NEXT GENERATION ALGORITHMS FOR RAILROAD CREW AND LOCOMOTIVE SCHEDULING

By

BALACHANDRAN VAIDYANATHAN

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2007

To my parents

ACKNOWLEDGMENTS

First, I thank my advisor Dr. Ravindra K. Ahuja who has been a great supervisor and mentor through my study at the University of Florida. His drive, enthusiasm, and sense of purpose have been exemplary, and his advice and support extremely helpful. I also express my gratitude to Dr. Joseph P. Geunes, Dr. Donald W. Hearn, and Dr. Sartaj Sahni for serving as my supervisory committee members and giving me thoughtful suggestions.

My special acknowledgement goes to my friend and professor Dr. G. Srinivasan who inspired me to take up Operations Research as a career. I thank all my collaborators, colleagues, and friends, especially, Dr. James B. Orlin, Dr. Jian Liu, Dr. Krishna Jha, Dr. Arvind Kumar, Dr. Guvenc Sahin, Mr. Larry A. Shughart, Sailash Mani, Ashwin Arulselvan, Suchandan Guha, Ibrahim Karakayali, Ismail Bakal, and Ashish Nemani.

I acknowledge my entire family and friends for their support. My utmost appreciation goes to my parents who taught me some of the most valuable lessons in life, to my sister whose energy and enthusiasm are limitless, and to my grandparents for always believing in me. Finally, I thank my wife, Sirisha, for her unwavering support which was instrumental in the completion of my research and dissertation.

TABLE OF CONTENTS

page

LIST OF TABLES

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

NEXT GENERATION ALGORITHMS FOR RAILROAD CREW AND LOCOMOTIVE SCHEDULING

By

Balachandran Vaidyanathan

August 2007

Chair: Ravindra K. Ahuja
Major: Industrial and Systems Engineering

Our research concerns optimal railroad crew and locomotive management. Crew scheduling entails assigning crews to trains while complying with union rules, so that the crew costs are minimal and train delays due to crew unavailability are minimized. Locomotive scheduling ensures the correct number and type of locomotives are attached to each train such that a train has sufficient pulling power while satisfying a variety of business constraints and minimizing locomotive costs. Locomotive routing involves routing each locomotive unit on a cyclic sequence of trains so that it can be fueled and serviced as necessary. These problems are difficult to solve NP-Complete problems and the North American locomotive and crew scheduling problems have not yet been solved satisfactorily. However, solving these problems has the potential to save railroads several million dollars a year.

First, we examine the North American railroad crew scheduling problem (CSP). We develop a network-flow based crew-optimization model and design efficient algorithms using problem decomposition and relaxation techniques. Next, we extend the earlier attempts to solve the locomotive planning problem (LPP) on several dimensions by considering new constraints desired by locomotive directors. We develop additional formulations necessary to transition solutions of our models to practice and report computational tests of these models on the data

provided to us by a major US railroad. Next, we study the locomotive routing problem (LRP). The LRP is a very large-scale combinatorial optimization problem and has previously been unstudied and unsolved. We formulate the LRP as an integer programming problem on a suitably constructed space-time network and develop fast aggregation-disaggregation based methods to solve this problem. Finally, we study the multicommodity flow problem (MCFP). MCFPs have found application in wide variety of domains. The LPP and the CSP are indeed special cases of the MCFP. However, the integer version of this problem is NP-Complete. We propose a novel aggregation-disaggregation framework to solve a class of large-scale integer MCFPs.

CHAPTER 1
INTRODUCTION

US railroads carry millions of railcars annually accounting for 40% of the nation's intercity ton-miles. US freight tonnage is expected to double in volume over the next 20 years (Florida Department of Transportation Report (2005)). Policy makers and corporate leaders are very concerned about the ability of the nation's infrastructure to handle this growth. Congestion has increasingly become a major economic barrier to the free flow of goods across the nation, with highways, airports, and railroads all laboring under record levels of volume, high energy costs, employee shortages, reduced funding, and additional challenges of security and severe weather. A number of recent studies from policy think tanks, government agencies, and academic associations have shown that the railroad industry must do more to avert crippling gridlock. It is thus incumbent on us as a society to work together to discover new and innovative techniques whereby all transportation providers, including the railroads, can improve the utilization, productivity, and reliability of the existing infrastructure. In this study, we have considered two very important problems related to railroad operations: crew scheduling and locomotive planning. We have developed robust optimization techniques to solve these problems quickly and to near optimality. In our research, we have placed a large emphasis on "implementability" of solutions, which refers to the ability of a solution to be used in real-life. Our algorithms for locomotive and crew scheduling are designed to produce implementable solutions which can be used by railroads to realize potential savings.

Railroads decompose their planning and scheduling problems into a number of stages. A blocking plan specifies how a set of cars with different origin-destination pairs be consolidated into relatively large groups (blocks) of cars to reduce handlings. These blocks then travel over the train network. The train schedule prescribes the number of trains to assemble, the origin, the

destination, the route of each train, the arrival and departure times of each train at each station, and the days of operation of each train. Once the train schedules have been finalized, locomotives and crews must be assigned to each train. Locomotive scheduling ensures the correct number and type of locomotives are attached to each train such that every train has sufficient pulling power, while simultaneously satisfying a variety of business constraints and minimizing locomotive costs. Crew scheduling entails assigning crews to trains in each crew district, while complying with union rules, so that the crew costs are minimal and train delays due to crew unavailability are minimized.

The railroad blocking problem and the train scheduling problem are well studied problems. Some recent references for the blocking problem are Newton, Barnhart, and Vance (1998), Barnhart, Jin, and Vance (2000b), and Ahuja, Jha, and Liu (2007). Ahuja, Jha, and Liu (2007) conduct the most comprehensive study of the blocking problem. They developed a very large scale neighborhood search (VLSN) algorithm to solve this problem. This algorithm was applied to data provided by several railroads producing excellent computational results. Their algorithms have also been licensed by two Class I railroads in the United States. Given a blocking plan, developing a train schedule is the next operational planning task faced by a railroad. The train scheduling problem is to identify train routes, their frequencies (how often they run in a week), and their timetables so as to minimize the cost of carrying cars from their origins to their destinations. The train schedule developed is repeated every period (mostly weekly). Some recent research on this problem is due to Farvolden and Powell (1994), Campbell (1996), Kraft (1998), Brannlund et al. (1998), and Jha (2004). Once the train schedule and blocks are fixed, then next problem that needs to be solved is the block-to-train assignment problem. This problem

has been studied by Crainic, Ferland, and Rousseau (1984), Haghani (1989), Keaton (1989, 1992), and Jha (2004).

In our research, we focus on the crew planning problem, the locomotive planning problem, and the LRP. These problems are large-scale NP-Complete problems which have not yet been solved satisfactorily. Solving these problems has the potential to save the railroads millions of dollars and make a significant impact on the bottomline. We develop novel optimization techniques to generate realistic and implementable solutions to these problems, employing robust algorithms. We have extensively applied theory from network optimization, linear programming, integer programming, and heuristic optimization to solve these problems.

CHAPTER 2
THE RAILROAD CREW SCHEDULING PROBLEM

2.1 Introduction

Crew scheduling problems consist of assigning crews to trains and creating rosters for each crew, while satisfying a variety of Federal Railway Administration (FRA) regulations and trade-union work rules. The objectives are to minimize the cost of operating trains on one hand and to improve the crew quality-of-life on the other hand. Improved quality of crew life leads to more productive employees, less employee turnover, and much safer operations. Although airline CSPs have been well studied and well solved, and railroad CSPs for European and Asian railroads have also been addressed to some extent, CSPs for North American railroads, due to various union and regulatory complexities, are unique and remain unsolved. Our focus is on developing efficient network flow optimization models that can form a backbone for all important aspects of crew scheduling for North American railroads: tactical, planning and strategic analysis. Henceforth, unless otherwise specified, the CSP is referred to in the context of North American railroads.

US freight tonnage is expected to double in volume over the next 20 years (Florida Department of Transportation Report (2005)). Railroad executives are very concerned about their ability to attract, train, and retain sufficient semi-skilled labor necessary to staff the increased number of train starts that will be needed to support this growth. Railroads pay train crew employees very high salaries (around $70,000 per year plus benefits) and yet have difficulty attracting a high quality work force. Operating a train as an engineer or managing a train as a conductor is not an easy job. This is further complicated by the fact that crews are seldom assigned to trains based on a fixed schedule. Generally the company telephones the next available crew and gives them their assignment two hours before a train departs. The crew takes

the train to an away location where they rest at a hotel and then return on another train as their turn is reached. Consequently, train crews do not know from one day to the next, let alone a week or a month ahead, when they will be working. Train crews spend inordinate amounts of time on call, waiting for assignment and away from their homes and families. The irregular work-style of railroad crews makes attracting potential employees to this career harder.

Also, railroads are not very profitable, typically earning less than 10% return on capital, and are constrained from raising already high wages to attract more employees. To close the supply and demand gap for train crews, railroads must raise productivity of their existing crews and change the historical pattern of operations to improve employees' quality-of-life. Success on both fronts will be required to ensure that railroads can continue to profitably grow their businesses. Labor costs, the largest component of a railroad's operating expense, account for a large percentage of the total revenue. Depending on the size of their network, each Class I railroad (a Class I railroad, as defined by the Association of American Railroads, has an operating revenue exceeding $319.3 million) employs around 15,000 to 25,000 locomotive engineers, conductors, and brakemen (Surface Transportation Board, US DOT, Bureau of Accounts). Consequently, improving the efficiency and effectiveness of train crews has the potential to dramatically reduce the cost of transportation. We propose a network-flow model and algorithms for assigning crews to trains that will make a significant impact on a railroad's on-time performance, crew utilization and productivity, while also improving both quality-of-lives for crew and railroad safety.

In a large Class I railroad, various divisions are tasked with analyzing train crews. Each group is interested in a different aspect of crew planning and scheduling. These perspectives can generally be characterized based on the planning horizon of the issue at hand. Crew issues faced

by railroads can be broadly classified into three categories: 1) Tactical – Decisions that must be made immediately to support real-time train operations. Tactical problems have a planning horizon of 24 to 48 hours. 2) Planning – Decisions that must be made as a part of the crew schedule design process. Typically, railroads make adjustments to their network operating plan every month, with significant changes two or three times a year to account for both long-run and seasonal changes in traffic patterns. 3) Strategic – Decisions that must be made well in advance (more than a year) of implementation to ensure sufficient lead time is available to properly prepare and implement a new business practice. The models and algorithms proposed have applications in all these areas of decision making.

Crew scheduling is one of the important mathematical problems in the rich set of planning and scheduling problems that can be modeled and solved using mathematical optimization techniques (Assad (1981, 1983), and Cordeau, Toth, and Vigo (1998b)). Crew scheduling is a well known problem in operations research and has been historically associated with airlines and mass-transit companies. Several papers on crew scheduling management have appeared in the past literature; most notable among these are due to Wren (1981), Bodin et al. (1983), Carraresi and Gallo (1984), Wise (1995), and Desrosiers et al. (1995). All these articles explore a set covering based approach to solve the CSP. Crew scheduling is conventionally divided into two stages: (1) Crew pairing: A crew pairing is a sequence of connected segments that start and end at the same crew base and satisfy all legality constraints. The objective is to find the minimum-cost set of crew pairings such that each flight or train segment is covered. (2) Crew rostering: The objective here is to assign individual crew members to trips or sequences of crew pairings. This pairing and rostering approach uses a set covering formulation and is generally solved using column generation embedded in a branch-and-bound framework (also called branch-and-price).

The pairing and rostering approach has gained wide acceptance and application in the airline industry. Gopalakrishnan and Johnson (2005) in a recent survey paper discuss the state-of-the-art in solution methodologies for the airline crew pairing and rostering problem. There have also been some applications of this approach in the railroad industry. Caprara et al. (1997), Ernst et al. (2001), and Freling, Lentink, and Wagelmans (2004) describe the application of this approach to railroad crew management. Caprara et al. (1997) describe the solution techniques adopted at an Italian railroad. They consider several business rules that are specific to European railroads and develop a heuristic algorithm to generate rosters. Ernst et al. (2001) consider the CSP faced by Australian railroads and develop an optimization model which constructs crew parings and rosters. While they consider several business rules, they still solve a relaxed version of the problem and mention the necessity for an exact method in their conclusions. Freling, Lentink, and Wagelmans (2004) develop a decision support system for airline and railroad crew planning using a branch-and-price solution approach to solve the integrated problem of pairing and rostering. They show that the integrated approach provides significant benefits over the sequential approach of solving the pairing problem and then the rostering problem. Other articles which describe applications of the pairing and rostering approach include Barnhart et al. (1994, 2003).

Other research in the area of railroad crew scheduling which uses a different approach is due to Chu and Chan (1998), and Walker, Snowdon, and Ryan (2005). Chu and Chan (1998) consider the problem of crew scheduling for light rail transit in Hong Kong. They decompose the problem into two stages, the first one involving partitioning of driving blocks into pieces, and the second one involving the combination of pieces into runs. With added localized optimization heuristics, they were able to solve the problem in less than half an hour of computational time.

However, their approach could not model the problem completely and the solution could only be used as a guideline for crew schedule generation. Walker, Snowdon, and Ryan (2005) develop an integer programming based method for simultaneous disruption recovery of train timetable and crew roster in real time in the context of New Zealand railroads. The crew rules that they consider are relatively simplistic and can be expressed in the form of integer programming constraints. They solve the problem using a column and constraint generation algorithm.

While there have been several papers devoted to the study of the railroad CSPs in Europe, Asia, and Australia, North American railroad CSPs are yet to be addressed satisfactorily. The only application of optimization methods to North American railroad crew scheduling is due to Gorman and Sarrafzadeh (2000). They studied crew balancing in the context of a major North American Railroad, Burlington Northern Santa Fe (BNSF) Railway and developed a dynamic programming approach to solve the problem. The major short-coming of their research was that they did not consider the possibility of different crew types; each governed by a different set of rules. Another drawback is that their approach could handle only a particular crew district configuration (single-ended crew district). While most crew districts in North America are single-ended, there are several which are double-ended or even more complex. The multicommodity network flow approach described models all the rules considered by Gorman and Sarrafzadeh (2000) and also handles the case where different crew pools have different sets of rules. It is also applicable to all the crew district configurations encountered in North America (in Section 2.2, we describe these configurations).

From our extensive review of the literature, we observe that the crew pairing and rostering approaches which use column generation have been the predominantly successful method to

solve CSPs. However, this approach cannot be used for North American railroads due to the following reasons:

- The rail network of North American railroad is divided into several crew districts. As a train follows its route, it goes from one crew district to another, picking up and dropping off crew at crew change terminals. Almost all crew districts consist of two or three terminals. Hence, a pairing and rostering approach is needlessly complex and not required since most pairings would consist of two trains, an outbound train from home to away and an inbound train from away to home. Also rail networks typically consist of 200-300 crew districts and the emphasis is on an approach which is simple and fast, and column generation techniques which are computationally very intensive are not appropriate.

- The FRA regulations governing North American railroads are extremely complex. The most complicating of these rules is First-In-First-Out (FIFO) requirement. FIFO constraints require that crews should be called on duty in the order in which they become qualified for assignment at a location. The reader may note that none of the past research handles constraints of this kind. While this constraint is extremely easy to state, explicitly modeling these constraints make the problem computationally intractable. The success of all approaches using column generation or branch-and-price algorithms is dependant on the ease of solving the sub-problem. Addition of the FIFO side constraints to the problem would spoil the special structure of the sub-problem and blow up the computational times. Since our model needs to be fast enough to be used in a real-time environment, this approach is once again not suitable.

While there has been significant work in the area of crew scheduling for European, Asian, and Australian railroads as well as in the area of airline crew scheduling, there is no modeling approach that is flexible enough to tackle CSPs faced by North American railroads. Our approach (Vaidyanathan, Jha, and Ahuja (2007a)), is the first of its kind and is a novel contribution to the application of innovative optimization techniques to solve real-world problems.

We model the CSP as a multicommodity network flow problem on an underlying space-time network. In this model, crew pools (set of crews governed by same business rules in a crew district) represent commodities, and the flow of individual crew represents their assignments. The space-time network is constructed in such a way that flow of crew automatically satisfies all FRA regulations and trade-union rules other than the First-In-First-Out (FIFO) requirement. We

formulate the CSP as an integer multicommodity flow problem on a space-time network and the FIFO constraints are modeled as side constraints to the multicommodity flow problem. We show that solving the integer programming formulation using standard branch-and-bound methodology is computationally intractable. On the other hand, the same problem with relaxed FIFO constraints can be solved very efficiently using branch-and-bound. We refer to the CSP with relaxed FIFO constraints as the Relaxed Problem, and a solution to this problem provides a lower bound to the optimal solution of the CSP. We develop an algorithm, called Successive Constraint Generation (SCG) algorithm, which starts with the solution of the Relaxed Problem and then iteratively adds constraints to remove FIFO violations. We also develop another algorithm, called Quadratic Cost Perturbation (QCP) algorithm, which perturbs arc costs in the space-time network to penalize FIFO violations, and we prove that this approach guarantees FIFO compliance. We also show that the QCP approach produces optimal solutions in most cases and less than 0.2% gap for a few cases, with running times in the order of minutes.

Our major research contributions are:

- We develop a space-time network so that the flow of crews on this network automatically satisfies all the FRA regulations and trade-union rules other than the First-In-First-Out (FIFO) requirement. The network-construction procedure is flexible enough to handle several combinations of rules and regulations and also various different crew district configurations. It is also flexible enough to handle costs which are non-linear functions of arc durations.
- We formulate the CSP as an integer-programming problem on the space-time network, enforcing the FIFO requirements by adding side constraints. We prove the one-to-one correspondence between solutions to this integer program and solutions to the CSP.
- We show that the FIFO requirement, if handled by the integer-programming approach, complicates the structure of the problem and makes it computationally intractable.
- We develop an exact algorithm called Successive Constraint Generation (SCG), which first solves the relaxed version of the integer program (without FIFO constraints) and then iteratively adds constraints in order to eliminate FIFO infeasibilities.

- We develop an approach based on a Quadratic Cost Perturbation (QCP) that perturbs the cost of arcs in the space-time network in such a way as to penalize violations of the FIFO constraints. We prove that this method guarantees FIFO compliance for the problem that we study and also show that it produces the optimal solution in most cases.
- We present extensive computational results and case studies of our algorithms on the real-life data.

2.2 Problem Description

2.2.1 Terminology

Crew District: The rail network of a railroad is divided into crew districts that constitute a subset of terminals (nodes). Each crew district is typically a geographic corridor over which trains can travel with one crew. A typical railroad network for a major railroad in the US may be divided into as many as 200 to 300 crew districts. As a train follows its route, it goes from one crew district to another, picking up and dropping off crew at crew change terminals. Contrary to the airline industry, where certain crews have the flexibility to operate over a large territorial domain, in the North American railroad industry, crews are qualified to operate only in certain specific geographic territories. The physics of operating a train depend on the track geometry, which is defined by the hills and curves in the route and by signaling and interlocking systems that control the movement of trains. A crew must be intimately familiar with all aspects of a route to safely operate a train on that route. Consequently, most crews are qualified to operate on a limited number of routes.

Crew Pools: Within a crew district, there are several types of crews called crew pools or crew types, which may be governed by different trade-union rules and regulations. For example, a crew pool may have preference over the trains operated in a pre-specified time window. Similarly, a crew pool consisting of senior crew personnel is assigned only to pre-designated trains so that crews in that pool know their working hours ahead of time. The multiple crew

pools within each district with different constraints make CSPs complex and difficult to model mathematically.

Home and Away Terminals: The terminals where crews from a crew pool change trains are designated either as home terminals or away terminals. The railroad does not incur any lodging cost when a crew is at its home terminal. However, the railroad has to make arrangements for crew accommodation at their away terminals. Different crew districts have different combinations of home and away terminals. A crew district with one home terminal and one away terminal is called a single-ended crew district. In such crew districts, typically, a crew operates a train from its home location to an away location, rests in a hotel for at least eight hours, operates another train back to its home terminal, rests for ten to twelve hours, and repeats this cycle. The other type of crew district is a double-ended crew district, in which more than one terminal is a home terminal for different crew pools. Some of the other crew district configurations are crew districts with one home terminal and several away terminals, and crew districts with several home terminals and corresponding sets of away terminals.

Crew Detention: Once a crew reaches its away terminal and rests for the prescribed hours, the crew is ready to head back to its home terminal. However, if there is no train, then the crew may have to wait in a hotel. According to the trade-union rules, once a crew is at the away terminal for more than a pre-specified number of hours (generally 16 hours), the crew earns wages (called detention costs) without being on duty. For example, if a crew is waiting for assignment at the away terminal for 18 hours, it is paid detention charges for two hours.

Crew Deadheading: Crew deadheading refers to the repositioning of crew between terminals. A crew normally operates a train from its home terminal to an away terminal, rests for a designated time, and then operates another train back to its home terminal. Sometimes, at the

away terminal, there is no return train projected for some time, or there is a shortage of crews at another terminal. Thus, instead of waiting for train assignment at its current terminal, the crew can take a taxicab or a train (as a passenger) and deadhead to the home terminal. Similarly, the crew may also deadhead from a home terminal to an away terminal in order to rebalance and better match the train demand patterns and avoid train delays. Crew deadheading is expensive as the crew is considered to be on-duty while deadheading and earns wages, and railroads also incur taxi expenses. Each year, a major freight railroad may spend tens of millions of dollars in crew deadheading.

On-duty and Tie-up Time: Whenever a crew is assigned to a train, it performs some tasks to prepare the train for departure, and hence crews are called on duty before train departure time. The time at which the crew has to report for duty is called the on-duty time. Similarly, a crew performs some tasks after the arrival of the train at its destination, and hence crews are released from duty after the train arrival. The time at which the crew is released from duty is called tie-up time. We refer to the duty duration before train departure as duty-before-departure and the duty duration after train arrival as duty-after-arrival. Hence, the total duty time (or duty-period) of a crew assigned to a train is the sum of the duty-before-departure, the duty-after-arrival, and the travel time of the train.

Duty Period: In most cases, duty-period of a crew assigned to a train is the total duration between the on-duty time and the tie-up time. In some cases when a crew rests for a very short time at an away location before getting assigned to a train, the rest time and the duration of the second train may also included in the duty period of the crew. (Section 2.2.2 describes calculation of duty period in more detail.)

Dead Crews: By federal law, a train crew can only be on duty for a maximum of 12 consecutive hours, at which time the crew must cease all work and it becomes dead or dog-lawed. Dead crews are a frequent consequence of delayed trains, congestion, mechanical breakdowns, etc. In these cases, crew dispatchers must send a relief crew by taxi or another train so that the dead crew can be relieved. The dead crew must then get sufficient rest before becoming available to operate another train.

Train Delays: When a train reaches a crew change location and there is no available crew qualified to operate this train, the train must be delayed. Each train delay disrupts the operating plan and causes further delays due to the propagating network effect. Train delays due to crew unavailability are quite common among railroads. These delays are very expensive (some estimate $1,000 per hour) and can be reduced significantly through better crew scheduling and train scheduling.

2.2.2 Regulatory and Contractual Requirements

Assignment of crews to trains is governed by a variety of Federal Railway Administration (FRA) regulations and trade-union rules. These regulations range from the simple to the complex. The regulations also vary from district to district and from crew pool to crew pool. We list below some examples of these kinds of constraints and their typical parameter values:

- Duty-period of a crew cannot exceed 12 hours. Duty-period of a crew on a train is usually calculated as the time interval between the on-duty time and tie-up time of the train.
- Whenever a crew is released from duty at the home terminal or has been deadheaded to the home terminal, they can resume duty only after 12 hours (10 hours rest followed by 2 hours call period) if duty-period is greater than 10 hours, and after 10 hours (8 hours rest followed by 2 hours call period) if duty-period is less than or equal to 10 hours.
- Whenever a crew is released from duty at the away terminal, they must go for a minimum 8 hours rest, except for these circumstances:
    - If the total time period corresponding to the last travel time from the home terminal followed by a rest time of less than 4 hours plus travel time of the next

assignment back home is shorter than 12 hours (in this case, duty-period = travel time on inbound train + rest time at away location + travel time on outbound);

- If the total time corresponding to the last travel time from the home terminal plus travel time of the next assignment back home is less than 12 hours when the rest time in between the assignments is more than 4 hours (in this case, duty-period = travel time on inbound train + travel time on outbound train)

- Crews belonging to certain pools must be assigned to trains in a FIFO order.
- A train can only be operated by crews belonging to pre-specified pools.
- Every train must be operated by a single crew.
- Crews are guaranteed a certain minimum pay per month regardless of whether or not they work.

As the regulations for crew assignment can vary from district to district and crew pool to crew pool, it is a mathematical challenge to build a unified model to formulate and solve this problem. This partly explains why these problems remain unsolved and no commercial optimization product has been deployed yet at railroads. Also, due to low margins in the railroad industry, investment in research funding is viewed as a luxury despite a potentially high return on investment in automated decision support systems.

2.2.3 Problem Inputs

The inputs that go into the mathematical formulation of the CSP include:

- **Train Schedule:** The train schedule provides information about the departure time, arrival time, on-duty time, tie-up time, departure location, and arrival location for every train in each crew district it passes through. We do not consider stochasticity in the train schedule and assume that train delays are only due to the unavailability of crew and not due to train cancellations or other disruptions.
- **Crew Pool Attributes:** This includes attributes of various crew types, namely their home locations, their away locations, minimum rest time, train preferences, etc.
- **Crew Initial Position**: This provides the position of crew at the beginning of the planning horizon and includes information of the terminal at which a crew is released from duty, the time of release, the number of hours of duty done in the previous assignment, and the crew pool the crew belongs to.

- **Train-Pool Preferences:** The train-pool preferences, if any, give us information about the set of trains that can be operated by a crew pool.
- **Away Terminal Attributes:** This gives us information about the away terminals for each crew pool. It also includes the rest rules and detention rules for each crew pool and at each away terminal.
- **Deadhead Attributes:** This gives us the time taken to travel by taxi between two terminals in a crew district.
- **Cost Parameters:** Cost parameters are used to set up the objective function for the CSP. They consist of crew wage per hour, deadhead cost per hour, detention cost per hour, and train delay cost per hour.

2.2.4 Constraints and Objective Function

The CSP involves making decisions regarding the assignment of crews to trains, deadheading of crews by taxi, and train delays. The constraints can be categorized into two groups: operational constraints and contractual requirements. The operational constraints ensure that every train gets a qualified crew to operate it while a crew is not assigned to more than one train at the same time. These also include assignment of certain crew pools to pre-specified trains. Assignment of crews to trains must, in addition, satisfy the contractual requirements described in Section 2.2.2. In our mathematical model, the operational constraints of the model are handled by the integer multicommodity flow formulation described in Section 2.3.2, and the contractual restrictions are honored in the network construction phase described in Section 2.3.1. The objective function of the CSP is to minimize the total cost of crew wages, the cost of deadheading, the cost of crew detentions, and the cost of train delays.

2.3 Mathematical Modeling

2.3.1 Space-Time Network

The CSP is formulated as an integer multicommodity flow problem with side constraints on a space-time network. We can decompose the CSP into an independent problem for each crew district and construct the space-time network for a particular crew district. In the network, each

node corresponds to a crew event and has two defining attributes: location and time. The events that we model while constructing the space-time network for the CSP are departure of trains, arrival of trains, departure of deadheads, arrival of deadheads, supply of crew, and termination of crew duty to mark the end of the planning horizon. All the arcs in the network facilitate the flow of crews over time and space. Figure 2-1 presents an example of the space-time network in a crew district. Note that for the sake of clarity, this network only represents a subset of all the arcs.

For each crew, we create a supply node whose time corresponds to the time at which this crew is available for assignment, and whose location corresponds to the terminal from which the crew is released for duty. Each supply node is assigned a supply of one unit and corresponds to a crew member. We also create a common sink node for all crews at the end of the planning horizon. This sink has no location attribute and has the time attribute equal to the end of the planning horizon. The sink node has a demand equal to the total number of crew supplied. The supply and sink nodes ensure that all the crews that flow into the system at the beginning of the planning horizon are accounted for and flow out of the system at the end of the planning horizon.

For each train (say l) passing through a crew district, we create a departure node (say l') at the first departing station of the train in the crew district and an arrival node (say l'') at the last arriving station of the train in the crew district. Each arrival or departure node has two attributes: place and time. For example, place (l') = departure-station (l) and time (l') = on-duty-time (l); and similarly, place (l'') = arrival-station (l) and time (l'') = tie-up-time (l).

In the network, we create a train arc (l', l'') for each train l connecting the departure node and arrival node of train l. We create deadhead arcs to model the travel of crew by taxi. A deadhead arc is constructed between a train arrival or crew supply node at a location and a train

departure node at another location. All the deadhead arcs which satisfy the contractual rules and regulations are created. We construct rest arcs to model resting of a crew at a location. A rest arc is constructed between a train arrival node or a crew supply node at a location and a train departure node at the same location. Rest arcs are created in conformance to the contractual rules and regulations. All rest arcs which satisfy the contractual rules and regulations are constructed. Since the contractual regulations are often crew pool specific, deadhead arcs and rest arcs are created specific to a crew pool. This implies that only crew belonging to a particular crew pool can flow on a particular rest arc or a deadhead arc. For example, suppose a supply node corresponds to crew belonging to crew pool A, then all the arcs which emanate from this node can only carry crew belonging to crew pool A. Finally, we create demand arcs from all train arrival nodes and crew supply nodes to the sink node. Each arc has an associated cost equivalent to the crew wages, deadhead costs, or detention costs, as the case might be. Also in the network, the time at the tail of an arc is always less than the time at the head of an arc, which ensures the forward flow of commodities on the time scale. It can be noted that all contractual requirements other than the FIFO constraint are easily handled in the network construction.

The space-time network described above models the flow of crews while honoring all the contractual constraints except the FIFO rule. However, it does not model the case when qualified crews are not available for assignment to a train causing train delays. Next, we present the construction of additional arcs incorporating train delays. At a location, we create rest arcs and deadhead arcs which do not honor the rest regulations and penalize them to ensure that flows on these arcs occur only when qualified crews are not available for assignment. The flows on these arcs denote that the train will be delayed until crew becomes qualified for train operation. However, as the delay of a train may have propagating effect in the availability of crews in

subsequent assignments, we assume here that the crew assigned to a delayed train has sufficient slack in the rest time at the train arrival node to make it qualified for subsequent assignments. Thus, the additional rest arcs and deadhead arcs model the train delays, with the assumption that the effect of train delays is only local.

The reader may also note that honoring contractual regulations while constructing the network reduces the number of constraints in the integer program significantly. Now, we present the multicommodity integer programming formulation of the CSP.

2.3.2 Integer Programming Formulation (IPF)

We formulate the CSP as an integer multicommodity flow problem on the space-time network described in Section 2.3.1. In our formulation, each crew pool represents a commodity. Crew enters the system at crew supply nodes, and hence every supply node corresponds to a supply of one crew. The crew takes a sequence of connected train, rest, and deadhead arcs before finally reaching the sink. While flow of more than one crew type can take place on a train arc, rest and deadhead arcs can have flow of only one type because the business rules for rest and deadhead are crew pool specific. Next we present the integer programming formulation of the problem.

We use the following notation in our formulation:

N: Set of nodes in the space time network

L: Set of train arcs in the network, indexed by l

D: Set of deadhead arcs in the network, indexed by d

R: Set of rest arcs in the network, indexed by r

A: Set of arcs in the space-time network, indexed by a

G(N, A): Space-time network

N_s: Set of crew supply nodes

N_d: Sink node

C: Set of crew pools in the system, indexed by c

i^+ : Set of outgoing arcs at node i

i^- : Set of incoming arcs at node i

i_c^+ : Set of outgoing arcs specific to crew pool c at node i

i_c^- : Set of incoming arcs specific to crew pool c at node i

A_r: Set of arcs on which flow will violate FIFO constraint if there is flow on rest arc r

f: Total number of available crew

M: A very large number

c_l^c : Cost of crew wages for crew pool $c \in C$ on train arc $l \in L$

c_d : Cost of deadhead arc $d \in D$

c_r : Cost of rest arc $r \in R$

tail(l): The node from which arc l originates

head(l): The node at which arc l terminates

The decision variables are:

x_l^c : Flow of crew pool $c \in C$ on each train arc $l \in L$

x_d : Flow on deadhead arc $d \in D$

x_r : Flow on rest arc $r \in R$

We formulate the CSP as follows:

$$\text{Min} \sum_{l \in L} \sum_{c \in C} c_l^c x_l^c + \sum_{d \in D} c_d x_d + \sum_{r \in R} c_r x_r \qquad (2.1a)$$

$$\sum_{c \in C} x_l^c = 1, \text{ for all } l \in L \qquad (2.1b)$$

$$\sum_{a \in i^+} x_a = 1, \text{ for all } i \in N_s \tag{2.1c}$$

$$\sum_{a \in N_d^-} x_a = f \tag{2.1d}$$

$$x_l^c = \sum_{a \in tail(l)_c^-} x_a, \text{ for all } l \in L, c \in C \tag{2.1e}$$

$$x_l^c = \sum_{a \in head(l)_c^+} x_a, \text{ for all } l \in L, c \in C \tag{2.1f}$$

$$\sum_{r' \in A_r} x_{r'} - M(1 - x_r) \leq 0, \text{ for all } r \in R \tag{2.1g}$$

$$x_l^c \in \{0,1\} \text{ and integer, for all } l \in L, c \in C \tag{2.1h}$$

$$x_d \in \{0,1\} \text{ and integer, for all } d \in D \tag{2.1i}$$

$$x_r \in \{0,1\} \text{ and integer, for all } r \in R \tag{2.1j}$$

Constraint (2.1b) is the train cover constraint, which ensures that every train is assigned a qualified crew to operate it. Constraint (2.1c) ensures flow balance at a crew supply node. Constraint (2.1d) ensures the flow balance at the sink node. Constraints (2.1e) and (2.1f), respectively, ensure flow balance at train departure and arrival nodes. The flow balance constraints at a train arrival node ensure that the crew which is assigned to a train is subsequently assigned to a rest arc, a deadhead arc, or a sink arc which emanates from the arrival node of the train. Flow balance constraints at a train departure node ensure that the crew which is assigned to the train has been assigned to a rest arc, a deadhead arc, or a supply arc which terminates at the departure node of the train. Constraint (2.1g) ensures that the crew assignment honors the FIFO constraint. Constraints (2.1h), (2.1i), and (2.1j) specify that all the decision variables in the model are binary. The objective function (2.1a) is constructed to minimize the total cost of crew

wages, deadheading, detentions, and train delays. Note that the detention and delay costs are taken into account while calculating the cost of rest arcs.

Now we show how Constraint (2.1g) enforces FIFO requirements. Figure 2-2 illustrates crew assignments in two situations: one in which FIFO is satisfied and the other in which FIFO is violated. In case (a), the crew on train 1-3 arrives at Terminal 2 first and also leaves first, and hence FIFO requirement is satisfied. In case (b), FIFO requirement is violated because the crew on train 1-3 enters terminal 2 first, but leaves after the other crew. Therefore in the solution, if there is flow on arc (4, 5), there should not be any flow on arc (3, 6).

Let us consider the following cases for Constraint (2.1g) with respect to flow on arc (4, 5):

Case 1: $x_{(4,5)} = 1$: The constraint becomes $\sum_{r' \in A_{(4,5)}} x_{r'} \leq 0 \Rightarrow x_{r'} = 0 \ \forall r' \in A_{(4,5)}$. This ensures that if there is flow on rest arc (4, 5), then there cannot be flow on any arc belonging to the prohibited set $A_{(4,5)}$, and hence there will not be any flow on arc (3, 6).

Case 2: $x_{(4,5)} = 0$: The constraint becomes $\sum_{r' \in A_{(4,5)}} x_{r'} \leq M \ \ \forall r' \in A_{(4,5)}$ which essentially means that the constraint is relaxed.

Let us now estimate the size of a typical instance of the CSP in a crew district. Most crew districts have two terminals, and a typical train schedule has around 500 trains running in a couple of weeks in a crew district. Each crew district could have two to four crew types and around 50 crews. Therefore, the space-time network could have around 50 + 2 x 500 = 1,050 nodes. The number of arcs in the network could be very large if we construct all feasible rest arcs and deadhead arcs. To restrict the number of arcs constructed, we place a limit on the maximum duration of rest arcs. For example, if the train schedule stretches over a period of ten days, it is unrealistic for a crew to rest for more than three days. In this case, we can restrict the maximum

rest arc duration to three days. After the space-time network of a typical problem is pruned based on this rule, the number of deadhead arcs is typically around 25,000, and the number of rest arcs is around 100,000.

Since the number of rest arcs for a typical problem is of the order of 100,000, and as each rest arc has one FIFO constraint, the number of FIFO constraints in the model would be 100,000, which is too large. We would therefore be losing one of the main advantages of the network flow formulation, which is, by honoring all business rules in the network construction phase; we keep the number of constraints small. Our computational results also confirm that handling FIFO constraints explicitly in this manner makes the problem computationally intractable.

Let us now consider the Integer Programming Formulation where we relax the FIFO Constraint (2.1g); we call this problem the Relaxed Problem. This problem typically has more than 100,000 variables and several thousand constraints, which make it a large optimization problem. Integer programs of this size are usually very difficult to solve to optimality or near-optimality in a reasonable amount of time. But we were able to solve this problem to optimality in a matter of minutes using commercial branch-and-bound based MIP solver provided by CPLEX 9.0. We believe that this is due to the special structure of the Relaxed Problem, which helps speed up the solution time significantly. All variables in the formulation are binary variables, and this leads to the MIP engine exploring fewer branches on the branch-and-bound tree compared to the case where variables are integer variables. Whenever the engine branches on a non-integer variable, the value on one branch is set to zero and on the other branch is set to one. Hence, at each level of the tree, one variable's value is prefixed and can be eliminated from the model. Consequently, it is very likely that a feasible integral solution is obtained early on,

and nodes in the branch-and-bound tree are fathomed much earlier than while solving a general integer program.

Another benefit of the network flow based approach is that even though we do not explicitly model each crew, the space-time network and the constraints are such that from the final solution of the model, we can easily extract the set of trains a crew takes over the entire planning horizon. In order to do this, we start at the supply node of a particular crew and identify a path from this supply node to the sink node that has positive flow on it. Note that due to the commodity specific flow balance constraints at each node, every crew will have a unique path with positive flow from its supply node to the sink node.

Theorem 2.1. There is a one-to-one correspondence between a feasible flow on the space-time network satisfying constraints (2.1a)-(2.1j) and a feasible solution to the CSP.

Proof: Consider a feasible flow on the space-time network. We have seen above how the path of each crew can be extracted from the solution using a simple run-through procedure. Due to the network construction methodology, the extracted path of each crew has to satisfy all the business and contractual rules. Hence, we see that every feasible solution on the space-time network corresponds to a feasible crew schedule. We can also show that the reverse transformation from a feasible crew schedule to a feasible flow on the space-time network is possible, hence establishing the result. ♦

Hence, we have shown the one-to-one correspondence between feasible solutions to the integer programming formulation and feasible solutions to the CSP and thus have established the validity of our integer programming approach. In the next section, we describe various algorithms to solve the CSP, which are centered on handling FIFO constraints in a computationally efficient manner.

2.4 Solution Approaches

In this section, we present our approaches to solve the CSP. As the FIFO constraints are the ones which complicate the nature of the integer programming formulation, our solution approaches are centered on effective ways to handle this constraint. We develop a constraint-generation based exact approach and a cost-perturbation based heuristic approach to solve the problem. While the constraint-generation based approach performs significantly better than the direct approach to solve the integer programming formulation, its application in a real-time environment may be restricted due to long running times. On the other hand, the cost-perturbation scheme produces good quality FIFO compliant solutions very efficiently and hence is better suited for the real-time environment.

2.4.1 Successive Constraint Generation (SCG) Algorithm

The SCG algorithm works by iteratively pruning out crew assignments which violate the FIFO constraints from the current solution of a more relaxed problem. We considered the following two methods for implementing constraint generation: (1) A branch-and-bound algorithm where constraints are added to the LP relaxation that is solved at each node of the branch-and-bound tree until FIFO violations are eliminated (branch-and-cut); and (2) An iterative method where we run branch-and-bound algorithm on the relaxed problem, solve it to optimality, and then add constraints to remove infeasibilities which is followed by another run of branch-and-bound on the more constrained problem and so on.

However, on further deliberation, we chose to implement the second method over the first for the following reasons:

- Since the LP relaxation of the relaxed problem can have fractional flows on the rest arcs, the number of rest arcs with positive flow in the LP relaxation will be more than the number of rest arcs with positive flow in an integral version. Also, larger the number of rest arcs with positive flow, greater is the possibility of FIFO violations. Hence, more FIFO constraints are likely to be added to a non-integral solution.

- SCG implemented using method (2) allows us to stop at any point when we feel that the level of FIFO infeasibility is reasonably small. We are able to do that because after the addition of a set of constraints, we obtain an integral solution at regular intervals of less than a minute. On the other hand, in the branch-and-cut method, the addition of constraints at a node on the branch-and-bound tree would only guarantee FIFO compliance of the LP relaxation which will not be an integral solution in general. Hence, we do not have the option to prematurely terminate until a point when we obtain at least one integral solution to the LP relaxation and consequently we do not have control over the quality of intermediate solutions in terms of number of FIFO violations.

- We show in our computational results that QCP does an excellent job in enforcing FIFO constraints for the current set of business rules, but we also mention that QCP does not guarantee FIFO compliance when there is priority in assigning crews to trains. We believe that the real benefit of SCG could come from being used in conjunction with QCP. In this approach, we would first apply QCP to obtain a solution with very few FIFO violations. SCG is then applied on this solution to prune out the small number of remaining infeasibilities. A branch-and-cut approach in this context would be unnecessarily complicated since when a small number of constraints are added, the problem can be re-optimized within a few seconds using SCG.

The SCG algorithm starts with the optimal solution of the Relaxed Problem, which may have several violations of the FIFO rule. In each iteration, the algorithm scans the rest arcs in the current solution which have positive flow, and for each such rest arc assignment which violates FIFO constraints, it adds the corresponding FIFO constraints. We then re-solve the problem and re-check for FIFO infeasibilities. This process is repeated until all FIFO infeasibilities are removed.

Algorithm-SCG

Step 1: Solve the Relaxed Problem. If a feasible solution exists, then proceed to Step 2. Otherwise STOP as the problem is infeasible.

Step 2: Examine all the rest arcs with positive flow in the solution of Step 1. Add FIFO constraints to the integer program on those rest arcs assignments which violate FIFO requirements.

Step 3: If FIFO constraints are added in Step 2, re-optimize the modified integer program and go to Step 2. Otherwise STOP as we have the optimal solution.

Note that the final solution of SCG satisfies all the constraints of the Integer Programming Formulation (IPF), and the constraints of SCG are a subset of the constraints of IPF. Hence, the SCG algorithm is an exact algorithm guaranteeing optimal solution to the original problem. However, in the worst case, SCG could add all the FIFO constraints to the integer program and would hence become an intractable approach. Fortunately this seldom happens in practice. Our computational results show that the number of constraints added is usually much less than the total number of rest arcs in the network.

While the SCG is an exact algorithm and produces provably optimal solutions, the running time of this algorithm could be quite high. In our computational experiments, in some instances, SCG had a running time in the order of minutes while in others it had a running time in the order of hours. While these running times are acceptable in the planning environment, they would restrict the applicability of this algorithm in the real-time environment. In the next section, we describe a cost-perturbation based algorithm which produces very good quality FIFO-compliant solutions with running times comparable to that of the Relaxed Problem.

2.4.2 Quadratic Cost-Perturbation (QCP) Algorithm

In the previous section, we describe a successive constraint-generation based approach to remove the FIFO violations iteratively. In this section, we present an algorithm which penalizes the FIFO violations in a solution. We show that this method guarantees zero FIFO violations in the case where there is no priority in assigning crews to trains and serves as a heuristic method for the other case when there are priority restrictions. Cost perturbation not only enforces FIFO constraints but also retains the special network flow structure of the problem leading to fast computational times. The basic intuition behind this approach is that we perturb the costs of arcs while solving the Relaxed Problem in such a way as to guarantee FIFO compliance.

We present our cost perturbation strategy through the illustration shown in Figure 2-3 for the case when there is only one crew pool type. In case 2-3 (a), crew assignments are made in a non-FIFO manner, and in case 2-3 (b), the assignments are made in a FIFO manner. Now consider the case when crews are detained at the Terminal 2. Then, due to the nature of detention costs, the cost of the assignment 2-3 (b) would definitely be less than or equal to the cost of assignment 2-3 (a), and hence the solution to the Relaxed Problem would honor FIFO constraints. On the other hand, suppose all the rest arcs had a cost of zero; then both the assignments would have the same cost, and the Relaxed Problem would have no cost incentive to choose assignment 2-3 (b) over assignment 2-3 (a). Thus, a solution to the Relaxed Problem may violate the FIFO constraints. In order to provide an incentive to the Relaxed Problem to choose case 2-3 (b) over case 2-3 (a), we perturb the cost assignments on rest arcs so that the solution of the Relaxed Problem has assignments of type 2-3 (b) and not assignments of type 2-3 (a).

The cost perturbation scheme that we use is a function of the duration of rest arcs. Suppose that the time duration between events corresponding to nodes 2 and 4, 4 and 5, and 5 and 7 are a, b, and c, respectively. Consider a cost assignment which is proportional to the square of the duration of rest arcs. The constant of proportionality is represented by k.

Then,

Cost of assignment 2-3 (a) = $k(\textit{duration of arc } (2,7))^2 + k(\textit{duration of arc } (4,5))^2$

$= k(a+b+c)^2 + kb^2 = k(a^2 + 2b^2 + c^2 + 2ab + 2bc + 2ca)$

Cost of assignment 2-3 (b) = $k(\textit{duration of arc } (2,5))^2 + k(\textit{duration of arc } (4,7))^2$

$= k(a+b)^2 + k(b+c)^2 = k(a^2 + 2b^2 + c^2 + 2ab + 2bc)$

It can be observed that the cost of assignments in case 2-3 (b) is less than that in case 2-3 (a). Hence, when the rest arcs have zero costs, the quadratic cost perturbation scheme in the

Relaxed Problem will give FIFO compliant assignments, when there will be only one crew pool type. The observation made here can also be generalized for multiple crew pools unless there is priority of crew pools in assignments to trains. If there is a priority assigned to crews in assignments to trains, then a crew can have FIFO-violated assignment to gain the priority assignments. We state our observations here as the following theorem.

Theorem 2.2. Quadratic Cost Perturbation applied to the Relaxed Problem guarantees FIFO compliant crew assignments, if there is no priority in assigning crews to the trains.

Proof: In the space-time network, rest arcs may have one of three costs assigned to them: (a) zero costs, (b) detention costs, or (c) train delay costs. For e.g., If all the rest arcs in Figure 2-3 had zero costs, then as shown above, the Relaxed Problem will choose the FIFO compliant assignment because it is cheaper. If the rest arcs in Figure 2-3(a) had detention costs on them, then the FIFO assignment shown in Figure 2-3(b) will either have the same level of detention or lesser detention. Hence, the perturbation scheme will work in this case too. A similar argument would also work for train delay costs because FIFO assignments will always have equal or lesser train delays than non-FIFO assignments. ♦

Since we do not want to change the cost structure of the original problem by a large extent, we set the value of k to a very small value and perturb the cost of each rest arc by a value which is computed as described above. Our computational tests in Section 2.6.1 show that this method works very well, and the solutions produced by Quadratic Cost Perturbation are indeed FIFO-compliant in the case where there are no priorities. The solution time of this method is very short and is comparable to that of the Relaxed Problem. Note that in the case where there are priorities, this approach could be used to obtain a solution with a small number of violations and then the SCG algorithm can be used to prune out these violations. Another interesting observation is that

for most of the instances tested, this method produces solutions with objective function values same as those for the Relaxed Problems. This implies that FIFO constraints can be satisfied with little or no impact on the solution cost. Hence, using this approach, we are able to obtain excellent quality of solutions using much less computational time. Due to its attractive running times and high solution quality, this method has the potential to be used in both the planning and the real-time environment.

2.5 Significances and Uses of the Model

In the introduction, we mention that the crew scheduling model has applications in the tactical, planning and strategic environments. In this section, we elaborate and provide specific examples of how the model can be used as an effective tool for decision making.

2.5.1 Tactical Crew Scheduling

The defining problem in tactical crew scheduling is to determine which crew should be assigned to operate each train. However, there are a number of sub-problems and issues that must be considered before assigning crews to trains. Railroads have around-the-clock crew calling centers with the responsibilities of monitoring the status of each crew and the status of each train and anticipating when a particular crew should be called to operate a particular train. A typical crew-calling center employs 200-300 clerks (crew callers) to call crews and answer inbound telephone queries from management and the crews. First, a crew caller looks at the projected lineup (crew assignment) of outbound trains at a particular crew change location. With a projection of train departure times, say 13:30, 15:00, and 16:00, the crew caller then goes through a number of checks before assigning a crew to a train: Is this train covered by a designated assigned pool, or is it to be covered by FIFO assignment from the general pool? When is the next qualified crew rested and available to operate this train? The actual rules are

very complex, and the combinations of solutions that must be considered can overwhelm a person.

Our model has several applications in the tactical scheduling environment. Some of these applications are given below:

Assignment of crews to trains: The output of our model tells us how to assign crews to trains.

Recommend which crews to place in hotels and which crews to deadhead home: When a crew arrives at an away terminal, the crew callers have to decide whether the crew should deadhead back home or go to a hotel for rest. The model can be used to mathematically look ahead and systematically make the trade-off between different cost categories of crew wages, deadhead costs, detention costs, and rest violation costs.

Minimize trains delayed due to shortage of crew: Train delays are potentially very costly because the delay of a train may lead to the unavailability of crew to operate another train in the future and may have a negative domino effect on network-wide operations. By creating several deadhead arcs while constructing the space-time network, we ensure that such a situation is avoided.

Disruption management: The crew scheduling model can be used as a tool to bring back disrupted operations to normalcy. Suppose at some point in time the operations are disrupted. The current state or snapshot of the system gives us the location of each crew and the hours of duty already done. Using this information and the information about the future train schedule, the crew scheduling models can be used to optimally re-assign crew to trains.

2.5.2 Crew Planning

The essence of the crew planning problem for operations or planning is to determine how many crews should be in each crew pool. It can be noted that as each position is guaranteed a

minimum number of work hours per month, it is quite costly to overestimate the number of positions required to staff a pool. Currently, railroads solve the pool sizing problem based on historical precedent and rules-of-thumb, through negotiation with the union, and by trial and error. The network flow model can satisfy the need for a structured approach that captures all of the considerations, quantifies the various costs, and recommends the best way to define and staff crew pools. Some of the applications of the model in the planning environment are described next.

Develop and evaluate crew schedules: The crew scheduling model can also be used to compare the current crew schedule used with the model-generated schedule on the basis of several criteria such as average rest time at the home location, average rest time at the away location, average deadhead time, etc. By suitably changing the model cost parameters, we can obtain schedules with different characteristics. For example, if we want to minimize detention, we can set the detention cost to a very large value and run the model.

Size of crew pools: Using the crew scheduling model, we can study the impact of varying the crew pool size on the solution quality. For example, suppose our objective is to minimize the number of crew used. While formulating the problem, we give large cost incentives to flow on the sink arcs from crew supply nodes to the sink node. This would lead to the model's producing a crew schedule which uses the minimum number of crew.

2.5.3 Crew Strategic Analysis

Strategic management involves development of policies and plans and allocating resources so as to implement these plans. The timeframe of strategic management extends over several months or even years. Strategic crew problems include forecasting future head-count needs and evaluating major policy changes such as negotiating changes to trade-union rules or changing the number and location of crew change points on a network. The railroad industry is now

experiencing unprecedented traffic growth. Therefore, it is very important to be able to quantify the expected impact on manpower needs as traffic grows since it takes 18 to 24 months to hire, train, and qualify train crew personnel. Recently, corporate strategists have been struggling with the trade-off between crew costs and train delays. Our model can be used to quickly calibrate efficient frontiers for each crew district and show what number of crews minimizes the sum of train delay costs and crew costs. If railroad management is dissatisfied with that level of train performance, one can simply increase the cost of train delay, and the model will request additional crews such that a new cost-minimizing solution is obtained.

Some of the applications of the network flow model in the strategic environment are listed below.

Determining the number of crew districts and territory of crew districts: We can use the crew scheduling model to re-optimize and test different crew district configurations. For example, suppose crew district 1 operates trains between location A and location B, and crew district 2 operates trains between location B and location C. Merging all three stations into a single crew district could give us better opportunity to optimize costs.

Effect of changing crew trade-union rules: The CSP is a complex optimization problem due to strict trade-union rules related to crew operation. The change of any of these rules will face a lot of resistance from the labor union. At the same time, change of any of these rules has the potential to impact crew costs substantially. Using the crew scheduling model, we can evaluate the impact of changing the trade-union rules on the crew cost. For example, suppose we want to know the impact of changing the mandatory rest time at home from 12 hours to 10 hours. We can run the model with the parameter setting of 10 hours and measure the change in crew cost.

Forecasting crew requirement: Based on the forecasted train schedule, we can use the model to help us forecast crew requirement. We first run the model assuming that a very large number of crew is available. Since the crew supply is much more than what is required, many crews will directly flow from the crew supply to the sink node. The total crew supply minus the number of unused crews will give an idea of the number of crews required based on the forecasted train schedule.

In this section, we have seen that the crew scheduling model has several real-life applications in the tactical, planning, and strategic environments. If put into production, the model has the potential to enable railroad professionals to improve their day-to-day operations and to plan effectively in order to achieve their long-term organizational goals.

2.6 Computational Results

In this section, we present computational results of our algorithms on several problem instances, and we also present a case study done on a representative instance. We implemented our algorithms in Visual Basic .NET programming language and tested them on the data provided by a major Class I railroad. We modeled our integer programs using Concert Technology 2.0 modeling language and solved them using the CPLEX 9.0 solver. We conducted all computational tests on a Pentium IV, 512 MB RAM and 2.4 GHz processor desktop computer.

2.6.1 Comparison of Algorithms

In this section, we compare the performances of the Relaxed Problem, the exact Integer Programming Formulation (IPF), the Successive Constraint Generation (SCG) algorithm, and the Quadratic Cost Perturbation (QCP) algorithm on several real-life instances. Our problem instances consist of train schedules over a period of one to four weeks. In one instance, the number of crew pools is one, making the problem a single-commodity flow problem. In the other

set of instances, the number of crew pools is two, and the problem is formulated as a multicommodity flow problem. For each instance, we measure the solution cost, the solution time, the number of FIFO constraints added to the formulation, and the number of FIFO constraints violated in the final solution. It can be noted that no FIFO constraints are added while solving the Relaxed Problem and the Quadratic Cost Perturbation (QCP). The results of our computational tests are presented in Table 2-1.

We draw the following conclusions from the computational results:

- The solutions to the Relaxed Problem have the highest number of FIFO violations. However, the solution times are also the fastest.
- The Integer Programming Formulation (IPF) has several thousand FIFO constraints. These constraints make the problem computationally intractable, and we could not obtain feasible solution for any of the instances in one hour of computational time.
- The Successive Constraint Generation (SCG) algorithm starts with the solution to the Relaxed Problem as the initial solution and progressively reduces the number of infeasibilities. However, the amount of computational time taken by this algorithm is still quite large. We were able to obtain a FIFO-compliant solution for two instances, and for all other instances, we terminated the algorithm when the iteration that was running at the 30^{th} minute of computational time was complete.
- The Quadratic Cost Perturbation (QCP) algorithm produces FIFO-compliant crew schedules for all instances. Also, in six instances out of eight, the objective function values are equal to that of the Relaxed Problem. Since the Relaxed Problem provides a lower bound to the optimal solution, QCP algorithm produces the optimal solution in six instances out of eight, and the optimality gap is less than 0.2% for the other two instances. This algorithm also has very fast solution times, which are comparable to that of the Relaxed Problem.

Thus, we conclude that the QCP algorithm outperforms the other algorithms in terms of both solution quality and solution time. It produces optimal or near-optimal solutions in a few minutes of running time, and it therefore has the potential to be used in both the planning and real-time environments.

2.6.2 Case Study

In this section, we conduct a case study to illustrate how the crew scheduling model can be used to derive useful information and drive decision making at a railroad. We perform the case study on a representative two-week data set which has 326 trains, 2 crew pools, and 48 crews, and we run the computational tests using the Quadratic Cost Perturbation (QCP) algorithm. The various aspects of the problem that we observe in this case study are as follows: (i) Effect of varying the number of available crews, (ii) Effect of varying deadhead cost, (iii) Effect of varying minimum rest time at the home location, (iv) Effect of varying detention time, and (v) Effect of varying detention cost.

Effect of varying the number of available crews

In this study, we quantify the effect of varying the number of available crews on the overall solution quality. We start with a set of 42 available crews and reduce the number of crews available until the problem becomes infeasible. Table 2-2 presents the computational results, and Figure 2-4 plots a chart between the number of available crews and solution cost.

We draw the following observations from this study:

- As the number of available crews decreases, the model tries to compensate for the lack of crews by increasing the level of deadheading and train delays.
- Initially, reducing the number of available crews does not have an adverse effect on the solution cost, but as more crews are removed, the solution cost rises steeply. For example, reducing the number of crews available from 42 to 26 (38% reduction) has an insignificant impact on the solution cost, but reducing the number of crews from 24 to 22 leads to the solution cost jumping up by more than $59,000 (20% increase).

The reader should note that the objective function in this case study is not a function of the number of crews used and is only a function of total deadhead, detention and delay. This explains why solutions using different number of crews have identical cost provided their total deadhead, detention and delay are the same.

Effect of varying deadhead cost

In this study, we quantify the effect of varying deadhead cost on the number of deadheads, total detention hours, total train delay hours, and overall solution cost. The default cost of deadheading used by the railroad is $144 per hour. We start with a deadhead cost of $0 per hour and then progressively increase deadhead cost while measuring the impact on the solution as shown in Table 2-3.

We make the following observations from this study:

- As the deadhead cost increases, the number of deadheads in the solution decreases. However, after a certain point, there is no significant decrease in the number of deadheads. For example, even for a very high deadhead cost of $10,000, the solution has 33 deadheads. From this observation we can conclude that there is an inherent imbalance in the system that necessitates deadheading.
- As the deadhead cost increases, the solution of the model has fewer deadheads and more train delays. This behavior of the model gives us the insight that if the deadhead cost increases at some point in time, then the railroad needs to adapt by allowing far more flexibility in terms of the train delays. Alternatively, the management can also negotiate with crew unions and reduce the minimum rest hour requirements.

Effect of varying minimum rest time at the home location

In this study, we quantify the effect of varying the minimum rest time at the home location on the average rest time at the home location, train delays at the home location, average rest time at the away location, train delays at the away location, and the overall solution cost. The default value of minimum rest time used by the railroad is 12 hours. We start with a minimum rest requirement of zero hours and progressively increase the value of this parameter while measuring the impact on the solution as shown in Table 2-4 and Figure 2-5.

From this study we observe that the minimum rest time at home can be increased to 16 hours without a significant increase in the solution cost. However, any increase beyond 16 hours leads to a steep increase in the solution cost. The railroad management can use these inputs to effectively negotiate rest times with the union. For example, if the union wants the minimum rest

time to be increased from 12 hours to 14 hours, then the management can use the model to quantify the impact of this change and negotiate appropriately.

Effect of varying detention cost

The railroad pays detention charges for each hour of crew rest beyond 16 hours at an away location. In this study, we quantify the effect of varying the detention cost on the total detention hours, number of deadheads, total train delay hours, and overall solution cost as presented in Table 2-5. The default value of detention cost used by the railroad is $140 per hour.

We make the following observations from this study:

- As the detention cost per hour increases, the number of detention hours in the solution decreases.
- As the detention cost per hour increases, the solution has a greater number of deadheads and train delays. This behavior of the model gives us the insight that if the detention cost increases at some point in time, then the railroad needs to adapt by allowing more flexibility in terms of train delays and crew deadheading.

Effect of varying detention time on the solution

In this study, we quantify the effect of varying the detention time (the minimum rest time at the away location after which a crew becomes eligible for detention allowance) on the average rest time at the away location, detention hours, and overall solution cost. The default value of detention time provided by the railroad is 16 hours. We present our results in Table 2-6 and Figure 2-6.

This study shows that increasing the detention time has an impact on the solution cost, but it diminishes as the detention time increases. We observe that increasing the detention time from 0 to 20 hours reduces the solution cost, but increasing it beyond 20 hours has almost no impact on the solution quality.

2.7 Summary and Conclusions

In our research, we develop a network flow-based approach to solve the railroad CSP in the context of North American railroads. The CSP for North American railroads is governed by several Federal Railway Administration (FRA) regulations and trade-union work rules. In order to develop a good crew schedule, in addition to satisfying these regulations, we also need to minimize the total wage costs, train delay costs, deadhead costs, and detention costs. The railroad divides the network into a number of crew districts, and each crew district has several crew pools. Each crew pool at a district could have a different set of operating rules. These factors make this a complex problem to model and solve.

The network flow formulation for the CSP that we develop is both flexible and robust and can be easily manipulated to represent each of the possibilities encountered in real-life. We formulate the CSP as an integer program on a space-time network. The network is constructed in such a way that all FRA regulations and trade-union work rules other than FIFO constraints are enforced during the network construction phase itself. The operational constraints are handled in the integer programming formulation. We develop two approaches to handle FIFO constraints. The first approach is a Successive Constraint Generation approach where constraints are generated iteratively to cut out FIFO violations. The second approach, which is called Quadratic Cost Perturbation, relies on perturbing the objective function to generate FIFO-compliant solutions. We provide extensive computational results comparing the performance of various approaches and show that the perturbation approach outperforms the other approaches both in terms of solution time and solution quality.

We believe that this research will eventually lead to the deployment of crew planning models and algorithms at North American railroads, replacing the current manual process and, in

doing so, make a significant impact on the railroad's on-time performance, crew utilization, and productivity, while also improving the quality-of-life for crew and improving railroad safety.

Table 2-1. Comparison of algorithmic performance.

# of weeks	# of crew pools	Relaxed Problem			2. Exact Integer Programming Formulation				3. Successive Constraint Generation				4. Quadratic Cost Perturbation		
		Cost($)	Time (Sec)	# of FIFO constraints violated	Cost ($)	Time (Sec)	# of FIFO constraints		Cost($)	Time (Sec)	# of FIFO constraints		Cost ($)	Time (Sec)	# of FIFO constraints violated
							Added	Violated			Added	Violated			
1	1	130,952	10.8	73	-	3,600	11,062	N/A	132,022	2,015	958	25	130,952	10.9	0
2	1	265,284	30.3	148	-	3,600	23,527	N/A	267,067	1,981	1,492	95	265,284	31.4	0
3	1	399,816	57.2	225	-	3,600	35,976	N/A	399,816	1,908	1,657	151	399,816	60.0	0
4	1	531,378	91.8	274	-	3,600	48,797	N/A	532,091	2,326	1,805	226	531,378	97.4	0
1	2	132,495	17.7	64	-	3,600	17,999	N/A	132,495	347	478	0	132,495	17.7	0
2	2	267,130	55.6	118	-	3,600	40,623	N/A	267,316	2,423	1,068	0	267,221	60.9	0
3	2	402,045	112.0	173	-	3,600	63,215	N/A	405,227	4,858	1,321	25	402,678	125.0	0
4	2	533,694	187.3	226	-	3,600	86,477	N/A	538,039	3,928	1,745	25	534,327	210.7	0

Table 2-2. Effect of varying crew pool sizes.

Num. of crew available	Num. of crew used	Num. of deadheads	Detention hours	Train Delay hours	Solution cost ($)	Increase in cost
42	31	38	37.00	8.77	262,838	-
40	30	38	37.00	8.77	262,838	0
38	29	38	37.00	8.77	262,838	0
36	29	40	37.00	7.85	263,340	502
34	29	40	37.00	7.85	263,340	0
32	28	40	37.00	7.85	263,340	0
30	28	41	37.00	7.85	263,697	357
28	28	41	37.00	7.85	263,697	0
26	26	43	30.65	30.38	268,704	5,007
24	24	43	17.50	154.83	295,486	26,782
22	22	44	6.37	417.12	354,610	59,115
20	-	-	-	-	Infeasible	-

Table 2-3. Effect of varying deadhead cost.

Deadhead Cost/hr ($/hr)	Number of deadheads	Detention Hours	Train delay hours	Solution cost ($)
0	42	34.55	5.45	253,079
100	38	37.00	8.77	260,051
200	38	37.00	8.82	266,396
300	37	40.33	9.48	272,733
400	37	40.33	9.57	278,918
500	36	40.33	13.13	284,955
600	36	40.33	13.13	290,955
700	36	40.33	13.13	296,955
800	36	40.33	13.13	302,955
900	36	40.33	13.18	308,967
1,000	35	36.80	22.95	314,935
10,000	33	36.80	55.13	813,771

Table 2-4. Effect of varying minimum rest time at home location.

Minimum rest (hrs)	Average rest at home (hrs)	Train Delays at home (hrs)	Average rest at away (hrs)	Train Delays at away (hrs)	Solution cost ($)
0	10.02	0.00	12.83	8.77	262,838
2	11.60	0.00	12.99	8.77	262,838
4	12.86	0.00	13.12	8.77	262,838
6	14.37	0.00	13.12	8.77	262,838
8	15.16	0.00	13.15	8.77	262,838
10	16.81	0.00	13.31	8.77	262,838
12	18.51	0.00	13.33	8.77	262,838
14	20.48	0.00	13.25	8.77	262,838
16	21.53	0.07	13.09	8.77	262,853
18	23.98	1.23	13.18	9.52	263,294
20	28.33	3.78	13.09	17.40	265,337

Table 2-5. Effect of varying detention cost.

Detention cost/hr ($/hr)	Detention hours	Number of deadheads	Train delay hours	Solution cost($)
0	305.45	35	1.17	254,840
40	64.57	37	3.32	258,630
80	40.33	37	8.77	260,528
120	37.00	38	8.77	262,098
160	34.55	39	8.77	263,542
200	30.77	39	11.97	264,904
240	23.32	39	18.50	265,849
280	23.32	39	18.50	266,782
320	18.67	39	24.18	267,534
360	11.10	39	35.53	268,167
400	1.93	40	49.23	268,452
500	1.93	40	49.23	268,645
600	1.93	40	49.23	268,838
700	1.93	40	49.23	269,032

Table 2-6. Effect of varying detention time.

Detention time (hours)	Avg. rest at away location	Detention hours	Solution cost ($)
0	8.95	1,136.60	460,558
4	8.95	633.18	390,079
8	10.41	331.60	324,478
12	11.12	83.17	280,491
16	13.33	37.00	262,838
20	14.20	3.67	256,561
24	14.78	1.52	255,620
28	15.82	0.00	254,840

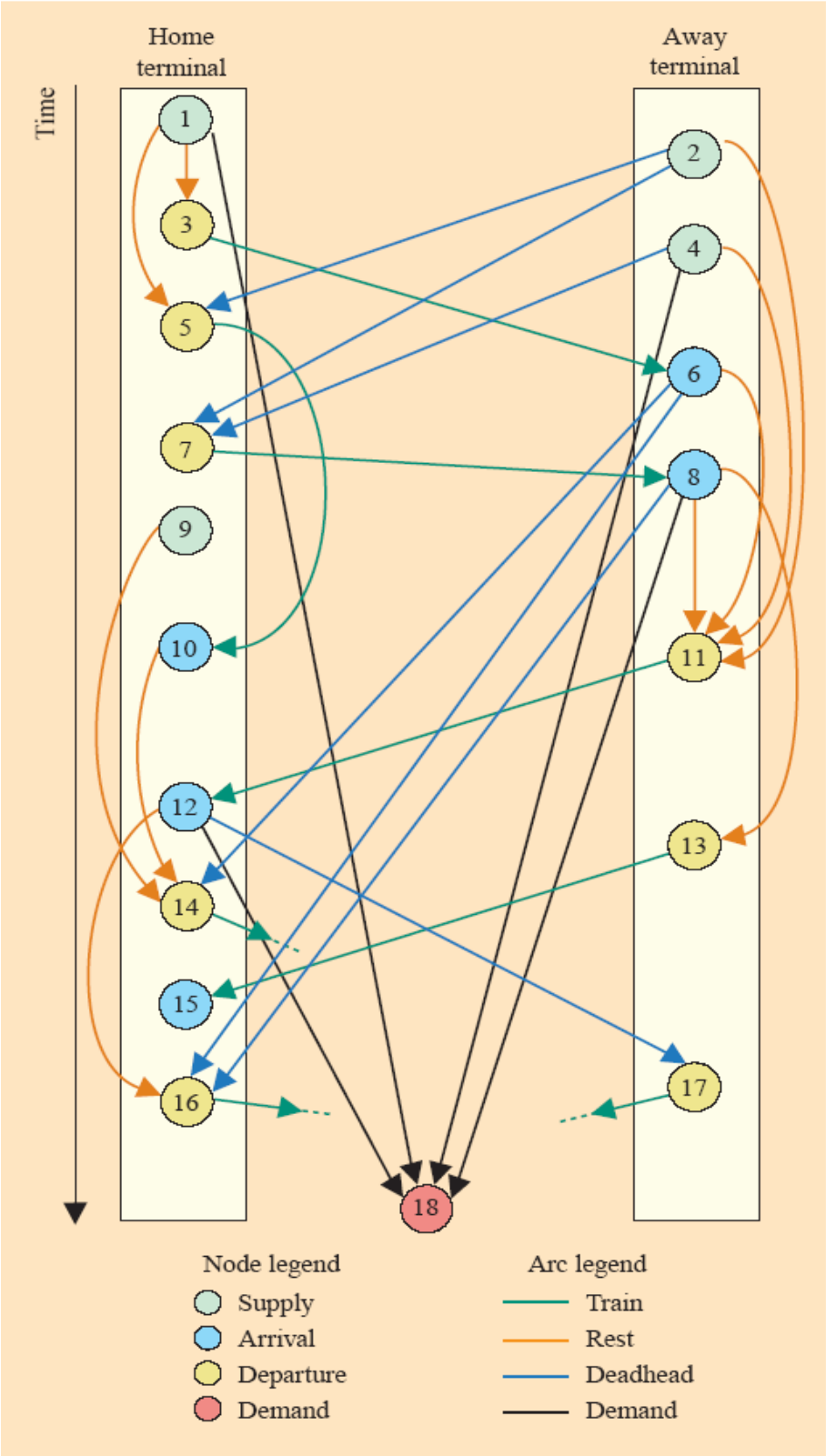

Figure 2-1. Space-time network for a single-ended district with a single crew type.

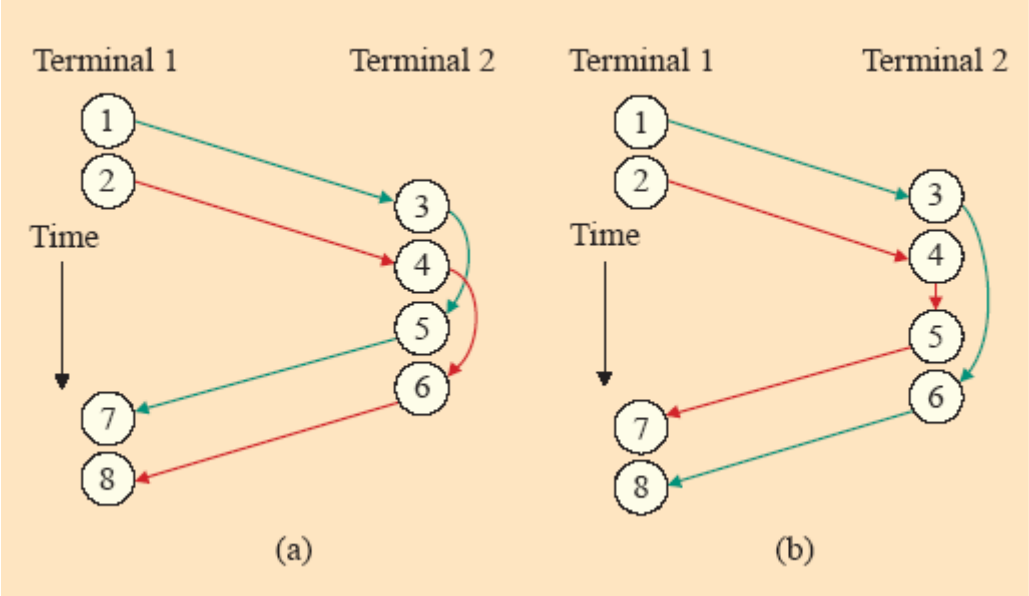

Figure 2-2. Illustration of the First-In-First-Out (FIFO) rule.

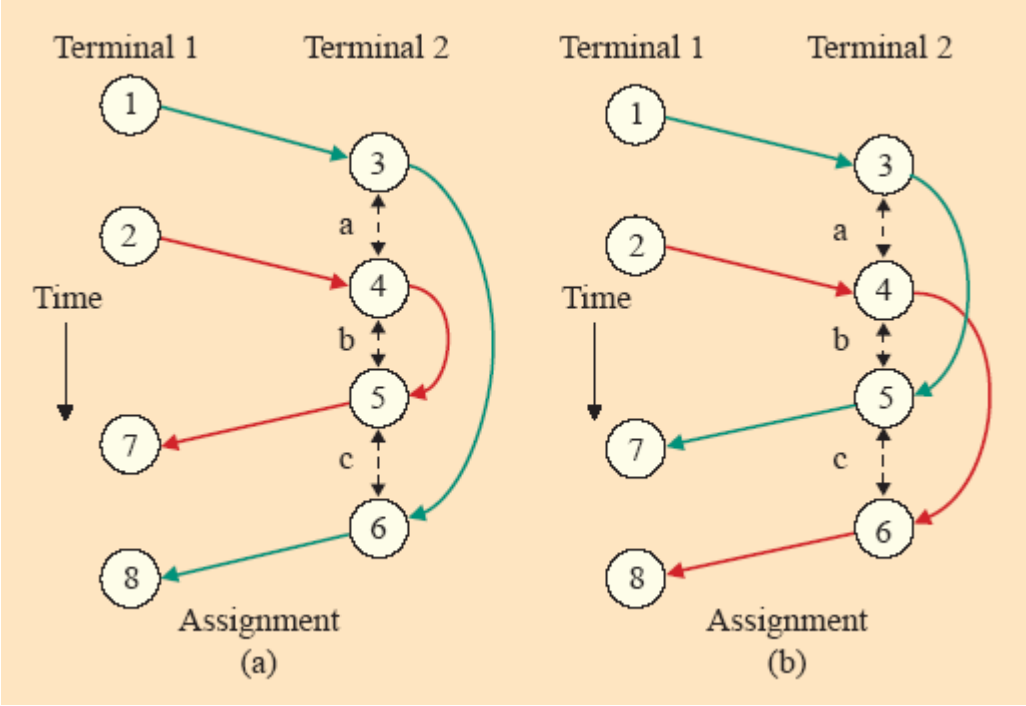

Figure 2-3. Crew assignments are made in (a) Non-FIFO manner, and (b) FIFO manner

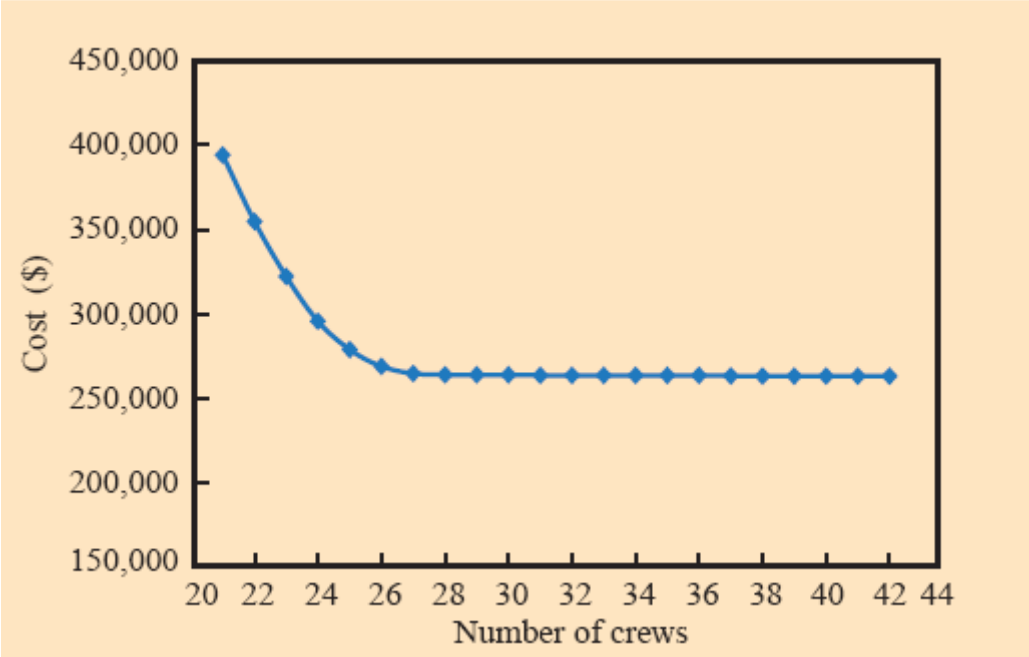

Figure 2-4. Solution cost vs. number of crews.

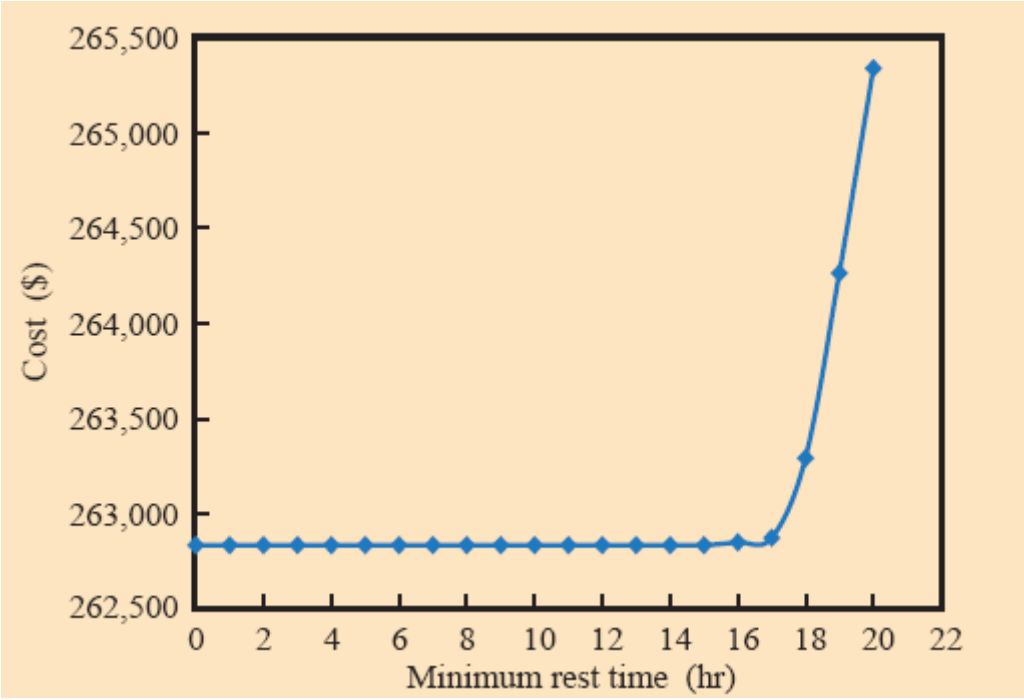

Figure 2-5. Solution cost vs. minimum rest time at home.

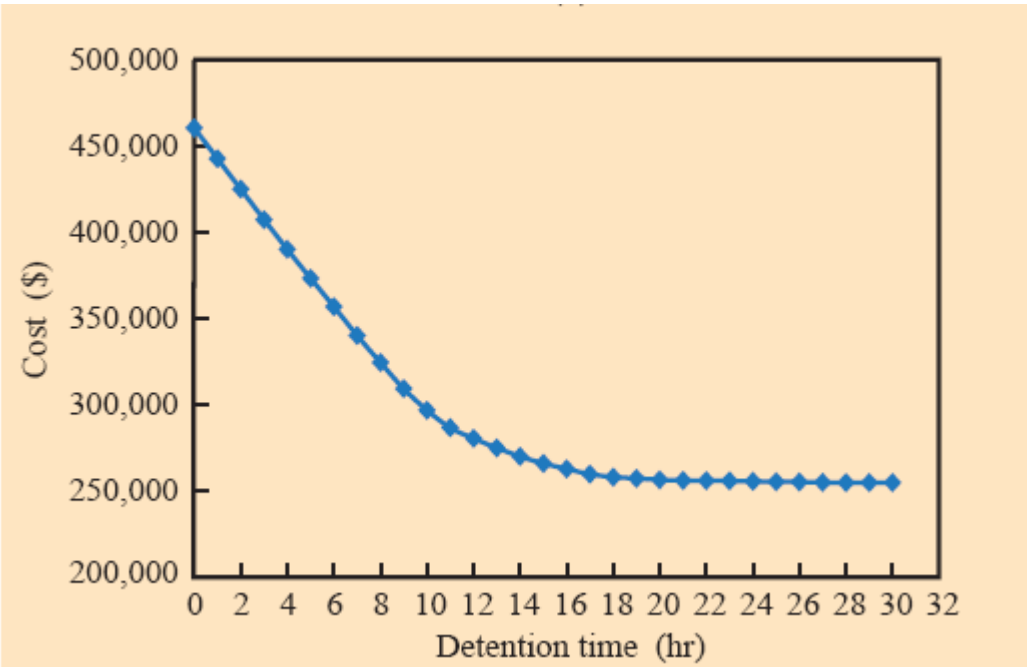

Figure 2-6. Solution cost vs. detention time.

CHAPTER 3
REAL-LIFE LOCOMOTIVE PLANNING: NEW FORMULATIONS AND COMPUTATIONAL RESULTS

3.1 Introduction

Locomotive scheduling involves optimally assigning a set of locomotives to trains satisfying a variety of business constraints and minimizing the total scheduling cost. Everyday, railroad managers must assign hundreds of locomotives to hundreds of different trains. These problems are notoriously difficult NP-hard (Ahuja et al. (2005)) transportation problems that have not yet been solved satisfactorily. But solving these problems effectively is of critical importance since a large US railroad has several billions of dollars of capital investment in locomotives. Our research is motivated by the need to develop effective and practical solution techniques to solve this complex optimization problem.

Locomotive scheduling problems arise at two levels: planning and dynamic assignment. A railroad has several locomotive types with different characteristics, such as, horsepower, tonnage pulling capacity, and costs. The locomotive planning problem (LPP) assigns sets of locomotives, called consists, to each train in a pre-planned weekly train schedule so that each train gets sufficient power to pull its load and the total cost of locomotive usage is minimized. The resulting plan must honor a variety of business rules, cannot require more locomotives than what is available in the total fleet, and must result in a plan that is relatively simple and repeatable. Another important feature of the LPP is that locomotives may deadhead on trains. Deadhead locomotives do not pull the train but are repositioned by the active locomotives from one place to another place. Deadheading plays an important role in locomotive planning, enabling extra locomotives to be moved from surplus locations to locations where locomotives are in short supply. Locomotives also light travel; that is, a set of locomotives forms a group, and one locomotive in the group pulls others from an origin station to a destination station. Light travel is

not limited by the train schedule thus making it much faster than deadheading. However, light travel is more costly as a crew is required and the move does not generate any revenue as there are no cars attached. A locomotive plan specifies which types of locomotives will pull each train, and how locomotives will deadhead or light travel to obtain overall network wide efficiency. The locomotive plan gives the "ideal locomotive assignment" in the absence of any disruptions, and this plan is further refined and adjusted in real-time to obtain the dynamic locomotive assignment. We focus on the planning version of the problem.

LPPs are notoriously hard combinatorial optimization problems. The paper by Cordeau, Toth, and Vigo (1998b) presents an excellent survey of existing locomotive planning models and algorithms for this problem. There are two kinds of locomotive planning models: Single and Multiple. Single locomotive planning models assume that there is only one type of locomotive available for assignment. These models can be formulated as minimum cost flow problems with side constraints; some papers on single locomotive planning models are due to Wright (1989), Forbes, Holt, and Watts (1991), Booler (1980, 1995), and Fischetti and Toth (1997). Single locomotive assignment models may be appropriate for some European railroads but are not suited for US railroads since most trains are assigned multiple types of locomotives. Such multiple locomotive assignment models have been studied by Florian et al. (1976), Ramani (1981), Smith and Sheffi (1988), Chih et al. (1990), Nou, Desrosiers, and Soumis (1997), Cordeau, Soumis, and Desrosiers (1998a), and Ziarati et al. (1997, 1999). The most comprehensive and recent multiple locomotive assignment models are due to Ahuja et al. (2005) which is the starting point of our current research.

In our past research, described in Ahuja et al. (2005), we formulate the LPP as a mixed integer multicommodity flow problem with side constraints on an underlying network and

develop solution methodologies that use problem decomposition, integer programming, greedy heuristics, and neighborhood search (Refer to Section 3.2.4 for an overview). About two years ago, we started working with the locomotive division at CSX Transportation (Class I US railroad) to get the locomotive flow-based model developed by Ahuja et al. (2005) implemented. The locomotive division was very interested but wanted some additional features to be added to our methodology to ensure that the solution is more implementable. Also, we were able to find instances of the LPP where the first approach did not yield feasible solutions in several hours of computational time. In order to eliminate these difficulties, we developed robust methods described to obtain realistic solutions that can be used for practical locomotive planning. Our major contributions are listed below:

- **Robust Approach:** We were able to find instances for the LPP where we could not obtain a feasible solution in over 10 hours of computational time using the locomotive flow-based formulation (Ahuja et al. (2005)). In our computational results, we show that using the consist-flow formulation, we can obtain near-optimal solutions to the same instances within a few minutes of computational time.

- **Minimize Consist Bustings:** Consist-busting involves mixing of locomotives from inbound trains and regrouping them to form new consists. This is undesirable due to several reasons described in Section 3.3.2. We propose a consist-based formulation where consists flow over the network instead of individual locomotives. This formulation reduces consist busting to zero. We also develop a second formulation called the hybrid formulation, which is used to benchmark the performance of the consist-based formulation.

- **Incremental Locomotive Planning**: Zero based approaches which develop a locomotive plan from scratch are almost impossible to implement. We describe how we can use the models developed for planning incrementally.

- **Cab Signal Requirements**: Federal law requires that trains operating in certain territories called cab signal territory must have a lead locomotive that is outfitted with the cab signal system. We describe a method to incorporate the cab signal requirements in our formulations.

- **Foreign Power Requirements**: Different railroads in North America often cooperate with one-another and allow direct trains, called run-through-trains, from the origin station on one railroad to the destination station on another railroad in order to reduce cost by eliminating unnecessary intermediate switching at the interchange stations. We describe a

method to structure the model to allow locomotives to go to and come from foreign railroads.

- **Computational Tests:** We present extensive computational results and case studies on several instances and derive meaningful conclusions.

Though our research has been conducted in collaboration with CSX Transportation, the models proposed are applicable in general (with possible minimal contextual changes) to all North American railroads.

3.2 Notation and Terminology

We give the notation and terminology related to the LPP. We also describe the constraints in terms of these notations. The constraints can be classified into two groups: hard constraints (which each locomotive plan must satisfy) and soft constraints (which are highly desirable but not always required to be satisfied).

3.2.1 Notation

Locomotives pull a set of trains from their origins to their destinations. The train schedule is a weekly cyclic schedule. Trains have different weekly frequencies; some trains run every day, while others run less frequently. We consider the same train running on different days as different trains; that is, if a train runs five days a week, we will consider it as five different trains for which all data is the same except that they will have different departure and arrival time on the seven day clock. Each train has a required horsepower (HP) and tonnage. Tonnage represents the minimum pulling power required to pull the train and the HP has a direct bearing on the speed at which the train can travel. Each train l has the following attributes associated with it:

dep-time(l): The departure time of train l

arr-time(l): The arrival time of train l

dep-station(l): The departure station of train l

arr-station(l): The arrival station of train l

T_l : Tonnage requirement for train l

β_l : HP per tonnage requirement for train l

E_l : The penalty for using single locomotive consist on train l

A railroad also has several types of locomotives. Let K denote the set of all locomotive types and k a particular locomotive type in this set. Locomotives which provide power and pull trains are called active locomotives and those which are being repositioned on a train are called deadheading locomotives. For every $k \in K$, we have the following attributes:

h^k : HP of a locomotive of type k

α^k : Number of axles on a locomotive of type k

G^k : Ownership cost of a locomotive of type k

B^k : Fleet-size of a locomotive of type k

c_l^k : Cost of assigning an active locomotive of type k to train l

d_l^k : Cost of deadheading a locomotive of type k on train l

t_l^k : Tonnage provided by a locomotive of type k to train l

The tonnage provided by a locomotive k depends on the train to which it is assigned. This is because different trains have different ruling grades or slopes and the pulling power of a locomotive depends on the ruling grade. Each train l has three sets of locomotives that could be assigned to it. MostPreferred[l] contains the locomotive types which are preferred on this train, LessPreferred[l] contains the locomotive types which are accepted on this train with a certain penalty and Prohibited[l] contains the locomotive types which are not allowed on this train. Locomotives are assigned to these three groups for each train based on the business rules of the railroad.

Light travel refers to the travel of locomotives on their own from one location to another. Light travel is a fast though expensive way to reposition locomotives. Each light travel has a fixed cost which is a function of the distance of light travel (since we need a crew) and it also has a variable cost based on the number of locomotives light traveling and the duration of light travel.

Consist is a collective term for the set of locomotives assigned to a train. Consist busting refers to the breaking up of an incoming consist at a station and the assignment of the locomotives in it to more than one outgoing train. Consist busting has a fixed cost B associated with it and also a variable component which depends upon the number of locomotives involved in consist busting. An efficient locomotive schedule can help us reduce consist busting greatly. A train-train connection refers to the direct transfer of a consist from an inbound train to an outbound train at the same station. Train-train connections are highly desirable since they reduce consist busting. Whenever a consist goes from one train to another train, it needs some minimum time to make this connection; we refer to this time as the minimum connection time.

3.2.2 Hard Constraints

Hard constraints are mandatory constraints which have to be satisfied for a locomotive assignment plan to be feasible.

Power requirement for trains: Each train must be assigned locomotives with at least the required tonnage and horsepower.

Locomotive class to train type: Each train type (e.g., auto train, or merchandise train, or intermodal train) is targeted to use specific classes of locomotive types. For example, we may specify that auto trains can be assigned only AC44 or AC60 type locomotives.

Geographic: Each geographic region permits the travel of only specific locomotive types. For example, we may specify that Atlanta can only use: C40, AC44, and AC60 locomotives.

Locomotive balance constraints: The number of incoming locomotives of each type into a station must be equal to the number of outgoing locomotives of that type at that station.

Active axle constraints: Each train must be assigned locomotives with at most 24 active axles. This business rule is designed to protect the standard couplers used in North America. Exceeding 24 powered axles may overstress the couplers and cause a train separation.

Consist size constraints: Each train can be assigned at most 12 locomotives including both the active and deadheading locomotives. This business policy reduces risk exposure if the train were to suffer a catastrophic derailment.

Fleet size constraints: The number of assigned locomotives of each type is at most the number of available locomotives of that type.

Repeatability of the schedule: The assignment of locomotives to trains should be such that the number of locomotives of each type at each station at the end of the week should be equal to the number of locomotives of each type at each station at the beginning of the next week so that the plan is repeatable every week.

3.2.3 Soft Constraints

These constraints are flexible constraints and they define characteristics which are preferred but not mandatory.

Consistency in locomotive assignment: A train should be assigned the same consist each day that it runs. Railroads believe that crews will perform more efficiently and more safely if they operate the same equipment on a particular route and train. As the crews learn the operating nuances associated with each combination, they will adjust their throttle and braking control accordingly.

Consistency in train connections: If locomotives traveling on a train to its destination station connect to another train originating at that station, then they should preferably make the

same connection on each day that both of the trains run. This allows station managers to optimize their sub-processes associated with arriving and departing trains.

Avoid consist busting: Since consist busting involves the use of more resources, it is preferable to avoid consist busting.

Avoid single locomotive consists: If a single locomotive is assigned to a train and this locomotive breaks down, then the train will get stranded.

3.2.4 Previous Research

In the previous research reported in Ahuja et al. (2005), we developed an integrated planning model that determines the set of active and deadheading locomotives for each train, light traveling locomotives from power sources to power sinks, and train-to-train connections (specifying which inbound train and outbound trains can directly connect). We first developed a mixed integer programming (MIP) formulation of the LPP. This MIP formulation, the locomotive flow model, is a multicommodity flow problem where each locomotive type represents a commodity and assignment of locomotives to trains is transformed into the flow of locomotives on a seven-day or weekly space-time train network. There were two major difficulties in solving the MIP formulation. First, the problem size was too large to be solved by the available commercial-grade MIP software as it contains hundreds of thousands of decision variables and comparable number of constraints. Secondly, the solution obtained using this formulation was not likely to be consistent; that is, the optimal solution may have different consist assignments to the same train on different days of the week. In a plan, railroads want the same consist assignment to a train each day it runs.

Our major contribution in the previous research was to handle both these difficulties effectively by proposing a two-phase approach. In the first phase, we assumed that trains run every day of the week, and determined the locomotive assignments under this assumption. (This

assumption is not a bad assumption in practice as most trains run every day). In the second phase, we considered the actual train schedule and re-optimized the locomotive assignment of the first phase by solving a sequence of single commodity flow problems. Assuming that trains run every day of the week reduces the weekly locomotive assignment problem to a daily (or one day) locomotive assignment problem (thus reducing problem size by a factor of seven thereby making it more tractable) and replicating the locomotive assignment each day of the week gives us a perfectly consistent assignment. We give a brief overview of this two-phase approach in Figure 3-1.

We formulate the problem is such a way that determining train-train connections is no longer a modeling issue. We focus our attention on the one-day LPP. While keeping the other features of the model identical to what they were before; we change our approach to model the one day LPP and consider flows of consists over the network instead of flows of individual locomotives. In the next section, we give a detailed description and motivation for the consist-based approach. In the rest of the chapter when we refer to LPP, we are referring to the one-day LPP.

3.3 Mathematical Modeling

3.3.1 Space-Time Network

The LPP is formulated as an integer multicommodity flow problem with side constraints on a space-time network. Each consist type defines a commodity in this network. We denote the space-time network as G (N, A), where N denotes the node set and A denotes the arc set. The space-time network is constructed as follows. We create a train arc (l', l'') for each train l; the tail node l' of the arc denotes the event for the departure of train l at dep-station(l) and is called a departure node. The head node l'' denotes the arrival event of train l at arr-station(l) and is called an arrival node. Each arrival or departure node has two attributes: place and time. For example,

place(l') = dep-station(l) and time(l') = dep-time(l). Similarly, place(l'') = arr-station(l) and time(l'') = arr-time(l). Some trains are called forward trains and some trains are called backward trains. Forward trains are those trains for which dep-time(l) < arr-time(l) and backward trains are those trains for which dep-time(l) > arr-time(l).

To allow the flow of locomotives from inbound trains to outbound trains, we introduce ground nodes and connection arcs. For each train arrival node, we create a corresponding arrival-ground node with the same place and time attribute as that of the arrival event. Similarly, for each train departure event, we create a departure-ground node with the same place and time attribute as that of the departure node. We connect each arrival node to the associated arrival-ground node by a directed arc called the arrival-ground connection arc. We connect each departure-ground node to the associated departure node through a directed arc called the ground-departure connection arc. Next, we sort all the ground nodes at each station in the chronological order of their time attributes, and connect each ground node to the next ground node in this order through directed arcs called ground arcs. (We assume without any loss of generality that ground nodes at each station have distinct time attributes) The ground nodes at a station represent the pool (or storage) of locomotives at the station at different instants of times. As trains arrive, they bring in locomotives to the pool through arrival-ground connection arcs. As trains depart, they draw locomotives from the pool through ground-departure connection arcs. The ground arcs allow inbound locomotives to stay in the pool as they wait to be connected to the outbound trains. We also connect the last ground node of the week at a station to the first ground node of the week at that station through a ground arc; this ground arc models the ending inventory of locomotives for a day becoming the starting inventory for the following day.

We also allow the possibility of light travel, that is, several locomotives forming a group and traveling on their own from one station to another. We assume that the set of light travel possibilities is given to us. We create a light arc in the weekly space-time network corresponding to each light travel possibility. Each light arc originates at a ground node (with a specific time and at a specific station) and also terminates at a ground node. Each light arc has a fixed charge which denotes the fixed cost incurred to assign a crew to that light travel. We denote this fixed charge for a light travel arc l by F_l. The light arc also has a variable cost which depends upon the number of entities light traveling on it.

The space-time network G = (N, A) has three types of nodes - arrival nodes (ArrNodes), departure nodes (DepNodes), and ground nodes (GrNodes); and four kinds of arcs - train arcs (TrArcs), connection arcs (CoArcs), ground arcs (GrArcs), and light travel arcs (LiArcs). Let AllNodes = ArrNodes $\cup$ DepNodes $\cup$ GrNodes, and AllArcs = TrArcs $\cup$ CoArcs $\cup$ GrArcs $\cup$ LiArcs. In Figure 3-2, we show a part of the space-time network at a particular station, which illustrates the various kinds of arcs.

3.3.2 Consist Flow Formulation for the LPP

Consist busting refers to the process of decomposing inbound consists at a station into several locomotives and the assignment of these locomotives to several outgoing trains. Railroad managers prefer to minimize consist-busting as much as possible due to the following reasons. (1) Consist-busting often results in outbound trains getting their locomotives from several inbound trains. If any of these inbound trains is delayed, the outbound train is also delayed, which potentially propagates further delays down the line; and (2) Consist busting is an extremely labor, cost, and time intensive operation. Additional crews are required to transfer (or hostle) locomotives between the rail yard and the shop facility where consists are busted and

reassembled. Moving locomotives across a station creates incremental moves for the yard masters to worry about; the hostlers block tracks so car switching activity must be paused; and train movements must be coordinated between transportation and mechanical departments (as opposed to train-to-train connections which are entirely within the purview of transportation). Consequently, consist busting consumes between two to six additional hours per locomotive within the station.

We develop a formulation for LPP that almost completely eliminates consist busting; we call this formulation the consist flow formulation. This formulation is an extension of the locomotive flow formulation described in Ahuja et al. (2005), which defines each locomotive type as a commodity and routes locomotives on the train network. In the consist flow formulation, we define each consist type (that is, a group of locomotives) to be a single commodity and route consists on the train network. Thus, the consist flow formulation differs from the locomotive flow formulation in the sense that locomotive types are replaced by the consist types. Observe that each feasible solution of the consist flow formulation has a corresponding feasible solution of the locomotive flow formulation with the same cost, but the converse is not true. Thus, the reader might be led to believe that consist formulation is quite restrictive and may produce significantly inferior solutions compared to the flow formulation. However, our computational results reveal that if the number and types of consists are judiciously chosen, then both formulations produce solutions with comparable quality. In the formulation described next, we assume that the consist types are pre-specified; we describe later in Section 3.3.4 how to determine good consist types.

The consist flow formulation is a multicommodity integer program formulated on the space-time network. We next describe the additional notations used in this formulation.

C: Denotes the set of consist types available for assignment and $c \in C$ represent a particular consist.

F_l : Fixed charge cost for using the light arc l.

c_l^c : Cost of assigning an active consist of type $c \in C$ to train l.

d_l^c : Defined for every arc $l \in AllArcs$. For a train arc $l \in TrArcs$, d_l^c captures the cost of deadheading a consist of type $c \in C$ on arc l. For a light arc $l \in LiArcs$, d_l^c captures the cost of traveling for a consist type $c \in C$ on arc l. For an arc $l \in CoArcs \cup GrArcs$, d_l^c captures the cost of idling for a consist type $c \in C$ on arc l.

α^{ck} : Number of locomotives of type $k \in K$ in consist type $c \in C$.

I[i]: Set of arcs entering node i.

O[i]: Set of arcs leaving node i.

S:Set of overnight arcs or arcs that cross the midnight timeline. (We choose midnight as the time for counting the number of locomotives used in our solution)

day(l): Number of times an arc l crosses the midnight timeline. Suppose, a train l leaves at 8 AM on day 1 and arrives at 2 AM on day 2, then day(l) = 1; whereas suppose it leaves at 8 AM on day 1 and arrives at 2 AM on day 3, then day(l) = 2.

The decision variables are:

x_l^c : Binary variable representing the number of active consists of type $c \in C$ on arc $l \in TrArcs$.

y_l^c : Integer variable representing the number of non-active consists (deadheading, light-traveling or idling) of type $c \in C$ on arc $l \in AllArcs$.

z_l : Binary variable which takes value 1 if at least one consist flows on arc $l \in LiArcs$ and 0 otherwise.

s^k : Integer variable indicating the number of unused locomotives of type $k \in K$.

We formulate the problem as follows:

$$\min z = \sum_{l \in TrArcs} \sum_{c \in C} c_l^c x_l^c + \sum_{l \in AllArcs} \sum_{c \in C} d_l^c y_l^c + \sum_{l \in LiArcs} F_l z_l - \sum_{k \in K} G^k s^k \quad (3.1a)$$

$$\sum_{c \in C} x_l^c = 1, \text{ for all } l \in TrArcs, \quad (3.1b)$$

$$\sum_{c \in C} \sum_{k \in K} \alpha^{ck} (x_l^c + y_l^c) \le 12, \quad \text{for all } l \in TrArcs, \quad (3.1c)$$

$$\sum_{l \in I[i]} (x_l^c + y_l^c) = \sum_{l \in O[i]} (x_l^c + y_l^c), \quad \text{for all } i \in AllNodes, c \in C, \quad (3.1d)$$

$$\sum_{k \in K} \sum_{c \in C} \alpha_{ck} y_l^c \le 12 z_l, \quad \text{for all } l \in LiArcs, \quad (3.1e)$$

$$\sum_{l \in S} \sum_{c \in C} \alpha^{ck} (x_l^c + y_l^c) day(l) + s^k = B^k, \quad \text{for all } k \in K, \quad (3.1f)$$

$$x_l^c \in \{0,1\}, \quad \text{for all } l \in TrArcs, c \in C, \quad (3.1g)$$

$$y_l^c \ge 0 \text{ and integer}, \quad \text{for all } l \in AllArcs, \ c \in C, \quad (3.1h)$$

$$z_l \in \{0,1\}, \quad \text{for all } l \in LiArcs, \quad (3.1i)$$

$$s^k \ge 0, \quad \text{for all } k \in K. \quad (3.1j)$$

Constraint (3.1b) ensures that every train l is assigned exactly one active consist. Constraint (3.1c) ensures that the locomotive flow upper-bound on each train arc is satisfied. Constraint (3.1d) ensures that flow is balanced at every node for every consist type. Constraint (3.1e) ensures that the locomotive flow upper-bound on each light arc is satisfied. Constraint (3.1f) ensures that the number of locomotives of each fleet type used is no more than the fleet size.

While the locomotive flow based formulation (Ahuja et al. (2005)) did not converge to feasible solutions for some instances in over 10 hours of computational time, the consist based

formulation was able to optimally solve the same instances within a few minutes of computational time. The reasons for this superior computational performance are: (1) In the locomotive flow formulation, we need to explicitly specify constraints that each train gets required tonnage, horsepower and does not exceed the 24-active axle requirement. However, these constraints are implicitly handled in the consist-based formulation using much lesser number of constraints. We handle the active axle constraint by not creating consists which have more than 24 active axles. We handle the tonnage and horsepower constraints implicitly in the following way; if assigning consist $c \in C$ as an active consist to train $l \in TrArcs$ violates the tonnage or horse power constraints, we set $x_l^c = 0$ thus disallowing the assignment of consist c to train l. Thus, the consist flow formulation has 2 |TrArcs| lesser number of side constraints compared to the locomotive flow formulation; (2) All active consist assignment variables, x_l^c, are binary variables, whereas in the locomotive flow formulation active flow variables are general integer variables. The MIP optimization engine, which is typically a branch-and-bound algorithm, is likely to explore lesser branches for a binary integer program compared to a general integer program as there are lesser options to pursue; and (3) Each time branching occurs on a binary variable it is either pre-fixed to zero or to one in the subsequent problems. Hence, as the algorithm progresses down the branch-and-bound search tree, the problem size and complexity of the linear programs solved at each node reduces.

The consist based formulation also has practical advantages. Railroads often impose complex rules on what locomotive types may be combined into ideal consists. Some locomotives do not work well together. Some railroads segregate AC powered locomotives and DC powered locomotives. These requirements are often very hard or impossible to honor in the locomotive flow formulation but are rather trivial in the consist flow formulation. Further, in the locomotive

flow formulation, an outbound train often obtains its planned consist from locomotives coming from multiple trains and if any of these inbound trains is delayed, the outbound train is delayed as well. But in the consist flow formulation, all outbound trains derive their active consist only from one inbound train (but may derive their deadhead consists from other trains) thus reducing the impact of train delays.

The consist flow formulation is superior to the previous locomotive flow formulation because (1) solution speed and robustness are greatly improved, (2) consist busting is reduced to zero, and (3) constraints are more easily incorporated, resulting in more practical solutions. Our computational tests show that while the consist flow formulation may have its optimal objective function value as much as 5% higher than that of the locomotive flow formulation, the solution is far superior in terms of consistency, simplicity, and robustness. Thus, it may be easier to comply with and may need overall fewer locomotives in practice (considering train delays, for example). And, the consist flow solution still requires far fewer locomotives than the manually generated plans currently in use by most railroads.

3.3.3 Hybrid Formulation for the LPP

In order to evaluate the performance of the consist based formulation, we need to ideally compare it to the solution of the flow formulation (Ahuja et al. (2005)). However, for some instances, the flow formulation does not converge to a feasible solution after several hours of computational time and this is hence not possible. We develop a hybrid formulation which can be used to benchmark the performance of the consist formulation. The hybrid formulation retains the structural properties of the consist formulation while also allowing total flexibility to bust consists at no additional cost. We realize that this is not a realistic assumption, but this model allows us to determine the impact on solution cost if consist busting is allowed. Hence, it serves

as a good benchmark for the solution of the consist flow formulation. Also, since it retains some of the structural properties of the consist formulation, it is also computationally robust.

In the hybrid formulation, we allow active locomotives on trains to travel as consists but the deadheading locomotives travel as individual locomotives. At the ground nodes, where inbound trains bring their consists, they are broken into individual locomotives. These individual locomotives flow on ground arcs. When outbound trains need active locomotives, they draw only the pre-specified consists from the ground pool; but when they need deadhead locomotives, they can draw individual locomotives. The reader can observe that the solution space of the hybrid formulation is a subset of the solution space of the locomotive flow formulation but a super set of the solution space of the consist flow formulation. We next give the MIP formulation of this hybrid version.

The decision variables are:

x_l^c : Binary variable representing the number of active consists of type $c \in C$ on arc $l \in TrArcs$.

y_l^k : Integer variable representing the number of non-active locomotives (deadheading, light-traveling or idling) of type $k \in K$ on arc $l \in AllArcs$.

z_l : Binary variable which takes value 1 if at least one locomotive flows on arc $l \in LiArcs$ and 0 otherwise.

s^k : Integer variable indicating the number of unused locomotives of type $k \in K$.

We formulate the problem as follows:

$$\min z = \sum_{l \in TrArcs} \sum_{c \in C} c_l^c x_l^c + \sum_{l \in AllArcs} \sum_{k \in K} d_l^k y_l^k + \sum_{l \in LiArcs} F_l z_l - \sum_{k \in K} G^k s^k \qquad (3.2a)$$

$$\sum_{c \in C} x_l^c = 1, \text{ for all } l \in TrArcs, \qquad (3.2b)$$

$$\sum_{c \in C} \sum_{k \in K} \alpha^{ck} x_l^c + \sum_{k \in K} y_l^k \le 12, \quad \text{for all } l \in TrArcs, \qquad (3.2c)$$

$$\sum_{c \in C} \alpha^{ck} x_l^c + y_l^k = \sum_{h \in O(head(l))} y_h^k, \quad \text{for all } l \in TrArcs, k \in K, \tag{3.2d}$$

$$\sum_{c \in C} \alpha^{ck} x_l^c + y_l^k = \sum_{h \in I(tail(l))} y_h^k, \quad \text{for all } l \in TrArcs, k \in K, \tag{3.2e}$$

$$\sum_{l \in I(i)} y_l^k - \sum_{l \in O(i)} y_l^k = 0, \quad \text{for all } i \in GrNodes, k \in K, \tag{3.2f}$$

$$\sum_{k \in K} y_l^k \leq 12 z_l, \quad \text{for all } l \in LiArcs, \tag{3.2g}$$

$$\sum_{l \in S} \sum_{c \in C} \alpha^{ck} x_l^c day(l) + \sum_{l \in S} y_l^k day(l) + s^k = B^k, \quad \text{for all } k \in K, \tag{3.2h}$$

$$x_l^c \in \{0,1\}, \quad \text{for all } l \in TrArcs, c \in C, \tag{3.2i}$$

$$y_l^k \geq 0 \text{ and integer}, \quad \text{for all } l \in AllArcs, \ k \in K, \tag{3.2j}$$

$$z_l \in \{0,1\}, \quad \text{for all } l \in LiArcs, \tag{3.2k}$$

$$s^k \geq 0, \quad \text{for all } k \in K. \tag{3.2l}$$

The constraints in this formulation are similar to those in the consist flow formulation. The only difference is we have to construct the flow balance equations for train arrival and departure nodes (3.2d) and (3.2e) differently (with multipliers) because active locomotives flow as consists on trains and locomotives flow as individual locomotives on the connection arcs and ground arcs. In this formulation also, we enforce the tonnage, active axles and the horse power constraints implicitly and hence share some of the advantages that the consist formulation has over the flow formulation.

The computational results presented in Section 3.5.1 show that consist flow formulation gives comparable solutions to the hybrid formulation, thereby demonstrating that disallowing consist busting does not come at a big cost. We also observe that hybrid formulation takes much more time compared to the consist flow formulation as the flexibility to break and reassemble

consists, increases the solution space significantly and the decision variables have multipliers associated with them which makes the MIP harder to solve.

3.3.4 Identifying Good Consists

During our computational testing of the consist-based and hybrid formulations, we observed that the performance critically depends on the number and types of consists. The greater the number of consists with different horsepower and tonnages, the better the quality of the solution since there are more potential matches between need and availability. However, from implementation perspectives, we would like to use the minimum number of consists as it keeps the solution simple and provides greater opportunities for substitution between different consist types which is done when trains run late. We are thus faced by the problem of finding a small set of good consist types which provide as good a solution as a large number of consist types. We show that using a minor change in our existing formulations, we can choose a small set of p good consist types from a given larger superset of potential consist types C. We accomplish it by introducing the following decision variables:

z_c = Binary variable which takes a value 1 if consist type $c \in C$ is used, and a value 0 otherwise

and add the following constraints to the formulations:

$$\sum_{l \in S} (x_l^c + y_l^c) \leq M z_c, \quad \text{for all } c \in C,$$

$$\sum_{c \in C} z_c = p,$$

where M is a sufficiently large number. The first constraint sets $z_c = 1$ whenever a consist type is used, and the second constraint ensures that no more than p such variables are set to 1, or no more than p consist types are used. These constraints will ensure that the model picks a set of p optimal consists from the pre-defined superset of consists C that minimizes the overall solution cost.

3.4 Incorporating Practical Requirements

We describe several additional requirements that a locomotive planning model must satisfy which we did not incorporate in our previous research reported in Ahuja et al. (2005). These features are necessary to generate an implementable locomotive plan.

3.4.1 Incremental Locomotive Planning

Our formulations described in Section 3.3 generate a locomotive plan for a given train schedule from scratch or zero-base. However, over time, train schedules change in response to shifts in traffic patterns, to free up tracks for periodic repair and replacement, and because of network outages such as hurricanes, floods, and winter blizzards. These changes include some trains being added or removed, or changes in train frequencies, train timings, and train tonnages. We can use our locomotive planning model to generate a brand new optimal locomotive plan for the modified train schedule. But, the former plan and the new plan could potentially be quite different requiring managers to reposition power from the old cycles to the new cycles causing much disruption to operations. Consequently, an additional challenge to locomotive planning is the development of implementation strategies for introducing new plans to an organization. Our research shows that the optimal (or near-optimal) frontier of solutions for the LPP is quite broad and quite flat, which means that there are a very high number of combinations of locomotive plans that produce similar values for the model objective function. Consequently, it would be desirable to choose a new locomotive plan that closely resembles the current locomotive plan, minimizing the number of tactical repositioning moves required to transition from one plan to another.

The incremental locomotive planning algorithm takes in the current locomotive plan and changes in the train schedule, and produces a new, incrementally different locomotive plan. The current weekly train schedules and their currently planned locomotive assignments are inputs to

this problem. Since the current plan need not conform to pre-specified consists, this model is a locomotive flow based model. The algorithm consists of the following steps:

- Take the current weekly plan and obtain a current daily plan. The trains which are considered a part of the daily problem are those trains which have a frequency greater than p (usually p = 4). The current daily flows for these trains are taken to be the most frequent locomotive assignment on the seven-day trains corresponding to this daily train in the current solution. For example, if a train is assigned an active flow of 2 SD40 for five days of a week and an active flow of 2 AC44 for one day, then the corresponding daily train will be assigned a flow of 2 SD40 in the current daily plan.
- Solve the incremental one-day flow formulation.
- Use the output of the incremental one-day flow problem and transition back to the seven-day problem by using the standard approach suggested in Ahuja et al. (2005). Since consistency is enforced while moving from the one-day to the seven-day solution; the incremental model's output is likely to be similar to the current weekly plan.

The incremental one-day flow formulation has the following decision variables:

x_l^k : Integer variable representing the number of active locomotives of type $k \in K$ on the arc $l \in TrArcs$.

y_l^k : Integer variable indicating the number of non-active locomotives (deadheading, light-traveling or idling) of type $k \in K$ on the arc $l \in AllArcs$.

α_l^{k+} : Positive deviation from the current flow for locomotive type k.

α_l^{k-} : Negative deviation from the current flow for locomotive type k.

z_l : Binary variable which takes value 1 if at least one locomotive flows on arc $l \in LiArcs$ and 0 otherwise.

s^k **:** Integer variable indicating the number of unused locomotives of type $k \in K$.

The objective function and constraints are:

$$\min z = \sum_{l \in TrArcs} \sum_{k \in K} c_l^k x_l^k + \sum_{l \in AllArcs} \sum_{k \in K} d_l^k y_l^k - \sum_{k \in K} G^k s^k + \sum_{l \in TrArcs} \sum_{k \in K} (\alpha_l^{k+} + \alpha_l^{k-}) P \qquad (3.3a)$$

We consider the same constraint set as that in the flow formulation (Ahuja et al. (2005)) with the following additional constraint:

$$\alpha_l^{k+} - \alpha_l^{k-} = x_l^k + y_l^k - (\overline{x_l^k} + \overline{y_l^k}), \text{ for all } l \in TrArcs, \tag{3.3b}$$

$$\alpha_l^{k+}, \alpha_l^{k-} \geq 0. \tag{3.3c}$$

where P is the penalty for deviating from the current daily plan's assignment, and $\overline{x_l^k}, \overline{y_l^k}$ are the current daily flows on arc l. This in effect sets α_l^{k+} and α_l^{k-} to the positive and negative deviations from the current daily plan respectively. By varying the value of parameter P, we can get solutions which differ from the current daily plan and hence the current weekly plan by different degrees.

We measure the difference between the output of the incremental planning model and the current plan as:

$$\% \text{ difference} = \frac{\text{Number of trains for which the model assignment differs from the current plan}}{\text{Total number of trains}} \times 100$$

Note that while it is not possible to directly control the percentage difference (above). By suitably modifying P we can obtain solutions which differ from the current plan by varying degrees (Increase in the value of P leads to a decrease in the percentage difference). Incremental planning enables the railroad to make a sequence of incremental improvements and arrive at the optimal locomotive plan without huge transition costs.

3.4.2 Cab Signal Requirements

Each locomotive in the fleet is equipped with specialized equipment that may enable it to operate in certain restricted territories while at the same time disqualify it from operating in other territories. Some of this equipment is required to enable the locomotive to be the first in a consist or the lead locomotive. Other equipment may be required by union rules or regulatory

rules unique to certain geography. Some of these constraints are handled during tactical (real-time) assignment of locomotives to trains. However, one particular constraint, which we call the cab signal constraint, must be a part of the locomotive plan.

Railroad corridors are equipped with signaling systems to control the movement of trains. Most corridors are outfitted with wayside signals only; the crew in the locomotive observes the signal and slows, stops, or proceeds depending on it. To increase safety, some corridors do not have wayside signals but instead are equipped with cab signal systems, where the signal is displayed inside the locomotive. Not only do cab signals aid the crew in foggy weather or in cases where the wayside signal may be obscured but the cab signal systems also interface with the locomotive throttle and brake and in cases where a crew does not honor the signal, the system will automatically slow down and then stop the train to avert a possible collision. Federal law requires that all consists operating in cab signaled territory must have a lead locomotive that is outfitted to interface with the cab-signal system. For planning purposes, railroads like to have all locomotives in a consist equipped with cab signals so that suppose the lead locomotive breaks down, the units can be swapped on the line of road and the train will continue; or, at the end of the line, the consist can reverse direction without being turned. Outfitting a locomotive with cab signals costs in excess of $100,000 and there are increased maintenance and inspection requirements as well. Consequently, railroads do not equip every locomotive with cab signals since that is too costly.

To incorporate the cab signal logic, we partition each consist type into two consist types, one with the cab signal capability and the other without the cab signal capability. For example, consist type 2-[SD40] can be decomposed into the categories: 2-[SD40-Normal] and 2-[SD40-Cab]. Similarly, we decompose the fleet-size requirements for SD40 locomotives into two parts:

SD40-Normal and SD40-Cab (this decomposition is an input provided by the railroad). For each train l, which operates in a cab signal corridor, we ensure that the all the consist flow variables on it, that is, x_l^c, y_l^c are zero when c is not a cab signal compatible consist. This change guarantees that all the trains in cab signal corridors are assigned cab signal consists, but those in normal corridors can be assigned either kind of consist. While this change does not change the nature of our model, it increases the number of consists which increases the number of decision variables and constraints proportionately, and leads to consequently longer run times. While the locomotive flow formulation (Ahuja et al. (2005)) which suffers from intractability cannot handle the increased problem size, the consist based formulation is also able to solve this larger problem quickly.

3.4.3 Foreign Power Requirements

Railroads often cooperate to run trains directly from the origin station on one railroad to the destination station on another railroad. These trains are called run-through trains. By allowing locomotives from the originating railroad to stay with the train through to destination, the cooperating companies eliminate queuing of trains and locomotives at the interchange stations that would otherwise occur as one railroad would have to move extra locomotives to the interchange point in anticipation of the train arrival, or the train would have to sit at the interchange waiting for locomotives. Over the week, several inbound run-through trains bring in foreign power into the network and several outbound run-through trains return foreign power to other railroad networks. However, the inbound and outbound run-through train schedules are not perfectly balanced and the foreign power often flows back to other interchange locations or on direct or indirect trains. But, at the end of the week, each company is obligated to return as many

locomotives that it receives from each connecting carrier in order to maintain the overall balance of power across North America.

The foreign power requirement is incorporated by making appropriate changes to the space-time network. The network is augmented in such a way that solving the LPP would automatically guarantee a plan which accommodates foreign power. We create the augmented space-time network in the following manner. We first create the space-time network for the LPP as described in Section 3.3.1. Then, we create a pseudo super station for each of the foreign railroads. All the trains which bring in foreign power into the system originate at their respective super stations and terminate at their respective destinations. All the trains which send foreign power out of the system originate at their respective origins and terminate at their respective super stations. Due to the flow balance constraints, the number of locomotives (or consists) of a particular type entering a super node will be equal to the number of locomotives (or consists) of the same type leaving the same super node. However, the model may use the super station as a short-cut between two geographic stations. For e.g. a consist that needs to travel between Chicago and Memphis may be routed from Chicago to a super station and then from the super station to Memphis because it is cheaper to do so. To prevent this kind of short-cutting, we set the costs on ground arcs at the super node to a large value. This ensures that the solution does not misuse the super node as a short-cut between two geographic stations and that only essential foreign power movement takes place between railroads.

3.5 Computational Results

We now present computational results on several problem instances. We also present the results of case studies and draw insightful conclusions. We implemented our algorithms in Visual Basic .NET programming language and tested them on the data provided by a major Class I railroad. We modeled our integer programs using Concert Technology 2.0 modeling language

and solved them using the CPLEX 9.0 solver. We conducted all computational tests on a Pentium IV, 512 MB RAM and 2.4 GHz processor desktop computer.

3.5.1 Comparison of Formulations

We compare the performances of the consist based formulation and the hybrid formulation on real-life instances while varying the pre-specified input consist set as shown in Table 3-1. The first row of the table shows three input consists. Each subsequent row adds two more consists to the previous set of consists. We ran both the consist and hybrid formulations with these consist compositions and noted the running time and solution cost on two different real-life scenarios given to us by the railroad. Table 3-2, Table 3-3, and Figure 3-3 present the results of the comparison.

We make the following observations from this study:

- As we increase the number of consists used, the solution becomes progressively better. An increase from three to nine consists leads to an improvement of more than 17% whereas an increase from nine to seventeen consists gives an improvement of less than 3% demonstrating the law of diminishing returns.
- The objective function of the hybrid formulation for all cases is only marginally (< 5 %) better than that of the consist formulation.

We also applied the original flow formulation developed in Ahuja et al. (2005) to these two scenarios but it did not converge to a feasible solution even after 10 hours computation time. From these tests, we conclude that the consist formulation is an excellent way to model the LPP and we need a small number of consists (less than 10) to obtain good solutions.

3.5.2 Identifying Good Consists

In Section 3.3.4 we described a method to select a set of p optimal consists from a larger candidate set of consists. Here, we consider an instance of the LPP and a set of 15 candidate consists. We then determine the optimal set of consists as described in Section 3.3.4, varying the parameter p from 3 through 15, and note the objective function values of the problem for each

case. We also compare these objective function values to the objective function value obtained when we use the set of eleven consist types specified by the railroad (blue line in the figure). Figure 3-4 presents these computational results.

We draw the following conclusions from these results:

- As the number of consists used increases, the solution quality improves. However, there are diminishing returns as we add more consists.
- Using just five consists, we are able to match the solution obtained by using the eleven consists specified by the railroad.
- Six consists, if chosen judiciously, can give as good a solution as fifteen consists. This is a gratifying observation as railroads desire five or six consists to ensure plan simplicity and plan compliance.

3.5.3 Incremental Locomotive Planning

In this study, we present computational results of the method described in Section 3.4.1 to solve the planning problem incrementally. In the incremental planning problem, we are given a possibly non-optimal feasible solution x^0 and we wish to incrementally optimize it. Let x^* denote the optimal solution of the LPP. We present the results of this test in Table 3-4. We start with solution x^0 and make incremental changes to it so that it gets closer to the optimal solution and we control the extent of changes. Observe that as we increase the extent of change allowed by decreasing the penalty, we progressively obtain improved solutions which differ from x^0 by a greater extent but use lesser number of locomotives. Also note that the solution when the penalty is zero is largely different from the current solution x^0.

3.5.4 Case Study

We conduct a case study to demonstrate the usefulness of the model to assist decision making. The various aspects of the problem that we test in this case study are as follows: (1) Effect of varying minimum connection time on solution cost, (2) Effect of varying transport

volumes on key transport performance characteristics; and (3) Effect of varying train times on key transport performance characteristics.

Effect of varying minimum connection time on solution cost

Freight trains do not run on time and often arrive later than their planned arrival time, which makes it difficult for locomotive dispatchers to adhere to the locomotive plan. One method commonly recommended to improve plan compliance is to increase the train-train minimum connection times (defined in Section 3.2.1). Although increasing the minimum connection time may improve plan adherence, it also increases locomotive costs as more locomotives will be held in inventory at terminals. We quantify the impact of increasing the minimum connection times. We performed the test for a representative instance where we varied connection times from zero hours to ten hours and noted the total cost incurred by each solution. This test was done twice, first with an input set of nine consists and then with an input set of eleven consists. We present the results of the test in Table 3-5 and Figure 3-5.

We draw the following conclusions from the study:

- The minimum connection time could have a significant impact on the number of locomotives used and the solution cost.
- The solution cost increases linearly with the increase in connection times at the rate of around $200,000 per hour.

Depending upon the train lateness and the willingness of railroad planners to improve locomotive plan compliance, appropriate connection times can be used. In this study, we assumed same connection times system-wide. Railroads may want to experiment with station-dependent connection times based on historical lateness performance of trains. As shown in this study, railroads can use the model to quantify the trade-off between decreasing minimum connection times and increasing solution costs.

Effect of varying transport volume on key performance characteristics

We measure the impact of varying transport volumes (or train tonnages) on the following key transport characteristics: number of locomotives used, solution cost, mean pulling power of a consist, and mean miles traveled per consist. We start with a solution with specified tonnages on each train. We then proceed to decrease tonnages by steps of five percent and measure the impact on transport characteristics. We then repeat the test but this time increasing tonnages by steps of five percent. The results of these tests are presented in Table 3-6 and Figure 3-6. The consist flow formulation was able to solve all instances in less than two minutes of computational time.

We draw the following conclusions from this study:

- As the transport volume (mean tonnage) increases, the solution cost tested increases as a quadratic function with an increasing slope
- The mean pulling power of each consist and the number of locomotives used increase as expected; however, the average number of miles traveled by each consist remains roughly the same.
- The consist flow formulation is robust since it solves all the instances in less than two minutes of computational time.

As we demonstrate, railroads can use the model as a generic approach for modeling the optimal number of locomotives needed or the cost as a function of the rail freight transport volume over the entire network.

Effect of varying train speed on key performance characteristics

We measure the impact of varying train speeds on the following key transport characteristics: number of locomotives used, solution cost, mean pulling power of a consist, and mean miles traveled per consist. We start with a solution with specified travel times on each train. We then proceed to decrease travel times by steps of five percent and measure the impact on the transport characteristics. We then repeat the test but this time increasing travel times by

steps of five percent. The results of these tests are presented in Table 3-7 and Figure 3-7. The consist flow formulation was able to solve all these instances in less than two minutes of computational time.

We draw the following conclusions from this study:

- As the train travel times increase, the solution cost increases as a quadratic function with an increasing slope.
- The number of locomotives used also increases as shown in the table; however, the mean miles per consist remains roughly constant.
- The consist flow formulation is robust since it solves all the instances in less than two minutes of computational time.

Increasing train speeds improves efficiency of the network and leads to lesser scheduling costs. However, it may also lead to an increase in other costs like fuel consumption cost, safety, etc. The railroad can use the model to perform cost-benefit analysis to trade-off the increase in cost of operations with the savings in scheduling costs and consequently determine their optimal strategy.

3.6 Summary and Conclusions

Our research builds upon our previous work on railroad locomotive scheduling. We focus on improving the planning version of the problem, where we assign locomotive types to various trains. We extend the approach on several dimensions by adding new constraints to the planning problem required by the railroads, and by developing additional formulations necessary to transition solutions of our models to practice. The major contribution is to suggest a new approach of routing consists (a group of locomotives) rather than routing individual locomotives. We show through our computational tests that even though routing of consists seems to restrict the solution space of the LPP, if we create good consists, we can get almost as good solutions as with the earlier formulations. This is a very important observation since solutions of the consist

formulation are much easier to implement, cause less train delays, and allow greater flexibility for mix-and-match for locomotive dispatchers. Further, we found that the consist formulation runs faster and is more robust than the earlier flow-based formulations. Another important contribution in the chapter is to enable a railroad to progressively move towards an optimal locomotive plan in several smaller and realistic steps by solving a sequence of incremental LPPs. Finally, we also added important practical features such as accounting for foreign power and cab-signal locomotives, which are critical to generating implementable locomotive plans.

The locomotive planning model creates a blueprint for use by managers to assist them in making day-to-day decisions regarding locomotive assignments. We demonstrate the usefulness of this model by performing extensive case studies to evaluate the impact of varying minimum connection times, freight transport volumes, and train speeds (or travel times). In the future, we hope to develop tools for use by tactical locomotive dispatchers to assign locomotives by specific tail numbers. This real-time model will strive to return the network to plan, while maintaining fluidity and minimizing overall costs. The locomotive planning models described will form the core of the real-time locomotive models.

Table 3-1. Set of input consists.

Index	# Consists	Consist Set	Horsepower
1	3	2[SD40], 3[SD40], 3[CW40-8]	6000, 9000, 12000
2	5	& 2[CW60AC], 2[CW40-8]+1[SD40]	12000, 11000
3	7	& 1[CW40-8] + 2[SD40], 2[CW44AC]	10000, 8800
4	9	& 2[CW40-8], 2[SD40] + 1[SD60I]	8000, 9800
5	11	& 2[CW40-8] + 1[CW60AC], 1[CW40-8] + 1[SD40]	14000, 7000
6	13	& 1[SD60I] + 1[GP40], 1[CW44AC] + 1[SD40]	6800, 7400
7	15	& 4[SD40], 2[GP40]	12000, 6000
8	17	& 2[SD60I], 3[GP40]	7600, 9000

Table 3-2. Comparison of formulations: Scenario 1

388 trains, 6 locomotive types, 87 stations

	Consist formulation				Hybrid formulation			
# of consists	Locos used	Cost ($)	Time (sec)	Status	Locos used	Cost ($)	Time (sec)	Status
3	1,376	9,991,949	2.9	Optimal	1,338	9,671,012	120.0	0.4% gap
5	1,330	9,789,957	45.0	Optimal	1,285	9,402,383	120.0	0.1% gap
7	1,183	8,597,982	10.8	Optimal	1,141	8,337,573	450.0	0.6% gap
9	1,051	8,221,321	4.8	Optimal	1,052	8,164,544	600.0	4.1% gap
11	1,047	8,118,454	9.1	Optimal	1,051	7,994,043	600.0	1.7% gap
13	1,045	8,062,075	92.6	Optimal	1,057	7,923,847	1,000.0	8.6% gap
15	1,047	7,995,065	114.0	Optimal	1,058	7,833,838	1,000.0	6.4% gap
17	1,047	7,995,065	120.0	0.1% gap	1,056	7,833,073	1,500.0	6.7% gap

Table 3-3. Comparison of formulations: Scenario 2

382 trains, 6 locomotive types, 87 stations

	Consist formulation				Hybrid formulation			
# of consists	Locos used	Cost ($)	Time (sec)	Status	Locos Used	Cost ($)	Time (sec)	Status
3	1,388	10,071,804	120.0	Optimal	1,361	9,813,393	347.0	Optimal
5	1,343	9,871,203	13.1	Optimal	1,312	9,573,504	450.0	0.4% gap
7	1,201	8,731,696	11.4	Optimal	1,166	8,517,248	450.0	1.4% gap
9	1,064	8,383,781	34.6	Optimal	1,060	8,282,831	600.0	2.6% gap
11	1,062	8,301,766	120.1	0.1% gap	1,056	8,179,846	600.0	2.3% gap
13	1,056	8,247,947	450.0	0.6% gap	1,066	8,112,917	1,200.0	8.1% gap
15	1,062	8,204,256	450.0	1.8% gap	1,071	7,990,316	1,200.0	4.2% gap
17	1,063	8,204,160	450.0	1.7% gap	1,071	7,990,300	1,800.0	4.9% gap

(c)

Table 3-4. Computational results of incremental locomotive planning.

Penalty Value	% Difference	Locos Used	Time (min)
1,800	5.5%	1,463	10.0
1,600	5.5%	1,463	10.0
1,400	6.3%	1,430	10.0
1,200	7.0%	1,393	10.0
1,000	7.4%	1,382	10.0
800	10.5%	1,352	10.0
600	19.4%	1,214	10.0
500	22.1%	1,175	10.0
400	26.4%	1,159	10.0
300	28.3%	1,141	10.0
200	36.8%	1,146	10.0
100	60.6%	1,130	10.0
0	81.0%	1,128	10.0

Table 3-5. Effect of varying the minimum connection time.

Min Connection Time (hrs)	Scenario 1: 9 Consist Types			Scenario 2: 11 Consist Types		
	Cost ($)	Locos Used	Time (sec)	Cost ($)	Locos Used	Time (sec)
0	7,497,798	903	19.8	7,408,113	903	38.3
1	7,734,067	947	7.5	7,628,916	943	3.3
2	7,926,831	977	4.0	7,832,533	973	16.3
3	8,112,888	1,015	2.6	8,010,123	1,009	5.5
4	8,330,598	1,049	2.9	8,218,477	1,047	4.2
5	8,506,699	1,081	10.0	8,392,475	1,083	4.2
6	8,732,819	1,127	14.3	8,617,059	1,131	9.7
7	8,896,129	1,159	7.2	8,778,249	1,164	34.6
8	9,163,605	1,221	23.2	9,042,940	1,221	86.8
9	9,277,919	1,245	20.6	9,165,863	1,245	57.7
10	9,439,117	1,277	12.3	9,306,191	1,274	37.7

Table 3-6. Effect of varying transport volumes.

% increase in tonnage	Mean tonnage of trains	Locos Used	Solution Cost ($)	Mean pulling power/consist	Mean miles/ consist
-20	6,183	1,026	7,502,802	10,099	405.05
-15	6,570	1,026	7,588,421	10,373	405.04
-10	6,956	1,042	7,726,579	10,464	404.17
-5	7,342	1,040	7,910,331	10,841	405.14
0	7,729	1,049	8,120,134	11,093	403.73
5	8,115	1,079	8,437,334	11,457	403.86
10	8,502	1,117	8,812,467	11,828	405.27
15	8,888	1,171	9,296,423	12,483	403.98
20	9,275	1,214	9,635,790	12,715	405.12

Table 3-7. Effect of varying train travel times.

% increase in travel time	Locos Used	Solution Cost ($)	Mean pulling power/consist	Mean miles/ consist
-20	915	6,959,179	11,237	403.20
-15	937	7,189,839	11,236	404.67
-10	975	7,531,424	11,199	407.52
-5	1,003	7,778,318	11,157	404.78
0	1,049	8,120,834	11,093	403.73
5	1,073	8,383,096	11,197	405.00
10	1,111	8,702,847	11,151	404.07
15	1,148	8,967,099	11,056	405.12
20	1,171	9,194,210	11,038	404.92

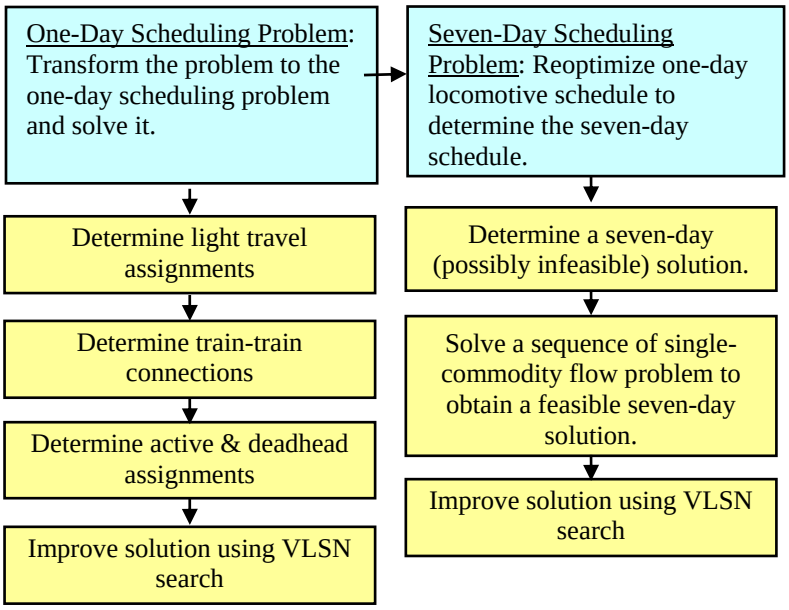

Figure 3-1. Overview of the locomotive scheduling algorithm

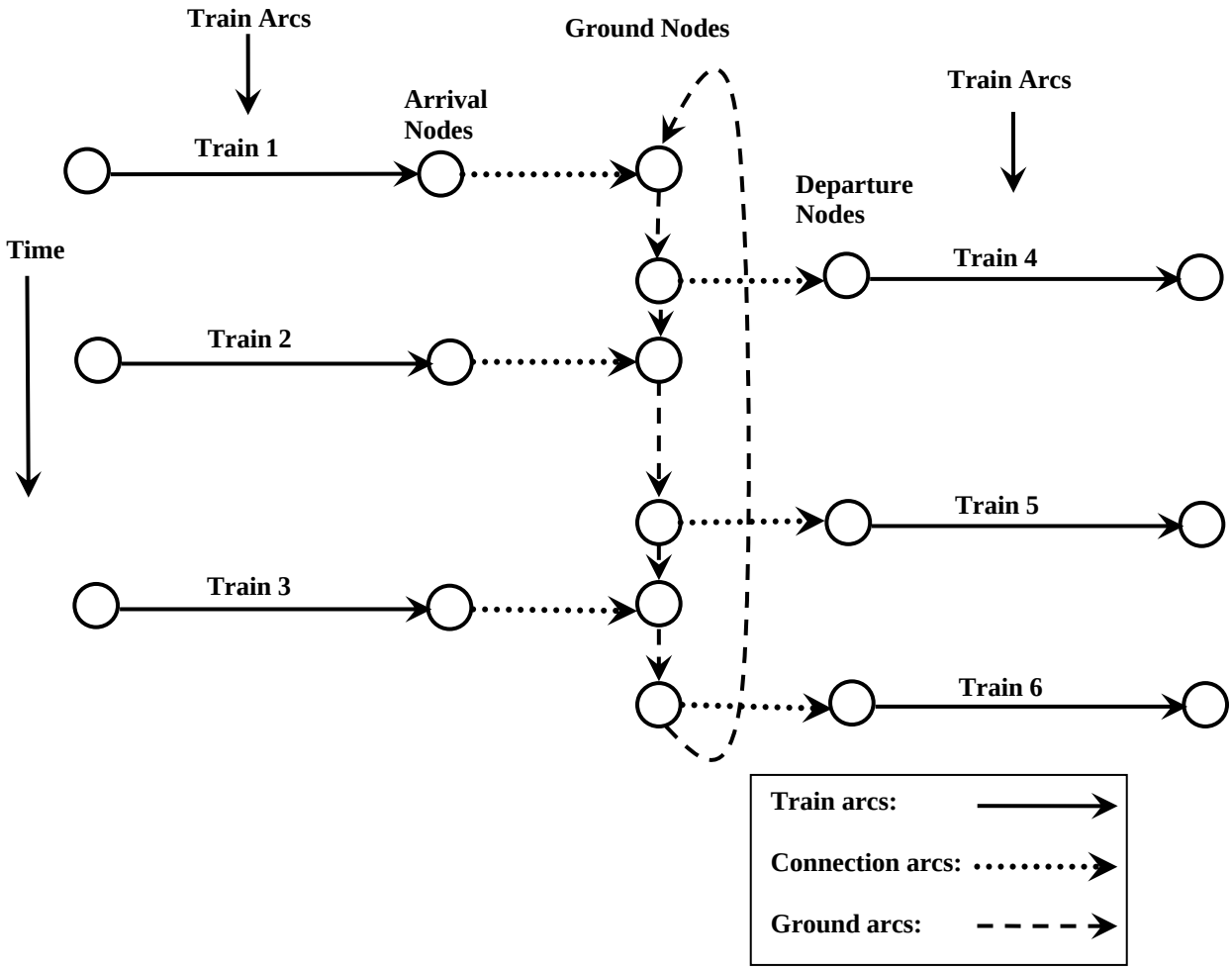

Figure 3-2. A part of the space-time network.

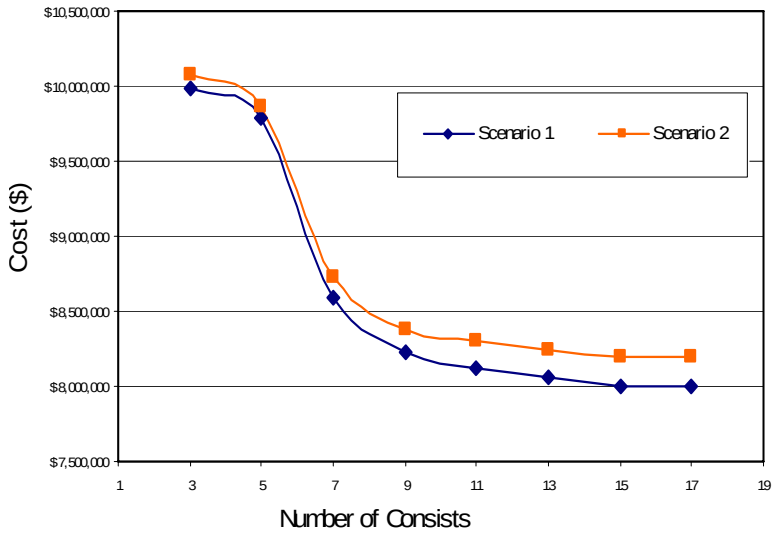

Figure 3-3. Solution cost of the consist formulation vs. number of consists.

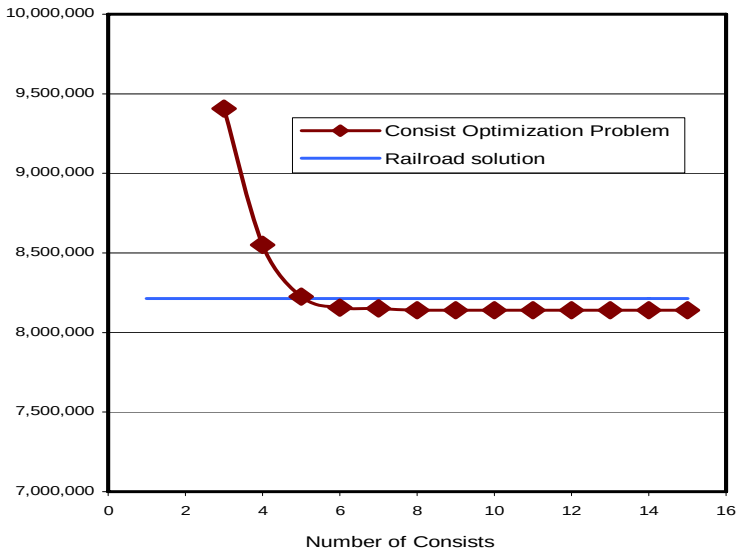

Figure 3-4. Solution cost vs. number of consists.

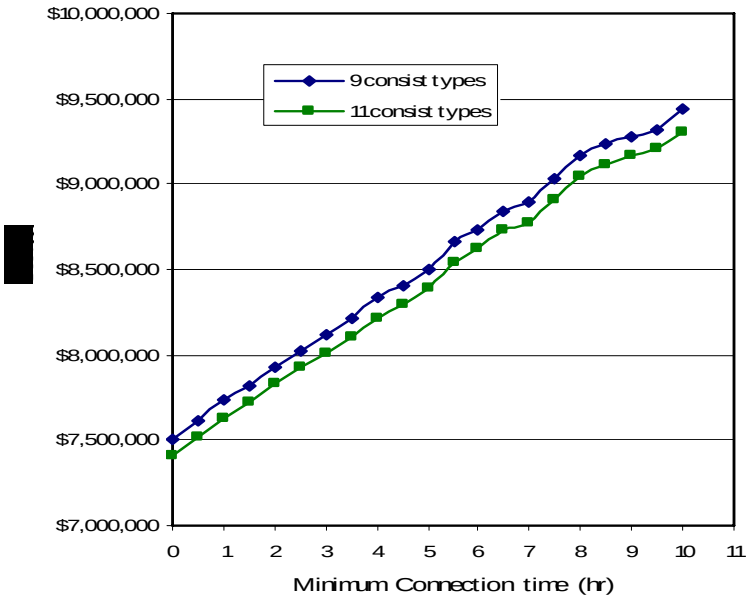

Figure 3-5. Solution cost vs. minimum connection time.

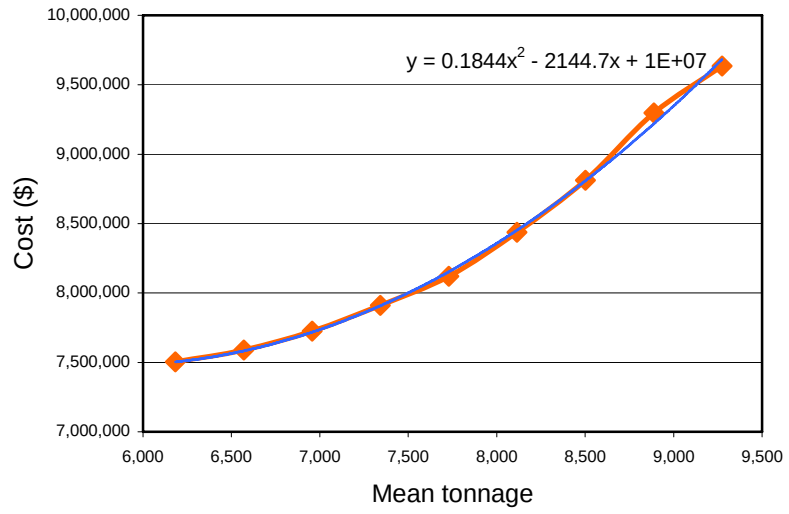

Figure 3-6. Solution cost vs. transport volumes.

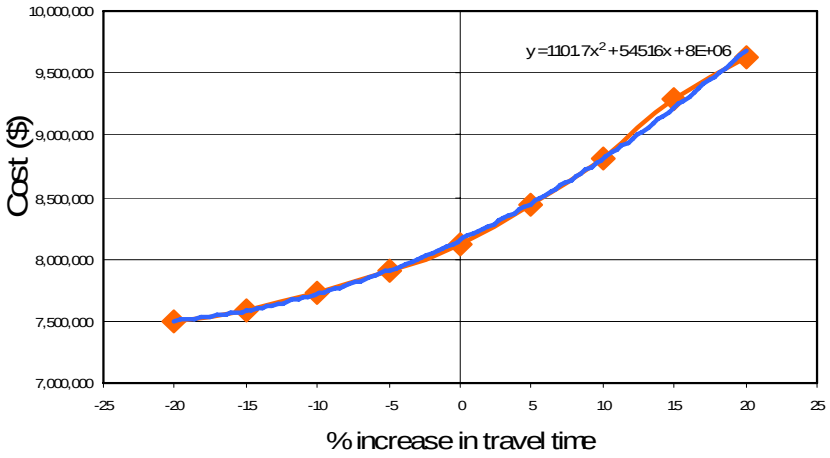

Figure 3-7. Solution cost vs. % increase in travel time.

CHAPTER 4
THE LOCOMOTIVE ROUTING PROBLEM

4.1 Introduction

The objective of the LPP is to assign locomotive types to trains in a cyclic train schedule, while honoring several operational constraints and minimizing the overall cost of assignment. Our models to solve the LPP, described in Chapter 3, assign locomotive types to trains in the schedule ensuring that every train in the weekly repeating schedule gets sufficient pulling power, locomotive flows are balanced, and several other operational constraints are satisfied; however, they do not account for the fueling and servicing feasibility of individual locomotive units. We bridge this gap and develop methods that can be used to route locomotive units on fueling and servicing friendly cycles. Locomotive plans or schedules are blue prints based on which the railroads manage their operations in real-time; our research is hence a vital cog in the area of locomotive management since it makes the blue-print more realistic and implementable.

The LRP involves routing each locomotive unit on a cyclic sequence of trains so that it can be fueled and serviced as necessary; while at the same time satisfying all other business and operational constraints considered in the LPP. We refer to the sequence of trains on which a locomotive unit travels as a locomotive route. Fueling feasibility requires that each locomotive route has sufficient fueling opportunities to ensure that the locomotive does not run out of fuel (out-of-fuel event). This translates to the constraint that every locomotive needs to have a fueling opportunity at least once for every F miles (900 miles) of travel. Similarly, servicing feasibility requires that each locomotive route has sufficient servicing opportunities and this translates to the constraint that every locomotive needs to have a servicing opportunity at least once for every S miles (3,000 miles) of travel. Suppose all the stations in the network support fueling and servicing, any feasible solution to the LPP could be trivially decomposed into a fueling and

servicing friendly routing of locomotives using a straight forward polynomial-time cycle decomposition algorithm (Ahuja, Magnanti, and Orlin (1993)). However, typically only 50% of the stations on the train network support fueling and 30% of the stations support servicing, and this makes the LRP an extremely complex combinatorial optimization problem. Indeed, this problem remains practically unstudied and unsolved, and our chapter reports the first attempt to model and solve the LRP using optimization methods.

The LPP (Ahuja et al. (2005), Vaidyanathan et al. (2007b)) is analogous to the well studied airline fleet assignment problem (Abara (1989), Talluri (1996), and Rexing et al. (2000)) and the LRP is analogous to the well studied aircraft maintenance routing problem (Clarke et al. (1997), Gopalan and Talluri (1998), Barnhart et al. (1998), and Talluri (1998)). Integrating airline fleet assignment and maintenance routing into a single problem makes the problem very hard to solve. Hence, the predominant approach adopted is sequential; i.e. the fleet assignment problem is solved first to fix the assignment of aircraft types to flights, and the aircraft routing problem is solved next to route individual aircrafts. Several researchers have worked on the aircraft routing problem. Clarke et al. (1997) present a mathematical formulation for the aircraft routing problem and discuss its similarity with the asymmetric traveling salesman problem. Gopalan and Talluri (1998), and Talluri (1998) describe methods to generate aircraft routings that satisfy the four-day maintenance requirement. In another study with a different objective, Feo and Bard (1989) study the maintenance location problem to find the minimum number of maintenance stations required to meet the specified 4-day maintenance requirements for a proposed flight schedule. The only study which considers the integrated fleet assignment and aircraft routing problem is due to Barnhart et al. (1998). They use a branch-and-price based approach to solve the integrated airline

fleet assignment and aircraft routing problem. However, even for small problems with 150-200 flights their approach takes 3-4 hours of computational time.

The study of various locomotive scheduling models has also been reported in the literature; however none of these consider the fueling and servicing requirements of locomotive units. In Chapter 3, we survey the pre-existing research on the LPP. The most comprehensive multiple locomotive planning models are due to Ahuja et al. (2005) and Vaidyanathan et al. (2007b). The only existing railroad research which considers maintenance requirements is due to Maroti and Kroon (2005). They consider the problem of routing locomotive units that require maintenance in the next one to three days and propose a multicommodity flow model to solve this problem. Their problem definition does not consider fueling requirements and this reason among several others makes their problem less general than the LRP that we consider.

The existing methods in the literature for aircraft routing cannot be directly applied to the LRP. This is because the LRP is more complex than the aircraft routing problem due to the following reasons: (a) Every airport on the airline network supports fueling, making fueling feasibility a non-issue; however every station on a railroad network does not support fueling; (b) Airlines do not operate during the night and this gives added flexibility to satisfy maintenance requirements whereas railroads operate round the clock; (c) Flight legs on the airline network are assigned to only one aircraft whereas trains on a railroad network are usually assigned several locomotive units; and (d) Aircraft routing problems are typically solved for a few hundred flights whereas the LRPs need to be solved for schedules with a few thousand trains. Also, note that the integrated approach used by Barnhart et al. (1998) takes 3-4 hours of computational time even for a problem with 150-200 flights; this is a very small problem from the railroad perspective. Due to these reasons, we adopt a two stage decomposition approach to solve the problem. In the

first stage we solve the LPP, using existing methods, to obtain an assignment of locomotive types to trains (locomotive schedule). In the second stage, we use the methods described in this chapter to solve the LRP and route locomotive units on fueling and servicing friendly cyclic routes (locomotive routes). While solving the LRP, we fix the assignment of locomotive types to trains based on the output of the LPP, and use the flexibility of varying the connections of locomotives between trains (train connections) to achieve fueling and servicing feasibility. We illustrate the importance of train connections through the following example. Consider a locomotive that arrives at a station on train A having traveled 500 miles after its last fueling. Let this locomotive have the option to connect to either train B or train C. While train B has a length of 300 miles and terminates at a fueling location, train C has a length of 500 miles and also terminates at a fueling location. If the locomotive connects from train A to C, then the total length of travel between fueling opportunities becomes 1,000 miles (> 900 miles) and leads to an out-of-fuel event. On the other hand, if the same locomotive connects from train A to B, the total length of travel between fueling opportunities is 800 miles and the locomotive route is fueling feasible. However, note this example considers a local single connection (or two trains) case but fueling and servicing feasibility is often related to more than one connection (or a sequence of trains). We formalize this notion by introducing the concept of strings. Strings enable us to account for the fueling and servicing feasibility of a sequence of trains on which a locomotive unit travels. Hence, each locomotive route is made up of a sequence of strings.

Since we fix the assignment of locomotive types to trains, the LRP can be solved as an independent problem for each locomotive type. Our solution approach for the LRP involves the following steps. First, we enumerate alternative fueling and servicing feasible paths (or strings) between service stations in the train network using dynamic programming. Once we enumerate

the strings, we formulate an integer program called string decomposition problem (SDP) on a suitably defined space-time network to decompose the locomotive schedule into flows on strings. The output of the SDP can then be assembled into fueling and servicing friendly cycles using the standard polynomial-time cycle decomposition algorithm (Ahuja, Magnanti, and Orlin (1993)). However, SDP is NP-Complete and has millions of decision variables. We develop an aggregation-disaggregation based method to solve this problem efficiently. Our aggregation-disaggregation method involves two steps. In the first step, we formulate and solve a much smaller aggregated SDP and in the second step we disaggregate the solution of the first stage to obtain a solution to the original problem. Finally, we develop necessary constraints that can be added to the LPP so that the planning model also considers fueling and servicing constraints to some extent.

Our major research contributions are:

- We develop a modeling framework and a multi-stage decomposition solution approach for the LRP which is a previously unstudied and unsolved problem. This problem is more complex than the well studied analogous aircraft routing problem.
- We formulate the LRP as an integer programming string decomposition problem (SDP) on a suitably defined space-time network and prove that the problem is NP-Complete.
- We develop efficient dynamic programming algorithms to enumerate strings which are the decision variables in the SDP.
- Due to the combinatorial nature of strings, the SDP has close to a million decision variables. We develop an aggregation-disaggregation based algorithm to solve this extremely large NP-Complete problem to near optimality within a few minutes of computational time.
- We present extensive computational results to validate the performance of our approach and present case studies of our algorithms on real-life data provided by a Class I US railroad.

We present the first comprehensive study of the LRP, which has remained practically unstudied and unsolved. We model the LRP using network based integer programming

techniques and design efficient algorithms to solve it. Our computational results demonstrate the efficacy of our algorithms; the algorithms that we present produce near optimal solutions within 5 minutes of computational time on several real-life instances obtained from a Class I US railroad. We also demonstrate the usefulness of our model to perform strategic analysis through suitable case studies. We believe that this research is a crucial step in our overall objective of enabling railroads to manage their locomotive resources more efficiently through optimization techniques.

4.2 Notations and Terminology

Train Schedule: The train schedule (or train network) contains the set of trains which operate every week. Each train has the following attributes: train ID, departure station, departure time of the week, arrival station, arrival time of the week, and duration. Railroads typically have a train schedule that is periodic. We assume that the train schedule is weekly periodic and hence we refer to trains in the weekly train schedule as weekly trains. Each weekly train is identified by the unique combination of its train ID and day of operation. For e.g. Train ID TR01 which operates on Monday, Tuesday, and Thursday corresponds to three weekly trains, TR01-Monday, TR01-Tuesday, and TR01-Thursday. TR01 is also referred to as a daily train.

Locomotive Schedule: The locomotive schedule specifies the assignment of locomotive types to all trains in the week such that all operational constraints, other than the fueling and servicing requirements, are satisfied. These constraints include pulling effort and horse power constraints for each train, flow balance for each locomotive type, fleet-size constraints, flow upper and lower bounds on each train, and weekly repeatability of locomotive assignments.

Decomposed Locomotive Schedule: The locomotive schedule can be decomposed with respect to the different locomotive types in the fleet. The decomposed locomotive schedule which corresponds to a particular locomotive type considers only the trains that carry that

locomotive type and also considers only assignments of that particular locomotive type. For e.g. Suppose we consider a schedule with three trains, TR01 has an assignment of one unit of type A and one unit of type B, TR02 has an assignment of two units of type A, and TR03 has an assignment of two units of type B. Then, the decomposed locomotive schedule corresponding to locomotive type A will contain TR01 with as assignment of one unit and TR02 with an assignment of two units; and the decomposed locomotive schedule corresponding to locomotive type B will contain TR01 with an assignment of one unit and TR03 with an assignment of two units.

Decomposed Train Schedule: The set of trains in a decomposed locomotive schedule of a particular locomotive type constitutes the decomposed train schedule of that locomotive type.

Fueling Station: A fueling station is a station that supports fueling of locomotives. Every fueling station has an associated fueling time and fueling cost per gallon.

Servicing Station: A servicing station is a station that supports servicing of locomotives. All servicing stations are also fueling stations. Each servicing station has an associated servicing time and servicing cost.

Train Connection: Refers to transfer of locomotives from an inbound train at a particular station to an outbound train at the same station.

Minimum Connection Time: This refers to the minimum time required for a train connection to be made. It is a function of the station at which the connection is made.

String: A string is a connected sequence of trains $t_1 - t_2 - ... - t_n$; i.e. the destination of train t_i is the same as origin of train t_{i+1}.

Fuel String: A fuel string is a connected sequence of trains such that a locomotive unit traveling on this sequence can be fueled feasibly.

Service String: A service string is a connected sequence of trains such that a locomotive traveling on this sequence can be fueled and serviced feasibly.

4.3 Overview of Our Approach

We present our approach to solve the fueling and servicing friendly LRP in Figure 4-1. This flowchart presents the various steps in our approach and how these steps relate to each other. In subsequent sections, we then elaborate on details. Note that step 1 involves solving the LPP and steps 2-4 constitute solving the LRP. The LRP is solved as an independent problem for each locomotive type

4.4 The Locomotive Planning Problem (LPP)

Our approach to solve the fueling and servicing routing problem is a two stage decomposition approach where in the first stage we solve the LPP and in the second stage we solve the LRP. The locomotive schedule obtained from the LPP is hence one of the inputs that goes into the LRP. In Chapter 3, we present our research on the LPP. The models developed there give us a locomotive plan that satisfies all constraints other than the fueling and servicing constraints.

We develop methods to incorporate fueling and servicing constraints. We fix the assignments of locomotive types to trains based on the solution to the LPP and solve the LRP which uses the flexibility of modifying train connections to enforce fueling and servicing feasibility of locomotive routes. Since we fix the assignment of locomotive types to trains after the LPP, the LRP can be solved as an independent problem for each locomotive type. In our description of the LRP in Sections 4.5-4.8, we therefore assume that there is only one locomotive type.

4.5 Fuel and Service String Enumeration Algorithms

In the introduction, we mentioned that the LRP uses the flexibility of switching train connections in order to achieve fueling and servicing feasibility. In this section, we formalize this notion by defining the concept of strings and also developing algorithms to enumerate strings. The objective of the LRP is to route locomotive units in cycles in such a way that each unit has sufficient fueling and servicing opportunities. Consider a unit that travels on one such cycle. This locomotive will visit several stations, some which are servicing stations, some which are fueling stations, and some which are neither. Suppose this cycle is a fueling and servicing friendly cycle. Then, by definition this would imply that the sequence of trains that the locomotive takes between fueling stops is a fuel string and the sequence of trains that the locomotive takes between servicing stops is a service string. This observation is the basis of our approach to solve the LRP. Our approach involves enumerating strings and then decomposing the locomotive schedule into flows on strings. In this section, we develop algorithms to enumerate strings efficiently. We next formally define fuel strings and then describe methods to enumerate them.

Fuel string

- A fuel string is a sequence of trains $t_1 - t_2 - \ldots - t_n$ which satisfies the following properties:
- It is connected i.e. arrival station of t_i = departure station of t_{i+1}.
- Departure station of t_1 and arrival station of t_n are fueling stations.
- Length of the sting is less than F miles.
- The string is minimal; i.e., a fuel string cannot pass through an intermediate fueling station or a fuel string cannot contain another fuel string as a sub-sequence.
- No train repeats in the string.

Valid fuel sequence and algorithmic logic

We define a valid fuel sequence as a sequence of trains which satisfies all the properties of a fuel string except that it does not terminate at a fueling station. Thus, a valid fuel sequence can be obtained from any fuel string by deleting the last train from it. Conversely, any fuel string can

be represented as a valid fuel sequence plus one last train which terminates at a fueling station. Our dynamic programming algorithm uses this property to enumerate fuel strings. The algorithm starts with a seed set of trivial valid sequences and iteratively builds longer sequences from this set. The algorithm is based on repeated application of the following inductive logic: Consider a situation when we have the set of all valid fuel sequences of length k where length represents the number of trains in the sequence. Consider a train that departs from the destination of the last train in the valid sequence. When this train is added to the valid sequence to create an augmented sequence, the following possibilities could occur:

- The augmented sequence is a valid fuel sequence in which case we store it in the set of valid fuel sequences of length k+1.
- The train which is being added to the valid fuel sequence is already a part of the valid fuel sequence in which case we do not store the augmented sequence in the set of valid fuel sequences of length k+1.
- The length of the augmented sequence is greater than F miles in which case we do not store it in the set of valid fuel sequences of length k+1.
- The augmented sequence is a fuel string in which case we store it in the set of fuel strings.

Fuel string enumeration

The fuel string enumeration (FSE) algorithm starts with a set of valid fuel sequences of length one. By definition the set of valid sequences of length one contains all trains which originate at a fueling station and do not terminate at a fueling station. The algorithm then iteratively generates longer fuel strings until all such possibilities have been enumerated. We now give the notation and description of the algorithm.

Let P^k denote the set of valid fuel sequences of length k, FS denote the set of fuel strings, and Trains denote the set of weekly trains in the schedule. Then, the fuel string enumeration algorithm can be formally written as shown in Figure 4-2.

The number of valid sequences and fuel strings is finite. Each execution of the inner-most loop produces a longer valid sequence, a fuel string, or fathoms that particular valid fuel sequence. Hence, FSE terminates in a finite number of steps. We now define service strings and extend the dynamic programming approach to enumerate service strings.

Service string

A service string is a sequence of trains $t_1 - t_2 - ... - t_n$ which is characterized by the following properties,

- It is connected i.e. arrival station of t_i = departure station of t_{i+1}.
- Departure station of t_1 and arrival station of t_n are service stations.
- Length of the string is less than S miles.
- The string is minimal; i.e., a service string cannot pass through an intermediate servicing station. This implies that a service string cannot contain another service string as a sub-sequence.
- No train repeats in the string.
- The string is also fueling feasible.

Since all service stations in the network also support fueling, it follows that for the last condition to be satisfied, every service string has to be constructed from one or more connected fuel strings.

Valid service sequence and algorithmic logic

We define a valid service sequence as a sequence of trains which satisfies all properties of a service string except that it does not terminate at a service station. Therefore, any service string can be represented as a valid service sequence plus one fuel string which terminates at a servicing station. Our dynamic programming algorithm uses this property to enumerate service strings. The algorithm starts with a seed set of trivial valid sequences and iteratively builds longer sequences from this set. Consider a situation when we have the set of all valid service sequences of length k where length represents the number of fuel strings in the sequence. Consider a fuel string that departs from the destination of the last fuel string in the valid

sequence. When this fuel string is added to the valid sequence to create an augmented sequence, the following possibilities could occur:

- The augmented sequence created is a valid service sequence in which case we store it in the set of valid service sequences of length k+1.
- A train which belongs to the added fuel string is already a part of valid service sequence of length k in which case we do not store the augmented sequence in the set of valid service sequences of length k+1.
- The length of augmented sequence is greater than S miles in which case we do not store it in the set of valid service sequences of length k+1.
- The augmented sequence forms a service string in which case we store it in the set of service strings.

Service string enumeration

The service string enumeration (SSE) algorithm starts with a set of valid fuel sequences of length one. By definition the set of valid sequences of length one contains all fuel strings which originate at a service station but do not terminate at a service station. The algorithm then iteratively generates longer service strings until all such possibilities have been enumerated. We now give the formal description of the algorithm.

Let P^k denote the set of valid service sequences of length k, FS denote the set of fuel strings, SS denote the set of service strings, Trains denote the set of weekly trains in the schedule, and Trains(l) denote the set of weekly trains that are a part of string l. Then, the fuel string enumeration algorithm can be formally written as shown in Figure 4-3.

The proof for finiteness of SSE follows in a similar manner to that of FSE. However, since the number of fuel and service strings is an exponential function of the problem size, both these algorithms are exponential time algorithms. Next, we formulate the string decomposition problem which is used to decompose a locomotive schedule into flows on service strings.

4.6 String Decomposition Problem (SDP)

In the previous section, we described methods to enumerate service strings. These service strings are the decision variables in the SDP. The objective of the SDP is to determine the minimum cost decomposition of a locomotive schedule into flows on service strings. Due to the manner in which service strings are constructed, it follows that a locomotive schedule when decomposed into flows on service strings will give fueling and servicing compliant locomotive routings. We formulate the SDP as an integer programming problem on a suitably defined space-time network. We first describe the construction of space-time network and then the integer programming formulation. We also show that string decomposition is NP-Complete by a polynomial-time reduction from the set partitioning problem which is known to be NP-Complete (Garey and Johnson (1979)).

4.6.1 Space-Time Network

We denote the space-time network by G (N, A) where N is the set of nodes and A is the set of arcs in the network. Each node in the network represents an event. The events that we consider are the arrival of weekly trains at service stations and departure of weekly trains from service stations. For each train arrival at a service station, we create a train-arrival node at that station. Once a locomotive arrives at a station, it needs a minimum connection time before being assigned to the next train; i.e., the time at which a locomotive on a train is available for future assignment is the arrival time of the train plus the minimum connection time. We call this time the ready time of the train. To each train-arrival node we assign a time attribute equal to the ready time of the corresponding train. For every train which departs from a service station, we create a train-departure node at the departure station of the train. To each train-departure node, we assign a time attribute equal to the departure time of the corresponding train.

We now describe the construction of arcs on the network. Let S be the set of service strings enumerated using the methods described in the previous section. Let first(s) denote the first train in string s and last(s) denote the last train in string $s \in S$. Since every service string $s \in S$, by definition, originates at a service station and also terminates at a service station, the space-time networks contains a train-departure node $n_1 \in N$ that corresponds to first(s) and a train-arrival node $n_2 \in N$ that corresponds to last(s). For each string s, we construct an arc between its corresponding nodes n_1 and n_2. The other set of arcs on the network are train connection arcs. At each service station we construct train connection arcs (CoArcs) between all combination of inbound train and outbound train to model the different possibilities of inbound locomotives being assigned to outbound trains.

The following important property directly follows from the construction of the space-time network.

Property 4.1. A locomotive which travels on a cycle in the space-time network given above can be fueled and serviced feasibly.

Proof: Consider a cycle on the space-time network G (N, A). The cycle will consist of a service string, a connection arc, another service string, and so on until it loops back to the first service string. Since by definition, a locomotive traveling on a service string can be fueled and serviced feasibly, the result follows. ◆

In Figure 4-4, we illustrate a part of the space-time network at a particular service station. This station has two incoming trains (two train-arrival nodes), two outgoing trains (two train-departure nodes), four incoming strings, seven outgoing strings, and four train connections. Note that the overall space-time network contains one such component at each service station and the strings connect these components to each other.

4.6.2 Integer Programming Formulation

We formulate the SDP as an integer programming problem on the space-time network. The objective of this formulation is to partition the locomotive assignment on each train (locomotive schedule) into flows on the service strings that pass through it while ensuring locomotive flow balance at each station. We now layout the notation and formally define this problem.

Trains: Set of all weekly trains. Indexed by l.

CoArcs: Set of all train connection arcs.

S: Set of service strings.

$e_{l,a}$: Event corresponding to the arrival of train l at a service station.

$e_{l,d}$: Event corresponding to the departure of train l from a service station.

S_l^- : Set of service strings ending with train l.

S_l^+ : Set of service strings beginning with train l.

$a_{l,s} = 1$ if train l is a part of string $s \in S$, 0 otherwise.

c_s : Cost of a locomotive unit traveling on string $s \in S$.

f_l : Number of locomotives assigned on train l in the locomotive schedule.

O: Set of arcs which cross the count-time. The count-time is a reference time at which we count the number of locomotives in the system. It is usually set as Sunday mid-night.

r_s : Number of times string $s \in S$ crosses the count-time.

The decision variables are as follows:

x_s : Number of locomotives assigned to string $s \in S$.

y_j : Number of locomotives assigned on train connection arc $j \in CoArcs$.

G: Ownership cost of a locomotive.

The objective function is:

$$Min \sum_{s \in S} c_s x_s + \sum_{s \in (O \cap S)} Gr_s x_s + \sum_{j \in (O \cap CoArcs)} Gy_j. \tag{4.1a}$$

The constraints are the following:

$$\sum_{s \in S} a_{ls} x_s = f_l, \text{ for all } l \in Trains, \tag{4.1b}$$

$$\sum_{s \in S_l^+} x_s = \sum_{j \in I(head(l))} y_j, \text{ for all } l \in Trains, \tag{4.1c}$$

$$\sum_{s \in S_l^-} x_s = \sum_{j \in O(tail(l))} y_j, \text{ for all } l \in Trains, \tag{4.1d}$$

$$x_s \geq 0 \text{ integer},$$
$$y_j \geq 0. \tag{4.1e}$$

Constraint (4.1b) is the set partitioning constraint that ensures that the locomotive assignment on each train is distributed completely on the service strings that pass through it. Constraints (4.1c) and (4.1d) are the flow balance constraints at each node on the space-time network and they ensure that the total flow entering a node is equal to the total flow leaving a node. The objective function (4.1a) is set up to minimize the total cost of fueling and servicing and also the ownership cost of locomotives used. Railroads operate in such a way that whenever a locomotive fuels, its tank is topped off. Based on this assumption and the average fuel burn rate of each locomotive, the cost of each service string c_s can be computed considering the cost of servicing and fueling at the final station in the string and the costs of fueling at the other fueling stations encountered en-route. Note that once we obtain a solution to the SDP, the arc flows can be converted into flows on cycles using a standard polynomial time cycle decomposition algorithm (Ahuja, Magnanti, and Orlin (1993)). From Property 4.1, it directly follows that each

of these cycles will be fueling and servicing friendly cycles and hence, constitute a solution to the LRP.

We now prove that the SDP is NP-Complete. Clearly, a certificate for the SDP can be verified in polynomial time. We next describe a polynomial-time reduction from the set partitioning problem which is known to be NP-complete (Garey and Johnson (1979)) to a special case of the SDP. The set partitioning problem is defined as follows "Given a universe U of n elements, a collection of subsets of U, $S = \{S_1, ..., S_k\}$, does there exist a sub-collection of S that covers all the elements of U exactly once?" This problem can be formulated mathematically as follows:

x_s : Binary variable which indicates whether a set $s \in S$ is chosen or not

Let $a_{ls} = \begin{cases} 1, \text{ if element } l \in U \text{ is contained in set } s \in S \\ 0, \text{ otherwise} \end{cases}$,

$$\sum_{s \in S} a_{ls} x_s = 1, \text{ for all } l \in U,$$

$$x_s \in \{0,1\}.$$

We reduce the set partitioning problem to a special case of the SDP as follows. For each $l \in U$ we create a train l which departs from and also arrives at service station A. Let the set of strings S be train sequences corresponding to the sets in S. For e.g., If $S_1 = \{l_1, l_2, ..., l_r\}$ then the corresponding string is the sequence of trains $\{l_1, l_2, ..., l_r\}$. We create a space-time network for these set of input trains and strings and set the flow (f_l) on each train to one. Now, note that for this specially constructed SDP, Constraint (4.1b) is identical to set partitioning constraint given above and the flow balance constraints are automatically satisfied because each string originates and terminates at the same station, and can therefore be relaxed. Hence, there is a one-to-one

correspondence between the set partitioning problem and the special case of SDP that we have constructed and the result follows. We state this result in the form of the following theorem:

Theorem 4.2. The string decomposition problem (SDP) is NP-Complete.

In the next section, we address the computational complexities of the problem and also describe algorithmic approaches to solve this problem in an efficient manner.

4.7 A Tractable Solution Approach: Aggregation and Disaggregation

A typical weekly train schedule has around 3,000 trains and the railroad network has around 100 stations out of which roughly 50% support fueling and 30% support servicing. For a problem of this size, the number of service strings runs into several million. Hence, a direct application of string enumeration to this large problem is not promising in terms of computational time and tractability. Another serious computational issue is the inherent complexity of the SDP. In addition to being NP-Complete, this problem has around a million decision variables, one corresponding to each service string. Hence, a direct branch-and-bound approach using commercial software to solve the SDP is not a viable approach.

In this section, we develop a tractable solution approach to solve the LRP. We use the concept of aggregation and disaggregation. The basic idea behind this approach is that instead of solving a large problem directly, we divide our algorithm into two stages. In the first stage, we construct a smaller aggregated model which can be solved quickly. Once we obtain the solution to the aggregated model, we use this information to disaggregate the solution and obtain a solution to the original problem. While this idea is quite intuitive to grasp, there are some issues which need to be considered to ensure that the solution produced using this approach in close to the global optimal solution. We next present our aggregation-disaggregation based algorithm and describe how we use this approach to effectively solve the LRP.

4.7.1 Aggregated Model

The aggregated model is an approximation of the original model. However, this approximate model is much smaller than the original problem and is hence solvable more quickly. This leads to the issue of trading-off between the degree of approximation and the ease of solution, i.e., an aggregate model may solve very quickly but may not produce good solutions or the aggregate model may be almost exact but may take a very long time to produce a good solution. The performance of an aggregated model depends on a few factors which need to be addressed while designing the aggregated model. The first question that needs to be asked while designing an aggregated model is "which entities to aggregate?" The second question that needs to be asked is "how to ensure that the aggregated model is as close as possible to the original problem?" The closer the aggregated model is to the original problem, the easier it will be to disaggregate the aggregate solution and obtain near optimal solutions to the original problem. In the rest of this section, we focus on answering these questions and developing a good aggregation model. We divide our discussion into two parts: we first focus on the feasibility aspect of the problem and in the process address the first question. Then, we focus on the optimality of the problem and in the process address the second question.

The locomotive schedule gives an assignment of locomotive types to weekly trains and is an input to the SDP. The weekly train schedule consists of a set of daily trains, each with a pre-specified frequency varying between one and seven. Note that all instances of a daily train in the week are similar in terms of origin station, destination station, departure time of the day, arrival time of the day, and duration. The only factors which differentiate these trains are the departure day of the week, arrival day of the week, and possibly their locomotive type assignments. Based on this observation, we aggregate all the instances of a daily train in the week (weekly trains) into a single aggregated daily train. Each aggregated train departs at the corresponding departure

time of the day and arrives at the corresponding arrival time of the day. The total flow on each daily aggregated train represents the total flow on all weekly trains corresponding to it. The locomotive schedule obtained after aggregation is called the aggregated locomotive schedule and the set of trains in this schedule constitute the aggregated train schedule. For e.g., suppose daily train TR01 has a weekly frequency of three and operates on Monday, Tuesday, and Thursday (TR01-Monday, TR01-Tuesday, and TR01-Thursday) and suppose each of these weekly trains carries a flow of one locomotive of type A and one of type B. Then in the aggregated locomotive schedule, aggregated daily train TR01 will have a flow of three units of type A and three units of type B. We refer to a string which is made up of daily trains as an aggregated string.

We formulate an aggregated string decomposition model to decompose the aggregated locomotive schedule into flows on aggregated strings. The decision variables (or aggregated strings) of the aggregated model are enumerated using the aggregated train schedule (instead of the weekly train schedule). We next list the steps to formulate the aggregated string decomposition model:

- Enumerate the set of service strings using the aggregated train schedule as input and the service string enumeration algorithm described in Section 4.4.
- Construct a space-time network as described in Section 4.5 but considering the aggregated strings (instead of strings) and the aggregated train schedule (instead of the train schedule).
- Formulate the aggregated string decomposition problem as described in Section 4.5 but using the aggregated locomotive schedule (instead of the locomotive schedule) and the aggregated strings (instead of strings).

Note that the aggregated model is identical in structure to the original model. However, the size of the aggregated model is several times smaller. The reason for that is aggregation happens at two levels: (1) decision variables, and (2) constraints. For e.g., consider an aggregated string made up of daily trains A, B, and C. Let each of these trains have a frequency of four. Then, there are 4 x 4 x 4 = 64 different strings possible in the original network using the weekly trains

corresponding to A, B, and C (Refer Figure 4-5). However, the aggregated formulation considers only one aggregate string instead of the 64 strings. Hence, the flow on the aggregated string in this case represents the cumulative flow on these 64 strings. The decision variables corresponding to the connection arcs (CoArcs) are also aggregated in a similar manner. Using this method of variable aggregation, we are able to reduce the number of variables to a few thousand. Also, in the aggregated model we have only one partitioning constraint for every daily train instead of one for every weekly train in the original model. Since the average frequency of daily trains is close to five, the number of partitioning constraints becomes roughly one-fifth that of the original model. Also, since the number of trains in the aggregated model decreases by a factor of one-fifth, the number of flow balance constraints also reduces by the same factor. Thus, the aggregation of variables and constraints results in a very compact formulation and consequently very fast run times.

We now proceed to state and prove two important properties relating the feasibility of the aggregated problem and the feasibility of the original problem.

Property 4.3. There exists a feasible solution to the string decomposition problem only if there exists a feasible solution to the aggregated string decomposition problem.

Proof: Consider a feasible solution F to the string decomposition problem. We can obtain a feasible solution to the aggregated string decomposition problem by assigning each aggregated string a flow equal to the total flow on all strings corresponding to it in F. Since F partitions the flow on all weekly trains and satisfies flow balance at each station, it follows that the assignment to each aggregated string computed above will partition the flow on each aggregated train and also satisfies flow balance at each station. The result follows. ◆

Property 4.4. There exists a feasible solution to the aggregated string decomposition problem only if there exists a feasible solution to the string decomposition problem.

Proof: Consider a feasible solution F to aggregated string decomposition. We can obtain a feasible solution to string decomposition by distributing the flow on each aggregated string in F on the strings corresponding to it in the original network in such a way that the flow on all weekly trains is partitioned. Since, we merely re-distribute the flows on the aggregated strings, the total flow entering a station and the total flow leaving a station in both the aggregated solution and the new solution will be the same. Hence, flow balance at each station is also satisfied and the result follows. ◆

We state the results of Property 4.3 and Property 4.4 in the form of the following theorem.

Theorem 4.5. There exists a feasible solution to the string decomposition problem if and only if there exists a feasible solution to the aggregated string decomposition problem.

We have thus showed the equivalence between the feasibility version of the SDP and the feasibility version of the much smaller aggregated SDP and have hence addressed the feasibility aspect of the problem mentioned at the start of this section. The other aspect we need to address is optimality. The objective function of the SDP minimizes total fueling and servicing costs and the ownership costs of locomotives. We now describe methods to ensure that the aggregated model takes these costs into consideration.

When we create an aggregated model for a problem, we lose some information about the original problem. The biggest approximation in the aggregated string decomposition model is that the information about departure day of the week and the arrival day of the week of weekly trains are lost. This could lead to the aggregated problem generating bad solutions in terms of locomotive usage. For e.g. Consider a weekly train A which arrives at a station on Monday at

10:00 AM and another weekly train B that departs from the same station at 11:00 AM on Sunday. The optimal solution of the SDP is not likely to use the connection between these weekly trains to route locomotives because the idling time of the locomotive would be extremely high (six days and one hour). On the other hand, in the aggregated train network, since we do not consider the arrival and departure days of trains and only consider the arrival and departure times, the connection time between the corresponding aggregated trains will be just one hour and hence it is very likely to be used in the solution since it seems to represent an efficient use of the locomotive resource. However, this is an incorrect representation of reality. We hence define an estimated connection time for each train connection on the aggregated network. The estimated connection time between daily trains A and B is computed to represent the most probable connection time if a locomotive connects between a weekly train corresponding to train A and a weekly train corresponding to train B. Note that we cannot compute this value exactly unless both A and B have a frequency of one. We represent the estimated connection time of a train connection j by t_j. Each string is made up of possibly several train connections. We estimate the duration of an aggregated string (t_s) by adding the exact durations of the trains belonging to it and the estimated connection times (t_j) for the connections that are a part of it. We next describe how to use these estimated durations to improve the aggregated model.

Consider the objective function of the SDP. The first term $\sum_{s \in S} c_s x_s$ represents the cost of fueling and servicing for the locomotives traveling on strings. The computation of c_s is a function of the stations encountered en-route, the cost of fueling or servicing at these stations, and the length of trains that belong to the string and this logic is hence directly applicable to the aggregated strings. The second part of the objective function $\sum_{s \in (O \cap S)} Gr_s x_s + \sum_{j \in (O \cap CoArcs)} Gy_j$

minimizes the total ownership cost of locomotives by counting the number of locomotives traveling on overnight arcs. Note that in the aggregated problem, it is not straight-forward to determine the set of overnight arcs or the number of times an aggregated string crosses the count-time (r_s). This is because the daily trains do not have arrival and departure day information. We handle this difficulty in the following manner. We first re-write the objective function of the SDP in a slightly different though equivalent form. Instead of minimizing the ownership cost of locomotives flowing on overnight arcs, we now construct the objective function in terms of minimizing the total cost of locomotive-minutes used on all arcs. In the aggregated SDP, since we cannot know the exc durations, we use the estimated duration of train connections (t_j) and the estimated duration of strings (t_s) described in the previous paragraph to set up the objective function. The objective function now becomes:

$$\text{Min } \sum_{s\in S} c_s x_s + \sum_{s\in S} G' t_s x_s + \sum_{j\in CoArcs} G' t_j y_j = \sum_{s\in S} c_s' x_s + \sum_{j\in CoArcs} G' t_j y_j .$$

Using this objective function in the aggregated model makes it a much better approximation of the original problem. Next, we focus on the second stage of our approach which concerns the disaggregation of the aggregated solution.

4.7.2 Disaggregation Model

We now consider the problem of disaggregating the solution to the aggregated problem to obtain a solution to the original problem. The proof of Property 4.4 describes a straightforward method to obtain a feasible string decomposition starting with a solution to the aggregated problem; however this method may give solutions which use a high number of locomotives and are hence not optimal. The method we describe in this section disaggregates the solution while minimizing the number of locomotives used. Note, that after the aggregation stage, the total fueling and servicing costs are already fixed and do not change in the disaggregation stage.

The inputs to the disaggregation model are:

- Locomotive schedule: This gives the assignment of locomotive types to weekly trains.
- Solution of the aggregated model: This gives the aggregated strings and their locomotive assignments.

The objective of the disaggregation model is to obtain a near optimal solution to the string decomposition problem; i.e. the objective is to re-distribute the flow on each aggregated string on the strings corresponding to it in the original network while ensuring that the assignment on each weekly train is covered and minimizing the total number of locomotives used. We next formulate the disaggregation model as an integral multicommodity flow problem with side constraints on a suitably defined network.

We denote the network as G (N, A) where N is the set of nodes and A is the set of arcs in the network. Each node on the network corresponds to an event. The events that we consider are the arrival of weekly trains at stations and departure of weekly trains from stations. We create a train-arrival node corresponding to each train arrival and a train-departure node corresponding to each train departure. We also construct one supply node and one demand node corresponding to each aggregated string with positive flow ($x_s > 0$) and define a commodity corresponding to this string. We set the supply of the corresponding commodity at each supply node to x_s and the supply of all other commodities to zero. Similarly, we set the demand of the corresponding commodity at each demand node to x_s and the demand of all other commodities to zero.

We now describe how to construct arcs. For each weekly train, we construct an arc connecting its train-departure node and train-arrival node. We also construct connection arcs (CoArcs) between train-arrival nodes and train-departure nodes at each station to model the different possibilities for train connections. Let first(s) be the set of weekly trains corresponding to the first daily train in aggregated string s, last(s) be the set of weekly trains corresponding to

the last daily train in aggregated string s, and trains(s) be the set of weekly trains corresponding to all the daily trains in s. For each aggregated string s with positive flow ($x_s > 0$), we connect its supply node to the departure nodes of all the trains in first(s) and we also connected the arrival nodes of all trains in last(s) to the demand node corresponding to s. The disaggregation model can be formulated as a multicommodity flow problem with side constraints (MCFSC) on the network constructed above.

N: Set of nodes in the network.

A: Set of arcs in the network. We represent each arc by (i, j).

Trains: Set of weekly trains in the network. Indexed by l.

CoArcs: Set of connection arcs in the network.

K: Set of commodities indexed by k. We define one commodity for each aggregated string s with positive flow.

s(k): Represents the aggregated string corresponding to commodity $k \in K$.

Trains(s): Set of weekly trains corresponding to the daily trains in s.

f_l: Total locomotive assignment on weekly train $l \in Trains$ in the locomotive schedule.

b^k: Supply/demand vector of commodity $k \in K$.

u_{ij}^k: Upper bound on the flow of commodity $k \in K$ on arc $(i, j) \in A$.

$$u_{ij}^k = \begin{cases} 0, \text{ if } (i,j) \in Trains \text{ and } (i,j) \notin s(k) \\ f_{ij}, \text{ if } (i,j) \in Trains \text{ and } (i,j) \in s(k) \\ \infty, \text{ otherwise} \end{cases}.$$

t_{ij}: Duration of arc $(i, j) \in A$.

N: Arc adjacency matrix of G(N,A).

The decision variables are:

x_{ij}^k: Flow of commodity $k \in K$ on arc $(i, j) \in A$.

The objective function is

$$\text{Min} \sum_{k \in K} \sum_{(i,j) \in A} t_{ij} x_{ij}^k. \tag{4.2a}$$

The constraints are:

$$\sum_{k \in K} x_{ij}^k = f_{ij}, \text{ for all } (i, j) \in Trains, \tag{4.2b}$$

$$Nx^k = b^k, \text{ for all } k \in K, \tag{4.2c}$$

$$0 \le x_{ij}^k \le u_{ij}^k, \; x_{ij}^k \text{ integer.} \tag{4.2d}$$

Note that the flow balance constraints at supply nodes, demand nodes, and transshipment nodes (Constraint 4.2c), and also setting flow upper-bounds on trains as described above ensures that the flow on each aggregated string is redistributed completely on the strings corresponding to it. Also, Constraint (4.2b) ensures that the total flow on each weekly train is partitioned among the disaggregated strings that pass through it. Hence, MCFSC is a valid formulation for the disaggregation problem. The objective function is set up to minimize the total number of locomotive minutes used.

Theorem 4.6 There is a one-to-one correspondence between feasible solutions to the disaggregation problem and feasible solution to MCFSC.

The MCFSC is an integral multicommodity flow problem and it is hence NP-Complete (Garey and Johnson (1979)). Note that Constraints (4.2) are the bundling constraints which link more than one commodity together and complicate the model and if these constraints were relaxed, then this problem reduces to |K| independent minimum cost flow problems. We hence adopt a sequential commodity-by-commodity approach to obtain a feasible solution to the disaggregation problem. We first sort the aggregated strings with positive flow in the decreasing

order of their lengths, where length is defined as the number of trains in the string. The reason we do this is that the aggregated stings with longer lengths have a larger impact on the objective function cost and we would like to optimize them first before moving to shorter strings. For each aggregated string s, we consider the sub-graph $G(N^{k(s)}, A^{k(s)})$ which contains only those train arcs for which $u_{ij}^{k(s)} > 0$ and $f_{ij} > 0$. We solve a minimum cost flow problem on this sub-graph, extract out the flows on the disaggregated strings, and update the total unaccounted flow on each train (f_{ij}). This is repeated until the flow on all aggregate strings is disaggregated. The disaggregation algorithm is formally stated in Figure 4-6.

The algorithm for disaggregation solves |K| independent minimum cost flow problems sequentially and is hence a polynomial time algorithm. The feasibility correctness of this algorithm follows from the fact that it is a special implementation of the general method described in the proof of Property 4.3 for obtaining a feasible solution to string decomposition problem starting from a feasible solution to the aggregated string decomposition problem. The following result is hence immediate.

Theorem 4.7. The disaggregation algorithm produces a feasible solution to the string decomposition problem starting from a feasible solution to the aggregated string decomposition problem, in polynomial time.

Our computational results show that the aggregation-disaggregation method solves the fueling and servicing problem very quickly and produces near optimal solutions.

4.8 Handling Infeasibilities

The infeasibilities that are encountered while solving the fueling and servicing LRP can be of two kinds: hard infeasibilities and soft infeasibilities. Hard infeasibilities are due to the nature of the train schedule and cannot be fixed without modifying the data. For example, consider a

situation where the train schedule is such that there does not exist a service string passing through a particular train. This implies that any locomotive unit which is assigned to this train cannot be serviced and fueled feasibly. Since fixing this infeasibility requires modifying the train schedule, we do not consider this kind of infeasibility in our research.

Soft infeasibilities are introduced due to the two stage decomposition approach to solve the fueling and servicing LRP, i.e. we first solve the LPP and then the LRP, instead of solving the integrated problem. Since, the LPP completely ignores the fueling and servicing constraints, this may lead to the locomotive schedule being fueling and servicing incompatible while solving the LRP. Soft infeasibilities can manifest themselves in the following two ways:

- A train which has a service string passing through it in the train network does not have a service string passing through it in the decomposed train network corresponding to a particular locomotive type assigned on it.
- A train has service strings passing through it but in the decomposed network but the problem is still infeasible due to the nature of the set partitioning constraints.

In order to handle this problem, we develop side-constraints that can be added to the LPP so that it accounts for fueling and servicing feasibility to a greater extent. These constraints ensure that some necessary conditions for fueling and servicing feasibility are satisfied. We refer the reader to Ahuja et al. (2005) and Vaidyanathan et al. (2007b) for more details about the locomotive planning model; here we describe the side-constraints that we add to the model.

- We enumerate a large number of service strings and only allow the set of train connections R contained in them while solving the LPP. Any locomotive which makes a train connection that is not a part of R cannot be fueled and serviced feasibly. Therefore, while solving the LPP, we ensure that only those connections which are a part of R are allowed. This approach eliminates local infeasibilities. For e.g. suppose the length of train A + the length of train B is more than the fueling threshold F, then the connection between these two trains will not be a part of R and will hence not be constructed.
- Consider a situation where a set of trains have only one string passing through them. Let train A be one such train and s be the string passing through it. From the first case, it is clear that all train connections which are a part s will be allowed in the planning problem. Note that if the flow on any of these train connections is less than the flow on train A, then

the locomotives assigned on train A cannot fueled and serviced feasibly. We therefore add side-constraints to ensure that this does not happen; i.e. we ensure that the flow on each connection in s is greater than the flow on train A.

- The first method handles local infeasibilities. However, it does not take into account higher level infeasibilities which are a function of more than one train connection. For e.g. Consider a train A which has two strings passing through it as shown in Figure 4-7. The first string contains the sequence 11–A–12 and the second string contains the sequence 21–A–22 as shown in the figure (numbers represent train connections and letters represent trains). Note that the first method would allow all four train connections to be used in the LPP and the LPP has no constraints which prevent a locomotive from being routed along the sequence 11–A–22 or 21–A–12, even though these sequences are not feasible sequences. We now formally define constraints that we add to the planning problem to ensure that flow does not occur on these kinds of infeasible sequences.

I(A): Set of incoming train connections at the tail node of train A.

O(A): Set of outgoing connections at the head node of train A.

C: Set of all locomotive types. Indexed by c.

y_j^c: Flow of locomotive type c on connection j.

FO(j): Set of fueling and servicing feasible outgoing connections in set O(A) which correspond to $j \in I(A)$.

FI(j): Set of fueling and servicing feasible outgoing connections in set I(A) which correspond to $j \in O(A)$.

We add the following constraints to the planning model for each train:

$$y_j^c \le \sum_{l \in FO(j)} y_l^c, \text{ for all } j \in I(A), c \in C,$$

$$y_j^c \le \sum_{l \in FI(j)} y_l^c, \text{ for all } j \in O(A), c \in C.$$

These constraints help eliminate higher level infeasibilities which are a function of two train connections.

Our computational results show that the necessary conditions described in this section are very effective in ensuring that the locomotive schedule can be decomposed into a fueling and

servicing friendly routing. In the next section, we present computational results to demonstrate the performance of all the algorithms developed and show that our approach is indeed a quick and effective approach to solve the locomotive fueling and servicing problem.

4.9 Computational Results

In this section, we provide computational results of our algorithms on several instances and also some case studies on a representative instance. We implemented our algorithms in the Microsoft Visual Basic programming language and tested them on the data provided by a major Class I US railroad. We modeled our integer programs using ILOG Concert Technology 2.0 modeling language and solved them using the CPLEX 9.0 solver. We conducted all our computational tests on an Intel Pentium 4, 512-MB RAM, and 2.4-GHz processor desktop computer.

4.9.1 Testing the Performance of the Algorithm

We test the performance of our locomotive routing algorithm on instances with different transport volumes. In order to estimate the performance of our algorithms, we first solve the LPP to obtain a locomotive schedule. Next, we solve the LRP using this locomotive schedule as a starting point and evaluate the locomotive routing cost as defined in the LPP (we do not consider fueling and servicing costs). Note that the solution cost of the LRP will always be greater than that of the LPP. However, the smaller the difference in cost between these two solutions, the better our algorithmic approach. We use this measure to estimate the performance of our algorithms and conduct our computational tests in the following sequence of steps for each instance:

- Solve the LPP to obtain a locomotive schedule
- Run the LRP on the locomotive schedule obtained from the LPP and compare the solution cost

Notice that, suppose the difference in cost between the solution of the LRP and solution of the LPP is small, then this implies that the fueling and servicing algorithms are performing well since we are able to enforce the fueling and servicing constraints without a significant increase in the solution cost. The percentage difference in cost hence serves as an upper bound on the optimality gap of the LRP.

In Table 4-1 and Figure 4-8, we present the results of our computational tests. From our computational tests, we draw the following conclusions:

- We are able to achieve a fueling and servicing friendly routing on all instances with less than 2.2% increase in the locomotive routing cost.
- The performance of our locomotive routing algorithm is consistent with a computational time of less than 5 minutes on all instances.
- The locomotive routing algorithm is able to achieve a 100% fueling and servicing friendly routing on all instances.

4.9.2 Case Study

We present a case study to demonstrate the uses of the locomotive routing model. We perform this case study on a representative instance with 2,824 trains, 77 stations, 39 fueling stations, 28 servicing stations, and 6 locomotive types. The various aspects of the problem that we study are reported in the following sections.

Effect of varying the servicing distance threshold (S)

In this study, we quantify the effect of varying the maximum distance that can be traveled before servicing (S) on the solution cost of the LRP. While fueling is a hard constraint since tank capacities of locomotives are well defined, servicing constraints are soft constraints since maintenance of units can usually be delayed to a later time. Table 4-2 presents the computational results, and Figure 4-9 shows the relationship between the servicing threshold and the solution cost.

From our computational tests we draw the following conclusions:

- As the servicing threshold is increased from 1,800 miles to 3,000 miles, the initial increase has a large impact on the solution quality, but subsequently the law of diminishing returns is observed.
- Beyond a servicing threshold of around 2,000 miles there is almost no impact of increasing the servicing threshold on the solution cost.
- The performance of our locomotive routing algorithm is consistent with a computational time of around 5 minutes on all instances.

Effect of adding/removing fueling facilities

In this study, we quantify the impact of varying the number of fueling facilities on the solution cost of the LRP. We start with the initial set of 39 fueling stations. To this set we add (or remove) fueling stations one-by-one and measure the impact on the cost of locomotive routing. When we add fueling stations, we add them in the decreasing order of number of inbound trains. On the other hand, when we remove fueling stations, we remove them in the increasing order of number of inbound trains. The reason we add and remove fueling stations in this manner is because when we add fueling stations, we want to first add those stations which are likely to have a greater impact on the solution cost. On the other hand, when we remove fueling stations, we want to first remove those stations which are likely to have a smaller impact on the solution cost. Table 4-3 presents the computational results, and Figure 4-10 shows the relationship between the number of fueling stations and the solution cost.

From our computational tests we draw the following conclusions:

- As the number of fueling stations is increased from 33 to 43, the initial additions have a large impact on the solution cost, but subsequently the law of diminishing returns is observed.
- Solution cost remains constant for the number of fueling stations varying between 35 and 40. This suggests that we could bring down the number of fueling stations from the current value of 39 to 35 without compromising much on the cost of locomotive routing. This could potentially translate into savings of several million dollars.

- The performance of our locomotive routing algorithm is consistent with a computational time of around 5 minutes on all instances.

Effect of adding/removing servicing facilities

In this study, we quantify the impact of varying the number of servicing facilities on the solution cost of the LRP. We start with the initial set of 28 servicing stations. To this set we add (or remove) servicing stations one-by-one and measure the impact on the cost of locomotive routing. When we add servicing stations, we add them in the decreasing order of number of inbound trains. On the other hand, when we remove servicing stations, we remove them in the increasing order of number of inbound trains. The reason we add and remove servicing stations in this manner is that when we add servicing stations, we want to first add those stations which are likely to have a larger impact on the solution cost. On the other hand, when we remove servicing stations, we want to first remove those stations which are likely to have a smaller impact on the solution cost first. Table 4-4 presents the computational results, and Figure 4-11 shows the relationship between the number of servicing stations and the solution cost.

From our computational tests we draw the following conclusions:

- As the number of servicing facilities is increased from 21 to 32, the initial additions have a large impact on the solution cost, but subsequently the law of diminishing returns is observed.
- Solution cost remains constant for the number of servicing stations varying between 24 and 28. This suggests that we could bring down the number of servicing stations from the current value of 28 to 24 without compromising much on the cost of locomotive routing. This could potentially translate into savings of several million dollars.
- The performance of our locomotive routing algorithm is consistent with a computational time of around 5 minutes for all instances.

4.10 Summary and Conclusions

Our previous research on locomotive planning focused on creating locomotive schedules which satisfy several business and operational constraints. While that research on the LPP has

several potential benefits, it has the drawback that it totally neglects fueling and servicing requirements and may hence generate solutions which are not implementable. We bridge this gap by considering fueling and servicing constraints and also develop optimization techniques in order to incorporate the fueling and servicing logic in our methodology to generate implementable locomotive routing.

Due to the inherent complexity of the integrated problem of locomotive planning and locomotive routing, we decompose our approach to achieve fueling and servicing feasibility into two stages. The first phase involves solving the LPP (LPP) using the algorithms described in Ahuja et al. (2005) and Vaidyanathan et al. (2007b). Once we have the solution to the LPP, we fix the assignment of locomotive types to trains and use the flexibility of train connections in order to enforce fueling and servicing feasibility in the second locomotive routing phase. We adopt the following approach to solve the LRP in the second phase. We enumerate the set of fueling and servicing feasible paths (or strings) on the network using dynamic programming algorithms and formulate the LRP as a string decomposition integer programming problem. Due to its prohibitive size and NP-Complete nature, this problem cannot be directly solved using commercial optimization software. We handle this computational complexity by developing an aggregation-disaggregation based approach to solve this problem. Our computational results on real-life instances demonstrate that our algorithms enforce the fueling and servicing requirements with a run-time in the order of minutes and with less than 2.2 % increase in the locomotive routing costs. We also demonstrate the usefulness of the model to perform strategic analysis and give examples of the kind of insights that can be drawn from such analysis.

We believe that this research is a vital stepping stone in our overall objective of enabling railroads to manage their locomotive operations more efficiently through optimization methods

which produce practical and implementable solutions. We believe that our collective research in this area will lead to the large scale deployment of optimization based locomotive management software at North American railroads.

Table 4-1. Performance of the locomotive routing algorithm.

SNo	Mean tonnage	Locomotive Planning Problem		Locomotive Routing Problem				% Increase in cost
		Cost ($)	# of locos	Cost ($)	# of locos	# of infeasibilities	Time	
1	5,700	6,068,767	1,520	6,192,415	1,566	0	3 m 41 s	2.04
2	6,080	6,444,686	1,591	6,565,983	1,632	0	4 m 07 s	1.88
3	6,460	6,861,699	1,646	6,982,995	1,687	0	3 m 59 s	1.76
4	6,840	7,161,102	1,683	7,317,678	1,733	0	4 m 20 s	2.19
5	7,220	7,598,937	1,688	7,755,512	1,738	0	4 m 07 s	2.06
6	7,600	8,025,096	1,738	8,184,696	1,785	0	3 m 55 s	1.99
7	7,980	8,501,004	1,717	8,687,484	1,774	0	4 m 07 s	2.19

Table 4-2. Effect of varying servicing distance threshold.

SNo	Threshold S (miles)	Cost ($)	# of locos	# of infeasibilities	Time
1	1,800	8,752,593	1,679	121	4 m 59 s
2	1,900	8,600,418	1,695	100	4 m 40 s
3	1,950	8,422,679	1,736	52	4 m 39 s
4	2,000	8,201,159	1,775	0	4 m 40 s
5	2,200	8,201,159	1,775	0	5 m 11 s
6	2,400	8,192,759	1,776	0	4 m 40 s
7	2,600	8,195,447	1,777	0	4 m 44 s
8	2,800	8,195,447	1,777	0	4 m 30 s
9	3,000	8,179,319	1,771	0	4 m 32 s

Table 4-3. Effect of varying number of fueling stations.

SNo	Number of facilities	Cost ($)	# of locos	# of infeasibilities	Time
1	33	8,253,831	1,767	14	4 m 56 s
2	34	8,253,831	1,767	14	4 m 59 s
3	35	8,179,319	1,771	0	4 m 45 s
4	36	8,179,319	1,771	0	4 m 40 s
5	37	8,179,319	1,771	0	5 m 05 s
6	38	8,179,319	1,771	0	5 m 12 s
7	39	8,179,319	1,771	0	4 m 41 s
8	40	8,179,319	1,771	0	5 m 03 s
9	41	8,173,943	1,769	0	4 m 36 s
10	42	8,168,567	1,767	0	4 m 47 s
11	43	8,168,567	1,767	0	5 m 10 s

Table 4-4. Effect of varying the number of servicing stations.

SNo	Number of facilities	Cost ($)	# of locos	# of infeasibilities	Time
1	21	8,490,764	1,748	44	4 m 22 s
2	22	8,319,852	1,770	16	4 m 40 s
3	23	8,306,412	1,765	16	4 m 26 s
4	24	8,182,704	1,763	4	4 m 35 s
5	25	8,179,139	1,771	0	4 m 47 s
6	26	8,179,139	1,771	0	4 m 50 s
7	27	8,179,319	1,771	0	4 m 36 s
8	28	8,179,319	1,771	0	4 m 32 s
9	29	8,158,151	1,759	0	3 m 27 s
10	30	8,147,063	1,759	0	3 m 28 s
11	31	8,130,935	1,753	0	3 m 29 s
12	32	8,112,119	1,746	0	3 m 43 s

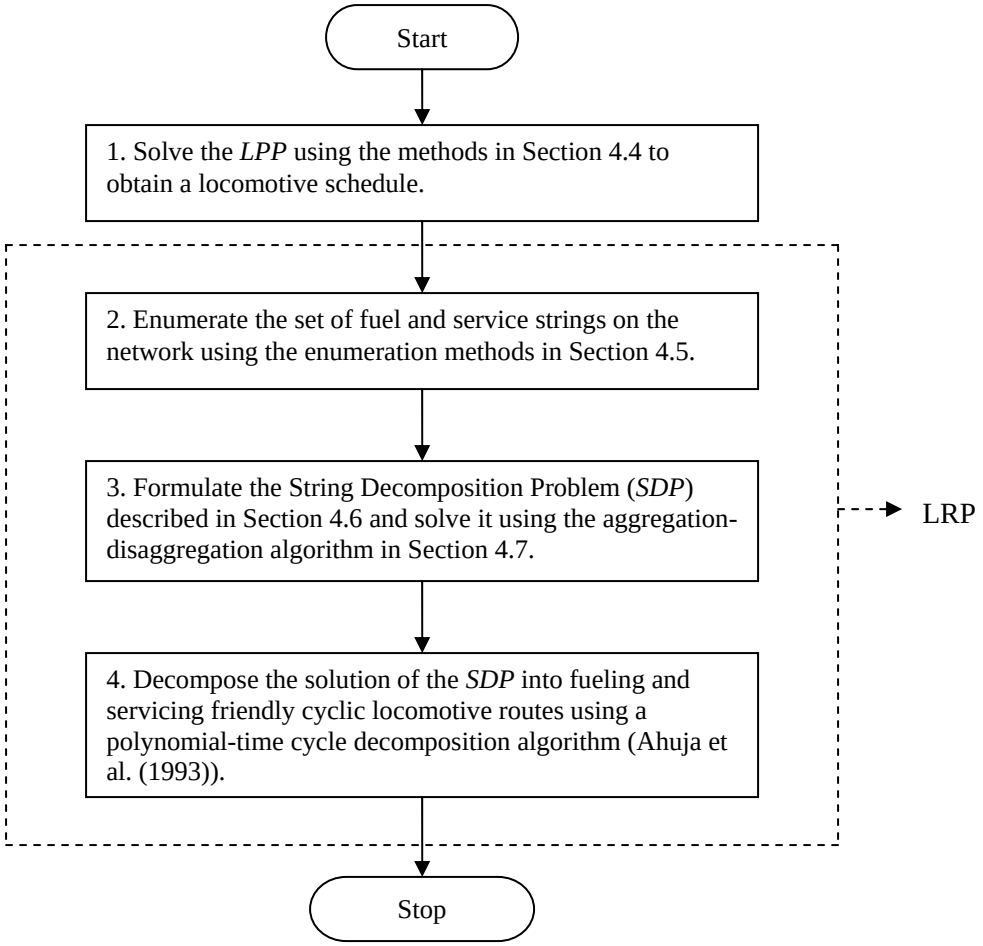

Figure 4-1. Flowchart of the fuel and service routing algorithm.

algorithm *fuel string enumeration*;
begin
  k =1;
  $FS = \phi$;
  $P^1 = \phi$;
  for each $l \in Trains$ **do**
    if l is a fuel string, $FS = FS + l$;
    else if *origin(l)* is a fueling station **then** $P^1 = P^1 + l$;
  while $P^k \neq \phi$ **do**
  begin
    $P^{k+1} = \phi$;
    for all $p \in P^k$ **do**
      let r = last train in valid sequence p;
      for all $l \in Trains : destination(r) = origin(l)$ **do**
        if $length(p \cup l) \leq F$ **and** $l \notin p$ **then**
        begin
          if $destination(p \cup l)$ is a fueling station **do**
            $FS = FS + (p \cup l)$;
          else
            $P^{k+1} = P^{k+1} + (p \cup l)$;
        end
      end-for
    end-for
    k=k+1;
  end
  return *FS*;
end

Figure 4-2. Fuel string enumeration algorithm.

algorithm *service string enumeration*;
begin
k =1;
$SS = \phi$;
$P^1 = \phi$;
Run the *FSE* algorithm to obtain the set of fuel strings *FS*;
for each $l \in FS$ **do**
if *l* is a service feasible string, $SS = SS + l$;
else if *origin(l)* is a servicing station **then** $P^1 = P^1 + l$;
while $P^k \neq \phi$ **do**
begin
$P^{k+1} = \phi$;
for all $p \in P^k$ **do**
let *r* = last train in valid sequence *p*;
for all $l \in FS : destination(r) = origin(l)$ and $Trains(l) \notin p$ **do**
if $length(p \cup l) \leq S$ **then**
begin
if $destination(p \cup l)$ is a servicing station **do**
$FS = FS + (p \cup l)$;
else
$P^{k+1} = P^{k+1} + (p \cup l)$;
end
end-for
end-for
k =k+1;
end
return *SS*;
end

Figure 4-3. Service string enumeration algorithm.

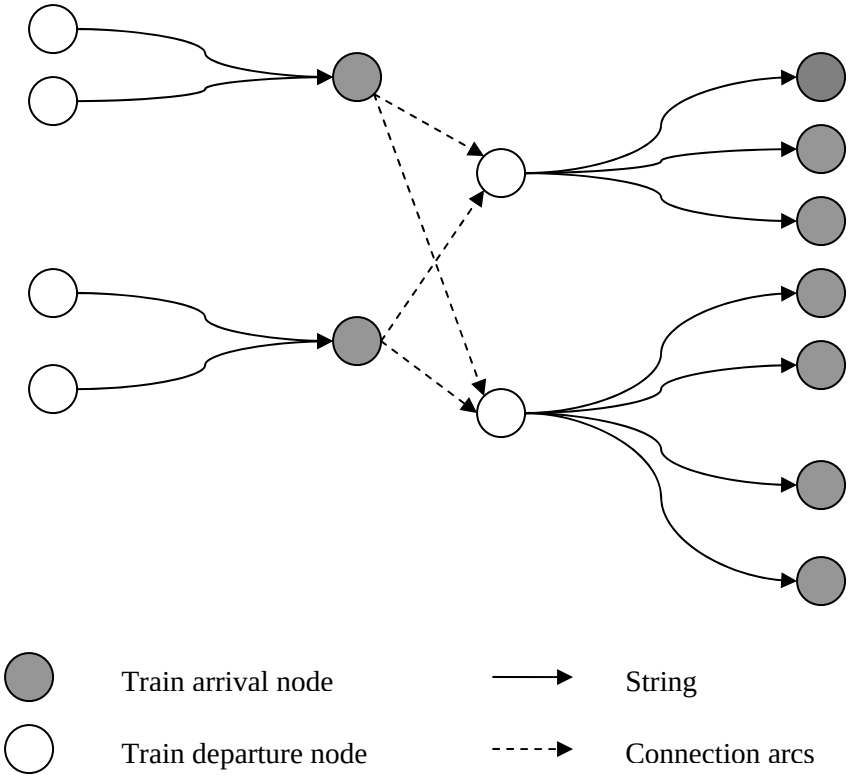

Figure 4-4. A part of the space-time network at a particular service station.

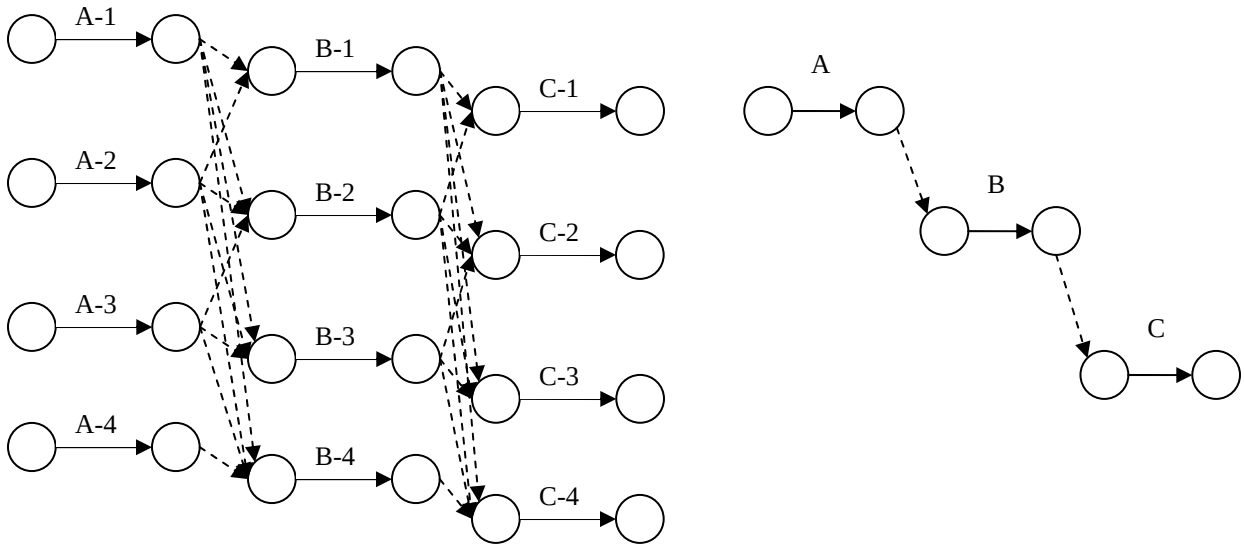

Figure 4-5. Disaggregated strings and corresponding aggregated string.

algorithm *disaggregation*;
begin
Let *S* represent the set of aggregated service strings with positive flow. Let this set be indexed by *s*;
Let $SS = \phi$ represent the set of disaggregated service strings;
Sort *S* in decreasing order of string length;
for each $s \in S$ **do**
Let *k = K(s)*;
Consider the sub-graph of *G(N,A)* induced by trains *l* for which $u_l^k > 0$ and $f_{ij} > 0$. Let this sub-graph be denoted by $G(N^k, A^k)$;
Solve a minimum cost flow problem on this sub-graph. Let the flow on each arc in the optimal solution be x_{ij}^k;
Decompose x_{ij}^k into flows on paths *P*. Let $SS = SS \cup P$;
Update the uncovered flow on each train $f_{ij} = f_{ij} - x_{ij}^k$, for all $(i, j) \in Trains$;
end for
return *SS*;
end

Figure 4-6. Disaggregation algorithm.

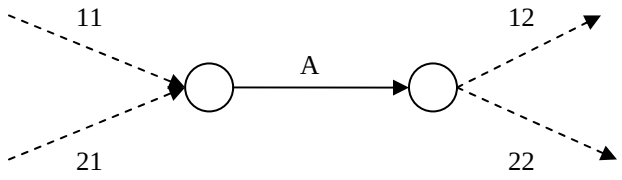

Figure 4-7. Example of infeasibility.

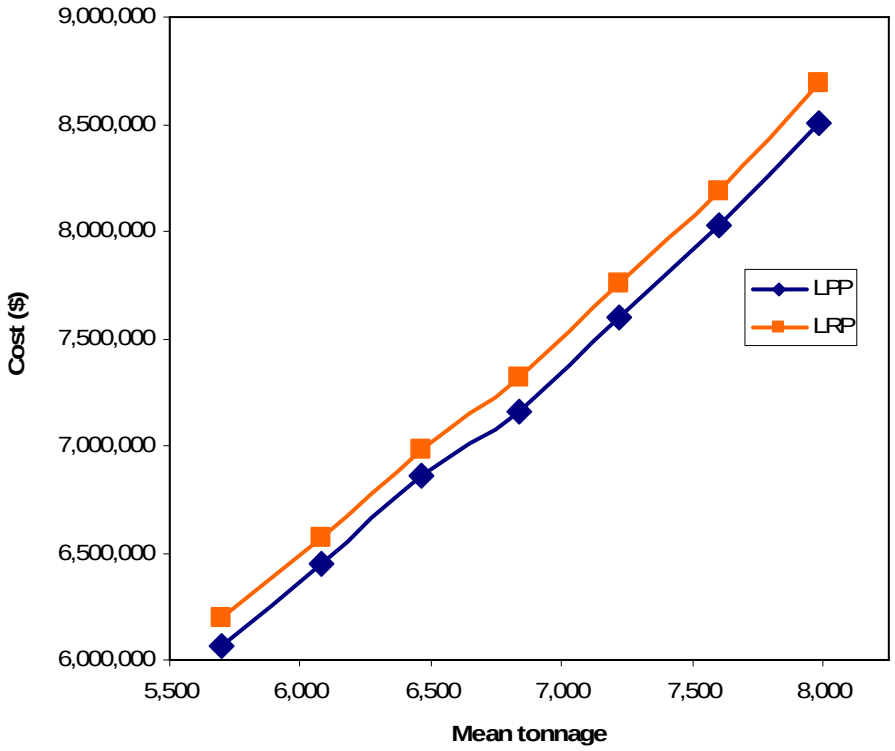

Figure 4-8. Performance of the locomotive routing algorithm.

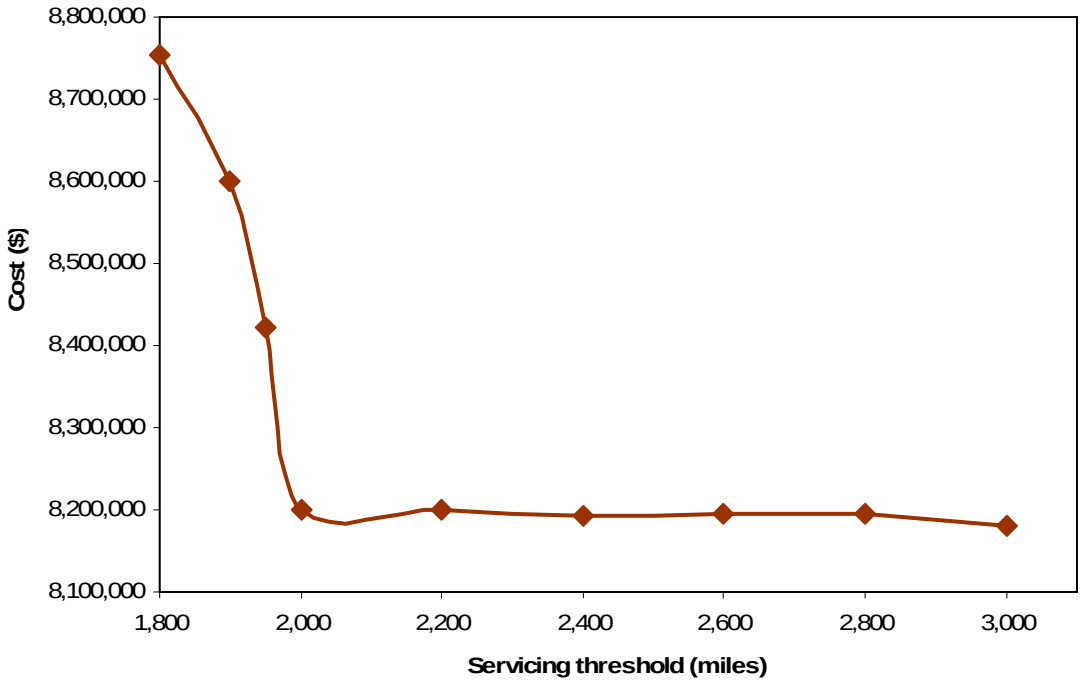

Figure 4-9. Solution cost vs. servicing threshold.

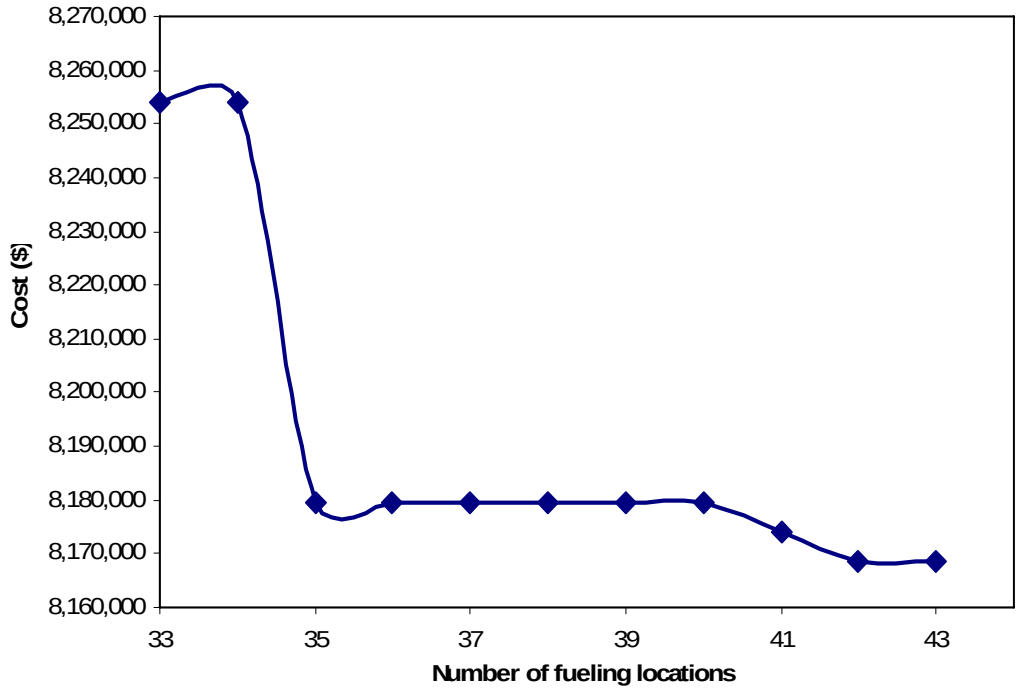

Figure 4-10. Solution cost vs. number of fueling locations.

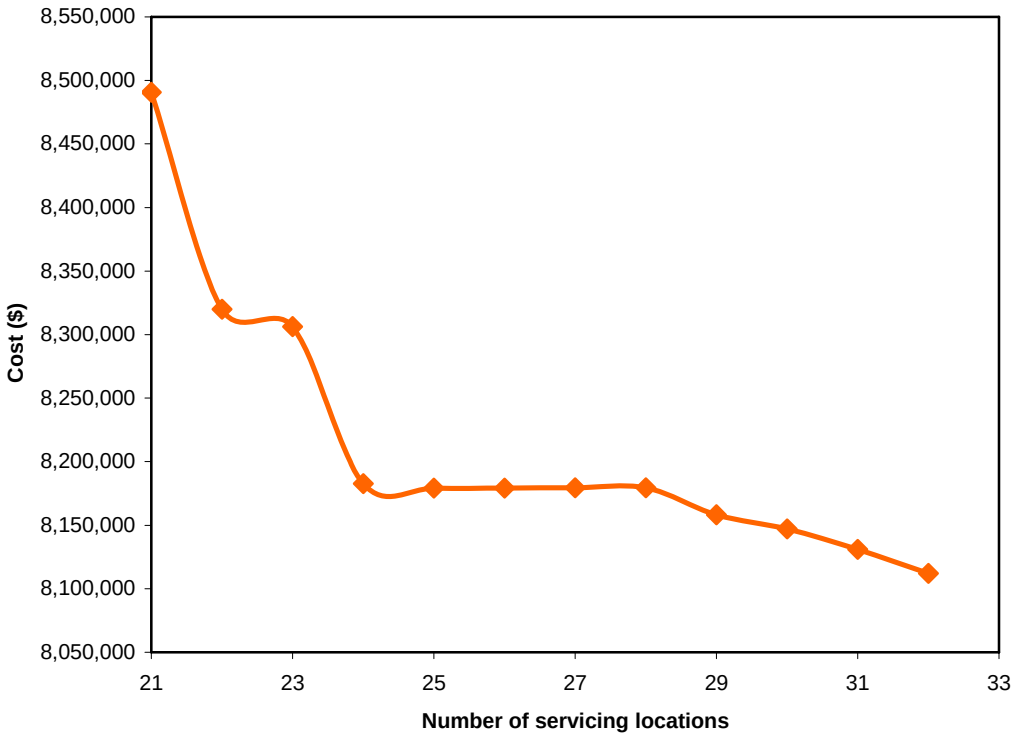

Figure 4-11. Solution cost vs. number of servicing locations.

CHAPTER 5
AGGREGATION FRAMEWORK TO SOLVE LARGE-SCALE INTEGER MULTICOMMODITY FLOW PROBLEMS

5.1 Introduction

The multicommodity flow problem (MCFP) is a generalization of the classical minimum cost flow problem. These problems are frequently encountered in application domains like communication networks, railroad transportation networks, and distribution networks where numerous physical commodities, vehicles, or messages, each governed by their own network flow constraints, share the same underlying network. The objective of the MCFP is to send several commodities that reside at one or more points in a network (sources or supplies) with arc capacities to one or more points on the network (sinks or demands), incurring minimum cost. The arc capacities are one way in which the flows of all commodities are bound together. Note that if the commodities do not interact with each other, then the integer version of the MCFP can be solved as several independent single commodity minimum cost flow problems. But, since this is not valid in general, this problem is a difficult problem and in fact belongs to the class of NP-Complete problems (Garey and Johnson (1979)). However, the MCFP problem has numerous applications and hence solving this problem efficiently has been an important area of research. We consider the integer version of MCFP, which we refer to henceforth by MCFP, and we propose an aggregation framework that can be used to efficiently solve a class of MCFPs.

Researchers have proposed a number of basic approaches for solving both the integral version and the continuous version of the MCFP. The three classical approaches, all based upon exploiting network flow substructure, and selected references are: (1) Price-directive decomposition algorithm (Cremeans, Smith, and Tyndall (l970), Swoveland (1971), Chen and Dewald (l974), Assad (1980a), Mamer and McBride (2000), Holmberg and Yuan (2003), and Larsson and Yuan (2004)); (2) Resource-directive decomposition algorithm (Geoffrion and

Graves (l974), Kennington and Shalaby (l977), and Assad (1980a)); and (3) Basis partitioning (Graves and McBride (l976), Kennington and Helgason (l980), and Castro and Nabona (1996)). Ford and Fulkerson (1958) and Tomlin (1966) first suggested the column generation approach. The first of these papers was the forerunner to the general Dantzig-Wolfe (1960) decomposition procedure of mathematical programming. The excellent survey papers by Assad (l978) and by Kennington (l978) describe all of these algorithms and several standard properties of MCFPs. The book by Kennington and Helgason (1980) and the doctoral dissertation by Schneur (1991) are other valuable references on this topic.

Other approaches to solve the MCFP are due to Gersht and Shulman (l987), Barnhart (l988), Pinar and Zenios (l990), Barnhart (1993), Barnhart and Sheffi (1993), Farvolden, Powell, and Lustig (l993), Frangioni and Gallo (1999), and Detlefsen and Wallace (2002). Schneur (l99l), and Schneur and Orlin (1998) applied scaling techniques to solve the MCFP. Matsumoto, Nishizeki, and Saito (l985) gave a polynomial-time combinatorial algorithm for solving a MCFP in s-t planar networks. Radzig (1997) developed approximation algorithms for the MCFP. Barnhart, Hane, and Vance (2000a) developed a column generation model and an integer programming based branch-and-price-and-cut algorithm for integral MCFPs, and Brunetta, Conforti, and Fischetti (2000) investigated polyhedral approaches and branch-and-cut algorithms for the integral version.

Multicommodity network flow models have found extensive applications in several areas. The routing of multiple commodities has been studied by Golden (1975), and Crainic, Ferland, and Rousseau (1984), and the multi-vehicle tanker scheduling problem by Bellmore, Bennington, and Lubore (1971). Other applications are due to Kaplan (1973), Evans (1977), Geoffrion and Graves (1974), Bodin et al. (1980), Assad (1980b), Korte (1988). In some recent research,

Gautier and Granot (1995) formulate the forest management problem as a multicommodity flow problem, Lin and Yuan (2001) construct a multicommodity flow network reliability model and apply it to the container loading transportation problem, Ouaja and Richards (2004) propose a hybrid algorithm for traffic engineering which uses Lagrangian relaxation and constraint programming, Ahuja et al. (2005) solve the LPP using using problem decomposition, integer programming, greedy heuristics, and neighborhood search, Vaidyanathan, Jha, and Ahuja (2007a) formulate the CSP as a integer multicommodity network flow problem and solve it using branch-and-bound based heuristic methods, Vaidyanathan et al. (2007b) formulate the LPP as an integer multicommodity flow problem on a space-time network and solve it using branch-and-bound methods, and Vaidyanathan, Ahuja, and Orlin (2007c) solve the LRP using an aggregation based algorithm. The book by Ahuja, Magnanti, and Orlin (1993) provides extensive coverage of applications of network and multicommodity flows.

The predominant approaches that have been adopted to solve the integer MCFP are column generation, Dantzig-Wolfe decomposition, Lagrangian relaxation, resource directive decomposition, integer programming (branch-and-bound and branch-and-price-and-cut), polyhedral approaches, very large scale neighborhood search (VLSN), and other heuristic methods. To the best of our knowledge, nobody has applied an aggregation based approach to solve the MCFP. We develop an aggregation framework to solve large-scale integral MCFPs which satisfy some special properties. Our approach to solve the MCFP works in the following manner. Instead of solving the large-scale MCFP directly, we decompose our approach into two stages. In the first stage, we consider an approximate aggregated MCFP which is several times smaller than the original problem. We construct the aggregated MCFP in such a way that it is a very good approximation of the original problem. Since the aggregated MCFP is much smaller

than the original problem, we can obtain the optimal solution to this problem very quickly. In the second stage, we use the solution to the aggregated MCFP as a starting point, and then disaggregate this solution to obtain near optimal solutions to the original problem. We demonstrate the efficacy of our approach by applying it to solve the LPP, which is a large-scale MCFP. Our computational tests demonstrate that the aggregation framework proposed produces near optimal solutions to the LPP in a fraction of the original computational times. Finally, we also outline methods to apply the framework to effectively solve other real-world optimization problems.

Our major contributions are:

- We develop a novel aggregation based framework to solve a class of MCFPs.
- We formulate an aggregated version of the MCFP which is several times smaller than the original problem and can hence be solved efficiently using standard branch-and-bound techniques.
- We discuss methods by which we can ensure that the aggregated MCFP is a very good approximation of the original problem.
- We develop fast disaggregation methods which are used to obtain a solution to the original MCFP from the solution of the aggregated problem.
- We discuss application of the aggregation framework to solve other integer programming problems like the airline fleet assignment problem, the airline CSP, the LRP, and the aircraft routing problem.
- We test the aggregation framework on the LPP and demonstrate that the approach produces near optimal solutions (less than two percent optimality gap) to the LPP while only taking a fraction (around one-fifteenth) of the original computational times.

Summarizing, we present the first attempt at developing an aggregation framework to solve a class of large-scale MCFPs. From our experience, we believe that the framework developed has the potential to be a useful tool to solve many real-world problems including airline fleet assignment, airline crew scheduling, railroad locomotive routing, aircraft maintenance routing, and ship scheduling.

5.2 Problem Definition, Notation, and Formulation

The aggregation framework proposed can be used to solve a class of MCFPs. In this section, we give the notation and terminology and also lay out properties of the underlying network, the constraints, and the objective function for this class of problems. We also formulate the MCFP as an integer programming problem on the network.

5.2.1 Network

In this section, we describe properties that need to be satisfied by the underlying network. We denote the network by G(N, A) where N represents the set of nodes and A the set of arcs. This network has only two types of arcs. We call the two types of arcs load arcs (L) and free arcs (F). Load arcs are arcs on which flow is constrained. For example, in the airline fleet assignment problem, the flight legs to which aircrafts need to be assigned are load arcs, and in the LPP, the trains to which locomotives need to be assigned to provide sufficient pulling power are load arcs. Free arcs are those arcs on which there is no restriction or constraint on the flow. Free arcs originate at the head node of a load arc and terminate at the tail node of another load arc. For example, in the railroad network, train connection arcs which are constructed between trains which arrive at a station and trains which depart from the same station are free arcs. Free arcs allow commodities to flow from one load arc to the next load arc. Each load arc l has the following four attributes associated with it: (1) A departure time represented by dep-time(l); (2) An arrival time represented by the arr-time(l); (3) A departure location represented by dep-location(l); and (4) An arrival location represented by arr-location(l).

We are now in a position to describe the construction of the network. For each load arc we create an arrival (head) node and a departure (tail) node. We represent the set of arrival nodes by N_a and the set of departure nodes by N_d. The arrival node corresponding to load arc l is at

location arr-location(l) and the departure node corresponding to a load arc l is at location dep-location(l). Next, we construct free arcs between all arrival nodes at a location and all departure nodes at the same location. Note that due to the manner in which we construct the network, any walk on the network G(N,A) will contain alternate load and free arcs. Also, by construction, both the in-degree of any arrival node and the out-degree of any departure node are one. Let C refer to the different commodities that need to be assigned to load arcs. These commodities need to be routed on G(N,A) in such a way as to satisfy a set of constraints while incurring minimal cost. In Figure 5-1, we present an example of the network that has five load arcs and eight free arcs. In this figure, the solid lines represent load arcs and the dotted lines represent free arcs.

5.2.2 Constraints

In this section, we describe the constraints governing the class of MCFPs.

Power requirement for load arcs: Each load arc l must be assigned a single unit of a particular commodity to provide sufficient power.

Commodity to load arc preferences: Only certain commodities are allowed to flow on certain load arcs.

Flow balance constraints: The total incoming flow of a particular commodity at each node on the network must be equal to the total outgoing flow of the same commodity at that node.

Arc flow lower bounds: Each load arc $l \in L$ should be assigned at least LB_l units of flow.

Arc flow upper bounds: Each load arc $l \in L$ can be assigned at most UB_l units of flow.

Fleet-size constraints: The number of units of each commodity used should be less than the fleet-size of that commodity B^c.

5.2.3 Formulation

In this section, we mathematically formulate the general class of MCFPs that can be solved using the aggregation framework. We formulate the MCFP on the network constructed as described in Section 5.2.1, and consider the constraints described in Section 5.2.2. Let I(i) denote the set of incoming arcs and O(i) denote the set of outgoing arcs at node $i \in N$. In order to count the total number of units of each commodity used in a solution, we set an arbitrarily chosen time as the count-time and count the total flow on all arcs that cross this count-time. We denote the set of arcs that cross the count-time by S. Each load arc $l \in L$ can have two kinds of flow of a particular commodity $c \in C$: active flow (x_l^c) which represents those units which provide power on the load arc, and deadhead flow (y_l^c) which represents those units which do not provide power but are repositioning on the load arc using the power of the active flow. We represent the cost of assigning an active unit of commodity $c \in C$ to load arc $l \in L$ by c_l^c, the cost of assigning a deadhead unit of commodity $c \in C$ to load arc $l \in L$ by d_l^c, the cost of assigning a unit of commodity $c \in C$ on free arc $f \in F$ by d_f^c, and the ownership cost of a unit of commodity $c \in C$ by G^c.

The decision variables in the MCFP are:

x_l^c: Binary variable representing the active flow of commodity $c \in C$ on load arc $l \in L$.

y_l^c: Integer variable representing the deadhead flow of commodity $c \in C$ on load arc $l \in L$.

y_f^c: Integer variable representing the flow of commodity $c \in C$ on free arc $f \in F$.

s^c: Number of unused units of commodity $c \in C$.

The MCFP is formulated as follows:

$$\min \sum_{l \in L} \sum_{c \in C} c_l^c x_l^c + \sum_{l \in L} \sum_{c \in C} d_l^c y_l^c + \sum_{f \in F} \sum_{c \in C} d_f^c y_f^c - \sum_{c \in C} G^c s^c \quad (5.1a)$$

$$\sum_{c \in C} x_l^c = 1, \text{ for all } l \in L, \quad (5.1b)$$

$$LB_l \le \sum_{c \in C} (x_l^c + y_l^c) \le UB_l, \text{ for all } l \in L, \quad (5.1c)$$

$$x_l^c + y_l^c = \sum_{f \in O[head(l)]} y_f^c, \text{ for all } l \in L, c \in C, \quad (5.1d)$$

$$\sum_{f \in I[tail(l)]} y_f^c = x_l^c + y_l^c, \text{ for all } l \in L, c \in C, \quad (5.1e)$$

$$\sum_{a \in S} (x_a^c + y_a^c) + s^c = B^c, \text{ for all } c \in C, \quad (5.1f)$$

$$x_l^c \in \{0,1\}, \text{ for all } l \in L, c \in C, \quad (5.1g)$$

$$y_a^c \ge 0 \text{ and integer, for all } a \in A, c \in C, \quad (5.1h)$$

$$s^c \ge 0, \text{ for all } c \in C. \quad (5.1i)$$

Constraint (5.1b), which is the power requirement constraint for load arcs, ensures that every load arc l is assigned one unit which provides sufficient power. Constraint (5.1c), which is a flow bound constraint, ensures that the flow upper and lower bounds on each load arc l are satisfied. Constraint (5.1d) and (5.1e), which are flow balance constraints, ensure that flow is balanced at each arrival and departure node for each commodity type. Constraint (5.1f), which is the fleet-size constraint, ensures that the number of units of each commodity used is no more than the available fleet-size of that commodity. Note that the Formulation (5.1) is the general formulation and several of the applications described are special cases or minor variations of this formulation. We now describe another important property that needs to be satisfied by the set of load arcs in the underlying network so that the aggregation framework can be applied to solve the MCFP.

5.2.4 Reducibility Property of the Network

The reducibility property is the most crucial property that needs to be satisfied by the set of load arcs L in the underlying network G(N,A) for the problem to fall in the class of MCFPs that can be solved using the aggregation framework proposed. The reason for this is that reducibility of the network is the property which allows us to aggregate and shrink the original problem. The set of load arcs L is said to satisfy the reducability property if it can be partitioned into several sets R such that the load arcs in each set $r \in R$ share several common properties. The load arcs in each set $r \in R$ have identical origin location, destination location, duration, power requirements, and flow bounds and only differ in their departure and arrival times. For each set $r \in R$ we consider an aggregated load arc and we denote the set of aggregated load arcs by L'. Hence, each aggregated load arc $l' \in L'$ has several arcs corresponding to it in the set L. Let the set of load arcs corresponding to aggregated load arc l' be represented by L(l').

5.3 Aggregation and Disaggregation Algorithm

The integer MCFP formulated in Section 5.2 is NP-Complete and hence large-scale MCFPs cannot be solved using commercial integer programming software. In this section, we develop an aggregation-disaggregation based approach to solve this problem. The basic idea behind this approach is that instead of solving a large problem directly, we divide our algorithm into two stages. In the first stage, we construct a smaller aggregated model which can be solved quickly. Once we obtain the solution to the aggregated model, we use this information to disaggregate the solution and obtain a solution to the original problem. While this idea is quite intuitive to grasp, there are some issues which need to be considered to ensure that the solution produced using this approach in close to the global optimal solution. We next present our aggregation-disaggregation based algorithm and describe how we use this approach to effectively solve the MCFP.

5.3.1 Aggregated Multicommodity Flow Problem

The aggregated MCFP is an approximation of the original MCFP. However, this approximate model is much smaller than the original problem and is hence solvable more quickly. This leads to the issue of trading-off between the degree of approximation and the ease of solution, i.e., (1) The aggregate model may be solvable very quickly but may not produce good solutions; or (2) The aggregate model may be almost exact but may take a very long time to produce a good solution. Hence, the performance of an aggregated model depends on a few factors that need to be addressed while constructing the model. The first issue that needs to be addressed while designing an aggregated model is which entities to aggregate. The second issue that needs to be addressed is how to ensure that the aggregated model is as close as possible to the original problem. The closer the aggregated model is to the original problem, the easier it will be to disaggregate the aggregate solution and obtain near optimal solutions to the original problem. In the rest of this section, we focus on answering these questions and developing a good aggregation model. We divide our discussion into two parts: we first focus on the feasibility aspect of the problem and, in the process, address the first question. Next, we focus on the optimality of the problem and, in the process, address the second question.

Consider a network G(N,A) that satisfies the reducibility property (defined in Section 2.4). The set of load arcs L on this network can be partitioned into several sets L(l') where $l' \in L'$. Note that the set of load arcs L can be written as $L = \bigcup_{l' \in L'} L(l')$. Similarly, the cardinality of the set of load arcs can be written as $|L| = \sum_{l' \in L'} |L(l')|$. We refer to the cardinality of set L(l') as the frequency of aggregated arc l' and denote it by $f_{l'}$. Hence, the load arcs can be considered to be made up of a set of aggregated load arcs, each with a pre-specified frequency. We refer to the set

of arcs in the set L(l') as the instances of aggregated load arc l'. Due to the reducability property of the network, all instances of an aggregated load arc are similar in terms of their origin location, destination location, power requirements, flow bounds, and duration. They only differ in their departure times and arrival times. Based on this observation, in the aggregated network, we aggregate all instances of an aggregated load arc in the original network into a single aggregated load arc. The flow on an aggregated load arc $l' \in L'$ represents the total flow on all the instances corresponding to it (L(l')). Also, the total power requirement of an aggregated loaded arc is computed by adding up the total power requirements on all the load arcs corresponding to it. For example, suppose an aggregated load arc has a frequency of four and all the load arcs corresponding to it have a power requirement of T, then the power requirement of the aggregated load arc is 4T. Similarly, we define the flow bounds on each aggregated loaded arc by adding up the flow bounds on all the instances corresponding to it.

We denote the aggregated network by G'(N' A') where N' represents the set of nodes and A' the set of arcs. This network has two kinds of arcs: aggregated load arcs (L') and aggregated free arcs (F'). The aggregated load arcs are obtained as described above. Each aggregated load arc l' has: (1) A departure location represented by dep-location(l'); (2) An arrival location represented by arr-location(l'). For each aggregated load arc we create an arrival (head) node and a departure (tail) node. We represent the set of arrival nodes by $N_a^{'}$ and the set of departure nodes by $N_d^{'}$. Next, we construct aggregated free arcs between all arrival nodes at a location and all departure nodes at the same location. Aggregated free arcs allow commodities to flow from one aggregated load arc to the next aggregated load arc. Note that due to the manner in which we construct the network, any path on G'(N',A') will contain alternate aggregated load arcs and aggregated free arcs. Also, the in-degree of any arrival node and the out-degree of any departure

node are one. We consider a set of commodities (C) which need to be assigned to aggregated load arcs and flow on the network to satisfy a set of constraints at minimal cost.

We use the following additional notation in our formulation:

$LB_{l'}$: Lower bound on total flow on aggregated load arc $l' \in L$. This lower bound is obtained by adding up the lower bounds on all instances of aggregated load arc l'.

$UB_{l'}$: Upper bound on total flow on aggregated load arc $l' \in L$. This upper bound is obtained by adding up the upper bounds on all instances of aggregated load arc l'.

$t_{l'}$: Duration of aggregated load arc $l' \in L'$.

$t_{f'}$: Estimated duration of aggregated free arc $f' \in F'$.

R^c: Total number of time units of commodity $c \in C$ available.

The decision variables in the aggregated MCFP are:

$x_{l'}^c$: Integer variable representing the active flow of commodity $c \in C$ on aggregated load arc $l' \in L'$.

$y_{l'}^c$: Integer variable representing the deadhead flow of commodity $c \in C$ on aggregated load arc $l' \in L'$.

$y_{f'}^c$: Integer variable representing the flow of commodity $c \in C$ on aggregated free arc $f' \in F'$.

s^c: Number of unused time units of commodity $c \in C$.

We formulate the aggregated MCFP as follows:

$$\min \sum_{l' \in L'} \sum_{c \in C} c_{l'}^c x_{l'}^c + \sum_{l' \in L'} \sum_{c \in C} d_{l'}^c y_{l'}^c + \sum_{f' \in F'} \sum_{c \in C} d_{f'}^c y_{f'}^c \quad (5.2a)$$

$$\sum_{c \in C} x_{l'}^c = f_{l'}, \text{ for all } l' \in L', \quad (5.2b)$$

$$LB_{l'} \le \sum_{c \in C} (x_{l'}^c + y_{l'}^c) \le UB_{l'}, \text{ for all } l' \in L', \quad (5.2c)$$

$$x^c_{l'} + y^c_{l'} = \sum_{f \in O[head(l')]} y^c_{f'}, \text{ for all } l' \in L', c \in C, \quad (5.2d)$$

$$\sum_{f \in I[tail(l')]} y^c_{f'} = x^c_{l'} + y^c_{l'}, \text{ for all } l' \in L', c \in C, \quad (5.2e)$$

$$\sum_{l' \in L'} t_{l'}(x^c_{l'} + y^c_{l'}) + \sum_{f' \in F'} t_{f'} y^c_{f'} + s^c = R^c, \text{ for all } c \in C, \quad (5.2f)$$

$$0 \le x^c_{l'} \le f_{l'} \text{ and integer, for all } l' \in L', c \in C, \quad (5.2g)$$

$$y^c_{i'} \ge 0 \text{ and integer, for all } i' \in A', c \in C, \quad (5.2h)$$

$$s^c \ge 0, \text{ for all } c \in C. \quad (5.2i)$$

Constraint (5.2b), which is the aggregated power requirement constraint, ensures that each aggregated load arc is assigned sufficient power. Constraint (5.2c), which is the aggregated flow bound constraint, ensures that the flow bounds on each aggregated load arc are honored. Constraint (5.2d) and (5.2e), which are flow balance constraint, ensures that flow is balanced at each arrival and departure node for each commodity type. Constraint (5.2f), which is the fleet-size constraint, limits the total number of commodity-time units used for by commodity type. In order to measure the total number of commodity minutes used, we cumulatively add up the commodity minutes used by commodities assigned on each aggregated load arc and the commodity minutes used by commodities assigned on each aggregated free arc. We then ensure that the total number of commodity minutes used is less than the total number of commodity minutes available.

The reader may note that the aggregated MCFP is similar in structure to the MCFP but could be several times smaller in size. Also, the fleet-size constraints (5.2f) in the aggregated model are constructed in an equivalent though different manner to the fleet-size constraint (5.1f) of the MCFP. The reason for this is that in the original network, it is quite straight forward to

determine whether a free arc crosses the count-time; however, in the aggregated network since we consider aggregated load arcs, it is not possible to state with certainty whether an aggregated free arc crosses the count-time. For the time being, we assume that we know the duration of aggregated free arc. In the later part of this section, we discuss the importance of having good estimates for the duration of aggregated free arcs.

The size of the aggregated MCFP is several times smaller than the MCFP. The reason for that is aggregation happens at two levels: (1) decision variables, and (2) constraints. For example, consider an aggregated load arc which has a frequency of four. In the MCFP, this aggregated load arc will have 4|C| active flow decision variables corresponding to it and 4|C| deadhead flow decision variables corresponding to it. However, the aggregated MCFP will have only |C| active flow decision variables and |C| deadhead flow decision variables. Hence, the flow on the aggregated load arc in this case represents the cumulative flow on the four load arcs that correspond to it. The decision variables corresponding to the free arcs are also aggregated in a similar manner. Using this method of variable aggregation, we are able to reduce the number of variables. Also, in the aggregated MCFP we have:

- One power requirement constraint (5.2b) for every aggregated load arc instead of one (5.1b) for every load arc in the MCFP.
- One set of flow bound constraints (5.2c) for every aggregated load arc instead of one (5.1c) for every load arc in the MCFP.
- One set of flow balance constraints (5.2d) and (5.2e) for each aggregated load arc arrival and departure node, instead of one for each load arc arrival and departure in the MCFP.

Suppose the average frequency of aggregated load arcs is close to five. Then, the total number of constraints in the aggregated model is roughly one-fifth that of the original model and the total number of variables is a fraction of that in the original model. Thus, the aggregation of

variables and constraints has the potential to result in a very compact aggregated formulation and consequently very fast solution times.

We now proceed to state and prove a few important properties relating the feasibility of the aggregated MCFP and the feasibility of the MCFP.

Property 5.1: For any assignment of flows on load arcs such that the total flow of a commodity entering a location is equal to the total flow of the same commodity leaving the same location and satisfying the power requirement constraints (5.1b) and flow bound constraints (5.1c), there exists a solution to the MCFP satisfying all constraints other than fleet-size constraint (5.1f).

Proof: Consider the assignment of flows on load arcs which satisfies the conditions stated above. Consider the set of inbound load arcs entering a particular location, the set of outbound load arcs leaving the same location, and the set of free arcs between these inbound load arcs and outbound load arcs. Since free arcs are constructed between all inbound load arcs and outbound load arcs at a particular location and since flow on these arcs is unrestricted, it follows that we can enforce flow balance constraints (5.1d) and (5.1e) at all nodes at that location by a simple run-through method. Repeating the same procedure at all locations in the network gives us a solution to MCFP satisfying all constraints other than possibly the fleet-size constraint (5.1f). ◆

Property 5.2: There exists a feasible solution to the MCFP only if there exists a feasible solution to the aggregated MCFP satisfying all constraints other than the fleet-size Constraint (5.2f).

Proof: We prove this property by construction. Consider a feasible solution to the MCFP. We can compute flows on arcs in the aggregated network by: (1) Assigning each aggregated load arc active flow equal to the total active flow on all load arcs corresponding to it; (2) Assigning

each aggregated load arc deadhead flow equal to the total deadhead flow on all load arcs corresponding to it; (3) Computing the flow on all aggregated free arcs using the procedure described in the proof of Property 5.1. Note that, due to the manner in which we formulate the aggregated MCFP, the constructed solution satisfies the aggregated power requirement constraint (5.2a) and the aggregated flow bound constraints (5.2b). Further, from Property 5.1, it follows that the assignment will also satisfy the flow balance constraints (5.2c) and (5.2d). The result is immediate. ◆

Property 5.3: There exists a feasible solution to the aggregated MCFP only if there exists feasible solution to the MCFP satisfying all constraints other than the fleet-size constraints (5.1f).

Proof: We prove this property by construction. Consider a feasible solution to the aggregated MCFP. We can compute flows on arcs in the original network by: (1) Distributing the total active flow on each aggregated load arc among the load arcs corresponding to it (one unit per load arc), (2) Distributing the total deadhead flow on each aggregated load arc among the load arcs corresponding to it in such a way that the flow bound constraints (5.1c) are not violated, (3) Computing the flow on all free arcs using the procedure described in the proof of Property 5.1. The constructed solution satisfies the power requirement constraints (5.1b) and the flow bound constraints (5.1c). Also, the total flow of a commodity entering a location is equal to the total flow of the same commodity leaving the same location. Hence, from Property 5.1, the result follows. ◆

We have thus showed the equivalence between the feasibility version of the MCFP and the feasibility version of the aggregated MCFP and have hence addressed the feasibility aspect of the problem mentioned at the start of this section. The other aspect we need to address is optimality. The objective function of the MCFP minimizes total active assignment cost, deadhead

assignment cost, idling cost, and ownership cost. We now describe methods to ensure that the aggregated model serves as a good approximation of the original problem.

In the aggregated MCFP, we do not consider the individual load arcs and only consider aggregated load arcs. Due to the abstraction of information, the aggregated MCFP is an approximation of the MCFP. Consider the aggregated MCFP. Suppose we were somehow able to accurately determine the time duration for each commodity assigned on each free arc accurately, then the aggregated MCFP would indeed model the original MCFP exactly. However, the only way in which this information can be obtained is from the solution to the MCFP which is what we set out to determine in the first place. Hence, it is not possible to accurately determine the durations accurately. Hence, our goal is to devise an intelligent method to provide good estimates for the duration of free arcs. Also, since the number of free arcs could be quite large, the method needs to run very quickly. The right way to estimate the durations is a function of the problem being studied. In Section 5.4.1, we describe a method to estimate the duration of free arcs (or train connections) while solving the LPP.

Note that the aggregated MCFP is similar in structure to the original MCFP. In other words, our method of aggregation shrinks the problem size while leaving the structure virtually unaltered. Hence, the existing state-of-the-art algorithms for the MCFP may often be directly applicable to solve the aggregated problem. We now describe the second stage of our approach which involves disaggregation of the solution.

5.3.2 Disaggregation

In this section, we show how the solution of the aggregated MCFP can be used to obtain a solution to the original MCFP. The approaches proposed use the solution of the aggregated MCFP to restrict the search space of the MCFP in order to speed-up computational times. We propose three different methods to disaggregate the solution and we now describe these methods.

Restricted multicommodity approach: Method 1

In the restricted multicommodity approach, we reduce the solution space of the MCFP formulated in Section 5.2.3 in the following manner. On every load arc, we allow only those commodities that are assigned as active commodities in the solution to aggregated MCFP to be assigned as active commodities. For example, suppose an aggregated load arc carries only commodities A and B, then all the load arcs that correspond to it are also restricted to only having those commodities as active commodities. By using this method, we can reduce the solution space of the MCFP by a large extent leading to very fast computational times.

Multicommodity approach which partitions the aggregated active and deadhead flows Method 2

Note that the solution to the aggregated MCFP satisfies flow balance at each location, i.e., in the solution to the aggregated MCFP, the total number of units of a particular commodity entering a location is equal to the total number of commodities of the same type leaving the location. In this method, we exploit this fact to solve the MCFP. Also, as long as the total flow on all instances of an aggregated train is equal to the total flow on the corresponding aggregated train, the total flow of a particular commodity between any pair of locations in the MCFP is equal to the total flow between the same pair of locations in the solution to the aggregate MCFP. Hence by Property 1, flow balance constraints for the MCFP are easily satisfied. Therefore, our goals are (1) To redistribute the active assignments on each aggregated load arc among the load arcs corresponding to it; (2) To redistribute the deadhead assignments on each aggregated load arc among all the load arcs corresponding to it.

Let L'(l) refer to the aggregated load arc corresponding to load arc l and L(l') represent all load arcs corresponding to aggregated load arc l'. We write these constraints mathematically in the following manner:

$$\sum_{l \in L(l')} x_l^c = x_{l'}^c, \text{ for all } l' \in L', c \in C,$$

$$\sum_{l \in L(l')} y_l^c = y_{l'}^c, \text{ for all } l' \in L', c \in C.$$

The right-hand side of these equations is obtained from the solution of the aggregated MCFP. These constraints, when added to the MCFP, restrict the solution space to a larger extent than the first method. Hence, the solution obtained using this method is slightly worse than that obtained using the first method; however, the computational times are faster.

Method 3

In this method, while the aggregated active flows are redistributed as described in the previous method, the aggregated deadhead flows are handled differently. Observe that as long as the total flow between any pair of locations in the MCFP is equal to the total flow between any pair of locations in the solution to the aggregate MCFP, the total number of consists of a particular type which enter a station will be equal to the total number of consists of the same type which leave the same station. By Property 5.1, flow balance constraints for the MCFP are easily satisfied. We use this fact to restrict the solution space of the MCFP.

First, we introduce some additional notation. Let F_{pq}^c be the total deadhead flow of commodity $c \in C$ between station p and station q in the solution to the aggregated MCFP and let H represent the set of all locations. Let L_{pq} be the set of all load arcs that originate from location p and terminate at location q. We then write these constraints mathematically in the following manner:

$$\sum_{l \in L(l')} x_l^c = x_{l'}^c, \text{ for all } l' \in L', c \in C,$$

$$\sum_{l \in L_{pq}} y_l^c = F_{pq}^c, \text{ for all } p \in H, q \in H, c \in C.$$

The right-hand side of these equations is obtained from the solution of the aggregated MCFP. These constraints are added to the MCFP to restrict the solution space and speed-up computational time. Note that the previous method is more restrictive and is also a special case of this method. This method gives less flexibility than the restricted multicommodity approach but more flexibility than the method 2.

5.4 Applications of the Aggregation Framework

As we mentioned in the introduction, the aggregation framework can be used an effective tool in solving several optimization problems. In this section, we outline some applications of the aggregation framework to solve large-scale real-world optimization problems. The problems considered include the LPP, the airline fleet assignment problem, the airline CSP, the LRP, and the aircraft routing problem. We go into details of how we use customize this framework to solve the LPP and present computational results to demonstrate the performance of the aggregation algorithm. We also outline how the aggregation framework can be used to solve all the other problems mentioned above but do not present computational studies of the performance of our algorithms on these problems.

5.4.1 Locomotive Planning Problem

The LPP involves the assignment of sets of locomotives, called consists, to each train in a pre-planned weekly train schedule so that each train gets sufficient power to pull its load and the total cost of locomotive usage is minimized. The resulting plan must honor a variety of business rules, cannot require more locomotives than what is available in the total fleet, and must result in a plan that is relatively simple and repeatable. Another important feature of the LPP is that locomotives may deadhead on trains. Deadhead locomotives do not pull the train but are repositioned by the active locomotives from one place to another place. Deadheading plays an important role in locomotive planning, enabling extra locomotives to be moved from stations

with a surplus to stations with a defecit. A locomotive plan specifies which types of locomotives will pull each train, and how locomotives will deadhead to obtain overall network wide efficiency.

We present an extensive review of research in the area in Chapter 3. The most comprehensive multiple locomotive planning models are due to Ahuja et al. (2005) and Vaidyanathan et al. (2007b). In Ahuja et al. (2005), we formulate the LPP as an integer MCFP with side constraints on an underlying space-time network and develop solution methodologies that use problem decomposition, integer programming, greedy heuristics, and neighborhood search. In Vaidyanathan et al. (2007b), we developed new formulations and approaches to make our planning model more implementable. The major outcome of that paper was a philosophy shift in the routing of locomotives. We showed that assigning consists (or a group of locomotives which are never broken up) to trains, instead of individual locomotives has the potential to generate implementable solutions without compromising much on the solution cost. We call this formulation of the LPP the consist-based formulation.

The algorithms that we developed in our research in the past were able to effectively solve weekly LPPs where the train schedule repeats every week. These weekly train schedules typically have around 3,000 trains and our algorithms were able to solve these problems within 10-60 minutes of computational time. However, subsequently railroads have expressed the need to solve much larger planning problems. One such problem is the 28-day LPP, where the train schedule repeats only once in 28 days. The reason for this is that the weekly repeating schedule assumption may not be a good assumption in certain cases. These 28-day train schedules have 3-5 times more number of trains than the weekly train schedules, and hence the associated LPPs are significantly larger optimization problems. Due to this fact and the additional fact that the

LPP is NP-Complete, our previous approaches are likely to be intractable when applied to these problems. Hence, there is a need for a novel scalable approach to solve the LPP and the aggregation framework addresses this need.

We consider the consist-based formulation of the weekly LPP introduced in Vaidyanathan et al. (2007b) and apply the aggregation framework to solve this problem. We now show how the LPP is a special case of the MCFP formulated in Section 5.2.3. The trains in the weekly train schedule are analogous to the load arcs and the train connections, which allow the transfer of locomotives from inbound trains at a station to outbound trains at the same station, are analogous to the free arcs. Hence, the LPP can be formulated on a network as described in Section 2.1 which contains only two kinds of arcs: train arcs, and connection arcs. The consists are the commodities which flow on this network. Also, the weekly train schedule is such that it can be considered to be made up of a set of aggregated trains, each with a pre-specified frequency varying between one and seven. These aggregated trains are analogous to aggregated load arcs. Note that all instances of an aggregated train in the week are similar in terms of origin station, destination station, departure time of the day, arrival time of the day, and duration. The only factors which differentiate these trains are the departure day of the week, and arrival day of the week. From these observations, it follows that the underlying network satisfies the reducability property. The LPP is also governed by power requirement constraints, flow bounds on all train arcs, flow balance constraints, and fleet-size constraints which are similar in structure to constraints (5.1b), (5.1c), (5.1d), (5.1e), and (5.1f) respectively. Hence, the LPP falls into the class of MCFPs that can be solved using the aggregation framework. We now proceed to estimate the size of a typical instance of the LPP.

Problem size

A typical weekly train schedule has around 3,000 trains and eight consist types. Hence, the space-time network will contain around 3,000 train arcs, and more than 500,000 connection arcs for a total of more than 500,000 arcs. The number of nodes on this network will be 3,000 train arrival nodes, 3,000 train departure nodes, for a total of 6,000 nodes. On each arc, we consider decision variables for the flow of each of the eight commodities. Hence, the number of variables in the formulation is around 500,000 x 8 = 4,000,000 variables. There are 3,000 train power requirement constraints (5.1b), 3,000 train flow bound constraints (5.1c), 6,000 x 8 = 48,000 flow balance constraints corresponding to each commodity at each node (5.1d and 5.1e), and 8 fleet-size constraints (5.1f) giving a total of around 55,000 constraints. Hence, the LPP is a large-scale integer MCFP with 55,000 constraints and millions of decision variables. Integer programming problems of this size are intractable when solved using commercial integer programming software. However, the aggregated LPP which is similar in structure to the aggregated MCFP, is much smaller. Suppose the average frequency of each train is five, then the aggregated LPP would roughly have only one-fifth the number of constraints and a small fraction of the number of decision variables as the original problem. The aggregation of variables and constraints results in a very compact aggregated formulation which can consequently be solved very quickly using commercial software.

In Section 5.3.1, we mentioned that the estimation of the duration of free arcs is crucial to ensure that the aggregated problem is a good approximation of the original problem. We now describe a method to estimate durations of train connection arcs. This step is a crucial part of our algorithmic approach and we empirically converged at the following method after a significant amount of research and computational testing of various possibilities.

Estimating train connection times

The biggest approximation in the aggregated LPP is that the departure day of the week and the arrival day of week information of trains are not considered. This could lead to the aggregated problem generating bad solutions in terms of locomotive usage. For e.g. Consider a weekly train A which arrives at a station on Monday at 10:00 AM and another weekly train B that departs from the same station at 11:00 AM on Sunday. The optimal solution of the LPP is not likely to use the connection between these weekly trains to route locomotives because the idling time of the locomotive would be extremely high (six days and one hour). On the other hand, in the aggregated network, since we do not consider the arrival and departure days of individual trains, the connection between the corresponding aggregated trains could be used in the solution. We handle this problem in the following manner. We define an estimated connection time for each train connection on the aggregated network. The estimated connection time between aggregated trains A and B is computed to represent the probable connection time if a consist connects between a instance of aggregated train A and an instance of aggregated train B. Note that we cannot compute this value exactly unless both A and B have a frequency of one. We represent the estimated connection time of a train connection j by t_j. Note that since the aggregated problem has several thousand train connections and the method used to estimate train connection times needs to be run once for every connection, this method has to be very efficient so as to not blow up overall computational times. Next, we describe the method used to estimate the train connection times.

Consider an inbound aggregated train i and outbound aggregated train j at a particular station. In order to compute the estimated connection time between these two aggregated trains, we consider the trains corresponding to them and the complete set of connections between these

trains. Each connection is assigned time duration equal to the duration of that connection on the original network. For example, suppose the inbound aggregated train has a frequency of four and the outbound aggregated train has a frequency three, then we have the possibilities for connections between corresponding seven day trains as shown in Figure 5-2.

In a general case, let the frequency of train i be represented by f(i). Let the actual connection times between the corresponding weekly trains be represented as matrix M of dimensions f(i) x f(j). We divide the computation of train connection times into two cases:

Case 1: $|f(i) - f(j)| \leq 1$

This case is when the inbound aggregated train and the outbound aggregated train have similar frequencies. In this case the estimated connection time is computed by solving an assignment problem between the inbound weekly trains and the outbound weekly trains. The connection times in the matrix M represent the objective function of this problem, i.e., the element M(i, j) represents the cost of assigning the ith inbound train to the jth outbound train. Let the optimal objective function of this problem be obj. We then compute the estimated connection time of the connection as obj/ min (f(i), f(j)).

Case 2: $|f(i) - f(j)| > 1$

This case is when the inbound aggregated train and outbound aggregated train have frequencies which are quite different. Suppose the method described in case 1 were used in this case, it may not give realistic estimates of connection times. For example, suppose the inbound aggregated train had a frequency of one and arrives at 11 AM on Monday, and the outbound aggregated train has a frequency of seven and departs at 11:30 AM every day of the week. Then, if we were to compute the connection time based on the method in case 1, then we will obtain an estimated connection time of 30 minutes. However, this value is over optimistic since there is no

guarantee that the connection between the inbound train on Monday and the outbound train on Monday will be made. Based on our empirical analysis, we found that in this case we using the average of all the connection times in M gives a good estimate of the connection time.

We were able to solve the aggregated LPP in a few seconds of computational time. We applied the three methods of disaggregation described in Section 3 with the following results: (1) The restricted multicommodity approach (method 1) produced solutions to the LPP that were less than 1% away from the optimal solution in around 10 seconds of computational time; (2) Method 2 produced solutions to the LPP that were around 1.75% away from the optimal solution in around 2 seconds of computational time; (3) Method 3 produced solutions to the LPP that were around 1.5% away from the optimal solution in around 2 seconds of computational time. Our computational results section demonstrates the success of the aggregation framework as an effective tool to solve the LPP. We were able to obtain near optimal solutions (less than two percent gap) while consuming only one-fifteenth of the original computational times.

5.4.2 Airline Fleet Assignment Problem

The objective of the airline fleet assignment problem is to assign aircrafts belonging to different fleet types to flight legs to minimize the assignment cost and subject to the following three types of constraints: (1) Covering constraints: Each flight leg must be assigned exactly one aircraft; (2) Flow balance constraints: An aircraft must land before take-off; and (3) Fleet-size constraints: For each fleet type, the number of aircrafts used must not exceed the number of aircrafts available. A model which solves the problem described above is called a basic fleet assignment model (FAM).

The fleet assignment problem has been well studied (Abara (1989), Hane et al. (1995), Talluri (1996), Rexing et al. (2000), Barnhart, Kniker, and Lohatepanont (2002), and Ahuja et al. (2004)). Predominant approaches to solve this problem include integer programming algorithms,

branch-and-bound methods, and very large scale neighborhood search techniques. However, to the best of our knowledge, none of the research of the past has employed aggregation methods to solve this problem. Most US airlines have the same flight schedule for five days of the week and a subset of the week day schedule in the weekend. Thus, airlines typically solve a daily fleet assignment model whose solution is then repeated on each day of the week. Their approach assumes that the flight schedule is the same on all days, which, strictly speaking does not hold for most airline schedules. Also, the additional constraint that the fleet assignment should repeat daily gives less flexibility to the model to assign aircrafts to flight legs and may hence lead to sub-optimal solutions.

US airlines typically operate 2,000 flights daily and the computational times using existing methods for daily fleet FAMs are of the order of hours. Given the fact that these problems are NP-Complete, it is very likely that the computational times will explode if the existing approaches were applied to problems say five times larger. In order to demonstrate the applicability of the aggregation based framework to solve larger FAMs, we consider the 5-day fleet assignment problem or the weekday fleet assignment problem. Note that the 5-day flight schedule can be considered to be made up of a smaller aggregated flight schedule where each aggregated flight has a pre-specified frequency varying between one and five.

We now show how the FAM is a special case of the MCFP formulated in Section 2.3. The flight legs in the flight schedule are analogous to the load arcs and the aircraft connections, which allow the transfer of aircrafts from inbound flights at an airport to outbound flight at the same airport, are analogous to the free arcs. Hence, the FAM can be formulated on a network as described in Section 2.1 which contains only two kinds of arcs: flight arcs, and aircraft connection arcs. The various aircraft types in the fleet are the commodities which flow on this

network. Also, the flight schedule, as mentioned in the previous paragraph, is such that it can be considered to be made up of a set of aggregated flight legs, each with a pre-specified frequency varying between one and five. These aggregated flight legs are analogous to aggregated load arcs. Note that all instances of an aggregated flight leg in the week are similar in terms of origin airport, destination airport, departure time of the day, arrival time of the day, and duration. The only factors which differentiate these flight legs are the departure day of the week, and arrival day of the week. From these observations, it follows that the underlying network satisfies the reducability property. The FAM is governed by covering constraints which are similar to the power requirement constraints (1b), flow balance constraints which are similar to constraints (1d) and (1e), and fleet-size constraints which are similar in structure to constraints (1f). Hence, the FAM is a special case of MCFPs that can be solved using the aggregation framework. The reader may also note that the FAM is in fact simpler than the LPP and the MCFP since aircrafts cannot deadhead on flight legs whereas multiple units can deadhead on load arcs. Also, note that the aggregated FAM will have a similar structure to the daily FAM. Hence, it is quite possible that the existing state-of-art algorithms to solve the daily FAM can be also be directly employed to solve the aggregated FAM.

5.4.3 Airline Crew Scheduling Problem

The objective of airline crew scheduling is to assign individual crew members to flights in a pre-specified schedule to minimize assignment costs and satisfy all governmental and contractual restrictions such as maximum daily working hours, minimum overnight rest period, maximum time away from a crew base, etc. In practice, airline crew scheduling is sequentially divided into two stages: crew pairing and crew rostering. Among all the airports served by airlines, some are called crew bases. Each crew belongs to a specific base. A crew pairing is a connected sequence of flight segments that starts and ends at the same base location and also

satisfies all the legality constraints. The crew rostering problem then assigns individual crew members to pairings.

The airline CSP is a well studied problem (Hoffman and Padberg (1993), Barnhart et al. (1994, 2003), Vance et al. (1997), and Gopalakrishnan and Johnson (2005)). Predominant approaches to solve this problem include column generation, zero-one programming, branch-and-cut, and other integer programming approaches. In this section, we give a description of the basic version of the problem and outline how the aggregation framework can be used to solve it.

Given a set of feasible crew pairings, the airline CSP can be formulated as the problem of finding the best collection of crew pairings such that each flight leg is covered exactly once. This problem is essentially a set partitioning problem and can be formulated in the following manner:

$$Min \sum_{j=1 \text{ to } n} c_j x_j$$

$$Ax = e_m,$$

$$x_j \in \{0,1\}, \text{ for all } j = 1 \text{ to } n.$$

where e_m is a vector having m entries equal to 1 and n is the number of crew pairings considered. The matrix A is an m x n matrix where each row represents a flight leg and the matrix is constructed in such a way that $a_{ij} = 1$ if flight leg i is covered by rotation j, and 0 otherwise. This problem is an NP-Complete problem with millions of decision variables and hence almost all existing research deals with handling this computational complexity in an efficient manner. To the best of our knowledge, none of the existing research has explored aggregation as a means to solve this problem. We next outline our aggregation based approach to solve this problem. Note that this problem does not exactly fall in the class of MCFP since there is no underlying network and it is also not a multicommodity flow problem. However, due to the

reducability property of the flight schedule, it still lends itself to the aggregation framework proposed.

In the previous section, we observed that 5-day flight schedule can be considered as an aggregated flight schedule where each flight has a pre-specified frequency varying between one and five. This gives us the intuition that instead of solving the larger CSP directly, we can adopt an aggregation approach. We aggregate all instances of a flight in the week into a single aggregated flight. Each aggregated flight now has a crew demand which is equal to its frequency in the week (instead of one in the original model). We also consider pairings which are now made up of aggregated flights and refer to these pairings as aggregated pairings. These aggregated pairings are the decision variables in the aggregated CSP. The aggregated CSP can be formulated as a set partitioning problem where the right hand side is now the frequency of the corresponding aggregated flight.

The aggregated CSP is identical in structure to the original model. However, the size of the aggregated model could be several times smaller. The reason for that is aggregation happens at two levels: (1) decision variables, and (2) constraints. The flow on each aggregated pairing represents the cumulative flow on several crew pairings in the original network. Also, in the aggregated model we have only one partitioning constraint for every aggregated flight instead of one for every flight in the original model. This aggregation of variables and constraints results in a very compact formulation and consequently has the potential to be solvable quickly. Once we obtain the solution to the aggregated model, we can use this solution to obtain a near optimal solution to the original problem using suitably designed disaggregation methods.

5.4.4 Locomotive and Aircraft Routing Problems

The LRP involves routing each locomotive unit on a cyclic sequence of trains so that it can be fueled and serviced as necessary; while at the same time satisfying all other business and

operational constraints considered in the LPP. We refer to the sequence of trains on which a locomotive unit travels as a locomotive route. Fueling feasibility requires that each locomotive route has sufficient fueling opportunities to ensure that the locomotive does not run out of fuel (out-of-fuel event). This translates to the constraint that every locomotive needs to have a fueling opportunity at least once for every F miles (900 miles) of travel. Similarly, servicing feasibility requires that each locomotive route has sufficient servicing opportunities and this translates to the constraint that every locomotive needs to have a servicing opportunity at least once for every S miles (3,000 miles) of travel. The LRP, until very recently, was a practically unstudied problem. Our recent research reported in Vaidyanathan, Ahuja, and Orlin (2007c) and Chapter 4 considers the LRP. In that paper and in Chapter 4, we develop an aggregation-disaggregation approach to solve the LRP. In this section, we give an overview of our research and refer the interested reader to that paper for more details.

We use the concept of strings to formulate the LRP. A string is a connected sequence of trains such that any locomotive which travels on this sequence can be fueled and serviced as necessary. We formulate the LRP as an integer programming string decomposition problem (SDP) on a suitably defined space-time network. The objective is to partition the locomotive assignment (obtained from the planning model) on each train in the schedule into flows on strings while simultaneously minimizing the solution costs. The SDP is essentially a set partitioning problem with side constraints and is NP-Complete. The number of strings (decision variables) in this problem runs to several million. Hence, a direct approach to solve this problem is not likely to be tractable. Note that the LRP like the airline CSP does not exactly fall in the class of MCFP described. However, due to the reducability property of the train schedule, it still lends itself to the aggregation framework. We next summarize our aggregation based approach to

solve this problem and refer the reader to Vaidyanathan, Ahuja, and Orlin (2007c) for further details.

The train schedule satisfies the reducability property. Hence, in a similar manner to the LPP, we aggregate all the instances of a train in the week into a single aggregated train. The flow on each aggregated train represents the total flow on all weekly trains corresponding to it. We also consider strings which are now made up of aggregated flights and refer to these strings as aggregated strings. These aggregated strings are the decision variables in the aggregated LRP. Using this method of variable aggregation, we are able to reduce the number of variables from close to a million, to several thousand. Also, in the aggregated LRP we have only one partitioning constraint for every aggregated train instead of one for every train in the original model. Since the average frequency of aggregated trains is close to five, the number of partitioning constraints becomes roughly one-fifth that of the original model. Also, since the number of aggregated trains is one-fifth the number of original trains, the number of flow balance constraints also reduces by the same factor. Thus, the aggregation of variables and constraints results in a very compact formulation and consequently very fast run times. We also developed a disaggregation algorithm using which this solution can be disaggregated to obtain near optimal solutions to the original problem. Our computational results demonstrated that the aggregation approach was able to solve a number of different instances of the LRP within five minutes of computational time and with less than a 2.2 % optimality gap.

Another routing problem of interest is the aircraft maintenance routing problem. Several researchers have worked on this problem (Feo and Bard (1989), Clarke et al. (1997), Gopalan and Talluri (1998), Talluri (1998), Barnhart et al. (1998)). Barnhart et al. (1998) use a branch-and-price based approach to solve the aircraft routing problem. Their formulation has a lot of

similarities to the formulation of the LRP described above. However, even for small problems with 150-200 flights their approach takes 3-4 hours of computational time. Since the aircraft routing problem is very similar in nature to LRP and is in fact simpler in several ways (described in Vaidyanathan, Ahuja, and Orlin (2007c)), we believe that the aggregation framework has the potential to be a useful speed-up tool to solve this problem too.

5.5 Computational Results

In this section, we provide computational results of our algorithms on several instances of the LPP with varying transport volumes. We implemented our algorithms in the Microsoft Visual Basic programming language and tested them on the data provided by a major Class I US railroad. We modeled our integer programs using ILOG Concert Technology 2.0 modeling language and solved them using the CPLEX 9.0 solver. We conducted all our computational tests on an Intel Pentium 4, 512-MB RAM, and 2.4-GHz processor desktop computer.

In order to test the performance of the aggregation framework, we first solve the LPP exactly using the consist based formulation proposed in Vaidyanathan et al. (2007b) and measure the solution cost and computational times. We then use the aggregation approach to solve this problem and tabulate the performance of the three disaggregation approaches described in Section 5.3. We present the results of our computational test in Table 5-1 and Figure 5-3.

From our computational tests we make the following conclusions:

- The aggregation based methods proposed were able to solve all instances of the LPP to less than 2% optimality gap in less than one-fifteenth the original computational time.
- The difference between method 1 and method 2 is close to 1% in terms of optimality gap. However, the difference in performance between methods 2 and 3 is only marginal.
- The aggregation based framework is able speed-up the solution times of all instances of the LPP from a few minutes to a few seconds.

5.6 Summary and Conclusions

We develop an aggregation framework that that can be applied to solve a class of large-scale integer multicommodity flow problems (MCFP). We define the terminology and describe the properties of this class of problems in terms of the underlying network, the objective function, and the constraints. The basic philosophy behind the aggregation approach is that instead of solving a large-scale problem directly, we divide our approach into two stages. In the first stage, we solve a much smaller aggregated model to optimality. Then, in the next stage we disaggregate the solution of the first stage to obtain a near optimal solution to the original problem.

We formulate the class of MCFP as an integer programming problem on a network and describe the constraints in this problem. We also identify the properties that the network has to satisfy for the aggregation framework to be applicable. We formulate the aggregated version of the MCFP and show that it is much smaller in size than the original problem in terms of number of decision variables and constraints. We establish the relationship between the feasibility of the original MCFP and the feasibility of the aggregated MCFP, and also describe the importance of estimating the duration of arcs to ensure that the aggregated model is a good approximation of the original model. Finally, we describe three methods by which we can disaggregate the solution to obtain the near optimal solutions to the original problem.

We demonstrate the applicability of the aggregation framework to solve several problems. We first describe an application of the aggregation framework to solve the LPP. We show that the LPP is a special case of the MCFP. Due to the nature of the aggregated LPP, its performance depends critically on the estimation of train connection times. While the train connection times in the original network can be computed trivially, in the aggregated network this information is not readily available. This is because, in the aggregated network, we abstract out the information

regarding the operating day of a train and only consider its frequency and arrival and departure times of the day. We describe methods to estimate train connection times on the aggregated network in such a way that the aggregated model is a very good approximation of the original model. Since the aggregated problem is much smaller in size than the original problem, it can be solved to optimality quickly. We provide extensive computational results on the LPP to demonstrate the efficacy of our approach. The computational results demonstrate that the aggregation algorithm is able to obtain near optimal solutions (less than two percent gap) while consuming only one-fifteenth the original computational times. We also outline applications of the aggregation framework to solve airline fleet assignment problem, airline CSPs, LRPs, and aircraft routing problems.

We believe that the aggregation framework described has the potential to be effectively applied to solve large-scale optimization problems. The main benefit of this approach is the fact aggregation shrinks the problem size leaving its structure virtually unaltered. Hence, the existing state-of-the-art algorithms may be directly applied to the aggregated problem. Due to this reason, it can be used as a tool to speed-up existing solution techniques to a wide variety of problems.

Table 5-1. Performance of the aggregation algorithm.

SNo	Tonnage	Exact Formulation			Aggregation method 1				Aggregation method 2				Aggregation method 3			
		Cost	Time	Con.	Cost	Time	Con.	Gap (%)	Cost	Time	Con.	Gap (%)	Cost	Time	Con.	Gap (%)
1	5,700	5,911,644	3m 20s	94.82	5,936,426	20.18s	94.82	0.42	5,998,630	5.54s	95.47	1.47	5,985,253	5.74s	95.14	1.23
2	6,080	6,325,823	8m 52s	95.14	6,369,353	27.31s	95.79	0.69	6,439,583	5.8s	94.89	1.80	6,414,983	6.22s	94.78	1.39
3	6,460	6,693,232	5m 57s	94.85	6,719,490	25.41s	94.46	0.39	6,825,278	6.87s	93.85	1.97	6,795,265	7.20s	93.92	1.50
4	6,840	6,994,761	6m 41s	94.93	7,035,412	16.79s	95.43	0.48	7,113,596	6.43s	94.82	1.70	7,088,953	8.44s	94.93	1.33
5	7,220	7,444,726	6m 41s	94.75	7,519,868	23.57s	94.10	1.01	7,588,132	5.57s	94.28	1.93	7,588,063	5.74s	94.21	1.89
6	7,600	7,848,320	7m 09s	94.96	7,895,066	34.47s	95.29	0.60	7,982,082	6.09s	93.81	1.70	7,965,902	6.29s	93.59	1.48
7	7,980	8,295,953	6m 01s	94.64	8,369,668	20.05s	93.99	0.89	8,458,740	6.09s	92.88	1.96	8,442,541	6.50s	93.31	1.74

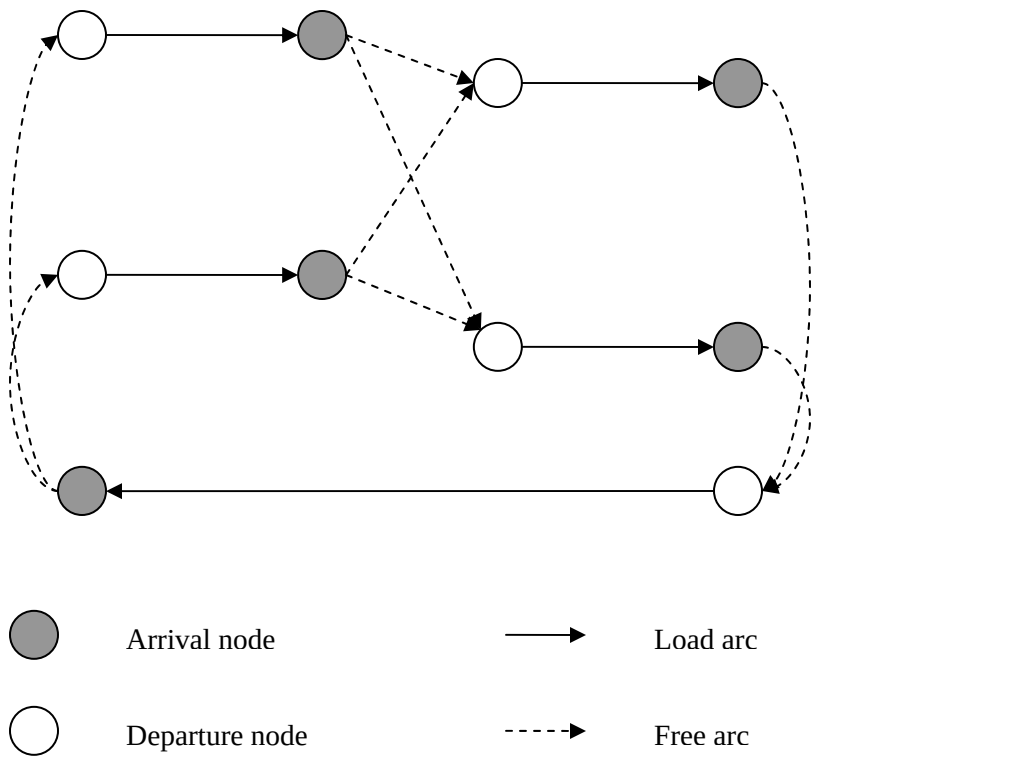

Figure 5-1. Network.

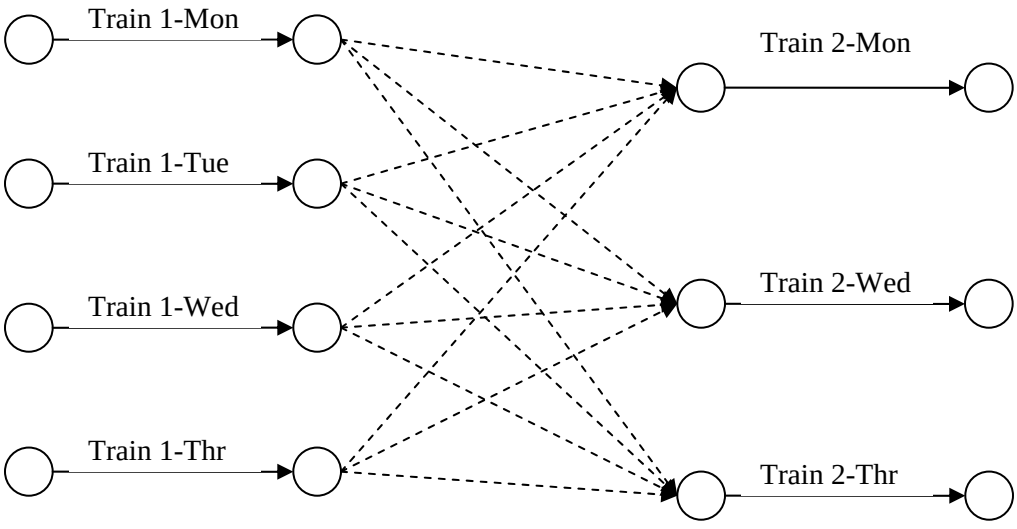

Figure 5-2. Potential train connections.

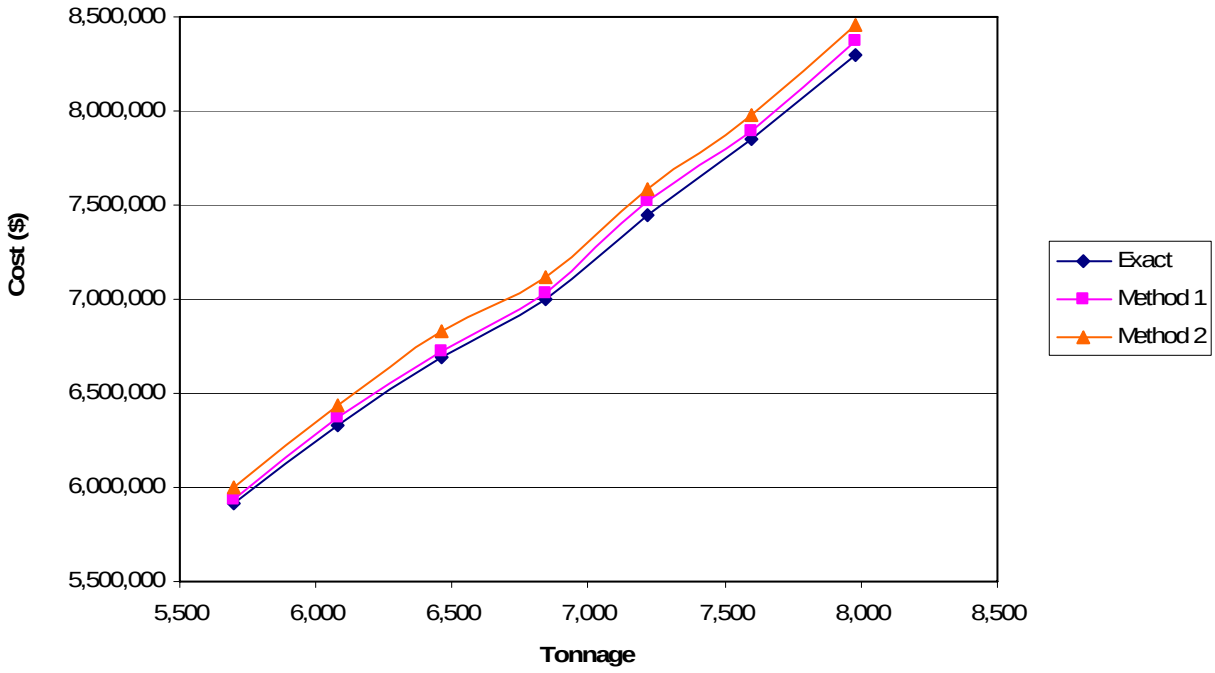

Figure 5-3. Performance of the aggregation algorithm.

LIST OF REFERENCES

Abara, J. 1989. Applying integer linear programming to the fleet assignment problem. *Interfaces* **19** 20-28.

Ahuja, R.K., K.C. Jha, J. Liu. 2007. Solving real-life railroad blocking problems. *Interfaces* forthcoming.

Ahuja, R.K., J. Liu, J. Goodstein, A. Mukherjee, J.B. Orlin. 2004. A neighborhood search algorithm for the combined through and fleet assignment model with time windows. *Networks* **44** 160-171.

Ahuja, R.K., T.L. Magnanti, J.B. Orlin. 1993. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, NJ.

Ahuja, R.K., J. Liu, J.B. Orlin, D. Sharma, L.A. Shughart. 2005. Solving real-life locomotive scheduling problems. *Transportation Science* **39** 503-517.

Assad, A.A. 1978. Multicommodity network flows - A survey. *Networks* **8** 37-91.

Assad, A.A. 1980a. Solving linear multicommodity flow problems. Proceedings of the IEEE International Conference on Circuits and Computers, 157-161.

Assad, A.A. 1980b. Models for rail transportation. *Transportation Research* **14A** 205-220.

Assad, A.A. 1981. Analytical models in rail transportation: an annotated bibliography. *INFOR* **19** 59-80.

Assad, A.A. 1983. Analysis of rail classification policies. *INFOR* **21** 293-314.

Barnhart, C. 1988. A network-based primal-dual solution methodology for the multicommodity network flow problem. Ph. D. Dissertation, Dept. of Civil Engineering, MIT, Cambridge, MA.

Barnhart, C. 1993. Dual-ascent methods for large-scale multi-commodity flow problems. *Naval Research Logistics* **40** 305-324.

Barnhart, C., N.L. Boland, L.W. Clarke, E.L. Johnson, G.L. Nemhauser, R.G. Shenoi. 1998. Flight string models for aircraft fleeting and routing. *Transportation Science* **32** 208-220.

Barnhart, C., C.A. Hane, P.H. Vance. 2000a. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Operations Research* **48** 318-326.

Barnhart, C., H. Jin, P.H. Vance. 2000b. Railroad blocking : A network design application. *Operations Research* **48** 603-614.

Barnhart, C., E. L. Johnson, R. Anbil, L. Hatay. 1994. A column generation technique for the long-haul crew assignment problem. T. Ciriano, R. Leachman, eds. *Optimization in Industry: Volume II*. John Wiley and Son, England, UK, 7-22.

Barnhart, C., E. L. Johnson, G. L Nemhauser, P. H. Vance. 2003. Crew scheduling. R.W. Hall, eds. *Handbook of Transportation Science.* Kluwer Academic Publisher, Norwell, MA, 493-521.

Barnhart, C., T.S. Kniker, M. Lohatepanont. 2002. Itinerary-based airline fleet assignment. *Transportation Science* **36** 199-217.

Barnhart, C., Y. Sheffi. 1993. A network-based primal-dual heuristic for the solution of multicommodity network flow problems. *Transportation Science* **27** 102-117.

Bellmore, M., G. Bennington, S. Lubore. 1971. A multivehicle tanker scheduling problem. *Transportation Science* **5** 36-47.

Bodin, L., B. Golden, A.A. Assad, M. Ball. 1983. Routing and scheduling of vehicles and crews: The state of the art. *Computer and Operations Research* **10** 63-211.

Bodin, L.D., B.L. Golden, A.D. Schuster, W. Rowing. 1980. A model for the blockings of trains. *Transportation Research* **14B** 115-120.

Booler, J.M.P. 1980. The solution of a railway locomotive scheduling problem. *Journal of the Operational Research Society* **31** 943-948.

Booler, J.M.P. 1995. A note on the use of Lagrangean relaxation in railway scheduling. *Journal of the Operational Research Society* **46** 123-127.

Brannlund, U., P.O. Lindberg, A. Nou, J.E. Nilson. 1998. Railway timetabling using Lagrangian relaxation. *Transportation Science* **32** 358-369.

Brunetta, L., M. Conforti, M. Fischetti. 2000. A polyhedral approach to an integer multicommodity flow problem. *Discrete Applied Mathematics* **101** 13-36.

Campbell, K.C. 1996. Booking and revenue management for rail intermodal services. PhD Dissertation. Department of Systems Engineering, University of Pennsylvania, Philadelphia, PA.

Caprara, A., M. Fischetti, P. Toth, D. Vigo, P.L. Guida. 1997. Algorithms for railway crew management. *Mathematical Programming* **79** 124-141.

Carraresi, P., G. Gallo. 1984. Network models for vehicle and crew scheduling. *European Journal of Operational Research* **16** 139-151.

Castro, J, N. Nabona. 1996. An implementation of linear and nonlinear multicommodity network flows. *European Journal of Operational Research* **92** 37-53.

Chen, H., C.G. Dewald. 1974. A generalized chain labeling algorithm for solving multicommodity flow problems. *Computers and Operations Research* **1** 437-465.

Chih, K.C., M.A. Hornung, M.S. Rothenberg, A.L. Kornhauser. 1990. Implementation of a real time locomotive distribution system. T.K.S. Murthy, R.E. Rivier, G.F. List, J. Mikolaj, eds. *Computer Applications in Railway Planning and Management*. Computational Mechanics Publications, Southampton, UK, 39-49.

Chu, C.K., C.H. Chan. 1998. Crew scheduling of light rail transit in Hong Kong: From modeling to implementation. *Computer and Operations Research* **25** 887-894.

Clarke, L., E. Johnson, G. Nemhauser, Z. Zhu. 1997. The aircraft rotation problem. *Annals of Operations Research* **69** 33-46.

Cordeau, J.F., F. Soumis, J. Desrosiers. 1998a. A Benders decomposition approach for the locomotive and car assignment problem. Technical Report G-98-35, GERAD, Ecole des Hautes Etudes Commerciales de Montreal, Canada.

Cordeau, J.F., P. Toth, D. Vigo. 1998b. A survey of optimization models for train routing and scheduling. *Transportation Science* **32** 380-404.

Crainic, T.G., J.A. Ferland, J.M. Rousseau. 1984. A tactical planning model for rail freight transportation. *Transportation Science* **18** 165-184.

Cremeans, J.E., R.A. Smith, G.R. Tyndall. 1970. Optimal multicommodity network flows with resource allocation. *Naval Research Logistics Quarterly* **17** 269-280.

Dantzig, G.B., P. Wolfe. 1960. Decomposition principle for linear programs. *Operations Research* **8** 101-111.

Desrosiers, J., Y. Dumas, M.M. Solomon, F. Soumis. 1995. Time-constrained routing and scheduling. M.O. Ball, eds. *Handbooks in OR & MS, Vol. 8*. Elsevier Science, 35-139.

Detlefsen, N.K., S.W. Wallace. 2002. The simplex algorithm for multicommodity networks. *Networks* **39** 15-28.

Ernst, A.T., H. Jiang, M. Krishnamoorthy, H. Nott, D. Sier. 2001. An integrated optimization model for train crew management. *Annals of Operations Research* **108** 211-224.

Evans, J.R. 1977. Some network flow models and heuristics for multiproduct production and inventory planning. *AIIE Transactions* **9** 75-81.

Farvolden, J.M., W.B. Powell. 1994. Subgradient methods for service network design problem. *Transportation Science* **28** 256-272.

Farvolden, J.M., W.B. Powell, I.J. Lustig. 1993. A primal partitioning solution for the arc-chain formulation of a multicommodity network flow problem. *Operations Research* **41** 669-693.

Feo, T.A., J.F. Bard. 1989. Flight scheduling and maintenance base planning. *Management Science* **35** 1415-1432.

Fischetti, M., P. Toth. 1997. A package for locomotive scheduling. Technical Report DEIS-OR-97-16, University of Bologna, Italy.

Florian, M., G. Bushell, J. Ferland, G. Guerin, L. Nastansky. 1976. The engine scheduling problem in a railway network. *INFOR* **14** 121-138.

Florida Department of Transportation Report. 2005. Freight rail component of the Florida Rail Plan. Technical Report.

Freling, R., R.M. Lentink, A.P.M. Wagelmans. 2004. A decision support system for crew planning in passenger transportation using a flexible branch-and-price algorithm. *Annals of Operations Research* **127** 203-222.

Forbes, M.A., J.N. Holt, A.M. Watts. 1991. Exact solution of locomotive scheduling problems. *Journal of Operational Research Society* **42** 825-831.

Ford, L.R., D.R. Fulkerson. 1958. A suggested computation for maximal multicommodity network flow. *Management Science* **5** 97-101.

Frangioni, A., G. Gallo. 1999. A bundle type dual-ascent approach to linear multicommodity min-cost flow problems. *INFORMS Journal on Computing* **11** 370-393.

Garey, M.R., D.S. Johnson. 1979. *Computers and intractability: A guide to the theory of NP-completeness*. W.H. Freeman, New York, NY.

Gautier, A., F. Granot. 1995. Forest management: A multicommodity flow formulation and sensitivity analysis. *Management Science* **41** 1654-1688.

Gersht, A., A. Shulman. 1987. A new algorithm for the solution of the minimum cost multicommodity flow problem. *Proceedings of the IEE Conference on Decision and Control* **26**, 748-758.

Goeffrion, A.M., G.W. Graves. 1974. Multicommodity distribution system design by Benders decomposition. *Management Science* **20** 822-844.

Golden, B.L. 1975. A minimum cost multicommodity network flow problem concerning imports and exports. *Networks* **5** 331-356.

Gopalakrishnan, B., E. Johnson. 2005. Airline crew scheduling: State-of-the-art. *Annals of Operations Research* **140** 305-337.

Gopalan, R., K.T. Talluri. 1998. The aircraft maintenance routing problem. *Operations Research* **46** 260-272.

Gorman, M.F., M. Sarrafzadeh. 2000. An application of dynamic programming to crew balancing at Burlington Northern Santa Fe Railway. *International Journal of Services Technology and Management* **1** 174-187.

Graves, G.W., R.D. McBride. 1976. The factorization approach to large scale linear programming. *Mathematical Programming* **10** 91-110.

Hane, C.A., C. Barnhart, E.L. Johnson, R.E. Marsten, G.L. Nemhauser, G. Sigismondi. 1995. The fleet assignment problem: Solving a large-scale integer program. *Mathematical Programming* **70** 211-232.

Haghani, A.E. 1989. Formulation and solution of a combined train routing and makeup, and empty car distribution model. *Transportation Research* **23B** 433-452.

Hoffman, K.L, M. Padberg. 1993. Solving airline crew scheduling problems by branch-and-cut. *Management Science* **39** 657-682.

Holmberg, K., D. Yuan. 2003. A multicommodity network-flow problem with side constraints on paths solved by column generation. *INFORMS Journal on Computing* **15** 42-57.

Jha, K.C. 2004. Very large-scale neighborhood search heuristics for combinatorial optimization problems. Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL.

Kaplan, S. 1973. Readiness and the optimal redeployment of resources. *Naval Research Logistics Quarterly* **20** 625-638.

Keaton, M.H. 1989. Designing optimal railroad operation plans: Lagrangian relaxation and heuristic approaches. *Transportation Research* **23B** 415-431.

Keaton, M.H. 1992. Designing optimal railroad operating plans: A dual adjustment method for implementing Lagrangian relaxation. *Transportation Science* **26** 262-279.

Kennington, J.L., R.V. Helgason. 1980. *Algorithms for Network Programming*. Wiley-Interscience, NY.

Kennington, J.L. 1978. Survey of linear cost multicommodity network flows. *Operations Research* **26** 209-236.

Kennington, J.L., M. Shalaby. 1977. An effective subgradient procedure for minimal cost multicommodity flow problems. *Management Science* **23** 994-1004.

Korte, B. 1988. Applications of combinatorial optimization. Technical Report No. 88541-OR. Institute für Okonometrie und Operations Research, Nassestrasse 2, D-5300, Bonn, West Germany.

Kraft, E.R. 1998. A reservation-based railway network operations management system. PhD Dissertation. Department of Systems Engineering, University of Pennsylvania, Philadelphia, PA.

Larsson, T., D. Yuan. 2004. An augmented Lagrangian algorithm for large scale multicommodity routing. *Computational Optimization and Applications* **27** 187-215.

Lin, Y., J. Yuan. 2001. On a multicommodity flow network reliability model and its application to a container-loading transportation problem. *Journal of Operations Research Society of Japan* **44** 366-377.

Lohatepanont, M., C. Barnhart. 2004. Airline schedule planning: integrated models and algorithms for schedule design and fleet assignment. *Transportation Science* **38** 19-32.

Mamer, J.W., R.D. McBride. 2000. A decomposition-based pricing procedure for large-scale linear programs: An application to the linear multicommodity flow problem. *Management Science* **46** 693-709.

Maroti, G., L. Kroon. 2005. Maintenance routing for train units: the transition model. *Transportation Science* **39** 518-525.

Matsumoto, K., T. Nishizeki, N. Saito. 1985. An efficient algorithm for finding multicommodity flows in planar networks. *SIAM Journal on Computing* **14** 289-302.

Newton, H.N., C. Barnhart, P.M. Vance. 1998. Constructing railroad blocking plans to minimize handling costs. *Transportation Science* **32** 330-345.

Nou, A., J. Desrosiers, F. Soumis. 1997. Weekly locomotive scheduling at Swedish State Railways. Technical Report G-97-35, GERAD, Ecole des Hautes Etudes Commerciales de Montreal, Canada.

Ouaja, W., Richards, B. 2004. A hybrid multicommodity routing algorithm for traffic engineering. *Networks* **43** 125-140.

Pinar, M.C., S.A. Zenios. 1990. Parallel decomposition of multicommodity network flows using smooth penalty functions. Technical Report 90-12-06, Department of Decision Sciences, Wharton School, University of Pennsylvania, Philadelphia, PA.

Radzig, T. 1997. Fast deterministic approximation for the multicommodity flow problem. *Mathematical Programming* **78** 43-58.

Ramani, K.V. 1981. An information system for allocating coach stock on Indian Railways. *Interfaces* **11** 44-51.

Rexing, B., C. Barnhart, T. Kniker, A. Jarrah, N. Krishnamurthy. 2000. Airline fleet assignment with time windows. *Transportation Science* **34** 1-20.

Schneur, R. 1991. Scaling algorithms for multicommodity flow problems and network flow problems with side constraints. Ph.D. Dissertation, Department of Civil Engineering, MIT, Cambridge, MA.

Schneur, R., J.B. Orlin. 1998. A scaling algorithm for multicommodity flow problems. *Operations Research* **46** 231-246.

Smith, S., Y. Sheffi. 1988. Locomotive scheduling under uncertain demand. *Transportation Research Records* **1251** 45-53.

Surface Transportation Board, US Department of Transportation, Bureau of Accounts, Employment Data, 2006.

Swoveland, C. 1971. Decomposition algorithms for the multi-commodity distribution problem. Working Paper No. 184, Western Management Science Institute, University of California, Los Angeles.

Talluri, K.T. 1996. Swapping applications in daily airline fleet assignment. *Transportation Science* **30** 237-248.

Talluri, K.T. 1998. The four-day aircraft maintenance routing problem. *Transportation Science* **32** 43-53.

Tomlin, J.A. 1966. A linear programming model for the assignment of traffic. *Proceedings of the 3rd Conference of the Australian Road Research Board* **3**, 263-271.

Vaidyanathan, B., K.C. Jha, R.K. Ahuja. 2007a. Multicommodity network flow approach to the railroad crew-scheduling problem. *IBM Journal of Research and Development* **51** 325-344.

Vaidyanathan, B., R.K. Ahuja, J. Liu, L.A. Shughart. 2007b. Real-life locomotive planning: new formulations and computational results. *Transportation Research B* forthcoming.

Vaidyanathan, B., R.K. Ahuja, J.B. Orlin. 2007c. The locomotive routing problem. Working paper, Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL.

Vance, P.H., C. Barnhart, E.L. Johnson, G.L. Nemhauser. 1997. Airline crew scheduling: A new formulation and decomposition algorithm. *Operations Research* **45** 188-200.

Walker, C.G., J. N. Snowdon, D. M. Ryan. 2005. Simultaneous disruption recovery of train timetable and crew roster in real time. *Computer and Operations Research* **32** 2077-2094.

Wise, T. H. 1995. *Column Generation and Polyhedral Combinatorics for Airline Crew Scheduling.* Ph.D. Dissertation, Cornell University, Ithaca, NY.

Wren, A. 1981. *Computer Scheduling of Public Transport*. North-Holland, Amsterdam.

Wright, M.B. 1989. Applying stochastic algorithms to a locomotive scheduling problem. *Journal of the Operational Research Society* **40** 187-192.

Ziarati, K., F. Soumis, J. Desrosiers, S. Gelinas, A. Saintonge. 1997. Locomotive assignment with heterogeneous consists at CN North America. *European Journal of Operational Research* **97** 281-292.

Ziarati, K., F. Soumis, J. Desrosiers, M.M. Solomon. 1999. A branch-first, cut-second approach for locomotive assignment. *Management Science* **45** 1156-1168.

BIOGRAPHICAL SKETCH

Balachandran Vaidyanathan is from India and he received his early education in the coastal city of Chennai. He received his bachelor's degree in naval architecture and ocean engineering from the Indian Institute of Technology, Madras, India in 2002. Since August 2003, he has been pursuing his doctoral degree at the Department of Industrial and Systems Engineering, University of Florida. His academic interests include mathematical modeling and design of algorithms to solve real-world problems. He has made contributions in the areas of railroad crew and locomotive scheduling and has been awarded INFORMS 2004 and 2006 Honorable Mentions in their annual student paper competition on Management Science in Railroad Applications. He also received the Graduate Student Best Paper Award 2007 from the Transportation Research Forum.