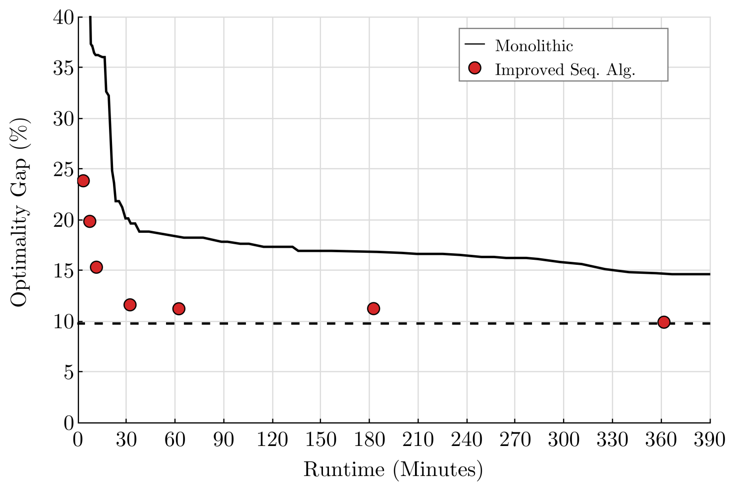
The Improved Heuristic Algorithm consistently out-performs the monolithic approach, even when the fleet size becomes very small. However, the effect of the reduced fleet size does start to show in the short-runtime cases for the 70% to 80% fleet size problem instances. The presence of pirate ships means that the algorithm is struggling a bit to find flyable solutions. However, it should be noted that the monolithic approach struggles *even more* to find flyable solutions as shown by its

Table 38: Optimality Gap Results of the Improved Sequential Algorithm with Reduced Fleet Size

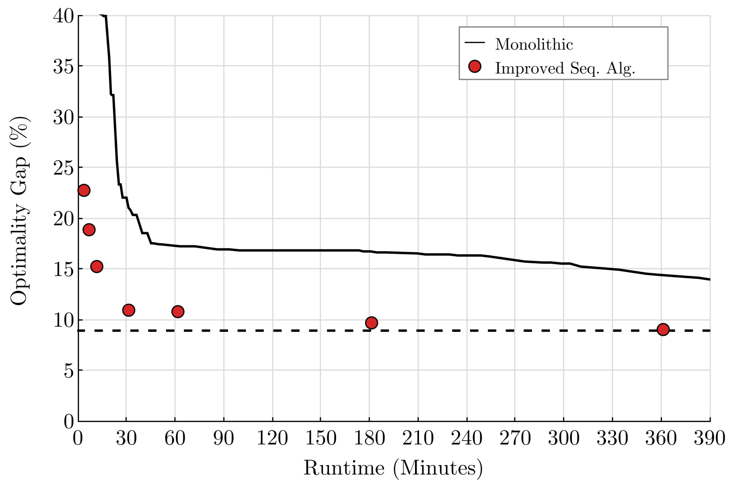
PRAP Runtime	95% Cap.	90% Cap.	85% Cap.	80% Cap.	75% Cap.	70% Cap.
2 min	23.3%	22.7%	25.2%	38.1%*	53.3%*	65.6%*
5 min	19.8%	18.8%	18.9%	20.0%	47.8%*	48.1%*
10 min	15.2%	15.2%	16.5%	17.5%	34.5%*	12.5%
30 min	11.6%	10.9%	13.5%	13.3%	15.5%	12.1%
1 hr	11.2%	10.8%	13.1%	12.4%	15.3%	12.3%
3 hrs	11.2%	9.6%	11.7%	11.0%	13.1%	10.9%
6 hrs	9.9%	9.0%	11.1%	11.0%	12.7%	10.2%

higher optimality gap for all of those cases.

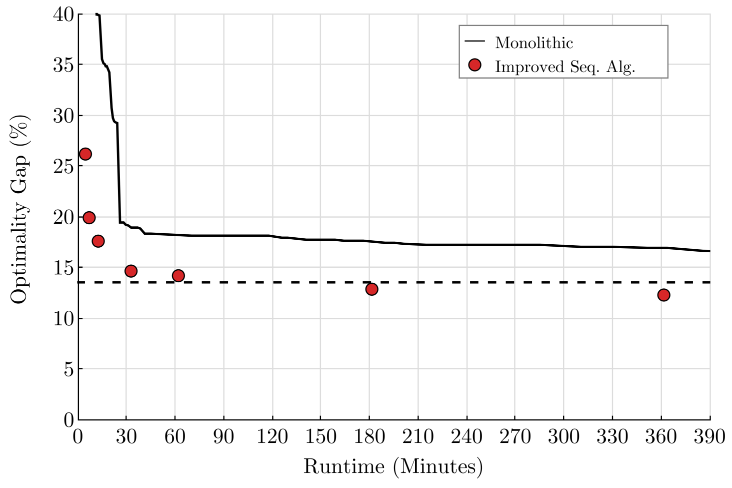
One would expect that cases in which the fleet size is very tight would cause problems for the Improved Sequential Algorithm since the Package Routing with Aircraft Problem does not include any kind of vehicle balance. When there is some flexibility in the fleet, this lack of balance is not a major issue since there are enough aircraft available. However, imagine the situation in which there are *exactly* enough aircraft available to move all of the packages. Since the PRAP does not balance vehicles, it may route packages expecting the ATL to MEM pick up flight to be flown by Aircraft A, and it may then expect the next delivery leg from IND to LAX to be also flown by Aircraft A. This is impossible and, as a result, *two* aircraft are needed. When there is some flexibility in the fleet, this is not a problem, but when the fleet size is very tight, the fictional *pirate ship* vehicles would be needed to fill in the additional capacity needs.



(a) 95% Fleet Size

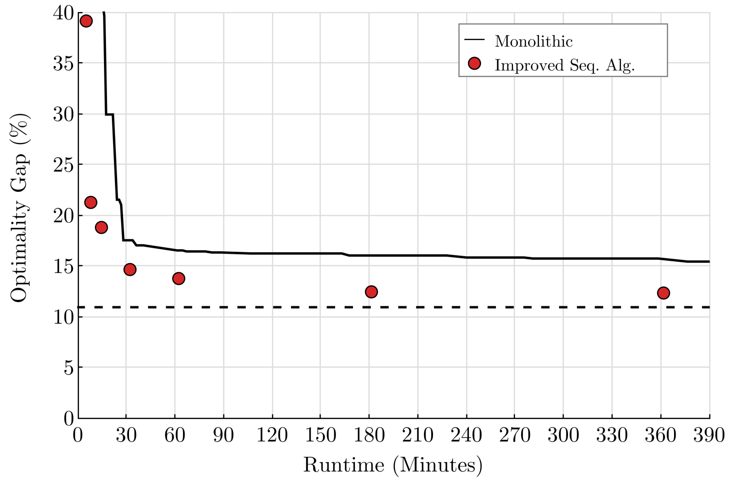


(b) 90% Fleet Size

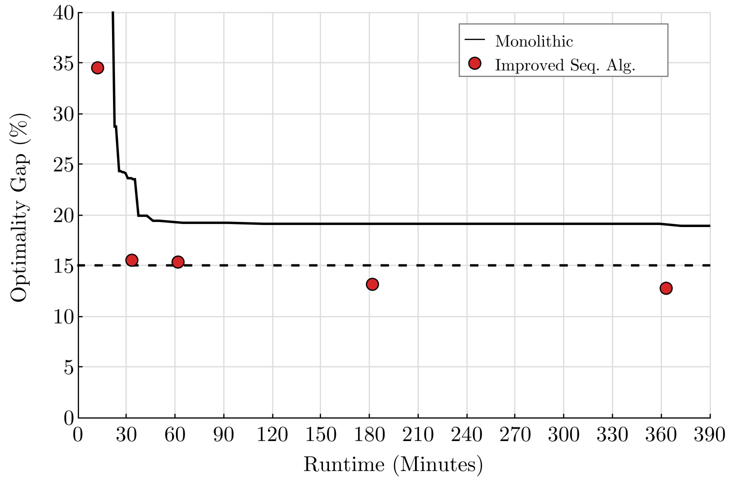


(c) 85% Fleet Size

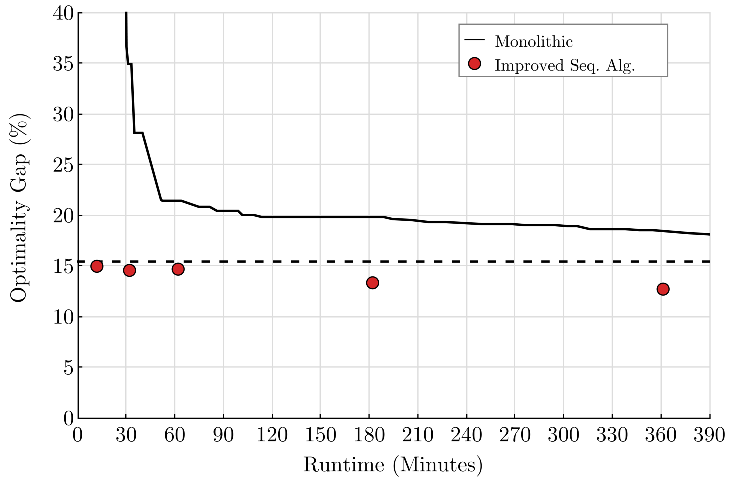
Figure 29: Fleet Size Sensitivity Study Results for the Improved Sequential Algorithm



(d) 80% Fleet Size



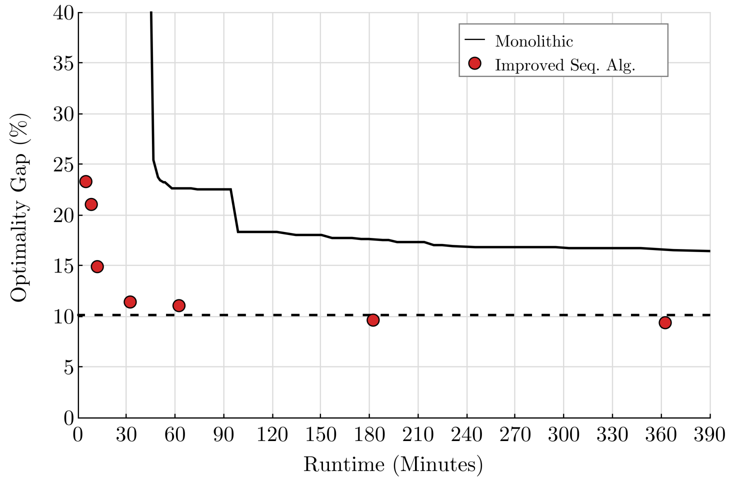
(e) 75% Fleet Size



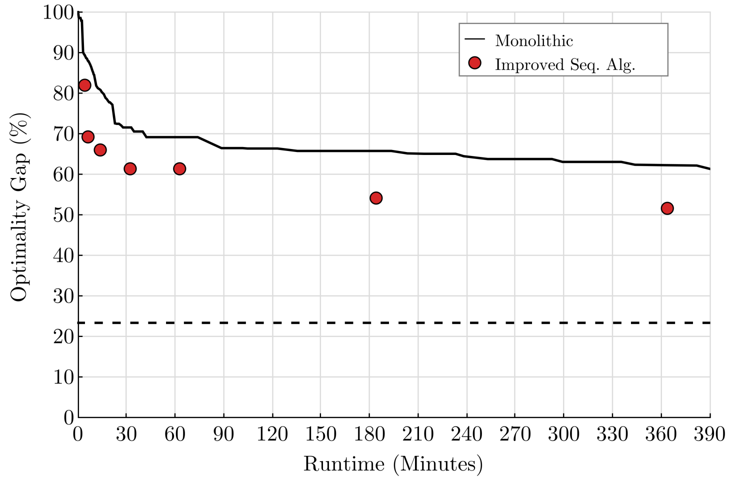
(f) 70% Fleet Size

Figure 29: Fleet Size Sensitivity Study Results for the Improved Sequential Algorithm (Cont.)

To further explore the performance of the Improved Sequential Algorithm, two additional reduced fleet problem instances were generated and tested. These two problem instances were created by using the smallest necessary fleets from two monolithic solutions. The first monolithic solution fleet was from the baseline problem instance. The second monolithic solution fleet was also run on the baseline problem instance, but a hull count penalty was included in the objective function so that the solver was incentivized to use as small of a fleet as possible. The results are shown in Figure 30. The monolithic approach was re-run using the tighter fleet size, not the original.



(a) Baseline Solution Fleet



(b) Hull Minimization Solution Fleet

Figure 30: Fleet Size Sensitivity Study Results for the Improved Sequential Algorithm

In both cases, the Improved Sequential Algorithm out-performed the monolithic approach for the considered time period, but for the minimum fleet problem instance, the solutions were significantly worse than the best-found monolithic solution. The poorness of the heuristic approach solutions is due to the presence of pirate ships in the final solutions. It was not capable of finding a solution without pirate ships for the minimal fleet problem instance. Interestingly, neither did the monolithic approach, despite a 48-hour runtime, but eventually the monolithic approach did produce a solution with *fewer* pirate ships. The optimality gaps of the individual cases are summarized in Table 39. The solutions with pirate ships are again marked with an asterisk.

Table 39: Optimality Gap Results of the Improved Sequential Algorithm with Reduced Fleet Size Based on Monolithic Solutions

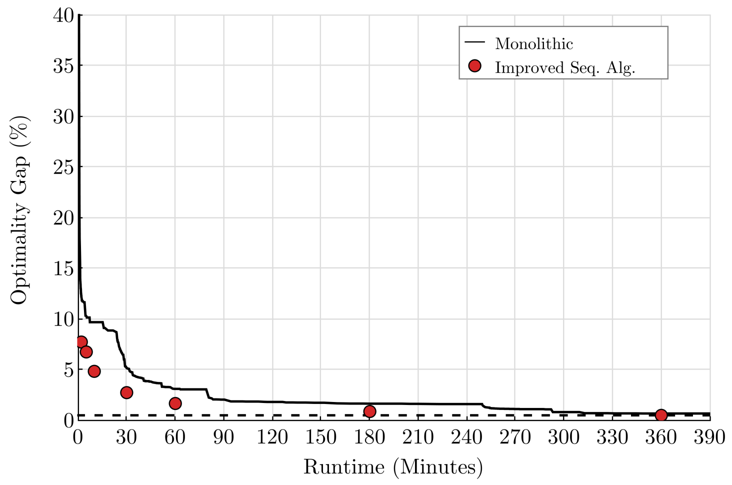
PRAP Runtime	Baseline Fleet	Hull Min Fleet
2 min	23.2%	81.8%*
5 min	21.0%	69.1%*
10 min	14.9%	65.8%*
30 min	11.4%	61.2%*
1 hr	11.0%	61.2%*
3 hrs	9.6%	54.0%*
6 hrs	9.3%	51.4%*

While every heuristic solution for the minimal fleet problem instance contained at least a dozen pirate ships, none of the solutions for the baseline solution fleet problem instance contained pirate ships. The baseline ESSND problem formulation does not include any direct incentive for minimizing the number of hulls and as a result, the fleet is larger and more flexible. Although the Improved Sequential Algorithm performs well compared to the monolithic approach for a given time period, for problems in which the fleet size is very tight, the heuristic algorithm will ultimately struggle to find flyable solutions. Its primary advantage over the monolithic approach, the lack of a balance constraint, is also its primary drawback for very tightly-constrained problems.

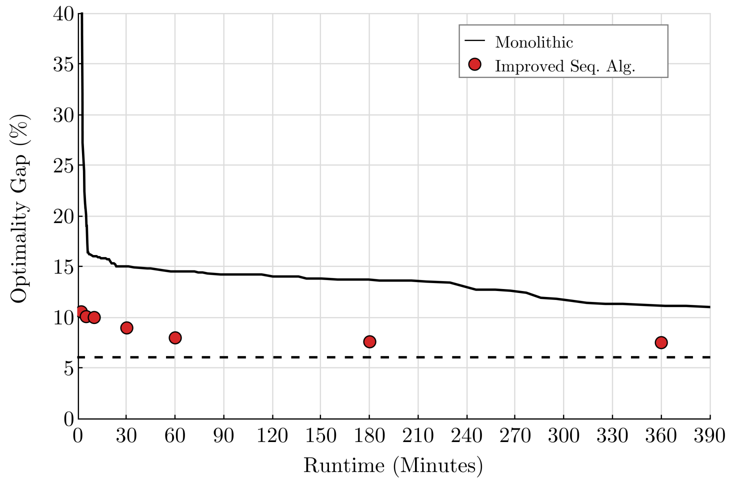
7.6.3 Airport Subsets

The airport subset sensitivity studies explore how the Improved Sequential Algorithm performs for problem instances that are *easier* than the baseline problem instance. The same randomly-generated subsets of airports that were used in the sensitivity studies for the monolithic approach in Section 6.5 were used in the sensitivity studies for the Improved Sequential Algorithm. In the previous airport subset sensitivity studies, the monolithic approach was found to be capable of solving the ten and twenty-five-airport problem instances in seconds and minutes, respectively. Realistically, for problems that can be solved so quickly to optimality, a heuristic approach is neither necessary nor appropriate. Therefore, the airport subset sensitivity studies were only performed on the fifty, seventy-five, and one-hundred-airport subset problem instances. Those results are shown in Figure 31.

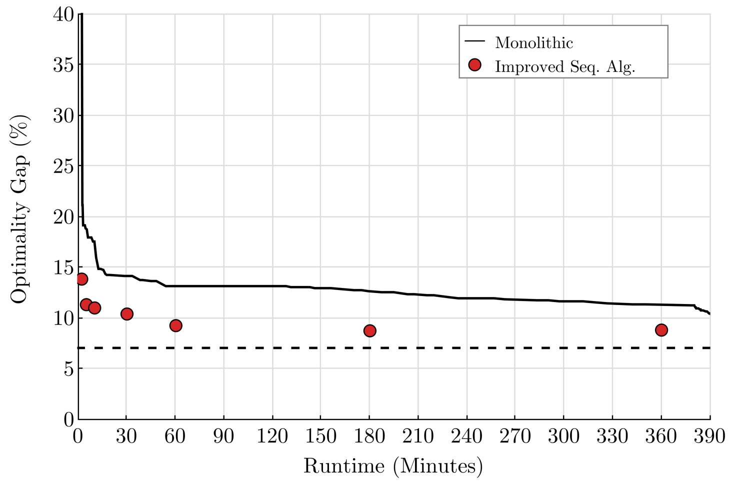
For the considered time period, the Improved Sequential Approach out-performs the monolithic approach, though it should be noted that the problem instance is easier for the heuristic approach to solve, *but it is also easier for the monolithic approach to solve as well.* Despite the increased performance of the monolithic approach, the heuristic approach produces better solutions for a given period of time in all cases. It does not, however, find a better solution in the considered time period than the best-found monolithic solution. This is likely a side-effect of the improved monolithic approach performance. The optimality gap results for each individual case is tabulated in Table 40.



(a) 50 Airport Subset



(b) 75 Airport Subset



(c) 100 Airport Subset

Figure 31: Airport Subset Sensitivity Study Results for the Improved Sequential Algorithm

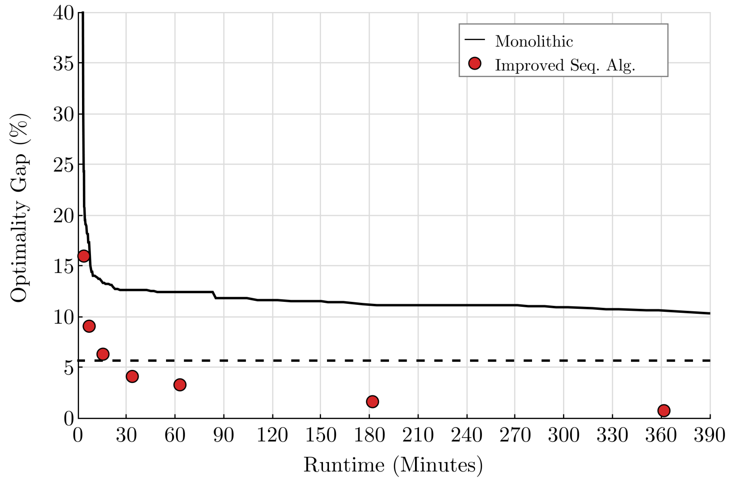
Table 40: Optimality Gap Results of the Improved Sequential Algorithm on Airport Subsets

PRAP Runtime	50 Airports	75 Airports	100 Airports
2 min	7.6%	10.5%	13.8%
5 min	6.7%	10.0%	11.3%
10 min	4.8%	9.9%	10.9%
30 min	2.7%	8.9%	10.3%
1 hr	1.6%	7.9%	9.2%
3 hrs	0.8%	7.6%	8.7%
6 hrs	0.4%	7.5%	8.8%

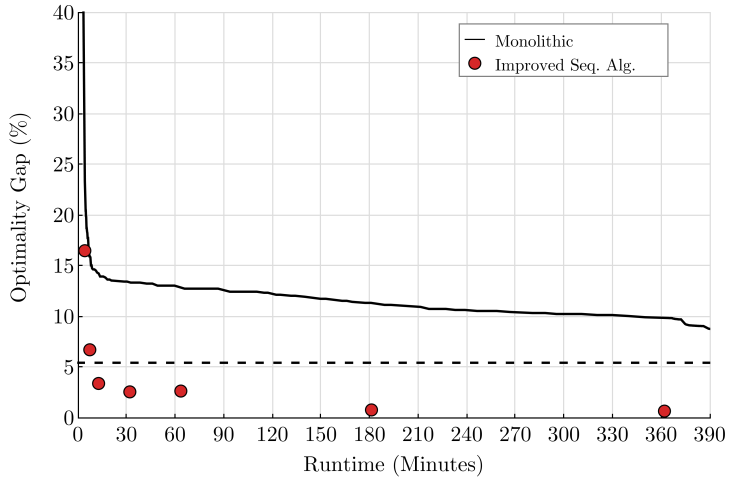
For the smallest problem instance, fifty airports, the Improved Sequential Algorithm found solutions with a less than 10% optimality gap in two minutes, and within three hours, it found solutions with a less than 1% gap. Its performance was also good on the larger problem instances. It found solutions with less than a 10% gap in ten minutes for the seventy-five-airport case and one hour for the one-hundred-airport case. This good performance of the Improved Sequential Algorithm shows that it is not only adept at handling more difficult problem instances (until the problem is very tightly-constrained), it is adept at handling easier problem instances in the realm where the monolithic approach is good, but not capable of producing optimal solutions in reasonable time frames.

7.6.4 Different Days of the Week

The different days of the week sensitivity studies also explore the Improved Sequential Algorithm's ability to solve easier problem instances than the baseline. The baseline problem instance was a *Monday.* Specifically, it encompassed the 24-hour period from the start of the night pick up activities on Monday evening to the end of the day delivery activities on Tuesday afternoon. Of the days of the week, Monday has the most volume and is the most difficult to solve. The weekend, Friday evening through Sunday evening, despite encompassing several calendar days, is the easiest to solve because significantly less volume enters the network and since there is more time to deliver the packages, more can be moved with trucks rather than aircraft. The results of the Improved Sequential Algorithm for each day of the week, from Tuesday through the weekend is shown in Figure 32.



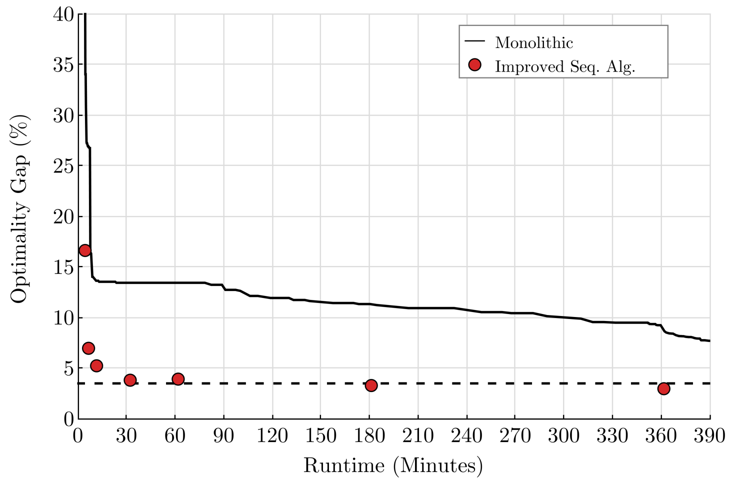
(a) Tuesday



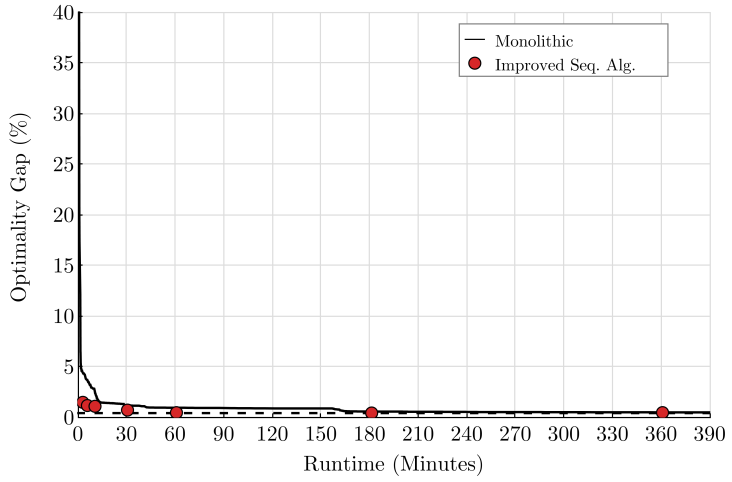
(b) Wednesday

Figure 32: Different Days of the Week Sensitivity Study Results for the Improved Sequential Algorithm

For all of the days of the week, the Improved Sequential Algorithm out-performs the monolithic approach. On the Tuesday and Wednesday problem instances, the heuristic algorithm finds better solutions than the best-found monolithic solution in under thirty minutes. It takes three hours for it to find a better solution than



(a) Thursday



(d) Weekend (Friday-Sunday)

Figure 32: Different Days of the Week Sensitivity Study Results for the Improved Sequential Algorithm (Cont.)

the monolithic for the Thursday case, and for the weekend case it never finds a better solution than the monolithic approach because the problem is so easy that the monolithic as very little trouble with it. The individual optimality gaps of the cases are shown in Table 41.

For all but the Thursday problem instance, the Improved Sequential Algorithm found solutions under a 1% gap, and for all problem instances it found solutions under a 10% gap in five minutes. Of the problem instances considered in this sensitivity study, the weekend case is clearly the easiest, with no solution having less than a

Table 41: Optimality Gap Results of the Improved Sequential Algorithm on Different Days of the Week

PRAP Runtime	Tuesday	Wednesday	Thursday	Weekend
2 min	15.9%	16.4%	16.6%	1.4%
5 min	9.0%	6.7%	6.9%	1.1%
10 min	6.3%	3.3%	5.2%	1.0%
30 min	4.1%	2.5%	3.8%	0.7%
1 hr	3.2%	2.6%	3.9%	0.4%
3 hrs	1.6%	0.7%	3.2%	0.4%
6 hrs	0.7%	0.6%	2.9%	0.4%

1.4% gap, and the Thursday problem instance is the most difficult. This is due to the Saturday-delivery products in addition to the usual commodities, which increases the number of commodities and therefore increases the problem size. Still, the heuristic approach found high-quality solutions quickly. These sensitivity studies show that the Improved Sequential Algorithm performs well for slightly easier problems beyond the baseline problem instance.

7.6.5 Temporal Scope

The expansion of the temporal scope sensitivity study explores the ability of the Improve Sequential Algorithm's ability to handle significantly more difficult problem instances. When these studies were performed on the monolithic approach, no usable solutions could be generated within two days of runtime. They were hopelessly intractable for a monolithic approach. However, the heuristic approach is more capable of solving these longer-scope problem instances. The results are shown in Table 42. The cases marked with the dagger symbol (†) are those in which the PRAP did not find a solution, aside from the starting heuristic solution, before being terminated.

Considering the intractability of these problems for a monolithic approach, the results are not bad. The optimality gaps for the six-hour cases are all between 23% and 28%. Although these gaps are not *as* good as the gaps for the single-day cases, the fact that solutions can be generated at all is a significant improvement over the

Table 42: Optimality Gap Results of the Improved Sequential Algorithm for Larger Temporal Scope

PRAP Runtime	2 Days	3 Days	4 Days	7 Days
2 min	37.0%†	38.0%†	37.7%†	36.8%†
5 min	27.5%	29.1%	37.7%†	36.8%†
10 min	26.2%	28.6%	37.7%†	36.8%†
30 min	26.2%	28.6%	37.7%†	36.8%†
1 hr	25.2%	28.6%	37.7%†	36.8%†
3 hrs	24.4%	26.7%	29.7%	25.9%
6 hrs	23.9%	26.7%	27.3%	25.9%

monolithic approach. These solutions would be valuable as rough estimates or as starting places for planners.

Since the week-long case is an important time period for express shipment planning, an alternative approach to generating week-long schedules was also explored. In this alternative approach, will be called the "reconstructed" approach, the PRAP of each day of the week was run individually, in parallel. Then, the assigned routings from each of these individual runs were combined and sent to the Aircraft Movement Problem to produce a full-week flyable solution. This approach is very similar to the approach employed in the Georgia Tech-FedEx collaboration for generating week-long schedules, except in that approach the whole ESSND problem is solved for each day. The results are significantly better than the previous approach in which a single PRAP encompassing the entire time period was solved. The results are shown in Table 43

With enough computational resources to run each PRAP in parallel, a solution with a 13.4% optimality gap can be generated for the entire week in less than one day of runtime (on average the AMP takes about thirteen hours). This is a major improvement over the monolithic approach and a significant improvement over the previous approach in which only a single PRAP instance was solved. Solutions with only a 13.4% optimality gap that could be generated in only a few hours would be

Table 43: Optimality Gap Results of the Improved Sequential Algorithm for the Reconstructed Week

PRAP Runtime	Opt. Gap
2 min	23.6%
5 min	19.1%
10 min	15.7%
30 min	14.7%
1 hr	14.3%
3 hrs	13.6%
6 hrs	13.4%

extremely valuable for an express shipment company.

7.6.5.1 Effect of Larger Temporal Scope on the Collapsed Product Set Sequential Algorithm

Although the performance of the Collapsed Product Set Sequential Algorithm was underwhelming for the single-day case, perhaps it's ability to reduce the PRAP problem size would be useful for problems with larger temporal scope. The CP-PRAP SAP only ever needs to be run for a single day's worth of overnight box volume (as long as that day has the largest volume), and the AMP is relatively easy to solve compared to the full formulation. As suspected, the Collapsed Product Set Sequential Algorithm *does* perform well on multi-day problem instances. The full summary of results are found in Table 44. For all cases, the CP-PRAP was used to generate package routings and it was run for ten minutes on the Monday overnight box products. The two-day volume was assigned to move with the daytime volume and was not flexible. Cases marked with an asterisk are those that broke sort capacity (sort capacity is not constant throughout the week).

The Collapsed Product Set Sequential Algorithm produces solutions under a 28% gap for all of the problem instances, and does so in well under a day of runtime. These results are on-par with those produced by the Improved Sequential Algorithm, though the CAP requires so much time to solve that the Collapsed Product Set Sequential

Table 44: Results of the Template-Based Sequential Algorithm for Multi-Day Problem Instances

Problem Instance	AMP Time	AMP Opt. Gap	CAP Time	Final Opt. Gap	Total Time
1 day	238 sec	22.26%	436 sec	21.48%	0.35 hrs
2 day	1666 sec	25.74%	4527 sec	25.26%	1.42 hrs
3 day	2584 sec	28.40%	18683 sec	27.91%	6.07 hrs
4 day	10675 sec	26.51%*	28176 sec	25.95%	10.96 hrs
7 day	11500 sec	23.88%*	43996 sec	22.71%	15.58 hrs

Algorithm is still inferior to the Improved Sequential Algorithm.

As with the single-day problem instances, the CAP added significant runtime in exchange for very little solution improvement. In fact, its runtime impact was even more severe than the single-day case, adding over a half-day of additional runtime for the full-week case. This makes the compelling case that while consolidation can be handled after the AMP, the CAP, as it is currently formulated, is not an effective method for doing so. Still, the challenges of the CAP should not detract significantly from the overall success of the approach.

Additionally, and this has not been mentioned up to this point since it was tangential to the discussion and not a *hard* requirement, express shipment companies enjoy schedules and routings with consistency even at the expense of absolute optimality. It is much easier to schedule crews and perform ground operations if, say, every night ATL sends one plane to MEM and one plane to IND, rather than a schedule in which the number of planes and the assignment of commodities to those planes changes each night. A consistent schedule may not be the mathematically optimal solution, but it may be much more *implementable*, which has significant, though difficult to quantify, value.

Overall, the ability of these heuristic approaches to produce solutions for an entire week of operations is an important contribution to both the literature and the express shipment community since it has not been (publicly) accomplished previously.

These heuristic approaches are able to overcome the challenges of solving such a large optimization problem and produce high-quality solutions for longer, more convenient time periods for express shipment companies.

7.7 Summary

In this chapter, several heuristic algorithms were developed and tested. Ultimately, the Improved Sequential Algorithm was found to be superior to the other heuristic algorithms and monolithic approach. Since heuristic algorithms are problem-specific and must be designed as such, research questions and hypotheses were posed as part of the development process of the algorithm. First, two implementation-based research questions were posed concerning the discipline definition and the discipline arrangement. Then, based on the results of exploratory experiments, the functional decomposition definition, which broke the express shipment problem into a Package Routing Problem and an Aircraft Movement Problem, was hypothesized to be appropriate for a heuristic decomposition algorithm. Next, a fixed point iteration-based approach was hypothesized as being an appropriate discipline arrangement, but experimental results led to the rejection of this hypothesis. Therefore, the package routing problem was altered to include aircraft movement variables and a sequential discipline arrangement was implemented and tested.

This sequential heuristic approach, called the Basic Sequential Algorithm, was capable of producing good-quality solutions more quickly than the monolithic approach for the first hour of runtime, thus supporting the fundamental hypothesis of this work. Its performance led to the next research question, which focused on whether the Basic Sequential Algorithm could be improved. This led to the Collapsed Product Set Sequential Algorithm, whose performance was inferior to the Basic Sequential Algorithm for the single-day baseline problem instance but was relatively good for problems of larger temporal scope. Research question 4 and the Basic Sequential Algorithm also led to the Improved Sequential Algorithm, which adds the phase-based hull count constraint to the Package Routing with Aircraft Problem. Adding a relaxed balance constraint, the sort-centric balance constraint, to the PRAP was also tested, but it was found to worsen the algorithm's performance. The final Improved Sequential Algorithm significantly out-performs the monolithic approach for the baseline problem instance. The Improved Sequential Algorithm was also tested in a number of sensitivity studies and it consistently out-performed the monolithic approach in those problem instances as well, showing that it can handle a variety of problems and maintain its superior performance. Importantly, it can also handle problems of larger temporal scope, including the full-week problem which is an important planning time frame for express shipment companies.

Based on the results of the algorithms developed in this chapter, particularly the Improved Sequential Algorithm, the fundamental and primary hypotheses of this work are supported. Additionally, thanks to their quick runtime and ability to provide high-quality solutions to previously intractable problems, these heuristic algorithms open the door for more tactical and strategic-level analyses for express shipment companies, potentially having an impact in the millions of dollars annually. The next chapter, Ch. 8 will introduce, implement and test the exact decomposition approach, Benders decomposition.

CHAPTER VIII

BENDERS DECOMPOSITION

In this chapter, an exact decomposition approach, Benders decomposition, will be described, implemented and tested. This is the next step in the investigation of hypothesis 0 which was first introduced in Chapter 5 in response to the fundamental objective of this work. That hypothesis is,

Hypothesis 0. *A heuristic approach or exact decomposition approach will able to produce solutions for the Express Shipment Service Network Design problem of comparable quality to the baseline monolithic approach in significantly less time.*

The heuristic decomposition approach was implemented and tested in the previous chapter so now it is time to implement and test an exact decomposition method. Based on a review of the literature (Ch. 4), Benders decomposition was determined to be an appropriate exact decomposition approach for the ESSND problem due to the presence of *complicating variables.* This is formalized as the research question,

Research Question 5. *Can Benders decomposition be used to solve the Express Shipment Service Network Design problem faster than the baseline monolithic approach?*

Since the literature suggests that Benders decomposition would be appropriate for the express shipment problem, my hypothesis in response to research question 5 is,

Hypothesis 5. *Based on the fact that Benders decomposition was specially developed to solved mixed integer programs, and it is commonly and successfully used for that purpose in the literature for similar transportation planning problems, I hypothesize* *that Benders decomposition will be capable of solving the Express Shipment Service Network Design problem faster than the baseline monolithic approach.*

The fundamental aim of Benders decomposition is to solve difficult problems that contain complicating variables, which, if fixed, make the problem significantly easier to solve. As a result, it is a very popular technique for solving challenging mixed integer programs, since MIPs reduce to linear programs if their integer variables are fixed. It was first introduced by J.F. Benders for linear and mixed-integer linear programs in 1962.[27] It has been since been expanded to additional problems such as to nonlinear and mixed-integer nonlinear programs with certain properties (*generalized Benders decomposition* [92]), and problems with logic relations, (*logic-based Benders decomposition* [115, 116]) which may be challenging to model as a "conventional" optimization problem.

Benders decomposition reformulates the original optimization problem into two problems: a relaxed master problem which only contains the complicating variables, and a subproblem in which the complicating variables are held fixed. The subproblem is then used to iteratively add constraints to the master problem until an optimal solution to the original problem is found. Since it iteratively adds constraints to the master problem, Benders decomposition is sometimes known as "row generation."

In this chapter, the theory-based mathematical basis for Benders decomposition is presented first. Then, a review of applications of Benders decomposition to airline and other freight problems are summarized. Next, Benders reformulation is applied to the ESSND problem, and methods for improving the classical implementation of Benders decomposition are discussed, including those tested for the ESSND problem. Finally, the ESSND implementation of Benders decomposition and the selected improvements are tested on the baseline problem instance and a smaller ten-airport subproblem.

8.1 Benders Reformulation and Decomposition

Consider the following mixed-integer program,

$$\begin{aligned} \min_{x,\,y} \quad & c^T x + d^T y \\ \text{s.t.} \quad & Ax + By \geq b, \\ & Dy \geq e, \\ & x \in \mathbb{R}_+, \\ & y \in \mathbb{Z}_+ \end{aligned} \tag{8.1}$$

If the integer variables (i.e. complicating variables), y, are fixed, the resulting optimization problem is,

$$\begin{aligned} \min_{x} \quad & c^T x \\ \text{s.t.} \quad & Ax \geq b - B\bar{y}, \\ & x \geq 0 \end{aligned} \tag{8.2}$$

Therefore, the original MILP can be rewritten as,

$$\min_{y \in Y} \left[d^T y + \min_{x \geq 0} \left\{ c^T x \mid Ax \geq b - By \right\} \right] \tag{8.3}$$

where $Y = \{y \mid Dy \geq e,\, y \in \mathbb{Z}_+\}$. The inner problem constitutes the decomposition *subproblem* (also sometimes called the *slave* or *worker* problem). When the integer variables are fixed, the subproblem is a continuous LP and can be dualized. Equation 8.3 can be rewritten as,

$$\min_{y \in Y} \left[d^T y + \max_{u \geq 0} \left\{ u^T (b - By) \mid u^T A \leq c \right\} \right] \tag{8.4}$$

where u is the appropriate vector of dual variables. Note that the space $U = \left\{ u \mid u^T A \leq c \right\}$ is independent of the choice of y variables. Assuming that U is not empty, it is composed of extreme points $u \in \mathcal{P}$ and extreme rays $u \in \mathcal{R}$.

Due to the duality relations, assuming that the original problem is not unbounded, when the dual subproblem is optimized over the space U, it will either be unbounded or optimal (corresponding to infeasibility or optimality in the primal, respectively). In the former case, the vector $u \in \mathcal{R}$ corresponds to a ray in the direction of the dual unboundedness and thus the primal infeasibility. In order to avoid this infeasibility, the master problem is restricted from these areas using *feasibility* cuts,

$$\bar{u}^T(b - By) \leq 0 \quad \forall u \in \mathcal{R} \subseteq U \tag{8.5}$$

where $\mathcal{R}$ is the set of the extreme rays of polyhedron $U = \{u^T A \leq c\}$. In the case in which the dual subproblem is not unbounded (or infeasible), the subproblem solution is an extreme point. This corresponds to a point with a feasible solution to the original MIP (8.1), but that solution is not optimal. Therefore, an auxiliary continuous variable, z, and (*optimality*) cuts,

$$\bar{u}^T(b - By) \leq z \quad \forall u \in \mathcal{P} \subseteq U \tag{8.6}$$

are used to cut off this sub-optimal solution and drive the relaxed master problem towards an optimal solution, where $\mathcal{P}$ is the set of the extreme points of the polyhedron U. The resulting formulation is called the Benders reformulation, which, using the notation thus far, is

$$
\begin{aligned}
\min_{y} \quad & d^T y + z && (8.7a)\\
\text{s.t.} \quad & \bar{u}^T(b - By) \le 0 \quad \forall u \in \mathcal{R} \subseteq U, && (8.7b)\\
& \bar{u}^T(b - By) \le z \quad \forall u \in \mathcal{P} \subseteq U, && (8.7c)\\
& y \in Y, && (8.7d)\\
& z \ge 0 && (8.7e)
\end{aligned}
$$

J.F. Benders showed that this reformulation is equivalent to the original MIP, formulation 8.1.[27] However, the sets $\mathcal{R}$ and $\mathcal{P}$ are prohibitively large for all but the smallest problems. Fortunately, only a subset of these constraints is active in the optimal solution. So, rather than create all of the constraints corresponding to $\mathcal{R}$ and $\mathcal{P}$, these constraints are iteratively added to the master problem.

Initially, the master problem only includes constraints 8.7d and 8.7e,

$$
\begin{aligned}
\min_{y} \quad & d^T y + z\\
\text{s.t.} \quad & y \in Y,\\
& z \ge 0
\end{aligned}
\tag{8.8}
$$

and the solution sets a lower bound. Then, the dual subproblem,

$$
\begin{aligned}
\max \quad & u^T(b - B\bar{y})\\
\text{s.t.} \quad & u^T A \le c,\\
& u \ge 0
\end{aligned}
\tag{8.9}
$$

is solved, keeping the $\bar{y}$ values found in (8.8) fixed. If the master problem solution found by (8.8) makes the subproblem (8.9) unbounded (i.e. the primal subproblem is *infeasible*), a feasibility cut, $\bar{u}^T(b - By) \le 0,\ u \in \mathcal{R}$, is added to the master problem using an extreme ray $u \in \mathcal{R}$ pointing in the direction of the dual unboundedness.

On the other hand, if the solution $\bar{y}$ found by the master problem (8.8) produces a feasible subproblem, then (8.9) produces an optimal extreme point $u \in \mathcal{P}$. This extreme point is used to create an optimality cut, $\bar{u}^T(b - By) \leq z$, which is added to the master problem. Additionally, the upper bound on the solution is updated using this new $\bar{u} \in \mathcal{P}$ solution.

Then, the master problem is solved again using the new constraint, and a new lower bound is calculated. If the lower bound is sufficiently close to the upper bound (i.e. within a user-set tolerance, ε), the algorithm is terminated. Otherwise, the subproblem is re-solved using the updated values for $\bar{y}$, and cuts are added until the termination criterion is met. The details of the algorithm are outlined in Algorithm 2.

Algorithm 2 Classical Benders Decomposition (adapted from Kalvelagen [123])

```
1: $LB \leftarrow -\infty,\ UB \leftarrow \infty$
2: while $UB - LB > \varepsilon$ do
3: Solve Subproblem:
4:     max $\left\{u^T(b - By) \mid u^T A \leq c,\ u \geq 0\right\}$
5:     if Unbounded then
6:         Get unbounded ray $\bar{u} \in \mathcal{R}$
7:         Add cut $\bar{u}^T(b - By) \leq 0$ to the master problems
8:     else
9:         Get extreme point $\bar{u}_p$
10:        Add cut $\bar{u}_p^T(b - By) \leq z$ to the master problem
11:        $UB \leftarrow \min\left\{UB,\ d^T\bar{y} + \bar{u}_p^T(b - By)\right\}$
12:    end if
13: Solve Master Problem:
14:    $\min\{d^T y + z \mid cuts, y \in Y\}$
15:    $LB \leftarrow \max\left\{\text{LB},\ d^T y + z\right\}$
16: end while
```

8.1.1 Formulation of ESSND Problem for Benders Decomposition

For the Express Shipment Service Network Design problem, the initial master integer program is formulated,

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in A} d_{ij}^f y_{ij}^f && && (8.10a)\\
\text{s.t.} \quad & \sum_{j:(i,j)\in A} y_{ij}^f = \sum_{j:(j,i)\in A} y_{ji}^f && \forall i \in N, f \in F \setminus \tilde{F} && (8.10b)\\
& \sum_{(i^+,j)\in A} y_{i^+j}^f \le n_f && \forall f \in F \setminus \tilde{F} && (8.10c)\\
& \sum_{f\in \hat{F}} y_{ij}^f \le 1 && \forall (i,j) \in A && (8.10d)\\
& y_{ij}^f \in \mathbb{Z}_+ && \forall (i,j) \in A, f \in F && (8.10e)
\end{aligned}
$$

where, in order, the constraints included are the vehicle balance, hull count and feeder limit constraints. The other constraints are captured in the primal feasibility problem which is,

$$
\begin{aligned}
\min \quad & \varnothing && && (8.11a)\\
\text{s.t.} \quad & \sum_{k\in K}\sum_{p\in P^k} \delta_{ij}^p b^k x_p^k \le \sum_{f\in F} u^f \bar{y}_{ij}^f && \forall (i,j) \in A && (8.11b)\\
& \sum_{p\in P^k} x_p^k = 1 && \forall k \in K && (8.11c)\\
& \sum_{k\in K}\sum_{p\in P^k} \delta_h^p n_k x_p^k \le e_h && \forall h \in H && (8.11d)\\
& 0 \le x_p^k \le 1 && \forall p \in P^k, k \in K && (8.11e)
\end{aligned}
$$

and contains, in order, the forcing, cover and sort capacity constraints. The dual of the primal formulation 8.11 is,

$$\begin{aligned}
\min \quad & \sum_{(i,j)\in A} \gamma_{ij}\lambda_{ij} + \sum_{k\in K}\mu_k + \sum_{h\in H} e_h\pi_h + \sum_{k\in K}\sum_{p\in P^k}\alpha_p^k && && (8.12a)\\
\text{s.t.} \quad & \sum_{(i,j)\in A} \delta_{ij}^p b^k \lambda_{ij} + \mu_k + \sum_{h\in H}\delta_p^h n_k \pi_h + \alpha_k^p \geq 0 && \forall k\in K, p\in P^k && (8.12b)\\
& \lambda_{ij} \geq 0 && \forall (i,j)\in A && (8.12c)\\
& \pi_h \geq 0 && \forall h\in H && (8.12d)\\
& \alpha_p^k \geq 0 && \forall k\in K, p\in P^k && (8.12e)\\
& \mu_k \in \mathbb{R} && \forall k\in K && (8.12f)
\end{aligned}$$

where $\gamma_{ij} \equiv \sum_{f\in F} u^f \bar{y}_{ij}^f$ for clarity, and the dual variable λ_{ij} corresponds to the forcing constraint, μ_k corresponds to the cover constraint, π_h corresponds to the sort capacity constraint, and α_k^p corresponds to the upper bound on x_k^p.

Since the ESSND problem objective function only contains design variables, the primal Benders subproblem, Eq. 8.11, is a *feasibility* problem. This means that only feasibility cuts will be added to the master problem. When the dual Benders subproblem returns an optimal solution, that indicates that an optimal master problem solution has been found which is feasible in the subproblem and therefore feasible and optimal in the original problem.

8.2 Relevant Applications of Benders Decomposition in the Literature

As shown in the comprehensive survey paper by Rahmaniani et al. (2017)[168], Benders decomposition (BD) has attracted significant attention in the mathematical programming community, particularly in recent years. It has been applied to a wide variety of problems and applications, including those outside the original scope of

Benders' 1962 paper, in large part due to the generalization by Geoffrion (1972)[92].

Costa (2005)[56] presents a survey of the application of Benders decomposition to fixed-charge network design (FCND) problems, including the capacitated FCND problem, which is relevant to many transportation problems including the ESSND problem. Costa found that "the relative ease of solving the auxiliary subproblem in network design formulations make of Benders decomposition one of the most appropriate approaches."

Rahmaniani et al. (2017)[168] present a thorough survey of the use of Benders decomposition in a variety of applications. In particular, they identify and classify the shortcomings of the approach as well as the various strategies that have been employed to overcome those shortcomings. Additionally, they discuss the common extensions of BD beyond its original LP and MIP scope.

Finally, Magnanti and Wong (1984)[146] provide an overview of the method, in addition to other methods, in the context of network design and transportation problems. They discuss the algorithm and some improvements to it, particularly their own "Pareto-optimal" Benders cuts, and illustrate the usefulness of the algorithm for transportation applications.

8.2.1 Applications to Airline Problems

To my knowledge, only Li et al. (2006)[137] have explicitly applied Benders decomposition to an airline problem involving cargo routing, though their application was for a combination carrier rather than a cargo-only airline. Additionally, Kim and Barnhart (1997, 1999)[129, 131, 132] present an approximate solution to the Service Network Design and Express Shipment Service Network Design problems that, while not stated explicitly, is similar to Benders decomposition.

Li et al. (2006) apply BD to a fleet assignment and cargo routing problem for a combination carrier airline with a single hub. They solve an integrated model that encompasses passenger fleet allocation and cargo routing, given a flight schedule.[137] In their application of BD, the cargo routing model constitutes the subproblem and fleet allocation model is the master problem. They present results comparing a state-of-the-art branch-and-bound solver, classical Benders decomposition, a BD variant employing Pareto-optimal cut generation (from Magnanti and Wong (1981) [151]), and a BD variant employing the ε-optimal approach (Geoffrion and Graves (1974) [93]). They found the classical Benders variant to be the most successful of the four approaches. It should be noted that this problem studied by Li et al. while displaying a number of similarities to the ESSND problem studied in this work, is significantly smaller and simpler thanks to a pre-set flight passenger flight schedule, a single-hub network, a prohibition on cargo consolidation outside of the main hub, a smaller vehicle set, and the overall size of the carrier.

Kim (1997)[129] and Kim and Barnhart (1999)[132, 131] present an approximate formulation for the service network design problem and extend that formulation to express shipment applications. Their approximate formulation implicitly models the commodity flows using cut-set inequalities rather than explicitly using flow variables. Including all of the potential cut-set inequalities to even a modestly-sized problem would be computationally prohibitive so they add a subset of the inequalities, thus leading to an approximate formulation and solution. Additionally, they present column generation, row generation, and synchronized column-and-row generation algorithms for solving the exact and approximate formulations. Although not explicitly stated, the row generation and synchronized column-and-row generation algorithms are similar to Benders decomposition, particularly since Costa, Cordeau and Gendron (2009)[57] later showed that cut-set inequalities are a subset of Benders feasibility cuts.

Other applications of Benders decomposition to airline problems have focused on passenger airlines, particularly the difficult problem of concurrently solving the fleet allocation, aircraft routing, crew scheduling and maintenance scheduling problems (or some subset). Traditionally, these problems are solved sequentially in an attempt to mitigate the computational intractability of an integrated model, which results in suboptimal solutions. Several researchers found a combination Benders decomposition/column generation approach to work well for these airline problems.

Cordeau et al. (2001)[53] solve the simultaneous aircraft routing and crew scheduling problem using an approach based on Benders decomposition. Due to the nature of their problem, the master problem solves the aircraft routing problem and the subproblem solves the crew scheduling problem. Column generation was used to solve both the master problem and subproblem, and a heuristic branch-and-bound algorithm was used to compute integral solutions. They found their method provided solutions that significantly reduced costs for a data set provided by an airline when compared to a sequential planning process.

Mercier, Cordeau and Soumis (2005)[156] solve the integrated aircraft routing and crew scheduling problem using an algorithm that, like the previous work of Cordeau et al. (2001), uses a combination Benders-column generation algorithm. They found that setting the crew scheduling problem as the master problem, rather than the aircraft routing problem as in their previous work, resulted in significant run time improvements. Additionally, they found the Pareto-optimal cuts introduced by Magnanti and Wong (1981)[151] to improve the convergence time of the algorithm.

Mercier and Soumis (2007)[157] revisited their problem once more, this time adding flight re-timing to their integrated problem. As in their previous work, they utilized a combination Benders-column generation approach but added an additional dynamic constraint generation procedure. They found their Benders decomposition-based methodology to significantly improve the solution time and solution quality of their problem compared to a sequential optimization approach and their previous work.

Papadakos (2009)[164] also applied a combination Benders-column generation algorithm to an airline scheduling problem which included fleet assignment, aircraft maintenance routing and crew scheduling problem. It was first solved using a sequential optimization approach, which produced suboptimal solutions. In order to find optimal solutions, an enhanced Benders decomposition combined with accelerated column generation algorithm was used. Papadakos found the combination BD-column generation algorithm to produce high-quality solutions to the problem, which was previously considered intractable for realistic problem instances.

8.2.2 Applications to other Transportation and Network Design Problems

Üster and Agrahari (2011)[188] use Benders decomposition to solve a freight-forwarding network design problem that includes the selection of consolidation and de-consolidation centers as well as line haul routing. In order to improve the quality of their Benders cuts, they utilize a two-phase approach similar to the approach presented by Van Roy (1986)[190]. Their two-phase solution method allows them to generate stronger Benders cuts compared to the classical BD method. They found their method to be an improvement over solving the same model using a state-of-the-art branch-and-bound solver (CPLEX), particularly when solving large models which quickly run out of available computational resources when using CPLEX.

Agarwal and Özlem (2008)[2] implement and compare the performance of a greedy heuristic, a column generation and a two-phase Benders decomposition to the liner shipping service network design problem that sought to simultaneously solve the ship scheduling and cargo routing problems. The column generation is a “pure” column generation approach, and the Benders decomposition uses column generation solve the master problem. They find that the greedy heuristic is the fastest, but it produces significantly worse solutions than the other two approaches. The pure column generation approach produced solutions on par with the quality of the Benders decomposition implementation but took longer to generate them.

Cordeau, Pasin and Solomon (2006)[52] develop an integrated framework for the deterministic logistics network design problem (LNDP), and present a simplex-based branch-and-bound approach and a Benders decomposition approach for solving the problem. They describe the LNDP as the problem that aims to "satisfy customer demands while minimizing the sum of fixed and variable costs associated with procurement, production, warehousing, and transportation." They find the two-phase Benders decomposition approach to be "somewhat" superior to the branch-and-bound technique, and the inclusion of valid inequalities to improve the performance of both solution methods significantly. Additionally, they found the use of Pareto-optimal Benders cuts (Magnanti and Wong (1981)[151]) improved the overall performance of the Benders decomposition even at the expense of increasing the subproblem runtime.

Huang, Lee and Xu (2006)[117] employ Benders decomposition to balance the *workload* at two air cargo terminals in order to improve the performance of the airline. Since the air cargo loads were found to be difficult to predict, they used a stochastic MILP to model the system. They then used Benders decomposition accelerated by a modified feasibility cut to successfully solve the problem, resulting in operating improvements for the carrier.

Geoffrion and Graves (1974)[93] applied a BD to a multicommodity distribution system design problem with minor modification. Namely, they relaxed the requirement that the master problem be solved to optimality at each iteration. Additionally, they found that problem formulation could play a large role in the performance of the algorithm. In particular, for their problem, they found that a formulation that reduced the number of constraints (which is generally favorable for LPs) performed significantly worse when solved using Benders decomposition because the resulting cuts were weaker than those of the other formulation. Overall, they found their application of Benders decomposition to satisfactorily solve their multicommodity distribution system design problem for large (for the day) realistic problem instances. They credited their success to the fact that relatively few Benders cuts were required for their problem, which they largely attributed to the tightness of their formulation.

Magnanti, Mireault and Wong (1986)[145] focus on the application of Benders decomposition to uncapacitated network design. They explore five Benders cuts beyond the "classical" Benders cut. Three of the cuts were from the literature and were derived as lower bounds for uncapacitated network design problems. Magnanti, Mireault and Wong show that these can be interpreted as Benders cuts. They additionally introduce a new Benders optimality cut, which they call a "strong" cut, and review the Pareto-optimal Benders optimality cut from their previous work. Additionally, a pre-processing method using the dual ascent heuristic is used to reduce the size of the original problem. They find that the BD application using "strong" cuts outperforms the classical BD application, and the Pareto-optimal cuts outperform other cuts on difficult problems.

The applications of Benders decomposition to a variety of problems are wide-ranging and extensive. Only a subset of the hundreds of papers is mentioned here. For a wider survey, the previously mentioned recent survey paper by Rahmaniani et al. (2017)[168] is an excellent place to start.

8.3 Enhancements to Benders Decomposition

Although Benders decomposition has been shown to provide improvements in solution quality over heuristic approaches and improvements in runtime over branch-and-bound approaches, as seen in the previous section, many past researchers have found it necessary to alter or enhance the original algorithm in order to realize these improvements. That is, for many problems Benders Decomposition as it was originally formulated does not perform well. Rahmaniani et al. (2017)[168] comment that "a straightforward application of the classical BD algorithm may require excessive computing time and memory." The same paper identifies several common problems with classical BD: "time-consuming iterations; poor feasibility and optimality cuts; ineffective initial iterations; zigzagging behavior of the primal solutions and slow convergence at the end of the algorithm; and upper bounds that remain unchanged in successive iterations because equivalent solutions exist."

This prompts the second Benders-based research questions,

Research Question 6. *Which, if any, enhancements to Benders decomposition from the literature improve its performance when solving the Express Shipment Service Network Design problem?*

To classify the approaches that previous researchers had taken to improve Benders Decomposition in light of these shortcomings, Rahmaniani et al. employ a four-category taxonomy. Those categories are,

1. *Decomposition Strategy*: the initial master and subproblem definitions.
2. *Solution Procedure*: the choice of algorithms for solving the master and subproblems.
3. *Solution Generation*: the method to generate values for the complicating variables.
4. *Cut Generation*: the strategy for obtaining optimality and feasibility cuts.

The first category, the decomposition strategy, encompasses approaches that alter the decomposition definition in an effort to improve the convergence behavior of the algorithm. In particular, Crainic et al. (2014,2016)[62, 63] comment that the traditional Benders decomposition "causes the [master problem] to lose all the information associated with the non-complicating variables," and "the problem structure associated with the linking constraints is lost, and thus many classical VIs [valid inequalities] are not applicable."[168] However, in practice, altering the decomposition definition is not very common. Rahmaniani el al. (2017) note that only four percent of the papers they reviewed implemented this technique.

The second category, the solution procedure, concerns methods for solving the master problem (MP) and subproblem. The most common way to solve the MP is to use branch-and-bound and the most common way to solve the subproblem is with simplex. However, in problems where either of these problems is particularly difficult to solve, these traditional approaches may be insufficient. Column generation is sometimes undertaken in either the MP or subproblem, but it requires a difficult integration if used in the master problem because of the simultaneous generation of columns *and* rows.[159] Other subproblem approaches typically focus on exploiting problem-specific features and the opportunity to split the subproblem into multiple subproblems that can be solved in parallel.

Other MP solution techniques include metaheuristics, constraint programming, and, of particular interest for this application, "single search tree" Benders decomposition. Single search tree BD, also called, "modern Benders," "one-tree Benders" and "Branch-and-Benders-cut (B&BC)," takes advantage of the properties of modern optimization solvers to avoid revisiting parts of the branch-and-bound tree in master problem. Rather than solving the MP to completion at every iteration, modern Benders generates Benders cuts for integer and fractional solutions within the branch-and-bound tree. This can save considerable computational effort since parts of the B&B tree are not visited again with each iteration.

The third category, solution generation, focuses on ways to improve the master problem so that it solves more quickly and/or produces better solutions. Rahmaniani et al. (2017)[168] categorizes these approaches as, 1) using alternative formulations, 2) improving the MP formulation, and 3) using heuristics to generation solutions or to improve those already found. The use of alternative MP formulations involves temporarily changing the MP. Notable approaches include the two-phase, cross decomposition and regularized decomposition approaches. The two-phase approach first solves a relaxed version of the MP, generates cuts, and then, at a certain point, re-introduces the integrality constraint. The cross decomposition approach combines Lagrangian relaxation and Benders decomposition. Finally, the regularized decomposition approach seeks to reduce instability that may arise in the algorithm by introducing a quadratic term in the MP objective function that keeps the subsequent solution close to a reference point.

The second subcategory, improvements to the MP formulation, is handled through the addition of valid inequalities to the master problem before the traditional BD algorithm begins. The types and effectiveness of these valid inequalities depends on the problem. Finally, the third subcategory, heuristics, includes approaches that warm-start the MP to obtain better initial cuts, or "repair" solutions so that optimality cuts, rather than the less-desirable feasibility cuts, can be added.[1]

Finally, the fourth and final category is concerned with cut generation. In the original Benders decomposition algorithm, the optimality and feasibility cuts are generated based on the solution point or ray generated in the subproblem. However, if the subproblem is degenerate, which is a common occurrence, there may be many possible cuts that could be added to the master problem, and some of these cuts will be more effective than others. In the classical Benders decomposition, the point or ray is chosen at random. Improved Benders cuts seek to choose the best (or at least a very good) possible Benders cut. The first and best-known cuts of this kind are the "Pareto-optimal" Benders cuts, which were introduced in the seminal paper by Magnanti and Wong (1981)[151]. Since that time, additional improved Benders cuts have been formulated. The primary drawback of these improved cuts is that

[1]Optimality cuts are more desirable than feasibility cuts because they improve the upper bound.

they often require more difficult subproblems or even additional auxiliary problems. However, despite these drawbacks, they are a very popular approach for improving the performance of Benders decomposition.

The ESSND problem is not immune to the shortcomings of a basic Benders decomposition implementation, as will be shown in Section 8.4. In an effort to improve the performance of the Benders decomposition, three modifications were implemented: a solution procedure improvement, a solution generation improvement, and a cut generation improvement. The solution procedure was modified by using the “modern” Benders decomposition algorithm rather than the classical. All implementations of BD in this work, including the baseline application, include this modification. Additionally, valid inequalities based on cut-set inequalities were added to the master problem to improve the solution generation. Finally, two types of improved Benders cuts were explored, 1) an updated version of Magnanti and Wong’s Pareto-optimal cuts by Papadakos (2008,2009)[163, 164], and 2) Minimal Infeasible Subsystem (MIS) cuts by Fischetti, Salvagnin and Zanette (2010)[82]. A decomposition strategy modification was not considered because it is a much less common approach than the other three.

8.3.1 Modern Benders Decomposition

The “modern” Benders decomposition algorithm, also called single search tree Benders decomposition and branch-and-Benders-cut algorithm, takes advantage of the ability of modern B&C solvers to accept additional constraints *during* the progression of the B&C algorithm. Additionally, Geoffrion and Graves (1974)[93] found that sub-optimal, and even fractional, MP solutions can be used to produce valid Benders cuts. These two factors form the foundation of the single-tree BD.

Unlike the classical Benders decomposition algorithm, in which the MP is solved to integrality at every iteration before the subproblem is solved, in the single search tree BD, the subproblem is solved whenever the MP B&C algorithm finds a new incumbent integral solution. Some implementations take this a step further and solve the subproblem for fractional solutions as well, particularly at the root node. This approach of adding constraints throughout the progression of the master problem branch-and-cut solution algorithm has several advantages. First, a large amount of rework can be avoided by only solving one B&B tree. In the classical BD, a new B&B tree is created and solved at every iteration. This means that there may be certain branches containing poor-quality solutions that appear in several trees and those branches must be revisited and pruned with every resolve of the MP. The single-tree BD algorithm avoids this potentially substantial rework.

Second, because the single-tree BD solves the subproblem each time a new incumbent solution is found, there is a chance that an optimal solution may be found that would have otherwise been pruned in the classical algorithm. In other words, in the classical BD, the true optimum solution of the original problem may be pruned from the MP B&B tree because it is suboptimal for that *that* particular iteration of the master problem, which is a relaxation of the original problem. It is not until the MP is sufficiently tightened with additional cutting planes that the previously "suboptimal" solution is recognized as optimal. In the single-tree BD, this solution can be identified more quickly.

The potential disadvantage of the single-tree BD is the fact that time may be spent generating cuts that would have otherwise been unnecessary if the MP were solved to "optimality" at every iteration. This consideration may be particularly important if the subproblem is difficult to solve, or if it results in such a dramatic increase in the size of the MP that it slows down the progress of the B&B algorithm. However, in most situations, this is not an issue thanks to the way that modern solvers handle user-generated cuts, and if it is an issue, the frequency of cut generation can be tailored in the algorithm itself.

In the single-tree BD, Benders cuts are added to the MP branch-and-bound through the use of so-called “lazy constraint” callbacks. Callbacks are opportunities provided by the solver to interact, either actively or passively, with the branch-and-cut algorithm. Passive callbacks are used to gather information about the state of the algorithm while active callbacks are used to change the progression of the algorithm, most commonly to add cutting planes.

These user-provided cutting planes come in two forms, 1) “user cuts,” and 2) “lazy constraints.”[2] They are both similar to the cutting planes added in the course of the typical B&C algorithm in that they are additional “constraints” that are added to the formulation with the purpose of cutting off parts of the LP hull that produce fractional solutions.

The difference between user cuts and lazy constraints lies in *how much* each cuts off. The user cuts act just like the general cutting planes presented in Section 4.1.2.1. They cut off parts of the LP hull in order shape it to be closer to the convex hull. User cuts are most commonly used when a problem has a certain structure that allows for specialized cutting planes to be added to the problem.

Lazy constraints cut off much more of the space, and specifically, may cut off *integer* solutions. This gives rise to the name – they are constraints that presumably *should have* been included at the start of the algorithm, but the user was “too lazy” to include them. Since they cut off previously “feasible” integer solutions, certain assumptions made in the branch-and-bound tree may no longer be valid so the handling of the branch-and-bound tree must be more sophisticated. One notable drawback of lazy constraints is that the valuable problem reductions typically performed by solvers prior to the creation of the branch-and-bound tree cannot be performed to the same extent for risk of producing a wrong solution. As a result, even if no integer solutions

[2]This terminology is specific to modern solvers, but fairly standardized across the most popular solvers today. Gurobi and CPLEX both use this terminology.

are cut off through the use of lazy constraints, the solver will typically require more time to solve the problem than if these cuts were added as "user cuts."

Since the Benders cuts are added to the MP for the express purpose of cutting off integer solutions, they are added as lazy constraints in the single-tree BD algorithm. In practice, the algorithm is similar to that presented for the classical BD, Algorithm 2 on page 240 except that the master problem is not solved to completion at every iteration and the Benders cuts are added when a new integer feasible solution is found. This is the way that the single-tree BD is implemented in this work, although some implementations additionally add cuts from fractional solutions and/or use a two-phase approach in which the algorithm is started, then stopped and restarted after a certain number of cuts have been generated.

Additionally, the implementation of single-tree BD in this work uses the *parallel* B&C algorithm in order to take advantage of the significant runtime improvements it offers. The primary drawback of using the parallel B&C algorithm is the potential difficulty in implementation. Thread safety must be handled by the user since it is not guaranteed by the solver. In cases where solving the subproblem may take a long time, this means that a separate subproblem must be maintained for each of the computational threads. Fortunately, in this implementation, the subproblem is solved very quickly and so simply putting a temporary "lock" on the thread was found to be sufficient without greatly stalling other threads.

8.3.2 Improved Benders Cuts

In the classical Benders decomposition, when the subproblem is solved and a cut is generated, there is no consideration of the relative quality of an optimality or feasibility cut relative to the other potential cuts that could have been generated from that subproblem. This difference in the relative quality of cuts arises as a result of degeneracy in the subproblem. *All* cuts generated from a particular subproblem *will* cut off the infeasible or suboptimal master problem solution from the space, but some cuts will *stronger* than others. The strength of the Benders cuts is important since it has a direct bearing on the total number of iterations that will be necessary for the algorithm to converge. Unfortunately, finding the strongest possible cut is typically not an easy problem. Approaches to finding the best possible Benders cuts typically feature an auxiliary problem or modify the original subproblem in such a way so as to perturb it towards a better solution. Some approaches are heuristic in nature, while others attempt to find the provably best cut.

8.3.2.1 Pareto-Optimal Cuts

By far, the best-known improved Benders cut is Magnanti and Wong's "Pareto-optimal" Benders cut.[151] If implemented exactly as it is outlined, it will produce the strongest possible optimality cut. Unfortunately, it can be very difficult to implement without significant computational expense.

To understand these Pareto-optimal cuts, three definitions must first be established (all three are restated from Papadakos (2008)[163]). First, the criteria for comparing two cuts,

Definition 1. *A cut, $u^T(b - By) \leq z$, $u \in \mathcal{P}$ corresponding to $u_1 \in U$, **dominates** that corresponding to $u_2 \in U$, if:*

$$u_1^T(b - By) \geq u_2^T(b - By), \quad \forall y \in Y,$$

with strict inequality for at least one point $y \in Y$.

This means that a cut that dominates another is stronger than the other. Second, this definition of dominance is used in the definition of Pareto-optimality for Benders optimality cuts,

Definition 2. *A cut is **Pareto-optimal** if it is not dominated by any other cut. Similarly, the point u corresponding to that cut is called Pareto-optimal.*

Finally, the definition of a core point,

Definition 3. *A point $y_0 \in ri(Y^c)$ is called a* ***core point*** *of Y, where $ri(S)$ and S^c are the relative interior and convex hull of set $S \subseteq \mathbb{R}^k$, respectively.*

That is, the core point is a point that is *strictly inside* the convex hull of Y. Using these definitions, Magnanti and Wong (1981) provide the following theorem (also restated from Papadakos (2008)[163]),

Theorem 1. *Let y_0 be a core point of Y, and let $\bar{u}$ be the optimal solution of the dual subproblem (8.9), then the optimal solution u_0 of the problem:*

$$\begin{aligned} \max \quad & u^T(b - By_0) \\ \text{s.t.} \quad & u^T(b - B\bar{y}) = z(\bar{u}), \\ & u^T A \leq c \end{aligned} \tag{8.13}$$

is Pareto-optimal.

The proof of this theorem can be found in the original paper.[151] The implication of this theorem is that after the subproblem is solved, an additional auxiliary problem (8.13) can be solved to produce the Pareto-optimal optimality cut. Using the core point y_0 and the previously found subproblem objective value $z(\bar{u})$, this auxiliary problem finds the strongest optimality cut (i.e. closest to the convex hull) to add to the Benders master problem.

From a theory-based point of view, Theorem 1 is a very important finding. It shows that the Pareto-optimal point can be found and provides a framework for how to find it. From an implementation point of view, however, it faces a number of challenges. First, the fact that an additional problem must be solved may outweigh the benefits of producing a superior cut. If solving the Benders subproblem is difficult or expensive, then solving a second, similar problem may not be worth the effort.

Second, to make matters worse, this secondary problem may suffer from instability, making it even more challenging. Third, finding core point y_0 may be very difficult and/or computationally expensive. In many situations, finding a core point may require yet another auxiliary problem, which must be carefully formulated so as to provide a point *inside* the convex hull. Approximate core points can be used, but the resulting cut is no longer guaranteed to be Pareto-optimal.

Finally, and this is the most critical for the ESSND application, Pareto-optimal Benders cuts are only *optimality* cuts as they were originally conceived. In Magnanti and Wong's improved Benders decomposition algorithm using these Pareto-optimal cuts, feasibility cuts, which are the *only* cuts produced in the ESSND implementation, are produced using the classical BD method. This is because the cut-generating auxiliary problem (8.13) uses the optimal value of the subproblem $z(\bar{u})$ to ensure that Pareto-optimal point is an optimal point. Since Benders feasibility cuts are only produced when the dual Benders subproblem is unbounded, no meaningful objective value exists. As a result, using Pareto-optimal cuts as they were originally formulated is impossible when solving the ESSND without altering the mathematical formulation

However, Papadakos (2008)[163] makes several observations and adjustments to the method for generating Pareto-optimal cuts that extends the approach to both optimality and feasibility cuts. Specifically, in his re-formulation of the Magnanti and Wong subproblem, the problematic equality constraint (i.e. $u^T(b - B\bar{y}) = z(\bar{u})$) is omitted. This results in the following theorem (restated from Papadakos (2008)[163]),

Theorem 2. *Let y_0 be a core point of Y, then the optimal solution u_0 of the following problem:*

$$\begin{aligned} \max \quad & u^T(b - By_0) \\ \text{s.t.} \quad & u^T A \leq c \end{aligned} \tag{8.14}$$

is Pareto-optimal.

The proof can be found in the original paper.[163] Papadakos (2008) calls (8.14) the

Independent Magnanti-Wong problem. Since the problematic equality constraint is excluded, this problem can be solved without first needing to solve the subproblem or even the master problem. Additionally, and critically for the ESSND BD implementation, this also means that Pareto-optimal *feasibility* cuts can be generated.

Thanks to the modifications made by Papadakos (2008) which allow for the creation of Pareto-optimal feasibility cuts, the impact of these improved cuts on the ESSND Benders decomposition implementation is explored. The results are found in Section 8.4.

The removal of the equality constraint allows for the creation of Pareto-optimal feasibility cuts, but two of the implementation challenges still remain – the potentially significant runtime detriment of solving the additionally auxiliary problem, and finding a core point. Fortunately, the subproblem and the Magnanti-Wong problem both solve quickly so that issue is not limiting. However, finding a core point is still challenging. To this end, another definition and theorem provide some assistance. First, is the definition of a *Magnanti-Wong point* as defined by Papadakos (2008),

Definition 4. *A **Magnanti-Wong point** is a point y_0 for which the optimal solution of the Magnanti-Wong problem (8.13) or the independent Magnanti-Wong problem (8.14) gives a Pareto-optimal u_0.*

This definition expands the scope of the possible points that can be used to find Pareto-optimal points. Although core points are still valid Magnanti-Wong points, Papadakos (2008) proves that core points are not the *only* points that can be used to find Pareto-optimal cuts. The precise details of what constitutes a Magnanti-Wong point can be found in the original paper.[163] Additionally, for cases in which it may still be challenging to find a Magnanti-Wong point, he provides this theorem,

Definition 5. *If y_0 is a Magnanti-Wong point, and $y \in Y$, then any convex combination of y_0 and y is also a Magnanti-Wong point.*

The proof of this theorem can be found in the original paper.[163] This theorem means that if one Magnanti-Wong (M-W) point is found, then the convex combination of that M-W point and the most recent MP solution will produce a valid M-W point. For this implementation of Pareto-optimal cuts for the ESSND problem, this process was approximated. The initial point was not chosen to be an M-W point, but rather a known feasible point. Once again, pirate ships are used as rule-breaking vehicles to produce a "feasible" solution. Convex combinations of this y_0^* and the most recent y were used to produce approximate M-W points. This approximation means that the cuts are not guaranteed to be optimal. Additionally, as in the Independent Magnanti-Wong algorithm of Papadakos (2008)[163], both classical Benders feasibility cuts as well as "Pareto-optimal" cuts are added to the master problem at each iteration.

8.3.2.2 Minimal Infeasible Subsystem (MIS) Benders Cuts

In addition to the Pareto-optimal cuts from Papadakos (2008), the effectiveness of *Minimal Infeasible Subsystem (MIS)* cuts from Fischetti, Salvagnin and Zanette (2010) was explored for the ESSND problem. MIS cuts were developed based on the observation that "finding a most-violated optimality cut is equivalent to finding an optimal vertex of a polyhedron with unbounded rays, which is a strongly $\mathcal{NP}$-hard problem."[82, 89] Additionally, they observe that "Benders separation can always be rephrased as a pure feasibility problem."[82] Using these observations, Fischetti, Salvagnin and Zanette (2010) alter the dual Benders subproblem to be similar to cut-generating linear program (CGLP) by Balas, Ceria and Cornuéjols[17],

$$\begin{aligned} \max \quad & u^T(b - B\bar{y}) - u_0\bar{z} && (8.15a) \\ \text{s.t.} \quad & u^T A \le u_0 c, && (8.15b) \\ & \sum_{i=1}^{m} w_i u_i + w_0 u_0 = 1, && (8.15c) \\ & (u, u_0) \ge 0 && (8.15d) \end{aligned}$$

Where $\bar{z}$ is the current value of z in the master problem and w is a vector of normalization coefficients. Several alterations are made to the classical Benders subproblem. The $u_0\bar{z}$ term in the objective function is a result of having made the Benders separation problem a pure feasibility problem. The normalization constraint, Eq. 8.15c, makes it possible to select a violated cut minimal infeasible subsystem. The w vector "models the quality measure that one wants to apply for a clever cut selection."[82] The case where $w_0 = 1$ and $w_1 = \cdots = w_m = 0$ produces the original Benders CGLP. Instead of the classical scheme, Fischetti, Salvagnin and Zanette (2010) suggest $w_0 = \cdots = w_m = 1$ in order to heuristically find a minimum cardinality MIS. They further refine their statement by observing that the multipliers, w_i, of "static" conditions (i.e. those with null rows in the matrix B), which are always active, should not be penalized and therefore those w_i should be set to zero. Finally, they make one additional refinement and set $w_0 = 1$. With these suggestions, Constraint 8.15c becomes,

$$\sum_{i \in I(t)} u_i + u_0 = 1$$

where $I(t)$ is the index set of the non-zero rows of B. The generated cut for a ray $(\bar{u}_0, \bar{u})$ from the MIS Benders separation problem (8.15) is,

$$\bar{u}^T(b - By) - \bar{u}_0 z \le 0 \qquad (8.16)$$

which reduces to the classical Benders feasibility cut when $\bar{u}_0 = 0$ and the classical Benders optimality cut when $\bar{u}_0 > 0$. This arrangement allows both optimality and feasibility cuts to be generated from the same framework. When compared with classical cuts and lightly improved classical cuts, Fischetti, Salvagnin and Zanette (2010)[82] find significant improvements with the use of MIS Benders cuts, particularly for problems with both feasibility and optimality cuts.

The MIS Benders cuts are attractive for the ESSND problem formulation because they, like Papadakos (2008)'s Pareto-optimal cuts, improve the quality of feasibility cuts. The improvements to optimality cuts in both of these approaches are irrelevant in the ESSND problem application, but approaches that improve the feasibility cuts are rare in the literature. Most cut improvement approaches focus on optimality cuts rather than feasibility cuts and are therefore irrelevant for the ESSND problem.

8.3.3 Valid Inequalities

Valid inequalities are part of the "solution generation" category according to the classification system by Rahmaniani (20017)[168]. In contrast to the single-tree BD modification and the improved cut generation modifications, which could be applied to a wide variety of Benders decomposition implementations, valid inequalities are tailored to the problem at hand. Their specific purpose is to improve the master problem relaxation so that fewer Benders cuts are required to find the optimal solution, and thereby reduce the total runtime. The goal of the most effective valid inequalities is to inform the master problem of qualities of the subproblem that have been "lost" to the master problem through the relaxation.

In the ESSND problem, the master problem, which is only concerned with aircraft movement variables, has lost all information about the package routing variables and the forcing constraint which links the two types of variables together. Consider the master problem relaxation prior to the addition of any Benders cuts. Since there is no forcing constraint informing the MP that packages need to be moved, the MP "optimal" solution will be to move zero vehicles since doing nothing is the cheapest solution. This means that *every* vehicle movement that needs to occur must somehow be captured in cuts added to the MP, and this will likely require many iterations. If, instead, the master problem relaxation can be strengthened prior to the addition of the first Benders cut, fewer cuts will likely need to be added.

Ideally, a valid inequality could be added to the MP to inform it of the package routing needs and forcing constraint. Fortunately, such a valid inequality has already been developed. Kim (1997)[129], develops valid inequalities for an approximate formulation of the SDNP and his version of the ESSNDP wherein the package flow variables are removed and the forcing constraint is replaced with valid inequalities. These valid inequalities are based on the *aggregate capacity demand inequalities* of Magnanti, Mirchandani and Vachani (1995)[150] and the cut-set inequalities described by Magnanti, Mirchandani and Vachani (1993,1995)[149, 150] and Magnanti and Mirchandani (1993)[148].

The valid inequalities added by Kim (1997)[129] are *cut-set* inequalities. That is, they are defined according to a *cut* in the graph $\{S, T\}$. As a minor aside, note that the term "cut" is used in two distinct ways in this section due to an unfortunate collision of terminology – a *cut* in the graph is used to make formulate a *cut* in the master problem. These are two separate meanings of the term "cut" that must, unfortunately, be used in close proximity. A *cut* in a graph is a partition of the vertices into two disjoint subsets. A *cut* added to a mixed integer program is a *cutting plane* that is used to strengthen a relaxation. Often this relaxation is the LP relaxation and the LP hull is strengthened to be closer to the convex hull. In Benders decomposition, Benders cuts strengthen the master problem relaxation to capture the requirements of the subproblem.

A *cut-set* is the set of edges in which one vertex is in one partition of the cut and the edge's other vertex is in the other partition of the cut. That is, for a cut $C = \{S, T\}$ in a graph $G = (V, E)$, the set $\{(u, v) \in E \mid u \in S, v \in T\}$ is a cut-set. In Figure 33, which depicts a cut $C = \{\{1\}, \{2, 3, 4\}\}$, the cut-set is shown as black edges, $\{(1, 2), (1, 3), (1, 4)\}$. Within the SNDP and ESSNDP, cut-set inequalities are

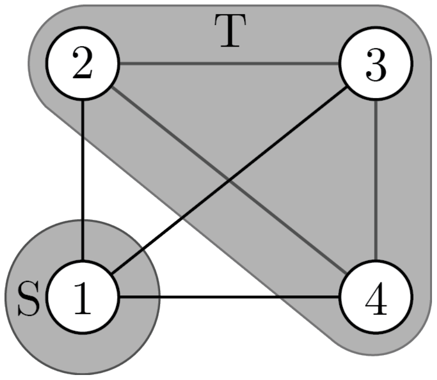


Figure 33: Notional Cut In a Graph of Four Nodes

generated based on the fact that across any cut in the graph, the total capacity assigned to the cut-set arcs must surpass the total demand traveling on those arcs. Mathematically, for the cut $C = \{S, T\}$, the valid inequality is,

$$\sum_{f \in F} u^f Y^f_{S,T} \geq D_{S,T} \tag{8.17}$$

where $Y^f_{S,T}$ equals $\sum_{(i,j) \in \{S,T\}} y^f_{ij}$, the total number of vehicles of type f assigned to arcs of the cut-set, u^f is the capacity of vehicle type f, and $D_{S,T}$ is the demand across the cut-set. Kim (1997)[129] defines $D_{S,T}$ as "the demand of all commodities with $O(k) \in S$ and $D(k) \in T$ or $O(k) \in T$ and $D(k) \in S$" where Ok and $D(k)$ represent the origin and desination airports of the commodity k, respectively. This definition is satisfactory for the purely overnight problem that Kim is solving, but for this application of the express shipment problem, a little more care is necessary due to the flexibility of two-day packages. Since two-day packages can be moved during either the night or the day, their demand should be included in the $D_{S,T}$ term only when they *must* travel on the arcs of that cut-set $\{S, T\}$ to avoid *over*-assigning capacity.

In this work these cut-set inequalities from Kim (1997)[129] were modified slightly, and valid inequalities defined over phase-based cut-sets were added to the Benders master problem. Additionally, to avoid an explosion in the number of inequalities added to the MP, only a limited number of cuts were made on the graph. One of the disadvantages of cut-set inequalities is that there are far too many possible cuts in a real-world graph to possibly enumerate them all. For any graph with N vertices, there are $2^N - 2$ possible cuts. Without even considering the time aspects of the express shipment problem, this means that $\sim 10^{34}$ cuts could be made on the graph of the baseline problem instance (Table 6) which includes 115 airports. This is clearly not a viable option.

Therefore, only a small fraction of these cuts were generated. Specifically, only the cut defined as one airport to all others was considered. That is, $C = \{\{X\}, \{A \backslash \{X\}\}\}$ where X is a single airport and A is the set of all airports. Additionally, cuts were generated based on the phases of the express shipment day.[3] As a result, inequalities were generated such that, for a pick up phase, the total vehicle capacity leaving a spoke must be greater than the minimum volume leaving that spoke, and similarly for delivery phases, the total vehicle capacity entering a spoke must be greater than the minimum volume terminating at that spoke. The structure of the inequality is the same as the aggregate demand inequality in Kim (1997)[129], Eq. 8.17, but the definition of $D_{S,T}$ is modified slightly to reflect this phase-based approach and the handling of two-day volume.

8.4 Results

In response to the findings of the review of enhancements to Benders decomposition in the literature, I hypothesize, in response to research question 6,

[3]The four phases of the day in the express shipment problem are: night pickup, night delivery, day pick up, and day delivery.

Hypothesis 6. *Based on this literature review of previously-developed methods for improving Benders decomposition, I hypothesize that implementing a single-tree Benders decomposition algorithm (i.e. improving the solution procedure), including the cut-set valid inequalities developed by Kim (1997)[129] (i.e. improving the solution generation), and implementing an approximate version of the Pareto-optimal Benders cuts from Papadakos (2008)[163] or the minimal infeasible subsystem (MIS) Benders cuts from Fischetti et al. (2010)[82] (i.e. improving the cut generation) will improve the performance of the Benders decomposition algorithm in solving the express shipment problem.*

Benders decomposition and the four enhancements discussed in Section 8.3 to it were implemented and tested on two problem instances, the small ten-airport problem instance used as a sensitivity study in Section 6.5, and the full-size problem instance outlined in Table 6. These experiments were performed on the experimental testbed described in Section 5.4 using the computational testbed described in Table 5. The time-space graph and package routing options were generated exactly as they were in the monolithic approach (refer to §6.1, §6.2 and §6.3 for details). This ensures that the results are solely the product of the solution method.

There was one important difference in the computational testbed. For the Benders decomposition experiments, CPLEX 12.7.1 from IBM was used rather than Gurobi. CPLEX, like Gurobi, is a state-of-the-art commercial mathematical programming solver. It was used for the Benders decomposition experiments instead of Gurobi because CPLEX recently included a built-in Benders decomposition.[118]

This CPLEX-developed Benders decomposition has the advantage of being implemented and rigorously tested by highly-skilled researchers and developers in the core inner workings of the solver. The disadvantage of this Benders implementation is that, like CPLEX's underlying branch-and-cut algorithm, the implementation is completely opaque. CPLEX does not indicate what kind of improvements to the classical Benders decomposition algorithm it may be using although it is clear that some improvements have been made. From the information that can be gleaned from the outputs, the CPLEX Benders implementation takes a multi-phase in which the algorithm is stopped, re-presolved and restarted. Additionally, it can be safely assumed that it is a single-tree implementation. Finally, given the popularity of using improved Benders cuts and the relatively few cuts it needs, it would be shocking if some or several improved Benders cuts were not used. The exact variety of those cuts, however, is unknown. Despite these disadvantages stemming from the opacity of the implementation, it still provides a valuable baseline for the Benders decomposition implementations.

The initial results from the CPLEX version of Benders decomposition were not as good as hoped - for realistic problem instances, only a few, poor solutions were found. Therefore, Benders decomposition was also implemented "from scratch" using CPLEX as the MIP and LP solver, and the improvements chosen from the literature. Including the CPLEX Benders implementation, there are four Benders varieties tested, 1) basic with classical feasibility cuts, 2) approximate Pareto-optimal cuts generated with the Papadakos (2008)[163] approach, 3) MIS cuts from the work of Fischetti, Salvagnin and Zanette (2010)[82], and 4) the CPLEX Benders implementation. All of the varieties are implemented using a single-tree Benders approach. Finally, the effect of the cut-set valid inequalities is also tested on these four Benders varieties. Of the two problem instances, the results of the ten-airport instance will be discussed first, followed by the full-sized problem instance.

8.4.1 Results of the Ten-Airport Problem Instance

The ten-airport problem instance is composed of the same ten airports as the airport subset sensitivity test described in Chapter 6, §6.5. Those airports are AUS, CLE, CVG, MEM, MSP, RST, SAT, SMF, STL, and SWF. They were randomly chosen and feature a mix of sizes and geographical diversity. The eight cases were run using this problem instance and the results are summarized in Table 45. Additionally, a monolithic case was run on this problem instance with continuous routing variables so as to have a commensurable baseline case.

Table 45: Benders Decomposition Results for Ten-Airport Problem Instance

Implementation	With VI	Total Run Time	Benders Cuts	Subproblem Time
Basic Benders	False	1784.62 sec	129	8.21 sec
Basic Benders	True	4339.57 sec	95	5.98 sec
Papadakos Benders	False	2262.00 sec	93	5.92 sec
Papadakos Benders	True	3213.40 sec	133	8.43 sec
MIS Benders	False	238.33 sec	82	5.33 sec
MIS Benders	True	443.29 sec	76	4.93 sec
CPLEX Benders	False	2.58 sec	37	-
CPLEX Benders	True	3.35 sec	48	-
Monolithic	False	5.86 sec	-	-
Monolithic	True	7.54 sec	-	-

All of the cases solved to optimality and produced the same solution value. There are a number of intriguing results in Table 45. First, the “basic” Benders decomposition implementation takes nearly 30 minutes to solve a small problem instance that the monolithic approach solves in under six seconds. To say there is a significant difference in runtime is an understatement. Aside from the CPLEX Benders cases, all of the Benders decomposition implementations took considerably longer to solve the problem than the monolithic approach, however, the minimal infeasible subsystem (MIS) Benders implementation was substantially faster than the basic and approximate Pareto-optimal (Papadakos) implementations. Ultimately, it is not altogether surprising that the monolithic approach outperforms the Benders decomposition in such a small problem instance. Benders decomposition has significantly more overhead and must iteratively refine the master problem before coming to a solution. The monolithic approach enjoys more powerful preprocessing algorithms which can have a dramatic effect on the runtime, and as shown in Chapter 6, the monolithic approach excels at small problems. What is somewhat surprising is the degree to which Benders decomposition lags behind the monolithic approach.

Additionally, it is surprising that the CPLEX Benders decomposition actually outperforms the monolithic approach for this problem instance. Unfortunately, the opaque nature of the CPLEX Benders decomposition implementation makes it nearly impossible to tease out the exact reason for this excellent performance, but the outputs do indicate that a multi-phase Benders decomposition is being implemented and the algorithm is stopped and restarted *twice* with additional preprocessing reductions being performed at each restart. These preprocessing steps could be responsible for significant speed up. Additionally, the CPLEX Benders implementation uses many fewer Benders cuts which indicates that they are probably stronger cuts and less time must be spent in the subproblem.

The other surprising result is that the valid inequalities which were supposed to improve the MP relaxation and thereby reduce the runtime actually made the runtime *worse* in every case, including the monolithic implementation. This result is somewhat confounding. The solution of every case in Table 45 has the exact same objective value regardless of whether valid inequalities were added or not so the valid inequalities are not leading to the wrong value. The other interesting aspect is that for the basic and MIS BD implementations, including the valid inequalities *did* reduce the number of cuts and the time spent in the subproblem, but still the total runtime was greater than the cases in which valid inequalities were not added. Since the valid inequalities are not cutting off the optimal solution and are therefore not wrong, the most probable explanation is that they are too weak to be meaningfully useful and instead their largest impact is to interfere with whatever preprocessing reductions and/or solution-finding heuristics are performed within the solver.

Finally, of the three types of Benders cuts implemented (classical, approximate

Papadakos and MIS), the MIS-generated cuts performed the best by far. This underscores the weakness of the classical Benders cuts for degenerate subproblems such as that of ESSNDP. Additionally, the poor performance of the approximate Pareto-optimal (as formulated by Papadakos) cuts do not perform better than the classical cuts. In fact, for the case without valid inequalities, the implementation of the Papadakos Pareto-optimal cuts using an approximate Magnanti-Wong point performs *worse* than the implementation using classical cuts. This strongly suggests that the approximate Magnanti-Wong point was poor. On a larger view, it also suggests that using an approximate Magnanti-Wong point is not without its risks and that the quality of the Magnanti-Wong point does have bearing on the quality of the generated cuts. As a whole, randomly choosing the extreme ray produced better cuts than producing approximate Pareto-optimal cuts with a bad Magnanti-Wong point.

8.4.2 Full-Problem Results

One major drawback of Benders decomposition, particularly for problems like the ESSNDP, is that every MP solution is infeasible until the optimal solution is found. This is not necessarily true for problems that add both optimality and feasibility cuts, since BD may find solutions that are feasible but sub-optimal with respect to the subproblem objective. For the ESSND problem, however, in which the sole role of the subproblem is to ensure that the master problem solution is feasible, if the algorithm is terminated before that feasible solution is found, the "solution" will be infeasible with respect to the original problem. When Benders decomposition is modified to the single-tree implementation, feasible suboptimal solutions may also be found, even when solving a problem that only adds feasibility cuts, because Benders cuts are generated whenever a new integer feasible solution is found, regardless of whether or not that integer solution is optimal for the MP.

Unfortunately, Benders decomposition, as it has been implemented in this work and even as it has been implemented in the version by CPLEX is ineffective at solving the full-size ESSND baseline problem instance. The results each Benders decomposition implementation are shown in Table 46. Since the valid inequalities produced worse results in the ten-airport case, they were omitted in these experiments. Of

Table 46: Benders Decomposition Results for Full-Size Baseline Problem Instance

Implementation	Total Run Time	Failure Cause	Optimality Gap
Basic Benders	45.48 hrs	RAM Limit	-
Papadakos Benders	21.53 hrs	RAM Limit	-
MIS Benders	0 hrs	Prob. Size	-
CPLEX Benders	6.23 hrs	RAM Limit	98.32%
Monolithic	48 hrs	Time Limit	13.3%

the Benders decomposition approaches, only the CPLEX implementation found any feasible solutions, but those solutions were so poor that they were completely useless. The other three Benders implementations were even worse. Not a single one found a feasible solution. The MIS Benders implementation curiously failed immediately due to an issue that seemed to stem from the problem size. When the total demand in the network surpassed a certain threshold, the approach failed immediately. The exact cause of this failure is unknown. The other three Benders implementations all failed as a result of surpassing the 240 GB of RAM available on the computational testbed. Of the three, the CPLEX Benders implementation was the most memory-hungry and it consumed its memory resources hours before the other two.

It is difficult to comment meaningfully on these results which fail to produce solutions despite generous computational resources and time, but what can be definitively determined is that for the ESSND problem, Benders decomposition is inferior to the monolithic approach. Ultimately, these Benders implementations were doomed by several factors. First, the baseline ESSND problem instance is very large. There are many commodities and aircraft that must be assigned in order to produce a feasible solution. Second, the master problem relaxation is extremely weak and when this was partnered with the large problem size, the algorithm could not cope. Third, one of the major disadvantages of Benders decomposition, particularly for problems such as the ESSND problem in which only feasibility cuts are added, there is no solution until the optimal solution. This is not strictly true for the single-tree Benders implementation, but the chances of finding a feasible solution to the ESSND through luck are slim. Finally, because it depends so heavily on the B&C algorithm to solve the master problem, Benders decomposition is still susceptible to the same pruning issues that hampered the monolithic approach. The convenience of the monolithic approach though is that the solutions are always feasible so the algorithm can be terminated early without resulting in a complete loss of effort.

8.4.3 Reflections on the Benders Decomposition Results

Based on the results of the Benders decomposition experiments, for the purposes of this work, hypothesis 5 is rejected. Four modifications to the classical Benders decomposition were implemented based on the findings in the literature in addition to the implementation recently included in the CPLEX optimization software. Although all of these implementations were capable of successfully finding the optimal solution for the small, ten-airport, problem instance, thus indicating that they were implemented correctly, even this small case took much longer to solve using Benders decomposition (except for the CPLEX implementation) compared to the monolithic approach. None of the Benders decomposition implementations were capable of producing solutions for the full-sized baseline problem instance.

Realistically, the implementation of Benders decomposition was an attempt to get the best of both worlds – an approach that produced the optimal solution *and* produced it quickly. Unfortunately, Benders decomposition struggled in the application to the express shipment problem, even with improvements from the literature. However, this does not mean that the research conducted in this chapter was conducted in vain. Several lessons can be learned from this experience, particularly from the results of the smaller ten-airport problem instance.

The improvements from the literature *did* have an effect on the performance of the algorithm. Of the custom-developed Benders decomposition implementations, the BD algorithm which used the MIS Benders cuts without the cut-set valid inequalities performed the best. This tells us several things about the enhancement strategies for Benders decomposition. First, "enhancements" do not always improve the performance of Benders decomposition. The valid inequalities based on those developed by Kim (1997)[129] actually hampered the progress of the algorithm, and the BD algorithm using the approximate Pareto-optimal cuts was *worse* than the classical Benders implementation in the case where there were no valid inequalities included in the master problem. In both of these cases, the implementation was weaker than the ideal. Due to the presence of two-day volume, the cut-set inequalities could not be implemented to the same strength as they were in Kim's work, who only solved the overnight problem instance.

In the case of the approximate Pareto-optimal cuts, finding a core point was very difficult so an *approximate* core point was used. This turned out to hinder the algorithm, leading the algorithm to choose *worse* cuts than the classical Benders implementation. Approximating the core point is not an uncommon practice in the literature considering how challenging it can be to generate a true core point, but based on the results of this work, care should be exercised when using an approximate core point because it could end up leading to worse performance.

Second, one of the primary drawbacks of Benders decomposition is that there are no intermediate solutions that are generated as the algorithm progresses, since every potential solution is infeasible until the optimal solution is found. This presents a serious issue since this means that the algorithm cannot be terminated early. For very large applied problems such as the express shipment problem, the expectation of finding true optimal solutions is, in reality, rather low. Applied problems are usually so large and/or complex that it is typically very difficult to obtain exact optimal solutions in a timely manner. As a result, perhaps a better approach would be to partner Benders decomposition with a heuristic that could perturb the infeasible solutions in order to find feasible intermediate solutions as the algorithm progresses.

Third, for the express shipment problem, improving the cut-generation scheme had the most positive impact on the Benders decomposition algorithm of the approaches implemented. The linear subproblem is extremely degenerate since there can be many feasible ways to assign packages to aircraft. As result of this high-degeneracy it is important that good Benders cuts be chosen. Unfortunately, there are very few improved *feasibility cuts* in the literature. The two most notable improved feasibility cuts were implemented in this work, and since the Benders subproblem is a feasibility problem, the improved optimality cuts, which are more common in the literature, are meaningless. In general, more research into improved Benders feasibility cuts could greatly improve the performance of a Benders decomposition algorithm for problems like the ESSND.

Finally, in the literature, it is typical for Benders decomposition implementations to be heavily modified from the original, "classical" algorithm. This is especially true for applied problems like scheduling for passenger airlines. These implementations are non-trivial and the development of one for the express shipment problem out of the scope of this work. However, from the results of the experiments in this work, perhaps this is what is necessary for applied problems to be solved exactly. For the purposes of this work, the original and moderately-modified Benders decomposition has been suitably investigated and found to be inferior to the monolithic baseline and heuristic decomposition approaches for the express shipment problem.

8.5 Summary

This chapter focused on the theoretical background, common enhancements and the application to the express shipment problem of Benders decomposition. The primary aim of this chapter was to investigate the performance of an exact decomposition approach so that it could be ultimately compared to the monolithic approach presented in Chapter 6 and the heuristic approach that was presented in Chapter 7. This investigation and comparison was focused on testing hypothesis 0, the fundamental hypothesis, that either a heuristic algorithm or exact decomposition algorithm would be capable of finding high-quality solutions to the express shipment problem more quickly than a monolithic approach. The Benders decomposition was found to be inferior to the monolithic approach for the full-sized baseline problem instance, but several lessons were learned in the process. The next and final chapter of this work, Ch. 9 will discuss the final conclusions and make suggestions for future work.

CHAPTER IX

CONCLUSIONS

9.1 Summary and Discussion of Results

The primary research objective of this dissertation was to develop a quick and effective algorithm that generates low-cost vehicle schedules to completely route one day's worth of domestic express shipment packages. To this end, the express shipment problem was mathematically captured as a mixed integer program called the *Express Shipment Service Network Design* problem (ESSND), and three classes of solution approaches were studied, implemented and compared. The three solution method classes were a monolithic approach, for which the Gurobi commercial optimization software was used, an exact decomposition approach, for which Benders decomposition was chosen, and a heuristic approach, which was designed in this work. Of these three approaches, the monolithic approach, which had been developed through a collaboration between Georgia Tech and FedEx, served as the baseline, and it was hypothesized that either a heuristic approach or an exact decomposition approach would be capable of producing solutions to the express shipment problem of comparable quality to the monolithic approach in less time. This was formally expressed as the fundamental hypothesis,

Hypothesis 0. *Based on the evidence in the literature that heuristic and exact decomposition approaches can out-perform monolithic approaches, I hypothesize that a heuristic approach or exact decomposition approach will able to produce solutions for the Express Shipment Service Network Design problem of comparable quality to the baseline monolithic approach in significantly less time.*

In order to test this fundamental hypothesis, all three classes of methods were implemented. The monolithic approach, which was developed as part of a collaboration between Georgia Tech and FedEx, served as the baseline approach and was presented first. In order to make the baseline single-day domestic problem tractable, a set of heuristics were used to generate the set of package routings, omitting the options that were unlikely to be present in an optimal solution. Additionally, the methods for generating the time-space graph and vehicle movement options were presented as well. This resulted in a problem that was solvable using a monolithic approach. These same methods for generating the package routing options, time-space graph, and vehicle movement options were used again in the Benders decomposition and heuristic algorithm implementations in order to ensure that the option space was the same for all three studied solution methods. The optimization itself for the monolithic approach was handled through the use of a commercial branch-and-cut-based optimization software (Gurobi 7.5).

The monolithic approach was found to produce high-quality solutions for the baseline single-day problem instance but the optimal solution remained elusive after two-days of runtime. Additionally, after an initial sharp improvement in solution quality, progress dramatically slowed, with only an improvement of 1.04 percentage points on the optimality gap for the entire second day of runtime. This slow progress is primarily due to the weakness of the LP relaxation of the ESSND problem and the size of the problem instance being solved. Sensitivity studies showed that all but the smallest problem instances failed to converge after running for forty-eight hours, and problem instances encompassing more than a single day of volume were hopelessly intractable.

Next, in Chapter 7, a heuristic approach was developed and implemented. Since sensitivity studies of the monolithic approach showed that smaller problem instances and relaxed formulations could be more easily solved than the full-sized, un-relaxed problem instance, it was hypothesized that a heuristic *decomposition* approach would be successful. The ESSND problem was decomposed functionally into a Package Routing Problem (PRP) and an Aircraft Movement Problem (AMP). Physical decompositions that partitioned the time-space graph were considered but deemed less appealing due to the type of coupling that would have to be reconciled between the subproblems, among other drawbacks.

However, there was still coupling between the Package Routing Problem and the Aircraft Movement Problem. Initially, it was hypothesized that a feedback-based approach was necessary in order to produce a consistent, feasible high-quality solution. Therefore, a fixed point iteration-based approach was implemented, with the subproblems solved using a monolithic approach. The results were unsatisfactory. It took over twenty minutes to find a solution with a 30% optimality gap. The iterative optimization approach did not drive itself towards a good solution as hoped, and attempting to provide master-level guidance to this FPI approach was unappealing because it would require many calls to a slow analysis method.

In response to the results of the feedback-based approach, a sequential algorithm was implemented. The FPI-based approach produced poor results because the package routings were assigned without regard to the need to fill whole aircraft. Therefore, the Package Routing Problem, the first of the two disciplines in the sequential algorithm, was modified to include aircraft movement variables, the constraint requiring those variable to be positive integers, and the feeder limit constraint. This new problem was called the Package Routing with Aircraft Problem (PRAP).

The addition of these variables and constraints dramatically increased the run time of the PRAP compared to the PRP. Therefore, the sequential algorithm terminated the PRAP early but solved the unmodified Aircraft Movement Problem after it to optimality. This sequential algorithm called the *Basic Sequential Algorithm* was found to be effective in solving the express shipment problem. It generated high-quality solutions more quickly than the monolithic approach for runtimes less than one hour and performed on par with the monolithic approach after that point. The success of the sequential approach to quickly generate high-quality solutions supported the fundamental hypothesis (Hy. 0) and showed that indeed a heuristic approach could produce high-quality solutions more quickly than the monolithic (or exact decomposition) approach.

This original sequential algorithm was only a basic implementation so although the fundamental hypothesis had been supported, further improvements to the algorithm were explored. This ultimately resulted in two improved sequential algorithms, 1) the *Collapsed Product Set Sequential Algorithm*, and 2) the *Improved Sequential Algorithm*. The Collapsed Product Set Sequential Algorithm sought to exploit the idea of "analogous commodities" in the problem to reduce the size of the Package Routing with Aircraft Problem. Rather than assign package routings to the full set of commodities in the problem as in the Basic Sequential Algorithm, the Collapsed Product Set Sequential Algorithm used the overnight priority box commodities, which constitute the most volume of all the products, as templates to assign the routings of all of their analogous commodities. Commodities that shared the same origin-destination pair were considered analogous. This allowed a considerably smaller PRAP to be solved, and resulted in an algorithm that was *much* less sensitive to problem size than any other algorithm studied in this work. Compared to the Basic Sequential Algorithm, it does not perform as well on the single-day baseline problem instance, though it does provide better solutions than the monolithic approach for the first hour of runtime. The real advantage of the Collapsed Product Set Sequential Algorithm is in its ability to solve problem instances encompassing much a much larger temporal scope. These problems are otherwise completely intractable, but now, with the Collapsed Product Set Sequential Algorithm, they are accessible.

The Basic Sequential Algorithm was also enhanced by improving the approximation of the PRAP to the original ESSND problem. In the original PRAP, only the vehicle variable integrality constraint and feeder limit constraint were included. The aim of including these aircraft-related considerations in the PRAP was to encourage the assignment of package routings that would lend themselves well to efficiently packaging aircraft. Although this *was* happening in the original PRAP, solutions only utilized the more cost-efficient newer aircraft, which resulted in massive over-use of these aircraft. As a consequence, in the AMP solution, which respected hull count and balance constraints, most packages were moved on different vehicles than the PRAP had used in its planning.

In order to correct this behavior, it was hypothesized that re-introducing a hull count constraint would improve the solution quality. The original hull count constraint could not be added to the PRAP because it was ineffective without the balance constraint. Therefore, the phase-based hull count constraint, which enforces the hull count for each *phase* was developed and added to the PRAP. This addition had a substantial impact on the solution quality. This version of the sequential algorithm, called the *Improved Sequential Algorithm*, was capable of finding a better solution to the baseline problem instance in an *hour* than the monolithic approach found in *two days*. The sort-centric balance constraint was also developed and tested but found to inhibit rather than help the PRAP find exceptionally high-quality solutions.

The most significant contributions of this work is the Improved Sequential Algorithm. It's ability to reduce the time to find high-quality solutions by *days* is massive improvement over the monolithic baseline. Additionally, the Template-Based Sequential algorithm opens the door to solving problems of a much larger temporal scope than before. Using this algorithm, good solutions to the full-week problem instance, which is much more useful for express shipment companies than the single day problem instance, can be found in relatively little time.

Finally, Benders decomposition was chosen and implemented as the exact decomposition approach due to the "complicating variables" found in the ESSND problem formulation. Benders decomposition is a "row generation" technique that attempts to exploit the fact that only a subset of constraints is typically active in an optimal solution. If these constraints can be efficiently identified and added to the problem, a smaller optimization problem can be solved. In order to identify these constraints, the original formulation is decomposed into a master problem and a subproblem. The master problem contains all of the complicating variables and the subproblem is used to generate constraints to add to the master problem.

In this dissertation, several modifications to the classical Benders decomposition algorithm were made from the literature based on known shortcomings of Benders decomposition. First, the single-tree Benders decomposition was implemented. In the single-tree Benders decomposition, a single branch-and-bound tree is made and Benders cuts are added to the master problem any time a feasible integer solution is found. Additionally, valid inequalities based on the cut-set inequalities presented by Kim (1997)[130] were added to the master problem too in order to better shape the otherwise poor master problem relaxation. Finally, two improved Benders cuts were implemented – the Pareto-optimal cuts as modified by Papadakos (2008)[163] to produce feasibility cuts, and minimal infeasible subsystem (MIS) cuts developed by Fischetti et al. (2010)[82].

These improvements to Benders decomposition and a commercial implementation of Benders decomposition by CPLEX 12.7 (a commercially-developed optimization software) were tested on a small ten-airport subproblem and the full-sized single-day subproblem. For the small ten-airport case, although optimal solutions were found by all of the Benders decomposition implementations, all except the CPLEX implementation took much longer than the monolithic baseline case. For the full-sized baseline problem instance, only the CPLEX implementation was able to find any solutions, but those solutions were extremely poor. Ultimately, Benders decomposition as it was implemented in this work was found to be ineffective.

Two primary factors were responsible for this poor performance. First, the Benders master problem relaxation is extremely poor and as a result, every aircraft movement must be expressed through a Benders cut. This requires a large number of cuts and iterations, leading to a long run time. Second, since only feasibility Benders cuts are added to the master problem in this application of Benders decomposition, all master problem solutions will be infeasible until the optimal solution is found. Technically, it would be possible in the single-tree Benders implementation for a feasible solution to be found and retained, but ultimately this is unlikely due to the poor master problem relaxation and the size of the problem.

Although the Benders decomposition implementation was not superior to the monolithic approach, these results are still a contribution. In particular, it is noteworthy that the minimal infeasible subsystem (MIS) Benders cuts provided the most significant improvement to Benders decomposition compared to the "basic" Benders implementation. Additionally, the cut-set valid inequalities and approximate Pareto-optimal Benders cuts actually hampered the performance of the algorithm rather than enhanced it, showing that "improvements" that cannot be fully or strongly implemented may end up being detrimental. Benders decomposition is currently enjoying a relatively large amount of popularity in the literature but is important to recognize that even popular approaches are not necessarily suitable for all problems. It is also important to recognize that most of these recent implementations of Benders decomposition modify the original algorithm, sometimes heavily. It is possible that with the recent buzz around Benders decomposition that a technique will be developed that makes it more capable of solving the express shipment problem. Additionally, it is possible that a more heavily-modified version of Benders decomposition, such as a hybrid column generation/row generation approaches would be more successful.

For the baseline single-day problem instance that was the focus of this work, the Improved Sequential Algorithm is, by far, the best approach. It is successful because it first adequately approximates the full ESSND problem formulation to assign package routings, and then it schedules aircraft to optimally move the package along those pre-assigned routings. In the monolithic approach, both the package routings and the aircraft movements are changed at the same time, which makes it more difficult to find good solutions. The Improved Sequential algorithm dramatically outperforms the monolithic approach, which ended up the second-best overall approach. It produces, in one hour, a solution of better quality than the solution that took the monolithic approach *two days* to generate.

This significant enhancement in performance makes the Improved Sequential Algorithm much more useful than the monolithic approach. By shaving hours off the analysis runtime, it is useful for a variety of applications for express shipment companies. It could be used at the operational level for contingency planning the case of disruptions to the normal network such as inclement weather. It could also be used at the tactical level to produce daily good-quality schedules. These solutions could be used as-is or taken as a high-quality starting place for the human planners. Finally, it could be used at the strategic level to run studies for major business decisions such as investments in new aircraft, sorting facility capacity upgrades, and the placement of new potential sorting facilities. Given the scope of express shipment companies, potential improvements to their operations could easily range in the millions or tens of millions of dollars annually.

9.2 Avenues for Future Work

There are several avenues for future research based on the findings of this work. First, *gateway* volume could be incorporated into the problem. Gateway commodities are those with one international city in its origin-destination pair and one domestic city. That volume is more difficult to accommodate because only part of the package routing is domestic, customs inspections must be considered for incoming volume, and incoming and outgoing international flights do not always follow the strong periodic nature of the purely domestic flights. However, including gateway volume would make the solutions more realistic and more useful. It would tie the domestic problem into the larger global network and thus be able to provide more directly implementable solutions.

Second, the Collapsed Product Set Sequential Algorithm currently produces good-quality solutions, but further research could potentially improve its performance. In particular, the consolidation assignment problem (CAP) as it is implemented in this work does not offer significant improvements considering its runtime. This is particularly problematic for problem instances with a larger temporal scope. For instance, the Collapsed Product Set Sequential Algorithm produces a solution to the full-week problem instance without consolidation in just over three and a half hours but requires an additional twelve hours to solve the CAP, which only reduced the optimality gap by 1.1 percentage points. For the single-day problem instance, the CAP only took slightly longer to run than the Aircraft Movement Problem, and this was acceptable, but for longer problem instances, a more scalable approach would be better. Perhaps a heuristic solution would be more successful than solving a subproblem of the ESSND with a B&C solver as it was implemented in this work. Additionally, it would be valuable to explore ways of improving the solution quality for the larger scope runs, perhaps through a local improvement heuristic.

Finally, as mentioned in the previous section, although the implementations of Benders decomposition in this work did not produce satisfactory results, it would be interesting to investigate whether a more heavily-modified version of Benders decomposition would be successful. In particular, the work of Cordeau, Mercier and Soumis (2001,2005,2007)[53, 156, 157] as well as Papadakos (2009)[164] for passenger airlines suggests that a hybrid column generation/row generation approach is a powerful approach for difficult to solve problems. However, implementing such an approach is not a trivial undertaking and would likely require quite a bit of sophistication and care.

In the end, the primary objective of this work, to develop an algorithm that quickly and effectively routed one day's worth of domestic express shipment volume and assigned the best possible vehicles to move them, was satisfied. The Improved Sequential Algorithm is capable of producing in one hour the high-quality solutions that the baseline monolithic approach requires days to find. This relatively simple heuristic is straight-forward to implement and capitalizes on the natural decomposition of the problem between the package-related decisions and the aircraft-related decisions. Together, these two considerations make the express shipment problem very difficult to solve, but by solving the problems semi-independently, high-quality solutions can be found much more quickly. Additionally, the full-week problem, a previously intractable problem that is more valuable than the single-day schedule for express shipment companies, can be solved with the Improved Sequential and Collapsed Product Set Sequential Algorithms. Together, these two heuristic algorithms greatly reduce the time it takes to produce high-quality solutions to the express shipment problem, and in doing so opens the door to more sophisticated planning. The scale of these companies means that even small improvements as a result of studies enabled by these heuristic algorithms could range in the millions of dollars.

REFERENCES

[1] Abara, J., "Applying integer linear programming to the fleet assignment problem," *Interfaces*, vol. 19, no. 4, pp. 20–28, 1989.

[2] Agarwal, R. and Ergun, Ö., "Ship scheduling and network design for cargo routing in liner shipping," *Transportation Science*, vol. 42, no. 2, pp. 175–196, 2008.

[3] Ahuja, R. K., Goodstein, J., Mukherjee, A., Orlin, J. B., and Sharma, D., "A very large-scale neighborhood search algorithm for the combined through-fleet-assignment model," *INFORMS Journal on Computing*, vol. 19, no. 3, pp. 416–428, 2007.

[4] Ahuja, R. K., Magnanti, T. L., and Orlin, J. B., *Network Flows: Theory, Algorithms and Applications.* Prentice Hall, 1993.

[5] Ahuja, R. K., Magnanti, T. L., Orlin, J. B., and Reddy, M., "Chapter 1 applications of network optimization," in *Network Models* (Ball, M., Magnanti, T., Monma, C., and Nemhauser, G., eds.), vol. 7 of *Handbooks in Operations Research and Management Science*, pp. 1 – 83, Elsevier, 1995.

[6] Alexandrov, N., "Multidisciplinary design optimization," *Optimization and Engineering*, vol. 6, no. 1, pp. 5–7, 2005.

[7] Alexandrov, N. M., Hussaini, M. Y., and others, *Multidisciplinary design optimization: State of the art*, vol. 80. SIAM, 1997.

[8] Allison, J. T., "Complex system optimization: A reivew of analytical target cascading, collaborative optimization, and other formulations," Master's thesis, The University of Michigan, 2004.

[9] Alumur, S. and Kara, B. Y., "Network hub location problems: The state of the art," *European Journal of Operational Research*, vol. 190, pp. 1–21, 2008.

[10] Andersen, J., Crainic, T., and Christiansen, M., "Service network design with asset management: Formulations and comparative analyses," *Transporatation Research Part C: Emerging Technologies*, vol. 17, no. 2, pp. 197–207, 2009.

[11] Armacost, A. P., *Composite Variable Formulations for Express Shipment Service Network Design.* PhD thesis, Massachusetts Institute of Technology, September 2000.

[12] Armacost, A. P., Barnhart, C., and Ware, K. A., "Composite variable formulations for express shipment service network design," *Transportation Science*, vol. 36, pp. 1–20, February 2002.

[13] Armacost, A. P., Barnhart, C., Ware, K. A., and Wilson, A. M., "UPS optimizes its air network," *Interfaces*, vol. 34, pp. 15–25, January-February 2004.

[14] Balakrishnan, A., Magnanti, T. L., and Mirchandani, P., "A dual-based algorithm for multi-level network design," *Management Science*, vol. 40, p. 567, 05 1994. Copyright - Copyright Institute for Operations Research and the Management Sciences May 1994; Last updated - 2014-05-19; CODEN - MNSCDI.

[15] Balakrishnan, A., Magnanti, T. L., and Mirchandani, P., "Modeling and heuristic worst-case performance analysis of the two-level network design problem," *Management Science*, vol. 40, p. 846, July 1994.

[16] Balakrishnan, A., Magnanti, T. L., and Mirchandani, P., "Heuristics, LPs, and trees on trees: Network design analyses," *Operations research*, vol. 44, p. 478, May 1996.

[17] Balas, E., Ceria, S., and Cornuéjols, G., "Mixed 0-1 programming by lift-and-project in a branch-and-cut framework," *Management Science*, vol. 42, no. 9, pp. 1229–1246, 1996.

[18] Balas, E., Ceria, S., Cornuéjols, G., and Natraj, N., "Gomory cuts revisited," *Operations Research Letters*, vol. 19, no. 1, pp. 1–9, 1996.

[19] Balesdent, M., Bérend, N., Dépincé, P., and Chriette, A., "A survey of multidisciplinary design optimization methods in launch vehicle design," *Structural and Multidisciplinary Optimization*, vol. 45, no. 5, pp. 619–642, 2012.

[20] Ball, M., Barnhart, C., Nemhauser, G., and Odoni, A., "Chapter 1 air transportation: Irregular operations and control," in *Transportation* (Barnhart, C. and Laporte, G., eds.), vol. 14 of *Handbooks in Operations Research and Management Science*, pp. 1 – 67, Elsevier, 2007.

[21] Barnhart, C., Hane, C. A., and Vance, P. H., "Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems," *Operations Research*, vol. 48, no. 2, pp. 318 – 326, 2000. Branch and bound;Branch and price and cut algorithm;Column generation model;Multicommodity;.

[22] Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., and Vance, P. H., "Branch-and-price: Column generation for solving huge integer programs," *Operations Research*, vol. 46, no. 3, pp. pp. 316–329, 1998.

[23] Barnhart, C., Krishnan, N., Kim, D., and Ware, K., "Network design for express shipment delivery," *Computational Optimization and Applications*, vol. 21, pp. 239–262, 2002.

[24] Barnhart, C. and Schneur, R. R., "Air network design for express shipment service," *Operations Research*, vol. 44, no. 6, pp. 852 – 863, 1996.

[25] Barnhart, C. and Shen, S., *Lecture Notes in Computer Science (LNCS)*, vol. 3616 LNCS, ch. Logistics service network design for time-critical delivery, pp. 86 – 105. Pittsburgh, PA, United states: Springer-Verlag Berlin Heidelberg, 2005.

[26] Bellmore, M. and Nemhauser, G. L., "The traveling salesman problem: a survey," *Operations Research*, vol. 16, no. 3, pp. 538–558, 1968.

[27] Benders, J., "Partitioning procedures for solving mixed-variables programming problems," *Numerische Mathematik*, vol. 4, pp. 238–252, 1962.

[28] Berge, M. E. and Hopperstad, C. A., "Demand driven dispatch: A method for dynamic aircraft capacity assignment, models and algorithms," *Operations Research*, vol. 41, no. 1, pp. 153–168, 1993.

[29] Bertsimas, D. and Tsitsiklis, J., *Introduction to Linear Optimization*. Athena Scientific Series in Optimization and Neural Computation, Athena Scientific, 1997.

[30] Bixby, R. E., "A brief history of linear and mixed-integer programming computation," *Documenta Mathematica*, pp. 107–121, 2012.

[31] Bland, R. G., Goldfarb, D., and Todd, M. J., "The ellipsoid method: A survey," *Operations Research*, vol. 29, no. 6, pp. 1039–1091, 1981.

[32] Blum, C. and Roli, A., "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Computing Surveys*, vol. 35, no. 3, pp. 268–308, 2003.

[33] Boland, N., Hewitt, M., Marshall, L., and Savelsbergh, M., "The continuous time service network design problem," *Operations Research*, vol. 65, no. 5, pp. 1301–1321, 2017.

[34] Bosch, R., "Opt art," *Math Horizons*, pp. 6–9, February 2006.

[35] Boussaïd, I., Lepagnot, J., and Siarry, P., "A survey on optimization metaheuristics," *Information Sciences*, vol. 237, pp. 82–117, 2013.

[36] Bowen Jr., J. T., "A spatial analysis of FedEx and UPS: hubs, spokes, and network structure," *Journal of Transport Geography*, vol. 24, pp. 419–431, 2012.

[37] Boyd, S., Xiao, L., Mutapcic, A., and Mattingley, J., "Notes on decomposition methods." Web Document, 2 June 2008. `https://stanford.edu/class/ee364b/lectures/decomposition_notes.pdf`.

[38] Büdenbender, K., Grünert, T., and Sebastian, H.-J., "A hybrid tabu search/branch-and-bound algorithm for the direct flight network design problem," *Transportation Science*, vol. 34, no. 4, pp. 364–380, 2000.

[39] Bunte, S. and Kliewer, N., "An overview on vehicle scheduling models," *Public Transport*, vol. 1, no. 4, pp. 299–317, 2009.

[40] Cantú-Paz, E., "A survey of parallel genetic algorithms," *Calculateurs paralleles, reseaux et systems repartis*, vol. 10, no. 2, pp. 141–171, 1998.

[41] Çetiner, S., "An iterative hub location and routing problem for postal delivery systems," Master's thesis, The Middle East Technical University, 2003.

[42] Çetiner, S., Sepil, C., and Süral, H., "Hubbing and routing in postal delivery systems," *Annals of Operations research*, vol. 181, no. 1, pp. 109–124, 2010.

[43] Chabrier, A., Danna, E., Pape, C. L., and Perron, L., "Solving a network design problem," *Annals of Operations Research*, vol. 130, pp. 217–239, 2004.

[44] Chan, Y. and Ponder, R. J., "The small package air freight industry in the United States: A review of the Federal Express experience," *Transportation Research Part A*, vol. 13A, pp. 221–229, 1979.

[45] Chvátal, V., *Linear Programming.* Macmillan, 1983.

[46] Clarke, G. and Wright, J. W., "Scheduling of vehicles from a central depot to a number of delivery points," *Operations Research*, vol. 12, no. 4, pp. 568–581, 1964.

[47] Clausen, J., "Branch and bound algorithms-principles and examples," *Department of Computer Science, University of Copenhagen*, pp. 1–30, 1999.

[48] Clerc, M., *Particle Swarm Optimization*, vol. 93. John Wiley & Sons, 2010.

[49] Cohn, A., Root, S., Wang, A., and Mohr, D., "Integration of the load-matching and routing problem with equipment balancing for small package carriers," *Transportation Science*, vol. 41, no. 2, pp. 238–252, 2007.

[50] Cohn, A. E. M., *Composite-Variable Modeling for Large-Scale Problems in Transportation and Logistics.* PhD thesis, Massachusetts Institute of Technology, 2002.

[51] Conejo, A. J., Castillo, E., Mínguez, R., and García-Bertrand, R., *Decomposition Techniques in Mathematical Programming: Engineering and Science Applications*. Springer Science & Business Media, 2006.

[52] Cordeau, J.-F., Pasin, F., and Solomon, M. M., "An integrated model for logistics network design," *Annals of Operations Research*, vol. 144, pp. 59–82, Apr 2006.

[53] Cordeau, J.-F., Stojković, G., Soumis, F., and Desrosiers, J., "Benders decomposition for simultaneous aircraft routing and crew scheduling," *Transportation Science*, vol. 35, no. 4, pp. 375–388, 2001.

[54] Cornuéjols, G., "Revival of the gomory cuts in the 1990s," *Annals of Operations Research*, vol. 149, no. 1, pp. 63–66, 2007.

[55] Cornuéjols, G., "The ongoing story of Gomory cuts," *Documenta Mathematica*, pp. 221–226, 2012.

[56] Costa, A. M., "A survey on Benders decomposition applied to fixed-charge network design problems," *Computers & Operations Research*, vol. 32, pp. 1429–1450, 2005.

[57] Costa, A. M., Cordeau, J.-F., and Gendron, B., "Benders, metric and cutset inequalities for multicommodity capacitated network design," *Computational Optimization and Applications*, vol. 42, pp. 371–392, 2009.

[58] Crabtree, T., Hoang, T., Tom, R., and Gildermann, G., "World air cargo forecast 2016-2017," tech. rep., The Boeing Company, 2016.

[59] Crainic, T. G., "Service network design in freight transportation," *European Journal of Operational Research*, vol. 122, pp. 272–288, 2000.

[60] Crainic, T. G., Hewitt, M., Toulouse, M., and Vu, D. M., "Service network design with resource constraints," *Transportation Science*, 2014.

[61] Crainic, T. G. and Laporte, G., "Planning models for freight transportation," *European Journal of Operational Research*, vol. 97, pp. 409–438, 1997.

[62] Crainic, T., Hewitt, M., and Rei, W., "Partial decomposition strategies for two-stage stochastic integer programs," in *Publication CIRRELT-2014-13*, Montréal, QC, Canada: Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, 2014.

[63] Crainic, T., Hewitt, M., and Rei, W., "Partial decomposition strategies for two-stage stochastic integer programs," in *Publication CIRRELT-2016-37*, Montréal, QC, Canada: Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, 2016.

[64] Croes, G. A., "A method for solving traveling-salesman problems," *Operations Research*, vol. 6, no. 6, pp. 791–812, 1958.

[65] Dantzig, G., Fulkerson, R., and Johnson, S., "Solution of a large-scale traveling-salesman problem," *Operations Research*, vol. 2, no. 4, pp. 393–410, 1954.

[66] Dantzig, G. B., *Linear Programming and Extensions.* Princeton University Press, 1963.

[67] Dantzig, G. B., "Reminiscences about the origins of linear programming," *Operations Research Letters*, vol. 1, pp. 43–48, April 1982.

[68] Dantzig, G. B. and Wolfe, P., "The decompostion algorithm for linear programs," *Econometrica*, vol. 29, pp. 767–778, Oct. 1961.

[69] Dejax, P. J. and Crainic, T. G., "Survey paper - a review of empty flows and fleet management models in freight transportation," *Transportation Science*, vol. 21, no. 4, pp. 227–248, 1987.

[70] Derigs, U. and Friederichs, S., "Air cargo scheduling: integrated models and solution procedures," *OR Spectrum*, vol. 35, pp. 325–362, 2013.

[71] Derigs, U., Friederichs, S., and Schäfer, S., "A new approach for air cargo network planning," *Transportation Science*, vol. 43, no. 3, pp. 370–380, 2009.

[72] Desaulniers, G., Desrosiers, J., and Solomon, M. M., eds., *Column Generation.* Springer, 2005.

[73] Droździel, P., Wińska, M., Madleňák, R., and Szumski, P., "Optimization of the post logistics network and location of the local distribution center in selected area of the lublin province," *Procedia Engineering*, vol. 192, pp. 130–135, 2017.

[74] Erera, A., Hewitt, M., Karacik, B., and Savelsbergh, M., "Locating drivers in a trucking terminal network," *Transportation Research Part E: Logistics and Transportation Review*, vol. 45, no. 6, pp. 988–1005, 2009.

[75] Erera, A., Hewitt, M., Savelsbergh, M., and Zhang, Y., "Improved load plan design through integer programming based local search," *Transporation Science*, vol. 47, pp. 412–427, 2013.

[76] Erera, A. L., Hewitt, M., Savelsbergh, M. W., and Zhang, Y., "Creating schedules and computing operating costs for ltl load plans," *Computers & Operations Research*, vol. 40, no. 3, pp. 691–702, 2013.

[77] Ernst, A. T. and Krishnamoorthy, M., "Efficient algorithms for the uncapacitated single allocation p-hub median problem," *Location science*, vol. 4, no. 3, pp. 139–154, 1996.

[78] Ernst, A. T. and Krishnamoorthy, M., "Exact and heuristic algorithms for the uncapacitated multiple allocation p-hub median problem," *European Journal of Operational Research*, vol. 104, no. 1, pp. 100–112, 1998.

[79] FedEx Corporation, "Annual Report Pursuant to Section 13 or 15(d) of the Securities Exchange Act of 1934." Form 10-K, 31 May 2017.

[80] FedEx Corporation, "FedEx extends express air transportation contract with united states postal service." Press Release via Web Document, 23 February 2017.

[81] Feo, T. A. and Resende, M. G., "Greedy randomized adaptive search procedures," *Journal of Global Optimization*, vol. 6, no. 2, pp. 109–133, 1995.

[82] Fischetti, M., Salvagnin, D., and Zanette, A., "A note on the selection of Benders' cuts," *Mathematical Programming*, vol. 124, pp. 175–182, July 2010.

[83] Fisher, M. L., "The Lagrangian relaxation method for solving integer programming problems," *Management science*, vol. 27, no. 1, pp. 1–18, 1981.

[84] Fisher, M. L., "An applications oriented guide to Lagrangian relaxation," *Interfaces*, vol. 15, no. 2, pp. 10–21, 1985.

[85] Fleuren, H., Goossens, C., Hendriks, M., Lombard, M.-C., Meuffels, I., and Poppelaars, J., "Supply chain-wide optimization at TNT Express," *Interfaces*, vol. 43, no. 1, pp. 5–20, 2013.

[86] Ford Jr, L. R. and Fulkerson, D. R., "Constructing maximal dynamic flows from static flows," *Operations research*, vol. 6, no. 3, pp. 419–433, 1958.

[87] Ford Jr, L. R. and Fulkerson, D. R., *Flows in Networks.* Princeton University Press, 1962.

[88] Frangioni, A., "About Lagrangian methods in integer optimization," *Annals of Operations Research*, vol. 139, pp. 163 – 193, 2005.

[89] Fukuda, K., Liebling, T. M., and Margot, F., "Analysis of backtrack algorithms for listing all vertices and all faces of a convex polyhedron," *Computational Geometry*, vol. 8, no. 1, pp. 1–12, 1997.

[90] Garey, M. R. and Johnson, D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, vol. 29. New York: W.H. Freeman and Company, 2002.

[91] Gendreau, M. and Potvin, J.-Y., *Handbook of Metaheuristics*, vol. 2. Springer, 2010.

[92] Geoffrion, A. M., "Generalized Benders decomposition," *Journal of Optimization Theory and Applications*, vol. 10, pp. 237–260, Oct 1972.

[93] Geoffrion, A. and Graves, G., "Multicommodity distribution system design by Benders decomposition," *Management Science*, vol. 20, no. 5, pp. 822–844, 1974.

[94] Glover, F., "Tabu search–part I," *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.

[95] Glover, F., "Tabu search–part II," *ORSA Journal on Computing*, vol. 2, no. 1, pp. 4–32, 1990.

[96] Glover, F., "Tabu search: A tutorial," *Interfaces*, vol. 20, no. 4, pp. 74–94, 1990.

[97] Glover, F. and Laguna, M., "Tabu search," in *Handbook of Combinatorial Optimization*, pp. 3261–3362, Springer, 2013.

[98] Glover, F. W. and Kochenberger, G. A., *Handbook of metaheuristics*, vol. 57. Springer Science & Business Media, 2006.

[99] Gomory, R. E., "Outline of an algorithm for integer solutions to linear programs," *Bulletin of the American Mathematical Society*, vol. 64, no. 5, pp. 275–278, 1958.

[100] Gomory, R. E., "An algorithm for the mixed integer problem," tech. rep., The RAND Corporation, Santa Monica, CA, 1960.

[101] Gomory, R. E., "An algorithm for integer solutions to linear programs," in *Recent Advances in Mathematical Programming* (Graves, R. and Wolfe, P., eds.), pp. 260–302, New York: McGraw-Hill, 1963.

[102] Gomory, R. E., "Early integer programming," *Operations Research*, vol. 50, no. 1, pp. 78–81, 2002.

[103] Gorman, M. F., Clarke, J.-P., Gharehgozli, A. H., Hewitt, M., de Koster, R., and Roy, D., "State of the practice: A review of the application of OR/MS in freight transportation," *Interfaces*, vol. 44, pp. 535–554, 2014.

[104] Grunert, T., Sebastian, H.-J., and Tharigen, M., "The design of a letter-mail transportation network by intelligent techniques," in *Systems Sciences, 1999. HICSS-32. Proceedings of the 32nd Annual Hawaii International Conference on*, pp. 16–pp, IEEE, 1999.

[105] Grünert, T. and Sebastian, H.-J., "Planning models for long-haul operations of postal and express shipment companies," *European Journal of Operational Research*, vol. 122, pp. 289–309, 2000.

[106] Hall, R. W., "Heuristics for selecting facility locations," *Transportation Logistics Review*, vol. 21, pp. 353–373, 1985.

[107] Hall, R. W., "Configuration of an overnight package air network," *Transport Management*, vol. 23A, no. 2, pp. 139–149, 1989.

[108] Hane, C. A., Barnhart, C., Johnson, E. L., Marsten, R. E., Nemhauser, G. L., and Sigismondi, G., "The fleet assignment problem: solving a large-scale integer problem," *Mathematical Programming*, vol. 70, pp. 211–232, 1995.

[109] Hansen, P. and Mladenović, N., "Variable neighborhood search: Principles and applications," *European Journal of Operational Research*, vol. 130, no. 3, pp. 449–467, 2001.

[110] Hansen, P. and Mladenović, N., "Variable neighborhood search," in *Handbook of metaheuristics*, pp. 145–184, Springer, 2003.

[111] Heinitz, F. and Meincke, P., "Modeling framework of origin and destination air cargo routing," *Transportation Research Record*, vol. 2336, pp. 83–90, 2014.

[112] Hemmecke, R., Köppe, M., Lee, J., and Weismantel, R., "Nonlinear integer programming," in *50 Years of Integer Programming 1958-2008*, pp. 561–618, Springer, 2010.

[113] Holland, J. H., "Adaptation in natural and artificial systems. an introductory analysis with application to biology, control, and artificial intelligence," *Ann Arbor, MI: University of Michigan Press*, pp. 439–444, 1975.

[114] Holland, J. H., "Genetic algorithms," *Scientific American*, vol. 267, no. 1, pp. 66–73, 1992.

[115] Hooker, J., *Logic-based methods for optimization: combining optimization and constraint satisfaction.* John Wiley & Sons, 2011.

[116] Hooker, J. and Ottosson, G., "Logic-based benders decomposition," *Mathematical Programming, Series A*, vol. 96, pp. 33–60, 2003.

[117] Huang, H. C., Lee, C., and Xu, Z., "The workload balancing problem at air cargo terminals," *OR Spectrum*, vol. 28, pp. 705–727, 2006.

[118] IBM ILOG CPLEX, "Benders algorithm in CPLEX V12.7.0." Web Document, December 2016. `https://www.ibm.com/support/knowledgecenter/en/SSSA5P_12.7.0/ilog.odms.cplex.help/CPLEX/ReleaseNotes/topics/releasenotes127/newBenders.html`.

[119] Jeong, S.-J., Lee, C.-G., and Bookbinder, J. H., "The European freight railway system as a hub-and-spoke network," *Transportation Research Part A*, vol. 41, pp. 523–536, 2007.

[120] Ji, P. and Chen, K., "The vehicle routing problem: The case of the hong kong postal service," *Transportation Planning and Technology*, vol. 30, no. 2-3, pp. 167–182, 2007.

[121] Jones, K. L., Lustig, I. J., Farvolden, J. M., and Powell, W. B., "Multicommodity network flows: The impact of formulation on decomposition," *Mathematical Programming*, vol. 62, no. 1-3, pp. 95–117, 1993.

[122] Jünger, M., Liebling, T. M., Naddef, D., Nemhauser, G. L., Pulleyblank, W. R., Reinelt, G., Rinaldi, G., and Wolsey, L. A., eds., *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*. Springer Science & Business Media, 2009.

[123] Kalvelagen, E., "Benders decomposition with GAMS," tech. rep., GAMS Development Corp., Washington D.C., 2002.

[124] Kalyani, D., "Couriers & local delivery services in the US," IBISWorld Industry Report 49222, IBISWorld, September 2016.

[125] Karmarkar, N., "A new polynomial-time algorithm for linear programming," in *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, pp. 302–311, ACM, 1984.

[126] Karp, R. M., "Reducibility among combinatorial problems," in *Complexity of computer computations*, pp. 85–103, Springer, 1972.

[127] Kennedy, J., "Particle swarm optimization," in *Encyclopedia of Machine Learning*, pp. 760–766, Springer, 2011.

[128] Kennington, J. L., "A survey of linear cost multicommodity network flows," *Operations Research*, vol. 26, no. 2, pp. 209–236, 1978.

[129] Kim, D., *Large Scale Transportation Service Network Design: Models, Algorithms and Applications*. PhD thesis, Massachuesetts Institute of Technology, 1997.

[130] Kim, D. and Barnhart, C., "Multimodal express shipment service design: Models and algorithms," *Computers and Industrial Engineering*, vol. 33, no. 3-4, pp. 685 – 688, 1997.

[131] Kim, D. and Barnhart, C., *Transportation Service Network Design: Models and Algorithms*, pp. 259–283. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999.

[132] Kim, D., Barnhart, C., Ware, K., and Reinhardt, G., "Multimodal express package delivery: A service network design application," *Transportation Science*, vol. 33, no. 4, pp. 391–407, 1999.

[133] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P., "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[134] Klotz, E. and Newman, A. M., "Practical guidelines for solving difficult linear programs," *Surveys in Operations Research and Management Science*, vol. 18, pp. 1–17, 2013.

[135] Kuby, M. J. and Gray, R. G., "The hub network design problem with stopovers and feeders: The case of Federal Express," *Transportation Research Part A*, vol. 27A, no. 1, pp. 1–12, 1993.

[136] Land, A. H. and Doig, A. G., "An automatic method of solving discrete programming problems," *Econometrica*, pp. 497–520, July 1960.

[137] Li, D., Huang, H., Morton, A., and Chew, E., "Simultaneous fleet assignment and cargo routing using benders decomposition," *OR Spectrum*, vol. 28, pp. 319–335, 2006.

[138] Li, Y., Miao, Q., and Wang, B., "US postal airmail routing optimization," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2334, pp. 21–28, 2013.

[139] Lim, C., *Wiley Encyclopedia of Operations Research and Management Science*, ch. Relationship Among Benders, Dantzig-Wolfe, and Lagrangian Optimization, pp. 1 – 6. John Wiley & Sons, 2010.

[140] Limbourg, S., Schyns, M., and Laporte, G., "Automatic aircraft cargo load planning," *Journal of the Operational Research Society*, vol. 63, no. 9, pp. 1271–1283, 2012.

[141] Little, J. D., Murty, K. G., Sweeney, D. W., and Karel, C., "An algorithm for the traveling salesman problem," *Operations Research*, vol. 11, no. 6, pp. 972–989, 1963.

[142] Louwerse, I., Mijnarends, J., Meuffels, I., Huisman, D., and Fleuren, H., "Scheduling movements in the network of an express service provider," *Flexible Services and Manufacturing Journal*, vol. 26, no. 4, pp. 565–584, 2014.

[143] Luke, S., *Essentials of Metaheuristics*. 2009. `https://cs.gmu.edu/~sean/book/metaheuristics/Essentials.pdf`.

[144] Lumsden, K., Dallari, F., and Ruggeri, R., "Improving the efficiency of the hub and spoke system for the SKF European distribution network," *International Journal of Physical Distribution & Logistics Management*, vol. 29, no. 1, pp. 50–64, 1999.

[145] Magnanti, T. L., Mireault, P., and Wong, R. T., "Tailoring Benders decomposition for uncapacitated network design," *Mathematical Programming Study*, vol. 26, pp. 112–154, 1986.

[146] Magnanti, T. L. and Wong, R. T., "Network design and transportation planning: Models and algorithms," *Transportation Science*, vol. 18, pp. 1–55, February 1984.

[147] Magnanti, T. L. and Mirchandani, P., "Shortest paths, network design and associated polyhedra," *Networks*, vol. 23, no. 2, pp. 103–121, 1993.

[148] Magnanti, T. L. and Mirchandani, P., "Shortest paths, single origin-destination network design, and associated polyhedra," *Networks*, vol. 23, pp. 103–121, 1993.

[149] Magnanti, T. L., Mirchandani, P., and Vachani, R., "The convex hull of two core capacitated network design problems," *Mathematical Programming*, vol. 60, pp. 233–250, 1993.

[150] Magnanti, T. L., Mirchandani, P., and Vachani, R., "Modeling and solving the two-facility capacitated network loading problem," *Operations Research*, vol. 43, no. 1, pp. 142–157, 1995.

[151] Magnanti, T. and Wong, R., "Accelerating Benders decomposition: Algorithmic enhancement and model selection criteria," *Operations Research*, vol. 29, no. 3, pp. 464–484, 1981.

[152] Mahapatra, S., "Analysis of routing strategies in air transportation networks for express package delivery services," Master's thesis, University of Maryland, College Park, MD, 2005.

[153] Marsten, R. E. and Muller, M. R., "A mixed-integer programming approach to air cargo fleet planning," *Management Science*, vol. 26, no. 11, pp. 1096–1107, 1980.

[154] Martins, J. R. R. A. and Lambe, A. B., "Multidisciplinary design optimization: A survey of architectures," *AIAA Journal*, vol. 51, no. 9, pp. 2049–2075, 2013.

[155] Mason, R. O., McKenney, J. L., Carlson, W., and Copeland, D., "Absolutely, positively operations research: The Federal Express story," *Interfaces*, vol. 27, no. 2, pp. 17–36, 1997.

[156] Mercier, A., Cordeau, J.-F., and Soumis, F., "A computational study of Benders decomposition for the integrated aircraft routing and crew scheduling problem," *Computers & Operations Research*, vol. 32, pp. 1451–1476, 2005.

[157] Mercier, A. and Soumis, F., "An integrated aircraft routing, crew scheduling and flight retiming model," *Computers & Operations Research*, vol. 34, pp. 2251 – 2265, 2007.

[158] Mladenović, N. and Hansen, P., "Variable neighborhood search," *Computers & operations research*, vol. 24, no. 11, pp. 1097–1100, 1997.

[159] Muter, I., Birbil, Ş. İ., and Bülbül, K., "Benders decomposition and column-and-row generation for solving large-scale linear programs with column-dependent-rows," *European Journal of Operational Research*, vol. 264, no. 1, pp. 29–45, 2018.

[160] Nemhauser, G. L. and Wolsey, L. A., *Integer and Combinatorial Optimization.* Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, 1989.

[161] O'Kelly, M. E., "A geographer's analysis of hub-and-spoke networks," *Journal of Transport Geography*, vol. 6, no. 6, pp. 171–186, 1998.

[162] Ouorou, A., Mahey, P., and Vial, J.-P., "A survey of algorithms for convex multicommodity flow problems," *Management Science*, vol. 46, no. 1, pp. 126–147, 2000.

[163] Papadakos, N., "Practical enhancements to the Magnanti-Wong method," *Operations Research Letters*, vol. 36, pp. 444–449, 2008.

[164] Papadakos, N., "Integrated airline scheduling," *Computers & Operations Research*, vol. 36, pp. 176–195, 2009.

[165] Papadimitriou, C. H. and Steiglitz, K., *Combinatorial Optimization: Algorithms and Complexity.* New York: Dover Publications, 1998.

[166] Poli, R., Kennedy, J., and Blackwell, T., "Particle swarm optimization," *Swarm intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[167] Powell, W. B. and Sheffi, Y., "The load planning problem of motor carriers: Problem description and a proposed solution approach," *Transportation Research Part A: General*, vol. 17, no. 6, pp. 471–480, 1983.

[168] Rahmaniani, R., Crainic, T. G., Gendreau, M., and Rei, W., "The Benders decomposition algorithm: A literature review," *European Journal of Operational Research*, vol. 259, pp. 801–817, 2017.

[169] Resende, M. G. and Ribeiro, C. C., "Greedy randomized adaptive search procedures: Advances, hybridizations, and applications," in *Handbook of Metaheuristics*, pp. 283–319, Springer, 2010.

[170] Rexing, B., Barnhart, C., Kniker, T., Jarrah, A., and Krishnamurthy, N., "Airline fleet assignment with time windows," *Transportation Science*, vol. 34, no. 1, pp. 1 – 20, 2000.

[171] Rios, J. and Ross, K., "Massively parallel Dantiz-Wolfe decomposition applied to traffic flow scheduling," in *AIAA Guidance, Navigation and Control Conference*, (Chicago, Illinois), AIAA, August 2009.

[172] Root, S. and Cohn, A., "A novel modeling approach for express package carrier planning," *Naval Research Logistics (NRL)*, vol. 55, no. 7, pp. 670–683, 2008.

[173] Rosenkrantz, D. J., Stearns, R. E., and Lewis, P. M., "An analysis of several heuristics for the traveling salesman problem," *SIAM Journal of Computing*, vol. 6, pp. 563–581, Sept 1977.

[174] SAVELSBERGH, M., “A branch-and-price algorithm for the generalized assignment problem,” *Operations Research*, vol. 45, no. 6, pp. 831–841, 1997.

[175] SCHENK, L. and KLABJAN, D., “Intramarket optimization for express package carriers,” *Transportation Science*, vol. 42, no. 4, pp. 530–545, 2008.

[176] SCHENK, L. and KLABJAN, D., “Intra market optimization for express package carriers with station to station travel and proportional sorting,” *Computers & Operations Research*, vol. 37, no. 10, pp. 1749–1761, 2010.

[177] SCHRIJVER, A., *Theory of Linear and Integer Programming.* John Wiley & Sons, 1998.

[178] SHEN, S., *Logistics Service Network Design: Models, Algorithms, and Applications.* PhD thesis, Massachusetts Institute of Technology, 2004.

[179] SHERALI, H. D., BISH, E. K., and ZHU, X., “Airline fleet assignment concepts, models, and algorithms,” *European Journal of Operational Research*, vol. 172, pp. 1–30, 2006.

[180] SKUTELLA, M., “An introduction to network flows over time,” in *Research Trends in Combinatorial Optimization*, pp. 451–482, Springer, 2009.

[181] SMILOWITZ, K. R., ATAMTÜRK, A., and DAGANZO, C. F., “Deferred item and vehicle routing within integrated networks,” *Transportation Research Part E: Logistics and Transportation Review*, vol. 39, no. 4, pp. 305–323, 2003.

[182] SOBIESZCZANSKI-SOBIESKI, J. and HAFTKA, R. T., “Multidisciplinary aerospace design optimization: survey of recent developments,” *Structural optimization*, vol. 14, no. 1, pp. 1–23, 1997.

[183] SRINIVAS, M. and PATNAIK, L. M., “Genetic algorithms: A survey,” *Computer*, vol. 27, pp. 17–26, June 1994.

[184] TALBI, E.-G., *Metaheuristics: from design to implementation*, vol. 74. John Wiley & Sons, 2009.

[185] TOSSERAMS, S., ETMAN, L. P., and ROODA, J., “A classification of methods for distributed system optimization based on formulation structure,” *Structural and Multidisciplinary Optimization*, vol. 39, no. 5, p. 503, 2009.

[186] UNITED PARCEL SERVICE, INC., “Annual Report Pursuant to Section 13 or 15(d) of the Security Exchange Act of 1934.” Form 10-K, 31 December 2016.

[187] UNIVERSITY OF WATERLOO, “Mona Lisa TSP Challenge.” Web Document, July 2012. `http://www.math.uwaterloo.ca/tsp/data/ml/monalisa.html`.

[188] ÜSTER, H. and AGRAHARI, H., “A Benders decomposition approach for a distribution network design problem with consolidation an capacity considerations,” *Operations Research Letters*, vol. 39, pp. 138 – 143, 2011.

[189] Van Laarhoven, P. J. and Aarts, E. H., "Simulated annealing," in *Simulated annealing: Theory and applications*, pp. 7–15, Springer, 1987.

[190] van Roy, T. J., "A cross decomposition algorithm for capacitated facility location," *Operations Research*, vol. 34, pp. 145–163, Jan.–Feb. 1986.

[191] Vanderbeck, F., "On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm," *Operations Research*, vol. 48, pp. 111–128, Jan.-Feb. 2000.

[192] Vanderbeck, F. and Savelsbergh, M. W. P., "A generic view of Dantzig-Wolfe decomposition in mixed integer programming," *Operations Research Letters*, vol. 36, pp. 296–306, 2005.

[193] Vanderbei, R. J., *Linear Programming: Foundations and Extensions.* Heidelberg: Springer, 4th ed., 2014.

[194] Vanderplaats, G. N., *Multidiscipline Design Optimization.* Vanderplaats Research & Development, Inc., 2007.

[195] Venter, G. and Sobieszczanski-Sobieski, J., "Particle swarm optimization," *AIAA Journal*, vol. 41, no. 8, pp. 1583–1589, 2003.

[196] Voss, S., Martello, S., Osman, I. H., and Roucairol, C., eds., *Metaheuristics: Advances and Trends in Local Search Paradigms for Optimization.* New York: Springer Science & Business Media, 2012.

[197] Voudouris, C., *Guided local search for combinatorial optimization problems.* PhD thesis, University of Essex, Colchester, UK, 1997.

[198] Voudouris, C. and Tsang, E., "Guided local search and its application to the traveling salesman problem," *European Journal of Operational Research*, vol. 113, no. 2, pp. 469–499, 1999.

[199] Voudouris, C., Tsang, E. P., and Alsheddy, A., "Guided local search," in *Handbook of Metaheuristics*, pp. 321–361, Springer, 2010.

[200] Wasner, M. and Zäpfel, G., "An integrated mulit-depot hub-location vehicle routing model for network planning of parcel service," *International Journal of Production Economics*, vol. 90, pp. 403–419, 2004.

[201] Wieberneit, N., "Service network design for freight transportation: a review," *OR Spectrum*, vol. 30, no. 1, pp. 77–112, 2008.

[202] Wolsey, L. A., *Integer Programming.* John Wiley & Sons, 1998.

[203] Wu, P. and Hartman, J., "Case study: Solving a rental fleet sizing model with a large time-space network," *The Engineering Economist*, vol. 55, pp. 71–104, 2010.

[204] Yaman, H., Karasan, O. E., and Kara, B. Y., “Release time scheduling and hub location for next-day delivery,” *Operations research*, vol. 60, no. 4, pp. 906–917, 2012.

[205] Yang, L. and Kornfeld, R., “Examination of the hub-and-spoke netowrk: A case example using overnight package delivery,” in *41st Aerospace Sciences Meeting and Exhibit, Aerospace Sciences Meetings*, 2003.

[206] Zäpfel, G. and Wasner, M., “Planning and optimization of hub-and-spoke transportation networks of cooperative third-party logistics providers,” *International Journal of Production Economics*, vol. 78, pp. 207–220, 2002.