New Aggregation Methods for Multicommodity Network Flow Problems: Theory and Applications to Locomotive Refueling

A thesis submitted for the degree of

Doctor of Philosophy

by

Ahmad Kazemi

Supervisors:
Prof. Andreas T. Ernst
Dr. Pierre Le Bodic
Prof. Mohan Krishnamoorthy

School of Mathematics
Monash University, Australia

September, 2021

To my wonderful parents

Copyright Notice

Declaration

This thesis is an original work of my research and contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

Signature:

Print Name: Ahmad Kazemi

Date: 07 September 2021

Acknowledgements

I would like to express my gratitude to my dear supervisors, Prof. Andreas Ernst, Dr. Pierre Le Bodic, and Prof. Mohan Krishnamoorthy. Thank you for your invaluable support, insights, and inspiration, which not only made this thesis possible but also helped me to grow and enjoy my PhD.

I would also like to express my thanks to John Chan for his continuous help from the very first day of my PhD and Dr. Asef Nazari for his insightful professional advice.

I would like to express my deepest gratitude to my family for their never-ending love and support: my beloved parents, Tayyebe and Mostafa, my kind-hearted sister, Saeedeh, my amazing brother, Alireza, my dear brother-in-law, Farid, and my sweet nephew, Farhan.

I would like to extend my sincere thanks to my treasured friends, Atabak Elmi, Malihe Rezaie, Mara Antic, Fahime Saleh, Mahsa Naseri, Timothy Chan, and Flora Tourchi, whom I met during my new life in Australia and have made me feel at home.

I would like to extend my sincere thanks to my lifelong, cherished friends, Mahmoud Shabani, Pedram Pishvaeian, Kamyar Pakdaman, Sajjad Kazemi, Mohammad Arab, Zabih Ghelichi, Iman Maghami, Amin Maghami, Navid Asmari, Reza Rezaee, and Amir Saadatjoo, whose love has never stopped by geographical barriers and are always in my heart.

This research was supported by an Australian Government Research Training Program (RTP) Scholarship. I would also thank Australia Research Council and Pacific National Pty. Ltd. for funding and motivating this research.

Abstract

Network flow problems appear in an extensive variety of applications in transportation, supply chain management, logistics, communication networks, and energy systems, to name but a few. A network structure is usually intrinsic in those problems, such as the network of the cities and their links in a routing problem. In addition to such applications, there exist problems that could be modelled as time- (or resource-) expanded networks although they may not exhibit an explicit network structure, for instance, a train scheduling problem over a set of stations, which adds a time dimension to the network of the stations to represent the time-related decisions. The application of network flow problems is significant for transportation problems as they are naturally expressed over physical networks.

Although network models are well studied, general approaches are not efficient when applied to special cases of large industrial problems. Particularly, while minimum-cost network flow problems are "easy" to solve, they are often embedded within network design type problems and grow very large when the need to track flow between many origin-destination pairs requires a large number of commodities to be included. This thesis is motivated by such a problem in rail transportation that is modelled based on a network representation. The problem consists in optimizing the refueling operations of transcontinental long-haul trains by Australia's largest railway company. The traditional approach to reduce the fueling costs is to fill the locomotives at inexpensive stations to bypass the more expensive stations. However, this approach is not applicable to rail networks with long-haul operations. To overcome this issue, railway companies have started adopting inline refueling tanks, a supplementary reservoir which can refuel locomotives during a trip, and can be refueled, attached, detached, and swapped at any station, independently of locomotives. This new technology offers many opportunities for railway companies, in particular to facilitate long-haul trips and to leverage fuel cost differences. As always, new opportunities mean new challenges, as inline tanks engender combinatorially many new possible refueling plans. Moreover, since the inline tanks are a substantial investment, the number of available inline tanks is limited. To tackle this, we design a first optimization model and develop solution algorithms that determine the efficient fuel plans for a fleet of inline tanks.

Computational experiments show that as the number of available inline tanks increases, even solving the LP relaxation of the the proposed model for the fuel management problem is significantly time-consuming. This is interesting as the same trend usually appears in solving multicommodity network flow problems, and here, we can consider inline tanks as commodities. Therefore, we propose novel aggregation techniques for mul-

ticommodity network flow problems that reduce the model size, and subsequently the LP relaxation computing time, while minimally degrade its LP bound quality. We apply the proposed aggregation techniques to the multicommodity fixed-charge network design problem as it is a well-known multicommodity network flow problem with an extensive application in transportation. These aggregation techniques result in a range of formulations with different trade-off in the LP relaxation computing time and the LP bound quality. We also apply such aggregation techniques to a variant of the real-world problem that has motivated this thesis to develop an exact solution algorithm.

The thesis makes a number of contributions, mainly:

1. It introduces a new class of fuel management problems in rail transportation, which is motivated by the operational issues of locomotive refueling in networks with long-haul travels. The problem is to efficiently plan a fleet of inline refueling tanks for a given train and locomotive schedule. We discuss the theoretical and the empirical complexity of this problem and show that this problem is strongly NP-hard even in its simplest form. This problem is conceptualized on a time-space network. A Mixed-Integer Programming (MIP) model is developed based on the underlying network, which can be viewed as an extended network design model. We apply the proposed MIP model to a real-life case study in Australia to investigate the model results and also evaluate the impacts of adoption of inline refueling on the ongoing operations of the railway companies. To evaluate the MIP model for scale-up, we apply it to a larger dataset from the USA, derived from an INFORMS railway application problem solving competition. As commercial solvers fail to provide good-quality solutions for large instances, we propose a heuristic algorithm that leverages a MIP solver and provides good-quality solutions in a reasonable time.
2. When solving hard multicommodity network flow problems using an LP-based approach, the number of commodities is a driving factor in the speed at which the LP relaxation can be solved, as it is linear in the number of constraints and variables. The conventional approach to improve the solve time of the LP relaxation of a MIP model that encodes such an instance is to aggregate all commodities that have the same origin or the same destination. However, the bound of the resulting LP relaxation can be significantly worse, which tempers the efficiency of aggregating techniques. This thesis introduces the concept of *partial aggregation* of commodities that aggregates commodities over a subset of the network instead of the conventional aggregation over the entire underlying network. This offers a high level of control on the trade-off between size of the aggregated MIP model and quality of its LP bound. We apply the concept of partial aggregation to two different MIP models for the multicommodity network design problem. We provide both theoretical results and empirical evidence

for the trade-off between the level of aggregation and LP bound tightness for the proposed models. Our computational study on benchmark instances confirms that the trade-off between solve time and LP bound can be controlled by the level of aggregation, and that choosing a good trade-off can allow the original large-scale problem to be solved faster than without aggregation or with full aggregation.

3. It applies the concept of partial aggregations to an abstract version of the real-world refueling problem introduced in this thesis. Partial aggregation results in formulations with tight bounds for this problem as well and hence, we develop an exact solution algorithm based on such aggregation schemes. The computational experiments on this algorithm demonstrate that utilizing partial aggregations can lead to a highly efficient for the problem investigated in this thesis, which is a variant of the multicommodity network design problem.

This thesis contributes to both theory and practice of the field of network optimization. It introduces a new type of network design problems with a multicommodity network flow sub-structure and an application in rail transport. Moreover, it proposes a novel and general approach, called *partial aggregations*, for a range of problems that contain a subset of variables and constraints which make up a multi-commodity flow component, including the industrial refueling problem that has motivated this thesis.

The main academic research outputs derived from this thesis are as follows:

1. Research paper: A. Kazemi, P. Le Bodic, A. T. Ernst, M. Krishnamoorthy. New partial aggregations for multicommodity network flow problems: An application to the fixed-charge network design problem, accepted and published in *Computers & Operations Research*, 136, 105505, 2021.
2. Research paper: A. Kazemi, A. T. Ernst, M. Krishnamoorthy, P. Le Bodic. Locomotive fuel management with inline refueling, accepted and published in *European Journal of Operational Research*, 293, 1077-1096, 2021.
3. Abstract paper: A. Kazemi, A.T. Ernst, M. Krishnamoorthy, P. Le Bodic. Locomotive fuel management with inline refueling, presented at *The Tenth Triennial Symposium on Transportation Analysis (TRISTAN X)*, Hamilton Island, Australia, June 2019.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Motivation

Railroad companies play a vital role in the freight transportation industry. Fuel costs constitute a significant part of the operating costs of these companies. It is the second largest cost type of the total operational costs of railroad companies in the USA (BNSF Railway, 2018; Union Pacific Corporation, 2018) and the third largest in Australia (Asciano, 2017). The annual fuel and fuel-related costs add up to over 2 billion dollars for the largest American railroad companies and over 200 million dollars for the largest Australian railroad companies. Therefore, even small improvements in fuel management of railroad companies are likely to contribute to annual savings of millions of dollars (Nourbakhsh and Ouyang, 2010). Furthermore, decreased transportation costs reduces the final price of goods for the end customers. This, in turn, makes railroad companies more competitive and could drive further demand for their services. Consequently, improving the fuel efficiency of railroad companies is critical.

Traditionally, locomotive fuel management consists of three interrelated problems (Nourbakhsh and Ouyang, 2010; INFORMS Problem Solving Competition, 2010):

1. decide the stations at which fuel contracts must be entered into, so as to facilitate refueling operations;
2. given a timetable, determining which locomotives must refuel at which particular station; and,

3. the amount of fuel to purchase at a station.

The fuel management cost structure is comprised of three cost types, and each of the interrelated problems corresponds to one. These three cost types are: (a) the fixed cost for using a station during the time horizon, (b) the the fixed cost of each refueling operation, the *stop cost*, and (c) the fuel price. Appropriate long term contracts may be entered into with suppliers of fuel at a fixed cost per refueling operation. The appropriate (and optimal) refueling choices also make use of the knowledge that fuel costs are likely to vary substantially between stations. By filling locomotives at inexpensive stations (and by bypassing stations at which fuel is relatively more expensive) – a strategy called *tankering* – the fuel purchasing costs can be substantially reduced. Although interrelated, these three problems are usually solved sequentially as they are together too difficult to solve with current optimization technology for the large instances encountered in practice.

In networks that have long distances between stations (relative to the locomotive tank capacity), the tankering strategy is likely to be of limited applicability as the tank capacity may not allow many successive trips. Furthermore, for some long-haul trips, railroad companies have to assign more locomotives than the number that pulling power requirement of a train implies, purely because of the limited tank capacity of the locomotives. Therefore, an alternative strategy may be required to address this challenge. One such example is the rail network in Australia, in which the distances between the stations are long, relative to tank capacity. In such cases, *inline refueling* may be the only technological option that offers alternative refueling plans. Inline refueling is a feature offered by *inline tanks* and allows the locomotives to take long-haul trips without the need to stop at the intermediate stations to refuel. An inline tank is a large fuel reservoir which connects to the locomotives through a pumping system and can transfer fuel to locomotives during a trip. An example of such inline refueling tanks is provided in Figure 1.1.

Figure 1.1: An example of inline refueling tanks. The picture courtesy of Pacific National Pty. Ltd., 2018.

Inline refueling is fundamentally different from traditional auxiliary refueling options, such as aerial refueling in the aviation industry and emergency refueling, as we will show

in this thesis. An aerial refueling operation transfers the fuel from an aircraft (tanker) to the aircraft that is in service (receiver) during its flight, and the tanker is usually located at a fixed base station. Emergency refueling operations are independent from each other and more expensive than the direct refueling at the fuel stations. On other hand, inline tanks can refuel locomotives during a trip, and can be refueled, attached, detached, and easily swapped at any station, independently of locomotives. Therefore, this new technology, particularly because of the possibility of switching inline tanks between locomotives while they carry fuel, offers many opportunities for railway companies. Such opportunities benefit the railroad companies to: (a) avoid assigning more locomotives to a train because of tank capacity limits and reduce the locomotive ownership cost, (b) facilitate long-haul trips and leverage fuel cost differences, (c) reduce the number of refueling operations, and (d) close the remote fuel stations and reduce the corresponding cost.

The opportunities offered by inline refueling come with additional complexity in the fuel management problem. Incorporating the inline refueling in the fuel management increase the number of possible fuel plans combinatorially. Moreover, since the inline tanks are a substantial investment, the number of available inline tanks is limited. Therefore, efficient planning of the inline refueling requires the choice of the best among a huge number of plans. Inline refueling links the fuel decisions of all locomotives in addition to the station location decisions of the traditional integrated problem. In contrast to the traditional integrated problem, multiple locomotives that operate on the same train service cannot be solved independently. These locomotives must be jointly considered since the locomotives that haul a train can share an inline tank. Such features together make the problem of locomotive fuel management with inline refueling complex in practice and in theory, as we will show.

1.2 Framework and Methodology

We first define and study the computational complexity of the fuel management problem with inline refueling and show it is a strongly NP-hard problem. We adopt the Mixed-Integer Programming (MIP) methods to tackle this problem as they have been successfully applied to such problems. We develop the first optimization MIP model to address the locomotive fuel management with inline refueling. The proposed model is based on a network representation, which is a frequent structure in transportation problems. Network optimization problems appear in an extensive variety of applications in transportation, supply chain management, logistics, communication networks, and energy systems, to name but a few. The network structure is usually intrinsic in those problems, such as the network of the cities and their links in a routing problem. The application of network optimiza-

tion problems is significant for transportation problems as they are naturally expressed over physical networks. Therefore, this thesis particularly focuses on network optimization techniques.

Although network optimization models are well studied, general approaches are not efficient when applied to large industrial problems. Particularly, while minimum-cost network flow problems are "easy", they are often embedded within network design type problems and grow very large when the need to track flow between many origin-destination pairs requires a large number of commodities to be included. The proposed MIP model for the fuel management problem is of the same type of such network optimization models and includes a multicommodity flow substructure. A well-known large-scale optimization method to reduce the size of large instances of such network optimization models is to aggregate all commodities with the same origin *fully* over the entire network. Although the full aggregation approach reduces a network optimization MIP model size, it significantly degrades the quality of its linear programming relaxation. We propose novel aggregation techniques for network optimization problems with a multicommodity substructure that aggregate the commodities over a subset of the network. These methods significantly reduce the model size while minimally degrading its linear programming quality. We investigate the significance and implications of partial aggregations for: (a) quality of alternative MIP models and their LP relaxations, (b) the performance of general MIP algorithms, and (c) development of specialized solution algorithms.

This thesis contributes to both theory and practice of the field of network optimization. It fills the gap in the fuel management literature by introduction and investigation of a new class of fuel management problems. It provides both theoretical results of the complexity of and algorithms to solve such problems and also studies the business implications of using the proposed models and methods on the ongoing operations of the railroad companies. Furthermore, it extends the state-of-the-art large scale optimization techniques to deal with large network optimization problems with a multicommodity flow substructure by introducing the concept of partial aggregations. It also empirically demonstrates the effectiveness of such aggregation schemes by applications to a classical network optimization problem and a variant of the practical fuel management problem introduced.

1.3 Main Contributions and Publications

This thesis contributes to the field of network optimization and applied operations research in several ways, through:

- The introduction of a class of fuel management problems in rail transport, the *locomotive fuel management problem with inline refueling* (LFMIR) and its variants.

- The complexity analysis of the LFMIR problem, which shows that unlike traditional locomotive refueling problems - which are only NP-hard if they include considerations on fuel station location or fuel volume discounts - LFMIR is NP-hard even if there are no decisions to be made about locating stations nor volume discounts.
- The development of optimization models and solution algorithms for the LFMIR problem.
- The introduction of the concept of partial aggregations for optimization problems with a multicommodity network flow substructure.
- The development of new formulations for the capacitated multicommodity fixed-charge network design problem based on the concept of partial aggregations.
- The theoretical results and empirical evidence for the trade-off offered by the new formulations in terms of their linear programming bound quality and computational difficulty.
- The utilization of the concept of partial aggregations to develop an efficient solution algorithm for a variant of the LFMIR problem.

The main academic research outputs derived from this thesis are provided below.

1. Research paper: A. Kazemi, P. Le Bodic, A. T. Ernst, M. Krishnamoorthy. New partial aggregations for multicommodity network flow problems: An application to the fixed-charge network design problem, accepted and published in *Computers & Operations Research*, 136, 105505, 2021.
2. Research paper: A. Kazemi, A. T. Ernst, M. Krishnamoorthy, P. Le Bodic. Locomotive fuel management with inline refueling, accepted and published in *European Journal of Operational Research*, 293, 1077-1096, 2021.
3. Abstract paper: A. Kazemi, A.T. Ernst, M. Krishnamoorthy, P. Le Bodic. Locomotive fuel management with inline refueling, presented at *The Tenth Triennial Symposium on Transportation Analysis (TRISTAN X)*, Hamilton Island, Australia, June 2019.

1.4 Thesis Structure

The remainder of this thesis is structured as follows.

Chapter 2 - Background This chapter gives an overview of the methodologies and algorithms adopted in this thesis as well as literature reviews on the multicommodity network flow and fuel management problems. This chapter is divided into four sections. The first section introduces the main modeling and algorithmic tools that are utilized in the rest of the thesis, including, Mixed Integer Programming (MIP), Linear Programming (LP), the

Branch-and-Bound algorithm (B&B), and Matheuristics. The second section reviews three main large-scale optimization techniques for large LP and MIP models: aggregation techniques, the column generation algorithm, and the Benders decomposition algorithm. The third section presents the formal definition of multicommodity network flow problems, particularly the multicommodity fixed-charge network design problem (MCND). This section also reviews the literature of MCND with a focus on the commodity aggregations for which we introduce novel techniques and formulations in Chapter 4. The fourth section gives a comprehensive literature review of the fuel management problems in various transportation applications. Here, we define the scope of the fuel management problem that this thesis investigates and demonstrate its novelties in comparison with the current studies in the field.

Chapter 3 - Locomotive Fuel Management with Inline Refueling (LFMIR) This chapter is based on the published paper Kazemi et al. (2021a). It concentrates on the real-world fuel management problem that motivated this work. We formally define the Locomotive Fuel Management problem with Inline Refueling (LFMIR) and study its theoretical and empirical complexity. We show that LFMIR is strongly NP-hard even in its simple forms with one resource (inline tank) available. In this chapter, we conceptualize the problem by a time-space network, and hence, propose a MIP model based on the network representation, which can be viewed as an extended network design model. The proposed model is then applied to two case studies from Australia and the USA to study the impacts of adopting inline refueling on the refueling operation of railway companies. We demonstrate that commercial solvers fail to provide good-quality solutions in a reasonable time for large instances. Therefore, we develop a heuristic algorithm, which is able to provide good-quality solutions in a reasonable time for industrial use.

Chapter 4 - Partial Aggregations for Multicommodity Network Flow Problems This chapter is based on the submitted paper Kazemi et al. (2021b) and proposes novel aggregation techniques for the multicommodity network flow problems that have an extensive application in logistics, telecommunication, and particularly transportation networks. When solving hard multicommodity network flow problems using an LP-based approach, the number of commodities is a driving factor in the speed at which the LP relaxation can be solved, as it is linear in the number of constraints and variables. The conventional aggregation approach to improve the solve time of the LP relaxation of a MIP model that encodes such an instance is to aggregate all commodities that have the same origin or the same destination. However, the bound of the resulting LP relaxation can significantly worsen, which tempers the efficiency of aggregating techniques. In this chapter, we introduce the concept of *partial aggregation* of commodities that aggregates commodities over a subset of the network instead of the conventional aggregation over the entire underlying network.

This offers a high level of control on the trade-off between size of the aggregated MIP model and quality of its LP bound. We apply the concept of partial aggregation to two different MIP models for the multicommodity fixed-charge network design problem. Our computational study on benchmark instances confirms that the trade-off between solve time and LP bound can be controlled by the level of aggregation, and that choosing a good trade-off can allow us to solve the original large-scale problems faster than without aggregation or with full aggregation.

Chapter 5 - From Partial Aggregations to Solution Algorithms In this chapter, we introduce an abstract version of the LFMIR problem, which is called Inline Fuel Delivery problem (*IFD*) and retains the main complexities of the LFMIR problem. We prove that IFD is NP-hard. This chapter applies the partial aggregations proposed in Chapter 4 to the IFD problem and develops a partial aggregation-based solution algorithm. Computational experiments demonstrate the effectiveness of this algorithm in providing good-quality solutions and also tight lower bounds in short computing times. The approach of this chapter not only provides effective methods to solve IFD but also gives insights on the utilization of the concept of partial aggregations to develop specialized solution algorithms for network optimization problems with a multicommodity flow substructure.

Chapter 6 - Concluding Remarks This chapter summarizes the key results of the thesis and outlines the most promising future research directions.

CHAPTER 2

Background

2.1 Mixed Integer Programming (MIP)

Mixed Integer Programming (MIP) refers to a set of modeling techniques and algorithms that solve the optimization problems that are expressed by a linear system where a subset of its variables is restricted to integer values. MIP models naturally fit numerous applications, and MIP methods have been successfully applied to a wide variety of classic and applied optimization problems. MIP methods provide powerful tools and frameworks to solve optimization problems, including exact, heuristic, and approximation algorithms. Since such algorithms are general-purpose tools, they may require tailoring and extensions for a better performance over specific problems. This thesis extends some of the MIP algorithms to solve practical problems in railroad fuel management and also a class of network flow problems with extensive applications in transportation. This section gives an overview of the main MIP modeling techniques and algorithms adopted in this thesis.

2.1.1 Linear Programming

Linear Programming refers to a set of modeling and solution methods that seek to optimize a linear optimization problem (also called linear program). A linear program (LP) generally consists of a set of decision variables with real-valued domains, a set of linear constraints that express the relations between decision variables, and possibly an objective function as a linear combination of decision variables. Let vector x be the set of decision variables

with n elements. Matrix $A \in \mathbb{R}^{m \times n}$ represents the coefficients of decision variables in the constraints, where each row corresponds to a constraint. Vectors $c, l, u \in \mathbb{R}^n$ determine the objective function coefficient, the lower bound, and the upper bound for decision variables, respectively. The general form an LP is as the model 2.1.

$$\begin{aligned} &\text{minimize} && c^T x && \text{(2.1a)} \\ &\text{s.t.:} && Ax \quad b && \text{(2.1b)} \\ & && l \quad x \quad u && \text{(2.1c)} \end{aligned}$$

The main driving factor of computational difficulty of solving an LP problem is its size, which can be stated in terms of number of variables n, number of constraints m, number of nonzeros in the constraint matrix A, or any combination of those. Therefore, smaller LPs have usually shorter computing times. Interior-point methods (Karmarkar, 1984), also known as barrier methods, have polynomial complexity for solving linear programs in terms of their size. The Simplex method (Dantzig, 1963) and its variant, Dual Simplex (Lemke, 1954), have exponential time complexity in the LP size (Vanderbei, 2014). However, Simplex algorithms are efficient in practice. Borgwardt (1982) and Smale (1983) show that the expected computational complexity of Simplex is polynomial in m and n on various distributions of random inputs. Moreover, Spielman and Teng (2004) prove the polynomial smoothed complexity of the Simplex algorithm.

Degeneracy is another factor that makes LP algorithms slower, particularly the Simplex algorithm. Simplex iteratively traverses on the extreme points of the polyhedron defined by the constraints and variable bounds of an LP until the optimal solution is obtained. Each extreme point corresponds to a feasible solution of the LP problem, say x . At each extreme point, at least n constraints out of m constraints are active, which means they are satisfied in the equality form by the solution x . Solutions that correspond to extreme points with more than n active constraints are degenerate. Such solutions make the Simplex algorithm slower as in those cases, it may cycle between solutions with a same objective function value. Bland (1977) proposes a simple rule to avoid Simplex from infinite cycling on degenerate solutions. Reducing the number of constraints is usually beneficial to regulate the degeneracy of an LP (Shetty and Taylor, 1987). Therefore, reducing the size of an LP is useful to improve its computing time as it may diminish the degeneracy, and also the complexity of LP algorithms is dependent on the LP size.

Since linear programming provides efficient solution algorithms for linear optimization problems, it has been hearth of many MIP algorithms, such as the Branch-and-Bound algorithm. We specifically discuss the impacts of using different linear programs on the

performance of the MIP algorithms in Chapter 4. See Vanderbei (2014) for a review of various linear programming modeling and algorithmic techniques, their applications, and their theoretical and empirical complexities.

2.1.2 MIP Modeling

In many practical applications, some of the decisions are intrinsically discrete, and hence, the domain of corresponding decision variables are restricted to integer values. Let x and y be the set of continuous and integer decision variables, respectively. The general form of a MIP model, provided below, describes the relations among continuous and discrete decision variables. In this thesis, we consider minimization optimization problems as any maximization problem can be easily transformed to a minimization problem. Vectors l and u determine the lower and upper bounds of the continuous variables, and Y encodes the possible integer values for the discrete decision variables. The textbook by Williams (2013) discusses the general principles of MIP modeling in a wide variety of applications.

$$\text{minimize} \quad c^T x + f^T y \tag{2.2a}$$
$$\text{s.t.:} \quad Ax + By \quad b_1 \tag{2.2b}$$
$$Cx \quad b_2 \tag{2.2c}$$
$$Dy \quad b_3 \tag{2.2d}$$
$$l \quad x \quad u \tag{2.2e}$$
$$y \; 2 \; Y \tag{2.2f}$$

Restriction of decision variables to integer values makes the problem complicated, and MIP models are in general NP-hard to be solved (Karp, 1972). Therefore, there is not yet an efficient general algorithm for solving MIPs in contrast to the LP problems. Two main approaches to solve the MIP models, the branch-and-bound (Land and Doig, 1960) and cutting plane (Gomory, 2010) algorithms, rely on solving the *LP relaxation* of a MIP model, which is obtained by relaxing the integer restrictions of the MIP model. The branch-and-bound algorithms recursively partition the problem into smaller subproblems that are easier to solve. Such algorithms solve the LP relaxation of each subproblem to obtain a lower/dual bound for the MIP model. Moreover, the solution of the LP relaxation of some subproblems are integer-feasible, in which case it gives an upper bound for the original MIP model. A branch-and-bound algorithm stops partitioning subproblems that have a lower bound worse than the best obtained upper bound. We discuss the branch-and-bound algorithms in details later in Section 2.1.3. Cutting plane algorithms iteratively modify the

LP relaxation of a MIP model by adding constraints until an integer-feasible solution is obtained.

In addition to exact methods, solving LP relaxations is an essential part of MIP-based heuristics such as Relaxation Induced Neighborhood Search (RINS) (Danna et al., 2004). RINS considers an LP relaxation solution and an upper bound and fixes the the values of the variables that have the same value in both. Then, the resulting MIP is solved to search for an improvement. Rounding heuristics transform the solution of the LP relaxation of a MIP model to an integer point. However, the nearest integer point is not always a feasible solution for the original MIP model. The optimal integer point might be also far from the optimal solution of the LP relaxation.

Solving LP relaxations is the main component of well-known approaches for solving MIPs as discussed. Therefore, the quality of the LP relaxation of a MIP model significantly affects the performance of a MIP algorithm. The quality of the LP relaxation is usually measured by its closeness to the corresponding MIP model (Bertsimas and Tsitsiklis, 1997). A MIP model is a *tight* formulation, if its LP relaxation is "close" to it. Of course, there is no standard threshold to consider a formulation as a tight formulation, and hence, the tightness of alternative MIP models of a problem are usually compared together. The tightness is usually measured and compared by the gap between the optimal value of a MIP model and the optimal value of its LP relaxation. Let z be the optimal objective function value of the MIP model f, and $\bar{z}$ be the optimal objective function value of its LP relaxation. Such gap is measured as $g_f = \frac{z - \bar{z}}{z}$ 100. Generally, MIP models with smaller g_f are considered as tighter formulations.

Let Q be the set of feasible solutions of a MIP model, and P be the polyhedron of its LP relaxation. *Convex hull* of Q, denoted by $Conv.(Q)$, is the smallest convex polyhedron that contains all points of the set Q. A MIP model is *perfect* if $Conv.(Q) = P$. A formulation is a *compact* formulation if it includes polynomial number of constraints in terms of its input data. In special cases, such as the assignment problem, there are compact, perfect formulations that encode the problem (Conforti et al., 2014). However, this is not usually the case, where only *extended* formulations, which have exponential number of constraints in terms of their input data, can "perfectly" encode their corresponding problem (Nemhauser and Wolsey, 1999). This implies that usually larger MIP models are tighter.

In summary, two important characteristics of a MIP model are its tightness and the computing time of its LP relaxation, which is dependent on the model size. These two characteristics are mostly confronting, which make the choice of the MIP model from a set of alternative formulations crucial. We discuss how the interaction of these two characteristics of various MIP models for a problem impact the performance of the MIP algorithms in Chapter 4.

2.1.3 The Branch-and-Bound Algorithm (B&B)

The B&B algorithm is a divide-and-conquer approach to solve MIP models (Land and Doig, 1960). This algorithm recursively partitions the problem into subproblems, which forms a search tree that implicitly enumerates all candidate solutions of a MIP model. Each node of the search tree is a MIP model that its solution space is a subspace of its parent node's solution space. The solution spaces of children nodes are an exhaustive partition of their parent node's solution space. Therefore, the root node of the search tree is the original MIP model that encodes the problem. At each node, the LP relaxation of the corresponding MIP model is solved which provides a lower bound (LB). Moreover, in the case that the solution of this LP relaxation is integer-feasible, it is recorded as an upper bound (UB) for the original MIP model. Upon solving the LP relaxation of a node, the node is *pruned* if certain conditions are met. Otherwise, it is further partitioned into subproblems by a *branching* strategy. A node is pruned if its LP relaxation is infeasible, its LP relaxation solution is integer-feasible, or its LP relaxation optimal value is worse than the best upper bound. Branching rules usually partition the corresponding MIP of a node based on the domain of an integer variable that has an optimal fractional value in the corresponding LP relaxation. The goal is to avoid such a fractional value for the selected variable in the children nodes' MIPs. Suppose an integer variable y_b has the fractional optimal value $\bar{y}_b$ in the solution of a node's LP relaxation. A typical branching rule partitions the MIP of this node into two children nodes, where in one node the upper bound of y_b is restricted to $\lfloor \bar{y}_b \rfloor$ and its lower bound is restricted to $\lceil \bar{y}_b \rceil$ in the other node. The algorithm terminates when the relative/absolute gap between the best LB and the best UB is within a given tolerance. The best LB corresponds to the best objective function achievable based on the LBs provided by the nodes that are not yet pruned. The best UB corresponds to the UB with the minimum objective function value among all integer-feasible solutions.

The development of a B&B algorithm requires the design of a branching strategy and a tree search strategy. The branching strategy determines the variable that is selected for branching and how the corresponding MIP is partitioned into subproblems. The tree search strategy determines the order of nodes on which the algorithm traverses the tree. Two typical strategies are the well-known Breadth-First Search (BFS) and Depth-First Search (DFS) strategies. Using different branching and search strategies result in different variants of the B&B algorithm.

B&B is the algorithm at the base of all efficient MIP solvers, such as commercial solvers CPLEX and Gurobi. Since an LP relaxation is solved at each node, its tightness and its computing time significantly affect the performance of the B&B. A tight LP relaxation with short computing time benefits the B&B algorithm to quickly reduce the gap between the upper and lower bounds. In modern MIP solvers, B&B is usually accelerated by a number of MIP

algorithms and methods, such as primal heuristics and cutting plane algorithms. Heuristics benefit the B&B by providing (better) upper bounds, while cutting plane algorithms tighten the LP relaxations to provide tighter lower bounds.

2.1.4 Matheuristics

Metaheuristics, such as Genetic Algorithms (García-Martínez et al., 2018), Simulated Annealing (Kirkpatrick, 1984), Variable Neighborhood Search (Mladenović and Hansen, 1997; Ranjbar and Kazemi, 2018), and Ant Colony Optimization algorithm (Dorigo et al., 2006), usually do not involve the development of a mathematical model to generate solutions. Matheuristics extend such heuristics and Metaheuristics and hybridize the mathematical programming methods and Metaheuristics. Matheuristics utilize a MIP solver in a heuristic framework with the aim of providing good-quality upper bounds. They usually consider a restricted MIP model of the original MIP model that can be solved easily by a MIP solver. This is particularly beneficial because one can search over a large space by solving a MIP model (Fischetti and Fischetti, 2016). Recent advancement in MIP solvers has facilitated the development of Matheuristics.

General-purpose MIP-based heuristics are a set of Matheuristics that are widely used in MIP solvers, such as local branching (Fischetti and Lodi, 2003), RINS, and diving heuristics (Berthold, 2006). Given an upper bound with the values $\bar{y}$ for the set of binary variables B, the local branching method adds a local branching constraint in the form of

$$\Delta(y,\bar{y}) := \sum_{j \in B:\bar{y}_j=0} y_j + \sum_{j \in B:\bar{y}_j=1} (1 - y_j) \leq k.$$

Constraint $\Delta(y,\bar{y})$ imposes that the values of the binary variables y can be at most in k variables different from the given solution $\bar{y}$. This results in a MIP with a smaller search space, which usually can be solved much faster the original MIP. This method is in fact a large neighborhood search.

Chapter 3 develops a heuristic as a variant of the diving heuristic for its fuel management problem. Diving heuristics in general explore a single path in the B&B tree to obtain a feasible solution quickly. As its name suggests, this heuristic dives down a branch of the B&B tree and keeps fixing the values of the integer variables. Here, the focus is more on fixing the values of the variables rather than improving the lower bound. A diving heuristic actually can be considered as a heuristic DFS strategy that is designed based on the problem structure for a better performance.

2.2 Large Scale Optimization

This section presents three well-known methods for solving LP and MIP models with a large number of variables and/or constraints. Section 2.2.1 discusses the aggregation methods which reduce the size of the model by aggregating sets of variables, constraints, or both. While such techniques reduce the model size, they usually increase the feasible space of a formulation. We demonstrate that the increase of the feasible space may cause that the aggregated model generate infeasible solutions for the original problem. Therefore, we also present aggregation/disaggregation based algorithms that utilize the aggregation methods in an algorithmic framework to generate feasible solutions for the original problem. Section 2.2.2 presents the column generation algorithm. The idea behind this algorithm is to not include all the variables in the model and add them in the model only when they can improve a solution. Section 2.2.3 discusses the Benders decomposition algorithm that is the dual of the column generation algorithm. The Benders algorithm starts with a reduced model that includes a subset of constraints of the original model, and iteratively adds extra constraints as required. All of these methods were originally proposed for LP models. However, they are extended also to the MIP models as we will discuss in this section.

2.2.1 Row and Column Aggregation

The size of an optimization model in both aspects, i.e. number of variables and number of constraints, directly affect its computing time. Aggregation techniques are useful tools to reduce the model size to tackle the challenges of solving large-scale optimization models. In Linear Programming and Mixed-Integer Programming, the model size can be reduced by aggregating constraints (rows) (Zipkin, 1980a), variables (columns) (Zipkin, 1980b), or both at the same time (Rogers et al., 1991). Aggregation techniques have been successfully applied to a wide variety of problems in network flow optimization (Evans, 1983), mining (Boland et al., 2009), environmental planning (Nazari et al., 2015), machine learning (Park and Klabjan, 2016), and location analysis (Francis et al., 2008), to name but a few.

A column aggregation approach replaces a set of variables with a variable as the weighted summation of the original variables. A row aggregation approach does the same with a set of constraints. Moreover, these two approaches can be applied simultaneously. An aggregated model, of course, includes less details about the problem and hence, its feasible region is generally larger than the original model. The variables and constraints of the aggregated model are usually called the aggregate variables and the aggregate constraints. For instance, consider two constraints $x_1 \quad u_1$ and $x_2 \quad u_2$ for the non-negative, continuous variables x_1 and x_2. A row aggregation approach may aggregate these two constraints into one aggregate constraint as $x_1 + x_2 \quad u_1 + u_2$. The feasible region of the

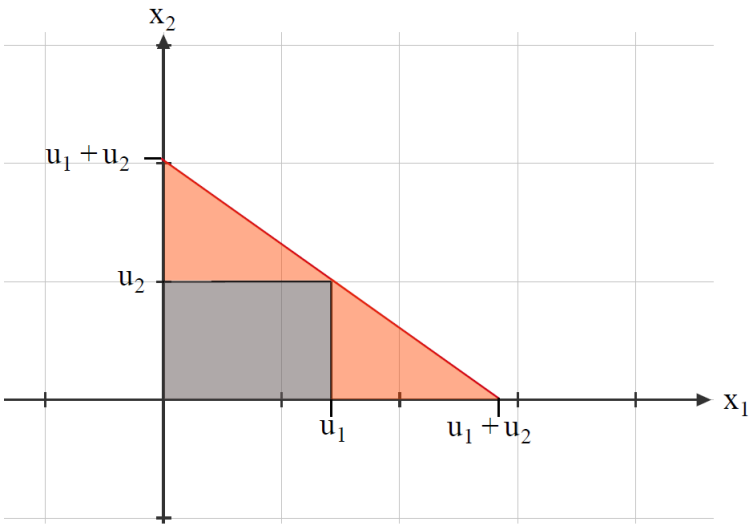


Figure 2.1: An example of the row (constraint) aggregation. Here, the constraints $x_1 \quad u_1$ and $x_2 \quad u_2$ are aggregated to $x_1 + x_2 \quad u_1 + u_2$. While this has reduced the number of constraints, it has increased the feasible region as shown.

original two constraints is depicted by the gray region in Figure 2.1. As this figure shows, the feasible region described by the aggregate constraint includes two extra regions shown by the red color. If the model in hand is a MIP model, the aggregation may only weaken the LP relaxation and not increase the integer solution space. If the augmented polyhedron of the LP relaxation of an aggregated MIP model does not include new integer points, the feasible solution space of the aggregated MIP model remains the same as the original model and only its LP relaxation is weakened. This is the case for some problems such as the multicommodity fixed-charge network design problem. Therefore, in such cases, we can have alternative MIP formulations with the same optimal integer solution but different LP relaxation tightness and computing times, where these two features are generally confronting.

As shown, aggregation methods offer a trade-off between the mathematical model size and the level of detail included (which affects the size of the feasible region). An aggregated MIP may have more feasible solutions than its disaggregated version, and the same holds for their LP relaxations, which leads in general to a weaker LP bound. On the other hand, size reduction translates to shorter computing times and possibly reduced degeneracy (Elhallaoui et al., 2008). The trade-off between quality of the bound and the time it takes to compute it is not always beneficial to a MIP algorithm such as B&B. Indeed, aggregation can allow nodes to be computed faster, but the weakening of the bound can lead to a much larger B&B tree, overall requiring more compute time.

In Chapter 4, we propose novel aggregation approaches for multicommodity network flow problems that provide much more control over this trade-off. We will show that our new aggregation approaches can significantly reduce the model size while only minimally degrading the quality of the LP relaxation, which benefits MIP algorithms to solve large instances.

We now formally define the general aggregation approaches for arbitrary optimization models. Consider the model (2.3) with m constraints and vector of n decision variables represented by x and the parameters represented by $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, and $A \in \mathbb{R}^{m \times n}$.

$$\begin{aligned} &\text{minimize} && c^T x && (2.3a)\\ &\text{s.t.:} && Ax \geq b && (2.3b)\\ &&& x \geq 0 && (2.3c) \end{aligned}$$

A variable aggregation scheme aggregates some variables and reduce the decision variables vector $x \in \mathbb{R}^n$ to the aggregate decision variables vector $\tilde{x} \in \mathbb{R}^{\tilde{n}}$. An aggregate decision variable is in general a weighted summation of some of the decision variables of the original model. For the k^{th} aggregate variable, we can define the vector v^k with n elements, each determines the weight of corresponding decision variable of the original model in the k^{th} aggregate variable. Matrix $V \in \mathbb{R}^{n \times \tilde{n}}$, where its k^{th} column is v^k, represents the transformation by the corresponding aggregation scheme for all variables. Therefore, the aggregated model is as the model (2.4).

$$\begin{aligned} &\text{minimize} && c^T V \tilde{x} && (2.4a)\\ &\text{s.t.:} && AV\tilde{x} \geq b && (2.4b)\\ &&& \tilde{x} \geq 0 && (2.4c) \end{aligned}$$

The goal of a constraint aggregation scheme is to aggregate m constraints of the original model 2.3 into $\tilde{m}$ aggregate constraints. Each aggregate constraint is a weighted summation of some of the constraints of the original model. Similar to variable aggregations, we define the matrix $R \in \mathbb{R}^{\tilde{m} \times m}$, where its l^{th} row is the row vector r_l, to represent the corresponding transformation of a constraint aggregation scheme. The row vector r_l has m elements, each determines the weight of the corresponding constraint from the original model in the l^{th} aggregate constraint. The aggregated model by such an aggregation scheme is as the model (2.5).

$$\begin{aligned}
&\text{minimize} && c^T x && && (2.5a)\\
&\text{s.t.:} && RAx \quad Rb && && (2.5b)\\
& && x \quad 0 && && (2.5c)
\end{aligned}$$

In some cases, it may be sensible to apply an extra constraint (variable) aggregation scheme after aggregating the model by a variable (constraint) aggregation scheme. For instance, consider example of Figure 2.1. Suppose that first, variables x_1 and x_2 are aggregated to the variable $\tilde{x} = x_1 + x_2$. In this case, the constraints $x_1 \quad u_1$ and $x_2 \quad u_2$ must be naturally aggregated to the constraint $x_1 + x_2 = \tilde{x} \quad u_1 + u_2$. The same happens if we first aggregate the constraints as we naturally have to aggregate variables x_1 and x_2 based on the aggregate constraint $x_1 + x_2 \quad u_1 + u_2$. A simultaneous variable and constraint aggregation scheme is defined by two transformation matrices V and R, and the aggregated model is the model (2.6).

$$\begin{aligned}
&\text{minimize} && c^T V\tilde{x} && && (2.6a)\\
&\text{s.t.:} && RAV\tilde{x} \quad Rb && && (2.6b)\\
& && \tilde{x} \quad 0 && && (2.6c)
\end{aligned}$$

The choice of transformation matrices, V and R, significantly affect the polyhedron of the aggregated model and subsequently, the effectiveness of the corresponding aggregation scheme. The aggregation techniques we propose in Chapter 4 aggregate the variables and constraints of the model simultaneously. The aggregate variables are the summation of corresponding original variables. Hence, the V matrix entries are either 0 or 1. The constraints of the same type for all aggregated variables are also summed up. Therefore, the R matrix entries are also either 0 and 1. These transformation matrices are selected based on the special structure of the multicommodity network flow problems and are shown to be effective in reducing the model size with a small expense in degradation of the model tightness. Moreover, such transformations ensure that the aggregated model is a lower bound for the original model if each variable and each constraints is present in at least on aggregate variable and one aggregate constraint, respectively.

Aggregation/disaggregation based algorithms

While aggregation techniques are useful to reduce the size of the model and obtain approximate solutions, in some cases they fail to provide exact solutions to the original problem. This is mainly because the feasible space of the aggregated problem is in general larger than the feasible space of the original problem as explained, which means the optimal solution of an aggregated model may be infeasible for its original model.

Aggregation/disaggregation based algorithms address this issue to obtain exact or heuristic solutions for the original problem using an aggregated problem. Such algorithms start from an aggregated problem and iteratively disaggregate the problem until the optimality is proved. The optimality is proved whenever the optimal solution of the aggregated model is feasible for the original model as the aggregated model is a lower bound/relaxation of the original model. In the worst case, the optimality is not proved until the problem is fully disaggregated. However, an efficient aggregation/disaggregation based algorithm usually finds the optimal solution of the original problem in a few iterations (Boland et al., 2009). The general framework of such algorithms consists of two main steps that are repeated until a proof of the optimality is obtained (Litvinchev and Tsurkov, 2003):

1. Solving the aggregated problem: this step gives an approximate solution that could be considered as a lower bound for the original problem and also could be used to generate a feasible solution (upper bound) for the original problem.
2. Re-aggregating the problem: based on the information obtained from the first step, this step may include re-aggregating of some variables or constraints or disaggregating some of the original variables and constraints. This translates to updating the transformation matrices V and R in the aggregated optimization model.

This framework involves solving a sequence of aggregated problems which their total computing time to obtain the optimal solution of the original model is usually less than the time of solving the original problem once. In general, aggregation/disaggregation based algorithms, such as the algorithms by Evans (1983), Boland et al. (2009), Nazari et al. (2015), and Francis et al. (2008), are not special cases of a general framework. In fact, the development of an aggregation/disaggregation based algorithm requires the design and implementation of problem-specific aggregation and disaggregation schemes. In Chapter 5, we apply the concept of partial aggregations introduced in Chapter 4 to an abstracted version of the problem introduced in Chapter 3. When such aggregation schemes are applied to this problem, the solution of the aggregated problem is not necessarily feasible for the original problem. Therefore, we propose a solution algorithm that utilizes the partial aggregations. The algorithm starts from a fully-aggregated model and constructs partially-aggregated models as it progresses to obtain the optimal solution of the original problem.

The computational experiments demonstrate the efficiency of the proposed algorithm in both finding good-quality upper and lower bounds in short time and proving the optimality in comparison with solving the original disaggregated problem in a single iteration.

2.2.2 Column Generation

Recall that the Simplex algorithm traverses the extreme points of the polyhedron of an LP problem to obtain the optimal solution. Each extreme point corresponds to a feasible solution which divides the variables into *basic* and *non-basic* variables. The set of basic variables is called the *basis*. Basic variables can take any value within their domain, while nonbasic variables are at their bounds. Traversing from an extreme point to another correspond to changing the set of basic variables and their values. At each iteration of Simplex, one basic variable exits from the basis and one nonbasic variable enters into it. The entering variable is a variable with the negative *reduced cost*. The reduced cost of a variable measures the change of objective function for one unit increase in the value of the corresponding variable. In practice, usually most of the variables never enter the basis (Bertsimas and Tsitsiklis, 1997). This inspires the idea of the *column generation* methods. The development of column generation algorithms dates back to the seminal works by Ford and Fulkerson (1958), Gilmore and Gomory (1961), and Appelgren (1969). Briefly, a column generation method starts with a subset of variables and add other variables only when they are required to enter into the basis.

A typical column generation algorithm decomposes the problem into a (restricted) master problem and a subproblem (Desrosiers and Lübbecke, 2005). The master problem is the same as the original model but with fewer variables. The subproblem, which is called the *pricing* problem, finds an entering variable. The column generation algorithm is similar to the Simplex algorithm, except that it finds the entering variable by an optimization problem instead of the enumeration that Simplex does. Consider the LP model (2.3). The master problem of a column generation for this model includes a subset of variables represented by the vector $\bar{x}$ and is as the model (2.7), where $\bar{c}$ and $\bar{A}$ show the parts of the original c and A that are related to variables $\bar{x}$. Moreover, dual variables of Constraint (2.4b) are represented by the p vector.

$$\begin{aligned}
&\text{minimize} && \bar{c}^T x && && (2.7a)\\
&\text{s.t.:} && \bar{A}\bar{x} \quad b && (p) && (2.7b)\\
& && \bar{x} \quad 0 && && (2.7c)
\end{aligned}$$

The pricing problem defined by the optimization problem (2.8) determines the entering variable. The coefficient of the corresponding variable of the column a in the objective function of the original problem is shown by $c(a)$. An entering variable exists only if $\bar{c} < 0$. Otherwise, the optimal solution of the restricted master problem is the same as the original model, and the algorithm terminates. The pricing problem searches among all columns $a \in A$ and adds the corresponding variable (to update $\bar{x}$) and column (to update $\bar{c}$ and $\bar{A}$) to the restricted master problem if it has a negative reduced cost.

$$\bar{c} := \min\{c(a) - p^T a \mid a \in A\} \tag{2.8}$$

The success of a column generation algorithm depends on the efficiency of the algorithm that solves the optimization problem (2.8). Such algorithms must solve the pricing problem without computing the reduced cost ($c(a) - p^T a$) for all potential columns $a \in A$. In fact, column generation is only effectively applicable to the cases that there is an efficient algorithm to solve the pricing problem due to the special structure of the problem, such as the cutting stock problem (Gilmore and Gomory, 1961).

The Dantzig-Wolfe decomposition

The Dantzig-Wolfe decomposition (DW) is a special case of the column generation algorithm that applies to LP problems with a block diagonal structure (Dantzig and Wolfe, 1960). A block diagonal LP includes k groups of variables. Variables within in each group are related together by a linear system separately from the variables in other groups. All variables are linked together by a set of linking constraints. The general form of such a model is provided as the model 2.9, where $x_i, c_i \in \mathbb{R}^{n_i}$, $A_{ij} \in \mathbb{R}^{m_i \times n_j}$, and $b_i \in \mathbb{R}^{m_i}$. These imply that the model (2.9) includes k groups of variables, where the i^{th} group has n_i variables that are related together by m_i constraints of the form of Constraint (2.9c), and there are m_0 linking constraints of the form Constraint (2.9b) that link all variables.

$$\text{minimize} \quad \sum_{i \in \{1,\ldots,k\}} c_i^T x_i \tag{2.9a}$$

$$\text{s.t.:} \quad \sum_{i \in \{1,\ldots,k\}} A_{0i} x_i = b_0 \tag{2.9b}$$

$$A_{ii} x_i = b_i \qquad \forall i \in \{1,\ldots,k\} \tag{2.9c}$$

$$x_i \geq 0 \qquad \forall i \in \{1,\ldots,k\} \tag{2.9d}$$

The DW method reformulates the model (2.9) based on Minkowski's theorem, which states that the polyhedron of any LP can be represented by a linear combination of its extreme points and extreme rays. Suppose that the feasible region of $Q_i = \{x_i \in \mathbb{R}^n_i : A_{ii}x_i = b_i, x_i \geq 0\}$ is bounded and has N_i extreme points as $\{v_{i1}, \ldots, v_{iN_i}\}$. Then, this feasible region can be represented by its extreme points as $Q_i = \{x_i = \sum_{j \in \{1,\ldots,N_i\}} l_{ij}v_{ij} : \sum_{j \in \{1,\ldots,N_i\}} l_{ij} = 1 \;\&\; l_{ij} \geq 0\}$. Therefore, the model (2.9) is reformulated as the model (2.10).

$$\text{minimize} \quad \sum_{i \in \{1,\ldots,k\}} \sum_{j \in \{1,\ldots,N_i\}} l_{ij}c_i v_{ij} \tag{2.10a}$$

$$\text{s.t.:} \quad \sum_{i \in \{1,\ldots,k\}} \sum_{j \in \{1,\ldots,N_i\}} l_{ij}A_{0i}v_{ij} = b_0 \tag{2.10b}$$

$$\sum_{j \in \{1,\ldots,N_i\}} l_{ij} = 1 \qquad \forall i \in \{1,\ldots,k\} \tag{2.10c}$$

$$l_{ij} \geq 0 \qquad \forall i \in \{1,\ldots,k\}, j \in \{1,\ldots,N_i\} \tag{2.10d}$$

The reformulated model (2.10) has significantly fewer constraints ($m_0 + k$) in comparison with the original model (2.9). However, it includes a huge number of variables as it includes extreme points v_{ij}. The DW method applies a column generation algorithm to the reformulated model to solve the original problem. In this algorithm, there is one master problem and k subproblems. As the focus of this thesis is not on the DW method we refer the readers to Dantzig and Wolfe (1960) and Lübbecke and Desrosiers (2005) for a detailed description.

The Branch-and-Price algorithm

The column generation algorithms can be embedded in a B&B algorithm to solve integer and MIP models. This generalizes the B&B method as it does not necessarily generate all columns at the root nodes. Such generalized B&B algorithms are usually called Branch-and-Price algorithms. The development of a Branch-and-Price algorithm requires sophisticated branching and search strategies to ensure that no subspace of the solution is left out. More information on Branch-and-Price algorithms can be found in Barnhart et al. (1998). Moreover, Vanderbeck (2000) discuses the extension of the DW method to the integer problems within the Branch-and-Price algorithms.

2.2.3 Benders Decomposition Algorithm

The Benders decomposition algorithm (Benders, 1962) is a well-known algorithm to solve large-scale optimization problems with so-called "complicating" variables. This algorithm generally decomposes the original problem into a master (that usually includes the complicating variables) and a subproblem, each can be solved much easier than the original problem. Then, these two decomposed problems are iteratively solved. The master problem is solved to obtain solutions. Then, the obtained solutions are evaluated by solving the subproblem and the information is fed back to the master problem in the form of cuts. The process is iterated until the master problem with the added cuts becomes equivalent to the original problem.

Consider the MIP model (2.11) in which y variables are the complicating variables.

$$\begin{aligned} &\textit{minimize} && c^T x + f^T y && \text{(2.11a)}\\ &s.t: && Ax + By = b && \text{(2.11b)}\\ & && x \quad 0 && \text{(2.11c)}\\ & && y \; 2 \; Y && \text{(2.11d)} \end{aligned}$$

One may decompose the problem (2.11) into a master problem that includes y variables and a subproblem that includes x variables. In this case, the MIP model (2.11) becomes equivalent to the optimization problem (2.12). Here, the subproblem is the inner LP problem in which y variables are fixed and is solved for x variables.

$$\min_{\bar{y}2Y} f f^T \bar{y} + \min_{x \quad 0} f c^T x : Ax = b \quad B\bar{y} g g \tag{2.12}$$

Let p be the dual variables for the constraint set $Ax = b \quad B\bar{y}$ of the inner LP problem of (2.12). Then, the LP subproblem can be dualized as $\max_p f p^T (b \quad B\bar{y}) : p^T A \quad c g$, and the MIP model (2.11) becomes equivalent to the optimization problem (2.13).

$$\min_{\bar{y}2Y} f f^T \bar{y} + \max_p f p^T (b \quad B\bar{y}) : p^T A \quad c g g \tag{2.13}$$

Independent from the $\bar{y}$ values, the inner optimization problem of (2.13) includes a set of extreme points E and a set of extreme rays Q. For an extreme ray $r_q \; 2 \; Q$, there may exist

a fixed solution $\bar{y}$ such that $r_q^T(b \quad B\bar{y}) > 0$. This makes the program $\max_p \{p^T(b \quad B\bar{y}) : p^T A \quad c\}$ unbounded and the primal subproblem infeasible. To avoid such infeasibilities, we can add cuts in the form of $r_q^T(b \quad By) \quad 0$ to the master problem for all extreme rays $r_q \in Q$. Moreover, among all extreme points $e \in E$, we are looking for the extreme point that maximizes $p_e^T(b \quad By)$. Having all extreme points and rays of the dual of the subproblem, we can add all the necessary cuts into the master problem as (2.14) to obtain an equivalent problem to the optimization model (2.11).

$$\begin{aligned}
&\textit{minimize} && f^T y + h && && (2.14a)\\
&s.t: && p_e^T(b \quad By) \quad h && \forall e \in E && (2.14b)\\
&&& r_q^T(b \quad By) \quad 0 && \forall q \in Q && (2.14c)\\
&&& y \in Y && && (2.14d)
\end{aligned}$$

Constraint sets (2.14b) and (2.14c) are usually called *optimality* and *feasibility* cuts, respectively. In general, there exists an exponential number of the optimality and feasibility cuts in the input size, which makes the MIP model (2.14) impractical for use. However, the Benders algorithm suggests an iterative framework that adds a subset of all feasibility and optimality cuts to obtain an equivalent model. This algorithm starts and solves a relaxed version of the model (2.14) with no optimality and feasibility cuts (i.e. (2.14a) s.t: (2.14d)) to obtain a solution $\bar{y}$. Then, this solution is used to solve the dual of the subproblem (i.e. $\max_p \{p^T(b \quad B\bar{y}) : p^T A \quad c\}$). In the case that the subproblem is unbounded, a feasibility cut can be added to the relaxed master model. Otherwise, an optimality cut can be added to the relaxed master model. The relaxed master problem gives an lower bound (LB) for the optimal solution. The objective of the subproblem plus the cost associated with the fixed solution gives an upper bound ($UB = p^T(b \quad B\bar{y}) + f^T\bar{y}$) for the optimal solution. Since there are finite number of the feasibility and optimality cuts, the algorithm converges with a termination condition such as $UB \quad LB < e$. Moreover, if Constraint (2.14d) implies integrality requirements, the Benders algorithm can be applied during the progress of the B&B algorithm. In the simplest form, it is enough to solve the subproblem to generate a cut whenever an integer feasible is found in the B&B tree. This algorithm is referred as Branch-and-Benders-Cut algorithm and has been widely employed for large-scale integer optimization problems (Gendron et al., 2016; Errico et al., 2017; Moreno et al., 2019).

Since its introduction, several methods have been proposed to accelerate and improve the Benders decomposition algorithm performance. Such methods mostly aim to improve the problem decomposition, the solution methods for the master and subproblems, heuris-

tics for obtaining better upper bounds, and the cut generation. See Rahmaniani et al. (2017) for more details. One effective acceleration technique is the Magnanti-Wong method (Magnanti and Wong, 1981) to generate Pareto-optimal cuts. They define the Pareto-optimal Benders cut as a cut that is not dominated by any other Benders cut. The cut corresponding to the vector p_1 dominates the corresponding cut of the vector p_0 if $p_1^T(b \quad By) \quad p_0^T(b \quad By)$ for all $y \ 2 \ Y$ with at least one strict inequality for a point $y \ 2 \ Y$. To generate Pareto-optimal cuts, they solve the subproblem twice at each iteration. First, they solve the dual of the subproblem as $Q \ = \max_p fp^T(b \quad B\bar{y}) : p^T A \quad cg$. Afterwards, an auxiliary subproblem as $\max_p fp^T(b \quad B\hat{y}) : p^T A \quad c, p^T(b \quad B\bar{y}) \ = \ Q \ g$ is solved to generate a Pareto-optimal cut. In the auxiliary subproblem, $\hat{y}$ is a *core point*, which is actually an interior point of the feasible space of the original problem. Because large LP problems usually suffer from the degeneracy, there are many cuts at each iteration of the Benders algorithm to choose from. The Magnanti-Wong method overcomes this issue by selecting stronger cuts and has became a standard method to include in Benders decomposition implementations (Jeihoonian et al., 2016; Fontaine and Minner, 2018; Mokhtar et al., 2019).

2.3 Multicommodity Network Flow Problems (MCF)

Multicommodity network flow problems seek to route a set of commodities K between an origin/source o^k and a destination/sink s^k for each $k \ 2 \ K$ over a network/graph $G \ = \ (N, A)$, usually with the goal of routing cost minimization. MCF problems apply to and are a useful modeling tool for a wide variety of problems in transportation, logistics, and communication networks (Assad, 1978). A commodity might be a group of passengers that must travel between two cities (Hane et al., 1995), a container that must be moved between two facilities (Crainic, 2000), or a data packet that must be transshipped in a communication network (Schwartz and Stern, 1980). To address various cost structures in different applications, multicommodity network flow problems have been employed together with linear objective functions, piecewise-linear objective functions (Croxton et al., 2007), network design problems with economies of scale (Andrews et al., 2010), network loading problem (Avella et al., 2007), and network design with nodes costs (Belotti et al., 2007). Moreover, there are applications that can be modelled as multicommodity time-space network although they may not exhibit an explicit network structure (Kliewer et al., 2006; Kazemi et al., 2021a), which extends the applicably scope of these types of problems.

The multicommodity network flow problem is a fundamental problem in network optimization, and the MCF structure is apparent in many practical and classical problems (Ahuja et al., 2014). Transportation problems naturally appear as a MCF problem since they occur on physical networks and involve a set of distinct commodities, passengers, or vehi-

cles. For instance, Desrosiers et al. (1995) and Desaulniers et al. (1998) propose classes of well-known MCF-based formulations for some vehicle routing and crew scheduling problems. A wide range of problems can be expressed on hypothetical networks, such as data communication problems (Assad, 1978). Furthermore, a variety of scheduling problems can be represented by a time-expanded network and hence, be modeled with a MCF substructure. See Árton P. Dorneles et al. (2017) and Kulkarni et al. (2018) as examples of MCF-based formulations for scheduling problems. The first investigates a school timetabling problem, and the latter studies a car scheduling problem. The MCF structure is often embedded in some other classical problems such as the network design problem (Magnanti and Wong, 1984), the hub location problem (Ernst and Krishnamoorthy, 1996), the Steiner tree problem (Wong, 1984), and the unsplittable flow problem (Barnhart et al., 2000), to name but a few.

A MCF problem becomes complicated when the commodities are interrelated, usually by bundling constraints such as capacity constraints and/or network design decisions. In such cases, the number of possible solutions grows exponentially with the increase in the number of commodities since all commodities are interrelated. In the context of MIP methods, when solving hard multicommodity network flow problems using an LP-based approach, the number of commodities is a driving factor in the speed at which the LP relaxation can be solved, as it is linear in the number of constraints and variables.

Various large scale optimization techniques introduced in Section 2.2 have been applied to different variants of the MCF problem. MCF problems are particularly attractive for such techniques because of the network substructure of the problem. Therefore, these techniques can leverage the efficiency of network optimization algorithms such as Network Simplex (Orlin, 1997), polynomial shortest path algorithms (Gallo and Pallottino, 1988), and the Ford–Fulkerson algorithm (for the maximum flow problem) (Ford and Fulkerson, 1956; Edmonds and Karp, 1972) to solve the resulting subproblems with special structures.

Ford and Fulkerson (1958) propose their idea for column generation method originally for the maximal multicommodity network flow problem, which maximizes the total flow of a set of commodities in a capacitated directed network. Barnhart et al. (1994) develop a tailored column generation algorithm for the MCF problem, which is particularly efficient over instances with many commodities. Barnhart et al. (2000) propose a Branch-and-Price-and-Cut algorithms for a MCF problem, in which flow of commodity may use a single path. Mamer and McBride (2000) develop a column generation algorithm with a special pricing procedure that solves a relaxed version of the original subproblem. To evaluate their proposed algorithm, they apply it to the MCF problem.

The MCF problem has been also attractive for the development and extension of various Benders decomposition algorithms in both practical and theoretical aspects. Geoffrion and Graves (1974) develop a Benders decomposition algorithm for the capacitated MCF

problem and successfully apply it to a real-life distribution problem. Magnanti and Wong (1981) apply their acceleration technique for the Benders algorithm to the MCF problem to show its efficiency. The application of the Benders decomposition algorithm to the MCF problem range widely from practical problems (see Sweeney and Tatham (1976), Oğuz et al. (2018), and Pishvaee et al. (2014) for a few examples) to classical variants such as the multicommodity fixed-charge network design (Costa, 2005).

This thesis focuses on and contributes to the aggregation methods for the MCF problems. The conventional aggregation approach to improve the solve time of the LP relaxation of a MIP model that encodes an instance of MCF is to aggregate all commodities that have the same origin or the same destination (Chouman et al., 2017). However, the bound of the resulting LP relaxation can significantly worsen, which tempers the efficiency of aggregating techniques. Commodity aggregation has been also applied to variants of MCF. Belieres et al. (2020) aggregate commodities to pass partial information to the master problem of their Benders decomposition algorithm for a service network design problem. Belieres et al. (2021) improve this algorithm by allowing the set of aggregated commodities to change during the progress of the Benders decomposition algorithm.

In Chapter 4, we introduce the concept of *partial aggregation* of commodities that aggregates commodities over a subset of the network instead of the conventional aggregation over the entire underlying network. This offers a high level of control on the trade-off between size of the aggregated MIP model and quality of its LP bound. Partial aggregations that we introduce are fundamentally different from those presented by Belieres et al. (2021). Our proposed partial aggregations change over the underlying network. On the other hand, their aggregation schemes consider a fixed set of commodities over the network, and it is only the partition of commodities into the subsets that may change during iterations of the algorithm.

2.3.1 Multicommodity Fixed-Charge Network Design Problem

To show the effectiveness of the proposed aggregation approaches, we apply these to the Multicommodity Capacitated fixed-charge Network Design problem (*MCND*). However, these approaches are not restricted to MCND and are applicable to any other problem that can be formulated with a multicommodity network flow subproblem in its variables and constraints. We choose MCND for three reasons. First, this problem is easy to understand, but difficult to solve. Second, MCND is well studied, has extensive applications (Magnanti and Wong, 1984), and is the basis for many other network design problems. Third, there are well-established benchmark instances for MCND in the literature (Crainic et al., 2001). Although, our aim is not to improve on the state-of-the-art for solving MCND, if that was

an outcome of the work we carry out, it would indeed be beneficial. What we are looking for, instead, is a generalized framework for solving problems that can be formulated with a multicommodity network flow subproblem in its variables and constraints, of which the MCND is a significant archetype.

The input of MCND is a directed graph $G = (N, A)$ and a set of commodities K. Each commodity $k \in K$ corresponds to a flow of d^k units that must be routed from an origin $o^k \in N$ to a destination $s^k \in N$. For each arc $(i,j) \in A$, a capacity u_{ij}, a per-unit-of-flow cost c_{ij} and a fixed cost f_{ij} are defined, all non-negative. The fixed cost is incurred if an arc is used/opened for routing flow.

It is well-known in the MCND literature that all commodities with the same origin (with the same destination) can be aggregated as one commodity with an origin and multiple destinations (with multiple origins and a destination) (Chouman et al., 2017). This leads to two formulations with the same optimal integer solution but with different LP bound qualities: (a) a disaggregated formulation (DA) that considers commodities based on origin-destination pairs; (b) a fully-aggregated formulation (FA) that aggregates all commodities with the same origin into a single commodity with that origin and multiple destinations, one per aggregated commodity. The same aggregation process can be used to instead aggregate commodities that have the same destination, but, without loss of generality, we only aggregate by origins in this thesis.

We employ the most common MIP model for MCND in the literature (Gendron et al., 1999). This model includes a continuous flow variable x_{ij}^k that corresponds to the amount of flow of commodity k on the arc (i,j), and a binary variable y_{ij} that determines if the arc (i,j) is opened. The binary input value o_i^k (s_i^k) is 1 if and only if the node $i \in N$ is the origin (destination) of the commodity k, and we partition the set of neighbors of a node i into the set of successor nodes $N_i^+ = \{j \in N \mid (i,j) \in A\}$, and the set of predecessor nodes $N_i^- = \{j \in N \mid (j,i) \in A\}$. The MCND MIP model is as follows:

Disaggregated Formulation (DA):

$$\text{minimize} \quad \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ij}^k + \sum_{(i,j) \in A} f_{ij} y_{ij} \tag{2.15a}$$

$$\text{s.t:} \quad \sum_{j \in N_i^+} x_{ij}^k - \sum_{j \in N_i^-} x_{ji}^k = (o_i^k - s_i^k) d^k \qquad \forall k \in K,\ i \in N \tag{2.15b}$$

$$\sum_{k \in K} x_{ij}^k \leq u_{ij} y_{ij} \qquad \forall (i,j) \in A \tag{2.15c}$$

$$x_{ij}^k \leq d^k y_{ij} \qquad \forall k \in K,\ (i,j) \in A \tag{2.15d}$$

$$x_{ij}^k \geq 0 \qquad \forall k \in K,\ (i,j) \in A \tag{2.15e}$$

$$y_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{2.15f}$$

Objective function (2.15a) minimizes the total cost, including the flow and fixed costs. Equation (2.15b) is the flow conservation constraint. Constraint (2.15c) enforces the capacity limits of arcs. Constraint (2.15d) links the binary and continuous decision variables in a disaggregated form of Constraint (2.15c). Constraint (2.15d) is not necessary for the validity of the MIP, however, it significantly improves the tightness of the LP relaxation bound (Gendron et al., 1999). This constraint is called a *Strong Inequality* (SI) in the literature (Chouman et al., 2017) and we adopt the same name throughout. The decision variables and their domains are defined in Constraints (2.15e)-(2.15f). This model corresponds to the DA formulation.

The full aggregation approach aggregates all commodities with a same origin. Suppose that the set $\tilde{N}$ includes the nodes $n \in N$ from which at least one commodity $k \in K$ originates. For each $n \in \tilde{N}$, the group of commodities K_n are aggregated together, where $K_n = \{k \in K | o_n^k = 1\}$. The FA formulation, based on the such aggregation approach, is developed on the DA formulation by: (a) summing up the flow conservation constraints (2.15b) over $k \in K_n$ for all $n \in \tilde{N}$, (b) summing up SIs (2.15d) over over $k \in K_n$ for all $n \in \tilde{N}$ and $(i,j) \in A$, and (c) and replacing the summation of decision variables $\sum_{k \in K_n} x_{ij}^k$ with a single aggregate variable $x_{ij}^{K_n}$ for all $n \in \tilde{N}$ and $(i,j) \in A$. The resulting formulation generally includes less variables and constraints and is as the model 2.16.

Fully-Aggregated Formulation (FA):

$$\text{minimize} \quad \sum_{n \in \tilde{N}} \sum_{(i,j) \in A} c_{ij} x_{ij}^{K_n} + \sum_{(i,j) \in A} f_{ij} y_{ij} \tag{2.16a}$$

$$\text{s.t:} \quad \sum_{j \in N_i^+} x_{ij}^{K_n} - \sum_{j \in N_i} x_{ji}^{K_n} = \sum_{k \in K_n} (o_i^k - s_i^k) d^k \qquad \forall n \in \tilde{N}, i \in N \tag{2.16b}$$

$$\sum_{n \in \tilde{N}} x_{ij}^{K_n} \leq u_{ij} y_{ij} \qquad \forall (i,j) \in A \tag{2.16c}$$

$$x_{ij}^{K_n} \leq \sum_{k \in K_n} d^k y_{ij} \qquad \forall n \in \tilde{N}, (i,j) \in A \tag{2.16d}$$

$$x_{ij}^{K_n} \geq 0 \qquad \forall n \in \tilde{N}, (i,j) \in A \tag{2.16e}$$

$$y_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{2.16f}$$

Gendron et al. (1999) showed that, in practice, the LP relaxation of DA provides tight bounds for MCND. On the other hand, FA is a smaller formulation with significantly shorter LP relaxation computing time, but looser bounds. The DA formulation has re-

ceived more attention particularly to develop MIP-based heuristics (Crainic et al., 2004; Rodríguez-Martín and José Salazar-González, 2010; Gendron et al., 2018). The FA formulation has been preferred in cutting plane algorithms by Bienstock and Günlük (1995, 1996), Bienstock et al. (1998) and Raack et al. (2011) because of the shorter computing time of its LP relaxation. Rardin and Wolsey (1993) studied the relationship between the aggregated and the disaggregated commodity representation for the uncapacitated fixed-charge network design with one origin and multiple destinations. The aggregated version of this problem includes one commodity while its disaggregated version includes one commodity per destination with nonzero demand. The authors derive an improved formulation by adding a family of so-called "dicut" inequalities to the FA formulation that has the same LP relaxation polyhedron as the DA formulation for this problem. These inequalities are specific to the problem they considered. Chouman et al. (2017) are the first who evaluated the effect of commodity representation, either disaggregated or fully-aggregated, on the performance of a Branch-and-Bound algorithm (B&B) with a specialized cutting plane algorithm. They showed that in general, the DA formulation outperforms the FA formulation over the instances with many commodities, but the FA formulation is more effective for instances with few commodities.

The model selection in the literature follows an "all-or-nothing" approach that employs either the DA or the FA formulation based on the application and the algorithm considered. This approach sacrifices one aspect, LP relaxation computational difficulty or LP relaxation bound quality, for the other aspect. In this thesis, we propose a spectrum of formulations that lie in within the range of the DA and FA formulations in this trade-off. This allows to select a formulation with the desired level of compromise between the model computational difficulty and the LP bound tightness. These formulations are introduced in Chapter 4 and are based on *partial* aggregations of commodities instead of a disaggregated or a conventional fully aggregated approach.

2.4 Fuel Management Problems

Fuel management problem in its simplest form is to determine the refueling plan of a vehicle. Such vehicle visits a set of fixed (fuel) stations during its route, and each trip leg between those stations requires a fixed fuel consumption. A refueling plan determines where a vehicle must refuel and how much fuel it must receive at each refueling operation.

Two types of costs are usually involved in a refueling operation: first, the fuel price which varies across different stations, and second, a fixed cost for each refueling operation, usually referred as *stop cost*. The general approach to reduce the fuel management costs, called *tankering* strategy, is to refuel the vehicle at inexpensive stations as much as possible

to avoid the need for refueling at the expensive stations. Therefore, an optimal refueling plan reduces the number of refueling operations as well as the refueling at expensive stations.

The simple version of the fuel management problem may be extended in different ways, including:

- **Stochastic fuel prices:** In cases where the planning horizon is long, it is sensible to incorporate uncertainty in the fuel prices.
- **Stochastic fuel consumption:** Another uncertain aspect that may impact the refueling plans is the fuel consumption that can affect both optimality and feasibility of a refueling plan.
- **Speed optimization:** Since the relation between speed and fuel consumption is usually non-linear, the simultaneous optimization of the speed and refueling plan of a vehicle results in more accurate solutions. The impact of the speed optimization is particularly significant in sea transportation.
- **Weight considerations:** Weight of the fuel that a vehicle carries can affect the refueling plan in two aspects. First, it may affect the fuel consumption of the vehicle. Second, high amounts of the carrying fuel can cause issues for the movement of the vehicle, in the aviation industry for instance.
- **Fuel management of a fleet of vehicles:** In case of planning a fleet of vehicles, there are usually features that interrelate the refueling plans of the vehicles, which makes the problem complicated. We next describe such features in details.

In the context of fuel management, linking features require the simultaneous optimization of refueling plans of all vehicles to avoid sub-optimal solutions. Considering the fuel planning of a vehicle as a special network flow problem and a vehicle as a commodity, the fuel management of a fleet of vehicles could be viewed as special case of the multicommodity network flow problem. Linking feature are typically considerations regarding:

- **Locating fuel stations:** management of a fleet of vehicles sometimes involves procurement of fuel at stations in advance. This could be done by contracts with fuel suppliers at the corresponding stations or by providing fueling facilities there. Both cases incur an additional fixed cost.
- **Demand assignment:** in some cases, the scheduling involves assignment of a set of costumers to a fleet of vehicles, which converts the problem into a routing problem with additional fueling considerations.
- **Volume discounts:** in some applications, such as sea transportation, it is common that fuel contracts with suppliers offer discounts in prices based on the total volume

purchased from the supplier. In such cases, simultaneous optimization of all vehicles' fuel plans is beneficial to leverage the volume discounts.

- **Fuel suppliers restrictions:** ignoring fuel suppliers restrictions may result in impractical solutions in which the planned amount of fuel purchase is higher than the supplier's capability or is lower than the supplier's minimum purchase requirement. Simultaneous optimization of the fuel plans of the fleet of the vehicles could avoid such solutions.

In rail transportation, the weight of the fuel a train carries is trivial in comparison with the total weight of the train. In addition, railway companies usually make contracts with the fuel suppliers to hedge the fuel price against the possible fluctuations. Such contracts usually incur a fixed contract fee. Therefore, among possible extensions for the fuel management problem in rail transport, locating fuel stations is prominent, and a typical locomotive fuel management involves the integrated problem of fuel planning of the locomotives and locating the fuel stations. Nourbakhsh and Ouyang (2010) are the first who studied the integrated locomotive fuel management problem. They employ a Lagrangian relaxation framework to optimize both fueling decisions and station location decisions and develop a cyclic plan for a fleet of locomotives in a transportation network. They also consider refueling capacities at refueling stations over the entire planning horizon, and *emergency refueling* as an auxiliary refueling option.

A vast majority of the locomotive fuel management literature has been motivated by the INFORMS Problem Solving Competition (2010). The problem is to simultaneously optimize the location and fuel decisions of a cyclic plan of a fleet of locomotives by considering a daily capacity for the stations. Moreover, each locomotive cannot be refueled at more than two intermediate stations during a trip between an origin-destination. This problem has been solved using integer programming with valid inequalities (Chiraphadhanakul and Figueroa, 2010; Raviv and Kaspi, 2012; Vermit, 2014; Kumar and Bierlaire, 2015) and heuristic algorithms (Ramsden, 2010; Nag and Murty, 2012; Schindl and Zufferey, 2013). Li et al. (2014) studied another version of locomotive fuel management. Considering fuzzy fuel prices to address uncertainty, the authors simultaneously optimize the speed and fuel decisions of a locomotive, since the speed of the locomotive affects the fuel consumption. However, decisions on the location of the stations are ignored. Therefore, the problem can be solved for each locomotive separately. In a different perspective, Vaidyanathan et al. (2008) studied the locomotive routing problem, in which the locomotives are assigned to a given train schedule by considering fueling and service constraints. However, fuel costs are not optimized, and they only ensure that no locomotive runs out of fuel during journeys between the stations. They developed an aggregation-disaggregation based algorithm to minimize the total cost of locomotives ownership and assignment.

The fuel management problem has also been investigated in other modes of transportation. The problem has been studied for around 40 years for the aviation industry (Darnell and Loflin, 1977). In addition to the tank capacity, the allowed weight of the aircraft during the take-off and landing restricts the amount of refueling. Moreover, the weight of the aircraft affects the fuel consumption during the flight. Considering these operational constraints, a tankering strategy, called *ferrying* in the aviation industry, has been investigated for a single aircraft (Teodorović, 1988; Philpott and Mudaliar, 1992; Zouein et al., 2002; Fregnani et al., 2013) and a fleet of aircraft. Additional considerations that make the fuel decisions of the airplanes interrelated include: (a) having volume discount strategies (Darnell and Loflin, 1977), (b) fuel suppliers' restrictions (Stroup, 1992), or (c) assignment of the airplanes to scheduled flights (Kheraie and Mahmassani, 2012). Studies on military aircraft fuel management are focused on fuel consumption optimization since they use a special type of refueling, called *aerial refueling*, which involves the transfer of fuel from an aircraft (tanker) to the aircraft that is in service (receiver) during its flight. After refueling, the tanker returns to its base station. The fuel management problem consists of finding the route with minimum consumption for both the receiver and tanker. Yamani (1986) studied the problem of fuel planning of a fleet of military aircraft. The integrated problem of routing and fueling of a single military aircraft with possibility of several aerial refueling operations is addressed by Bush (2006); Kannon et al. (2015); Ferdowsi et al. (2018). To the best of our knowledge, the only research that studies the fuel costs for military aircraft is by Hubert et al. (2015). They consider a single aircraft in which fuel can be ferried only for one subsequent trip. The remaining fuel can be offloaded and sold at the destination and aerial refueling is not available.

Fuel management for sea transportation is considered for two types of services: *Liners* with fixed routes and *Trampers* where the set of ports visited is also optimized. Besbes and Savin (2009) investigated the fuel management problem of a single vessel for both liners and trampers. Zhen et al. (2017) optimized fuel plans for a single liner under stochastic fuel price and consumption. Since the speed of ships significantly affects the fuel consumption, the integrated problem of fuel and speed decisions of a single liner vessel is studied to obtain additional savings (Yao et al., 2012; Sheng et al., 2014, 2015). In situations where there is a fleet of trampers (Vilhelmsen et al., 2014; Meng et al., 2015) or where fuel suppliers offer volume discounts (Plum et al., 2014; Wang and Meng, 2015), the fuel decisions of different vessels are interrelated. Therefore, the fuel plans of all of the vessels should be optimized simultaneously to avoid sub-optimal solutions. Zhen et al. (2017) addressed the problem of fuel management of a fleet of ships when the speed, refueling, and fleet size are optimized simultaneously with distribution-free stochastic fuel price. Reinhardt et al. (2020) investigated the speed and fuel consumption optimization for a fleet of liners incorporating

several other business constraints. De et al. (2019) studied a dynamic fuel management problem for a fleet of liners with fluctuating fuel prices and consumption. Moreover, Wang and Chen (2017) studied the integrated problem of speed and fuel decisions of a fleet of liners while restricting total carbon emission. The problem of liner fuel optimization with fixed speeds is mathematically equivalent to locomotive refueling without inline refueling.

In another application, Sundström and Binding (2010) and Hu et al. (2011) addressed optimizing recharging decisions for a fleet of Electric Vehicles (EV) while incorporating battery behavior and electricity supply constraints. Wang et al. (2016) studied the same problem as the INFORMS competition but for electronic vehicles. They considered one additional constraint which states that the amount of recharging at a station is dependent on the time that the EV stays at the station. A mathematical model proposed by Sassi and Oulamara (2017) lexicographically assigns a fleet of EVs to predetermined tours and optimizes recharging costs with electricity supply constraints. In the context of motor vehicles, Suzuki (2012) proposed a decision support system (DSS), which jointly optimizes routing and refueling decisions of a motor-carrier vehicle by considering operational constraints on the amount of fuel purchased. Suzuki et al. (2014) extends this work by incorporating the effect of the fuel's weight on the vehicle's fuel consumption.

There are also studies that focus on the location of refueling stations. In these studies, instead of considering demands as nodes, they are stated as origin-destination flows that must pass at least one supply node during their shortest path. This approach of modeling is firstly introduced by Hodgson (1990) and Berman et al. (1992) as *Flow-Capturing* models. Kuby and Lim (2005) are the first who used flow-capturing models to determine the location of the fuel stations. Given the number of stations, they maximized the number of the covered fuel demand flows while ensuring the vehicles do not run out of fuel during their journey. Using flow-capturing models, Wang and Lin (2009) and MirHassani and Ebrazi (2013) investigated the set-covering version of the fuel station location problem. Several extensions to the basic flow-capturing models of the fuel station location problem are addressed in the literature: (a) allowing a specified deviation from the shortest path for refueling (Kim and Kuby, 2012), (b) capacitated fuel stations (Upchurch et al., 2009), (c) a multi-period model (Chung and Kwon, 2015), (d) deviation-allowed flows considered together with capacitated stations (Hosseini et al., 2017), and (e) simultaneously flow routing and station locating (Yildiz et al., 2016). Anjos et al. (2020) proposed a framework for locating recharging stations of EVs considering the effects of new stations on EV adoption.

This thesis introduces a new class of fuel management problems in which a new linking feature, inline refueling, is incorporated. As explained in Chapter 1, inline refueling is operated by special supplementary fuel reservoirs, inline tanks. Incorporating the inline refueling in the fuel management offers a degree of freedom and flexibility for fuel

planning. Hence, it brings saving opportunities for the fuel planning with the expense of additional complexity. We formally define and study the fuel management problem with inline refueling considerations and develop the first optimization model for planning a fleet of inline tanks.

To the best of our knowledge, the type of inline refueling considered in our work has not yet been studied for any mode of transportation. Emergency refueling, aerial refueling, and fuel offloading, which are studied in the literature, involve a form of inline refueling but are fundamentally different. Based on Nourbakhsh and Ouyang (2010), emergency fuel price is higher than the fuel prices in the stations. However, inline refueling is performed with the same fuel prices as the stations, since the inline tanks are refueled at the stations. Moreover, the emergency fuel cannot not be used by other locomotives and is limited to the locomotive's tank capacity. In contrast, the fuel in the inline tank can be used by different locomotives by switching to different trains, and it has its own capacity in addition to the locomotives' tanks. Aerial refueling, studied for military aircraft (Yamani, 1986; Bush, 2006; Kannon et al., 2015; Ferdowsi et al., 2018), is different from inline refueling for three reasons. Firstly, the optimization is on fuel consumption, not fuel price. Secondly, it is assumed that the tanker aircraft returns to its base station. This assumption implies that the aerial refueling is the same as refueling at the stations with an extra cost for fuel supplied via the tanker aircraft. Finally, the fuel received by aerial refueling can be only used by the receiver aircraft. Hubert et al. (2015) considered fuel offloading as taking out the extra fuel from an aircraft and selling it to the station. They considered fuel ferrying only for one extra subsequent trip, while we do not restrict the number of trips for fuel tankering. Moreover, they considered a single aircraft but we optimize fuel plans for a fleet of locomotives. Most importantly, they assumed that the fuel can be sold at stations. This assumption is impossible for many cases and fuel can be transferred only if another transportation unit of our fleet can use it in future.

CHAPTER 3

Locomotive Fuel Management with Inline Refueling

3.1 Overview

This chapter investigates the fuel management problem with inline refueling in the context of rail transport. Section 3.2 defines the problem formally. Theoretical complexity of the problem and its restricted versions is proved in Section 3.3, where we show that this problem, even in its simplest form, is strongly NP-hard even if only one inline tank is available. Section 3.4 conceptualizes the problem as a time-space network. Therefore, the problem can be modelled as a special case of the multicommodity network design problem based on the time-space network conceptualization. The base model presented in Section 3.4 is extended in Section 3.5 to incorporate the operational restrictions of locomotive refueling. Section 3.6 applies the proposed model to a case study in Australia to evaluate and investigate the adoption of inline refueling on the operations of the railway companies. Section 3.7 evaluates the model for a larger dataset from the USA. As commercial solvers fail to provide good-quality solutions, we develop a heuristic algorithm in this section which provides good-quality solutions in a reasonable time for large instances.

This chapter contains three main contributions:

1. It introduces a new class of locomotive fuel management problems and analyzes their theoretical and empirical complexity.

2. It proposes a MIP model based on a time-space network representation of the problem.
3. It presents two case studies from Australia and the USA to evaluate various scenarios and also develops a heuristic for large instances.

3.2 Problem Description

We introduce the *General Fuel Management problem with Inline Refueling*, GFMIR, which seeks to minimize the total fuel-related costs when train schedules and locomotive routes are given. A train schedule is a sequence of stations that the train visits. The input data includes the set of stations S, the set of locomotives L, the time horizon, and the set of train schedules during the time horizon. We derive the individual trip legs set T from all trains schedule. Each trip leg $t \in T$ defines a single trip with the fixed departure time, origin station, arrival time, and destination station. Moreover, the corresponding train of each trip leg is hauled by a set of assigned locomotives $L_t \subseteq L$. Each locomotive $l \in L$ has a fixed route and a fixed tank capacity P_l. The route of a locomotive indicates the sequence and the time of stations that the locomotive visits during the time horizon, and is derived from the given assignments of locomotives to train trip legs. The set of inline tanks I are similar to locomotives in that they have a fixed capacity P_i for all $i \in I$, but the assignment of these tanks to train trips is to be determined and optimized. The schedule is assumed to be cyclic which requires that the same fuel tanks (inline tanks or locomotives' tanks) must return to the initial location with the same fuel level. The train schedules and locomotive routes are cyclic in the given input, and only the fuel plans of locomotives must be ensured to be cyclic. However, as the routes of inline tanks are not given as the input, both their routes and fuel plans must be ensured to be cyclic. The cyclic requirement is due to the common usage of cyclic schedules in railroad companies. The fuel consumption of train trip legs is predetermined and deterministic and is equally divided between the locomotives that haul the corresponding train. The extra weight of the tankered fuel may increase the fuel consumption. However, since the extra weight is trivial compared to the train weight, the extra fuel consumption due to fuel tankering is ignored as is common in fuel optimization problems in rail transportation (Nourbakhsh and Ouyang, 2010). The extra weight could also be modeled by a slight reduction of the technical fuel capacity.

Three cost types are incorporated in the locomotive refueling operations, namely, the fuel purchasing cost, the refueling operation fixed cost, and the fixed cost of stations' services. The fuel prices C_s^F vary at different stations $s \in S$. Each refueling operation of the locomotives incurs a fixed cost C_s^H which is called the *stop cost*. Similarly, the refueling of inline tanks at the stations incurs the same fixed cost. Using a station $s \in S$ for refueling

purposes during the time horizon incurs another fixed cost C_s^O. If the company owns the station, this fixed cost is associated with the costs of maintenance of the facilities and the labor that are related to the refueling operation. Otherwise, it is a fixed cost to be paid to the fuel supplier as a contract fee. In addition, it is assumed that the refueling at a station is possible for a locomotive or an inline tank only if it waits at the station more than the required time of a refueling operation.

We consider two options for locomotive refueling: The first option, which is studied in the literature, is the conventional direct refueling of the locomotives at stations. The second option, which is not studied in the literature so far, is the option of inline refueling, which is carried out by inline tanks. This allows locomotives to travel longer distances without having to stop at intermediate stations for refueling. Inline tanks assigned to a train are mounted on a wagon and connect through one pumping system to all locomotives that haul the train. Therefore, since they use same inline tanks, locomotives on the same train are not independent (in contrast to the conventional refueling case which they can be considered separately regardless of trains they haul). Inline refueling reduces total fuel management costs since it enables the company to plan to buy more fuel from inexpensive stations (and perhaps, to close down some fuel stations too). Moreover, it decreases the number of refueling stops. However, the available inline tanks should be managed efficiently due to limited budget to procure these inline tanks. Given a number of available inline tanks, in addition to decisions of the traditional integrated problem, which are selecting the fuel stations locations and refueling scheduling (where to refuel) and fuel planning (the amount of fuel purchase) for locomotives, we also need to determine the assignment of the available inline tanks to the train trips. This, accordingly, determines the routes of the inline tanks during the time horizon. Furthermore, each train is hauled by one or several locomotives collaboratively. Hence, the locomotives, among such collaborating locomotives, that use the assigned inline tanks to the train, are to determined. Additionally, the fuel plan for inline tanks should be optimized in a manner similar to that of the locomotives.

In practice there are a number of other restrictions that need to be considered, leading to different variants of the basic problem. These considerations include limiting the number of locomotives that may be refueled from an inline tank, limited number of inline tanks per train trip, balancing requirements between inline tanks used on the same trip, and restrictions on the transfer of fuel from inline tanks to locomotives and between inline tanks both during trips and at the stations. However, as we shall see in the next section, the problem is already hard even when considering only the most basic model of inline refueling. Such extra considerations are incorporated in the proposed model in Section 3.5.

Since the GFMIR problem is complicated, as we will show theoretically and empirically, we limit it to deterministic input. Although several parameters of the problems, such

as the fuel price and the fuel consumption, are intrinsically uncertain, the limitation to deterministic input is quite reasonable indeed. We consider a weekly or fortnightly time horizon, which is common in the railroad industry for train and locomotive planning. Railroad companies usually hedge fuel prices over months, and hence, considering a fixed, deterministic fuel price over two weeks aligns with the practice. Moreover, uncertain fuel consumption can be handled with a small cost by considering safety fuel levels as we will show. The limitation to deterministic input allows us to study the proposed models and algorithms in various scenarios, which provides a basis to extend the proposed models and algorithms to more sophisticated cases, including the GFMIR problems with uncertain parameters.

3.3 Complexity of the Problem

In this section, we prove that GFMIR is NP-hard. Moreover, we demonstrate that the restricted version of the problem, in which we ignore the decisions related to selecting the fuel stations locations and the decisions related to where to refuel, is also NP-hard. This is interesting, as the restricted problem is simply a minimum-cost network flow problem to which inline tanks have been added. Clearly, the fuel management problem with inline refueling is a generalization of the problem set in the INFORMS Problem Solving Competition (2010), which has been proved to be NP-hard by Raviv and Kaspi (2012). Therefore, the locomotive fuel management problem with inline refueling is NP-hard:

Corollary 3.3.1 (Proposition 1 Raviv and Kaspi (2012))**.** *GFMIR is NP-hard.*

Next, we consider the restricted version of GFMIR by ignoring the decisions related to selecting the fuel stations locations (hence all stations are considered opened) and where to refuel (no fixed costs for stops). Moreover, we consider three other simplifications from the problem described in Section 3.2 to obtain the simplified problem RGFMIR1 for the complexity proof. The simplifications are: (a) dropping cyclicity requirement for locomotives and inline tank, (b) initial fuel level of locomotives and inline tanks are fixed to full capacity as the given input, and (c) initial location of inline tanks are fixed based on the given input. Subsequent to the complexity proof of RGFMIR1, we cancel these three relaxations for the problem and show that the resulting problem is still NP-hard. Finally, we prove that the restricted problem is NP-hard even if one inline tank is available.

Problem 3.3.1 (RGFMIR1)**.** *Input: A directed graph $G = (V, A)$ in which nodes correspond to the stations with fuel price $c(v)$, a set L of locomotives with tank capacity P_l for all $l \in L$, a set I of inline tanks with capacity P_i and the initial location $v(i)$ for all $i \in I$. All locomotives and inline tanks start with full capacity. A set T of train trip legs, where a trip $t \in T$ is defined as an arc a_t with departure time $a(t)$, arrival time $w(t)$, and the set $H(t) \subseteq L$ which determines the locomotives that*

haul the train. Based on the information of all train trip legs, each locomotive l visits a sequence of stations Q_l in time order W_l, and hauls the corresponding trains for a sequence of trip legs T_l. The consumption D_t denotes the fuel consumption of each locomotive that hauls the corresponding train of trip leg t.

Solution:

- *Assignment of each inline tank $i \in I$ to a set of train trip legs $T_i \subseteq T$ which translates to the sequence of stations Q_i it visits in time order W_i.*
- *For each trip leg that inline tank i is assigned to, i.e. for all $t \in T_i$, a set of locomotives $A_i^t \subseteq H_t$ that connect to and use this inline tank during the trip. On the other hand, set A_l^t indicates the inline tanks that are connected to locomotive l during trip leg t. Inline tanks assigned to a same train are mounted on a wagon and connected to the selected locomotives through a shared pumping system. Therefore, with I_t denoting the inline tanks assigned to the trip leg t, we have either $A_l^t = I_t$ (locomotive l is connected to the pumping system) or $A_l^t = \emptyset$ (locomotive l is not connected to the pumping system) for all $l \in H_t$.*
- *Fuel purchase plan for each locomotive l as F_l^q that determines the amount of fuel purchased for this locomotive at the q^{th} station it visits for all $q \in Q_l$.*
- *Fuel purchase plan for each inline tank i as F_i^q that determines the amount of fuel purchased for this inline tank at the q^{th} station it visits for all $q \in Q_i$.*
- *Inline refueling plan for each inline tank i as Y_i^{lt} that indicates the amounts of fuel transferred from inline tank i to locomotive l during trip t for all $t \in T_i, l \in A_i^t$.*
- *Let T_l^q (T_i^q) denote the trip legs that locomotive l (inline tank i) has taken to arrive at its q^{th} station in its sequence Q_l (Q_i). Fuel purchase and inline refueling plans determine the fuel level of each locomotive l over the sequence of stations it visits as $f_l^q = P_l + \sum_{j=1}^{q} F_l^j - \sum_{t \in T_l^q} D_t + \sum_{i \in A_l^t} \sum_{t \in T_l^q} Y_i^{lt} : \forall q \in Q_l$ and the fuel level of each inline tank i over the sequence of stations it visits as $f_i^q = P_i + \sum_{j=1}^{q} F_i^j - \sum_{l \in A_i^t} \sum_{t \in T_i^q} Y_i^{lt} : \forall q \in Q_i$.*
- *The assignments, fuel purchase plans, and inline refueling plans must be with the minimum cost while the fuel level of the locomotives and inline tanks is always non-negative and does not violate their corresponding capacity.*

Theorem 3.3.1. *RGFMIR1, which is the restricted version of GFMIR, is strongly NP-hard.*

Proof. This is a reduction from the classic 3-SAT problem where there is an inline tank per Boolean variable, and it is assigned one of two possible routes, meaning TRUE of FALSE. Each clause will require one inline tank to be feasible.

Consider an instance of 3-SAT with n Boolean variables $v_1, \ldots, v_n$ and m clauses, i.e.,

$c_1 \wedge c_2 \wedge \quad \wedge c_m$, each of which contains three literals, i.e., $c_i = (l_{i1} \_ l_{i2} \_ l_{i3})$, where each literal is a Boolean variable v_k or its negation. Moreover, $s(j,v)$ returns the index of the clause of the j^{th} appearance of variable v in the clauses (assuming clauses are ordered with respect to their index number). $p(i,v)$ shows the location of variable v in clause c_i as a literal (assuming literals within a clause are ordered by their index). $s(j,v)$ and $p(i,v)$ distinguish the appearance of a variable v_k and its negation $\bar{v}_k$. h_k and $\hbar_k$ represents the number of times variable v_k and its negation has appeared in the clauses, respectively. Without loss of generality, we suppose that each variable appears in each clause at most once and appears in at least one clause.

In order to reduce the 3-SAT instance, we construct the RGFMIR1 instance as follows. In the reduced instance, all trains are hauled by exactly one locomotive. Therefore, trains are not discussed since each trip leg can be viewed as a separate train which does not affect the inline tanks assignments. We present different gadgets in the reduction that each makes a part of the reduced instance and have different characteristics.

- Clause locomotives: for each clause, there is a clause locomotive with capacity 2 that takes three trips, each corresponding to a literal. Specifically, locomotive i, corresponding to clause c_i, takes three trips $\{t_{i1}, t_{i2}, t_{i3}\}$, each consuming one unit of fuel ($D_{t_{ij}} = 1 : 8j \in \{1,2,3\}$). Locomotive i over trip t_{ij} departs station S_i^j at time $a(t_{ij}) = 10i + 2j$ and arrives at station S_i^{j+1} at time $w(t_{ij}) = 10i + 2j + 1$. The fuel price at all such stations is 1 ($c(S_i^j) = 1 : 8j \in \{1,2,3,4\}$).
- Literal connection locomotives: these locomotives sequentially link the presences of a variable in the clauses (separately for each variable and its negation). This leads to the set of literal connection locomotives L_{v_k} where $|L_{v_k}| = h_k \quad 1$ for variable v_k and the set of literal connection locomotives $L_{\bar{v}_k}$ where $|L_{\bar{v}_k}| = \hbar_k \quad 1$ for variable $\bar{v}_k$. Locomotive $j \in L_{v_k}$ with the capacity of 2 takes two trips t_1^{jk} and t_2^{jk}, each consuming one unit of fuel. Trip t_1^{jk} departs station $S_{s(j,v_k)}^{p(s(j,v_k),v_k)+1}$ at time $a(t_1^{jk}) = 10s(j,v_k) + 2p(s(j,v_k),v_k) + 1$ and arrives at station $S_j^{v_k}$ at time $w(t_1^{jk}) = 10s(j,v_k) + 2p(s(j,v_k),v_k) + 2$. Trip t_2^{jk} departs station $S_j^{v_k}$ at time $a(t_2^{jk}) = 10s(j,v_k) + 2p(s(j,v_k), v_k) + 3$ and arrives at station $S_{s(j+1,v_k)}^{p(s(j+1,v_k),v_k)}$ at time $w(t_2^{jk}) = 10s(j+1,v_k) + 2p(s(j+1,v_k),v_k)$. Fuel price at station $S_j^{v_k}$ is zero. Literal connection locomotives for variable $\bar{v}_k$ are added similarly.
- Inline tanks: There are n inline tanks, each corresponding to a Boolean variable. Each inline tank $k \in \{1,\ldots,n\}$ is initially located at station S_k^0 with the capacity of one unit of fuel ($P_k = 1$). Fuel price at station S_k^0 is zero.

- Boolean assignment locomotives: there are two locomotives with the capacity of 1 unit of fuel per Boolean variable k as l_k^0 and $\bar{l}_k^0$. Locomotive l_k^0 takes one trip t_k^0, consuming one unit of fuel. Trip t_k^0 departs station S_k^0 at time $a(t_1^k) = 0$ and arrives at station $S_{s(1,v_k)}^{p(s(1,v_k),v_k)}$ at time $w(t_k^0) = 10s(1,v_k) + 2p(s(1,v_k),v_k)$. This locomotive essentially links the initial location of inline tank k to the trip corresponding to the first presence of the variable v_k in the clause locomotives' trips. Similarly, trip $\bar{t}_k^0$ of locomotive $\bar{l}_k^0$ links the initial location of inline tank k to the trip corresponding to the first presence of the variable $\bar{v}_k$ in the clause locomotives' trips. As we will show, the assignment of inline tank k to locomotive l_k^0 ($\bar{l}_k^0$) translates to setting $v_k = 1$ ($\bar{v}_k = 1$).
- Terminal locomotives: to ensure that inline tank assigned to a variable is not freely switched between trips related to different variables, we introduce terminal locomotives. There is one terminal locomotive per each Boolean variable k, n terminal locomotives in total. A terminal locomotive with the capacity of 1, corresponding to the k^{th} binary variable, takes two trips t_1^{km} and t_2^{km}, each trip consuming 1 unit of fuel. Trip t_1^{km} departs station S_1^{km} at time $a(t_1^{km}) = max\{10s(h_k,v_k) + 2p(s(h_k,v_k),v_k) + 1, 10s(\hbar_k,\bar{v}_k) + 2p(s(\hbar_k,\bar{v}_k) + 1\} + 10$ and arrives at station S_2^{km} at time $w(t_1^{km}) = a(t_1^{km}) + 1$. Trip t_2^{km} departs station S_2^{km} at time $a(t_2^{km}) = w(t_1^{km})$ and arrives at station S_3^{km} at time $w(t_2^{km}) = a(t_2^{km}) + 1$. Fuel price at stations S_1^{km}, S_2^{km}, and S_3^{km} is equal to 1.
- Terminal connection locomotives: per each Boolean variable k, there is one terminal connection locomotive for v_k and one for $\bar{v}_k$. Such locomotives link the last presence of variables v_k and $\bar{v}_k$ to the corresponding terminal locomotive of k^{th} Boolean variable. The connection are made with the same logic as literal connection locomotives (a locomotive with the capacity of 2 taking 2 trips consuming 1 unit of fuel that pass from an intermediate station with the fuel price of zero, and with appropriate time schedule that arrive to the corresponding station before the relevant terminal locomotive leaves).

Figure 3.1 shows the reduction of an instance of 3-SAT as $(v_1 \_ v_2 \_ \bar{v}_4) \wedge (v_2 \_ v_3 \_ \bar{v}_4) \wedge (\bar{v}_1 \_ v_2 \_ \bar{v}_3)$. In this figure, each arrow indicates a locomotive of the types mentioned before that may pass some intermediate stations. The sequence of locomotives that visit a same station is shown on a same node that the preceding locomotive is an incoming arc to that node and the subsequent locomotive is an outgoing arc to the same node. Clause and terminal locomotives are shown in colors black and blue, respectively. Other types of locomotives are shown in red and shown only for the ones that correspond to variable v_2 to have a clear figure. Inline tanks are located at the red square nodes on the left side of the figure. The fuel price of stations with nonzero price, are labeled above the corresponding node. The dotted line represent the time that a locomotive stays in a station idle.

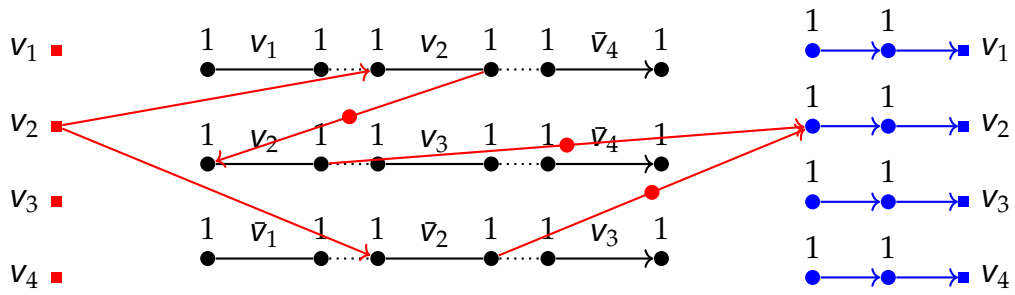


Figure 3.1: Reduction of a 3-SAT instance

Considering this reduction, an instance of 3-SAT is satisfiable if and only if there is an optimal solution with total cost of zero for its corresponding instance of RGFMIR1. Clearly, when an instance of 3-SAT is satisfiable, we can translate the solution of the 3-SAT instance to assignments of inline tanks to the corresponding trips to have the total cost of zero.

Conversely, assignments of inline tanks to trips corresponds to setting a variable to TRUE ($v_k = 1$) or FALSE ($\bar{v}_k = 1$). These assignments are equivalent to a solution of 3-SAT because of the three reasons. First, the inline tanks cannot be swapped between trips that correspond to the TRUE and FALSE value of the same variable since there is no link between the TRUE and FALSE values of the same variable. Second, the inline tanks cannot be swapped between trips that correspond to different variables without incurring cost to the fuel plan. Because the schedule of such trips are mutually exclusive. Also, between each two successive trips there is a non-zero idle time. Therefore, if an inline tank is swapped from a trip corresponding to a literal to a trip corresponding to another literal, it cannot be swapped back, which results in at least one gadget (terminal locomotives) having a cost greater than zero. Third, locomotives that correspond to each clause cannot take all trips with cost of zero unless an inline tank can be assigned to them at least once, which is equivalent to saying the clause is satisfied. Hence, by finding a fuel plan with cost of zero we can construct a satisfiable solution for the 3-SAT instance.

In addition, RGFMIR1 is in the class NP since every solution for this problem is a flow in the network for which it can be certified in polynomial time whether all of the demands are satisfied or not. Furthermore, the reduction is polynomial with respect to the number of variables and clauses. Therefore, RGFMIR1 is strongly NP-hard. □

RGFMIR1 is a simplified version of the problem described in Section 3.2. We considered three relaxations at the beginning of this section to obtain a simplified problem for the complexity proof. Next, we undo such relaxations step by step and show RGFMIR1 with those considerations is also strongly NP-hard.

Corollary 3.3.2. *The variant of RGFMIR1, in which the fuel plan of the locomotives and the inline*

tanks must be cyclic, is strongly NP-hard.

It is enough to consider an extra trip for each locomotive in the reduction of the proof of Theorem 3.3.1 to ensure the cyclic plan of the locomotives. The destination of this extra trip is the initial location of the locomotive and consumes no fuel.

For the inline tanks, we assume that the final destination of terminal node that correspond to inline tank k (corresponding to k^{th} variable) is station S_k^0.Furthermore, by considering cyclic schedules, determining the initial fuel level of the locomotives and the inline tanks can be incorporated in the problem.

Corollary 3.3.3. *The variant of RGFMIR1, in which the initial location of inline tanks is not fixed as a given input, is strongly NP-hard.*

By this generalisation, it is not ensured that the optimal solution of the problem has exactly one inline tank assigned to each variable. However, we ensure that exactly one inline tank arrive to station S_k^0 for all $k \in \{1,\ldots,n\}$. To do this, we consider a locomotive schedule per each inline tank that needs an assignment of an inline tank to allow a zero cost solution (similar to clause locomotives). These schedules start at the same time, so one inline tank cannot be assigned to more than one of them. The locomotive corresponding to inline tank k terminates to station S_k^0.

Considering Corollaries 3.3.2 and 3.3.3 and the proof of Theorem 3.3.1. RGFMIR2, which is described in Corollary 3.3.4 is also NP-hard since we can consider Corollaries 3.3.2 and 3.3.3 at the same time in the reduction. In the rest of this chapter, "*restricted problem*" refers to RGFMIR2.

Corollary 3.3.4 (RGFMIR2)**.** *This problem which is based on RGFMIR1 while considering cyclic schedules for the locomotives and the inline tanks and assuming that the initial location of the inline tanks and initial fuel levels are not fixed as the given input is strongly NP-hard.*

Finally, we prove that RGFMIR1 and its variants are NP-hard even if there is only one inline tank available. In the reduction, instead of considering n inline tanks, each located initially at station S_k^0 for all $k \in \{1,\ldots,n\}$, consider one inline tank initially located at station S_1^0. Then, the terminal locomotive k is connected to station S_{k+1}^0 by a locomotive similar to literal connection locomotives, and time schedules are adjusted accordingly. Since the resulting connections are acyclic, we can create train schedules that correspond to those connections. Moreover, this construction can be used along with the gadgets that have been introduced for Corollaries 3.3.2 and 3.3.3.

Corollary 3.3.5. *RGFMIR1 and RGFMIR2 are strongly NP-hard even if only one inline tank is available.*

3.4 Mathematical Modeling

We model GFMIR on a time-space network which determines the location of fuel stations, the fuel plans of the locomotives and the inline tanks, and the assignments and the routes of the inline tanks. In this section, first we define a time-space network to capture all considerations of the problem. Time-space networks have been successfully employed for various transportation applications (Kliewer et al., 2006; Steinzen et al., 2010; Guedes and Borenstein, 2018). In contrast to the traditional locomotive fuel management problem, GFMIR involves routing decisions for inline tanks. Time-space networks have been utilized for routing problems due to their advantages in modeling possible connections between trips. A time-space network connects all trips related to a same location with a timeline instead of explicitly pairing all possible combinations of trips. Next, a Mixed-Integer model is proposed based on this network. In the proposed model in this section, we do not include the details of a specific railroad application. The aim is to develop a general model that can be applied to different rail companies.

3.4.1 Network Representation

In this section, we conceptualize the GFMIR problem as a time-space network. Components of this network represent and relate activities that are involved in fuel management operations: (a) opening/closing fuel stations, (b) direct refueling to locomotives and inline tanks at fuel stations, (c) movement and fuel level of inline tanks, where movements are determined by their assignments to train trip legs, (d) inline refueling to locomotives by the assigned inline tanks to the corresponding train, and (e) movement and fuel plan of locomotives. In contrast to inline tanks, movement of locomotives is a given input.

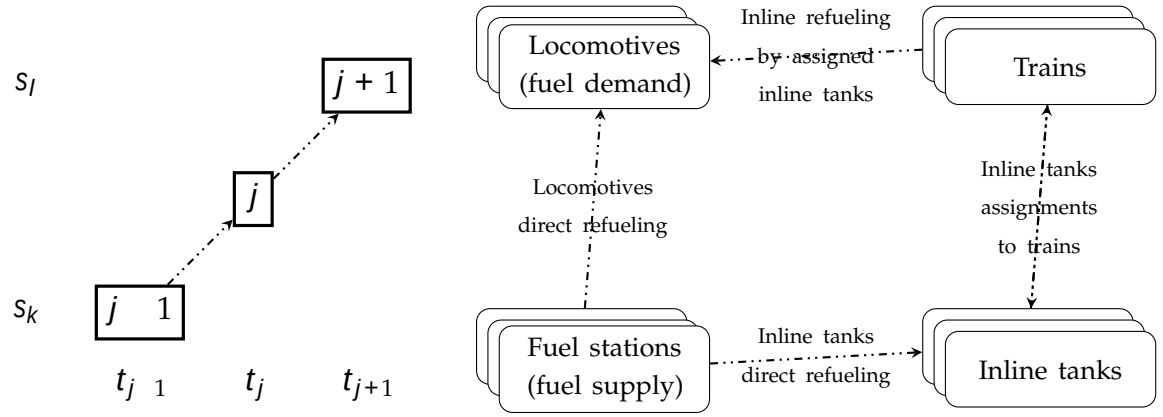
In the proposed network, nodes correspond to a locomotive or an inline tank staying at a station over a time period, taking a trip leg, or operation of a fuel station during the time horizon. On the other hand, arcs represent a specific instant of time that an activity starts/ends or a refueling operation. Flow on arcs indicate the amount of fuel at that time, which can be the fuel level of a locomotive or an inline tank at the time or the amount of fuel transfer by a refueling operation. Arcs connect relevant activities and are of two types: First, movement connections that join the sequence of trips and idle periods of a locomotive or an inline tank during the time horizon. Second, fuel circulation connections that can be direct refueling at a fuel station or inline refueling. Movement connections are based on the time sequence and location match of activities of a locomotive or an inline tank. Direct refueling arcs connect a fuel station to the node representing a time period that an inline tank or a locomotive stays at that station. Inline refueling connects the nodes representing the same trip for a locomotive and the inline tanks. Since each node corresponds to a locomotive,

an inline tank, train trip legs, or fuel stations, the time-space network can be viewed as different layers. Each layer corresponds to a specific entity, for instance a locomotive, and includes only the corresponding nodes. Different layers are connected by fuel circulation connections. However, there is only one commodity, fuel, flowing in the network.

Figure 3.2 sketches the general structure of the time-space network. Figure 3.2a illustrates the movement connections for a simple example. Suppose three activities $f j \quad 1, j, j + 1 g$ are defined for a locomotive (or an inline tank). Note that the logic for representing the activities of the locomotives and inline tanks are the same as their activities are either staying at a station during a time period or taking a trip between two stations during a time period. These activities correspond to staying at station s_k during time period $t_{j \ 1}$, taking a trip from station s_k to station s_l during time period t_j, and staying at station s_l over time period t_{j+1}, respectively. Moreover, these time periods imply that the activities occur as the sequence $f j \quad 1, j, j + 1 g$. Therefore, two arcs are added to show the sequence. By this illustration, flow on the arc from $j \quad 1$ to j indicates the fuel level of the locomotive (or the inline tank) before taking trip leg j, and the arc from j to $j + 1$ indicates its fuel level after taking trip leg j. Fuel circulation connections join different layers between fuel stations, inline tanks, train trip legs, and locomotives. Figure 3.2b illustrates the general relation between layers. Supply of fuel is at fuel stations, whereas locomotives demand fuel. Fuel can be transferred to locomotives directly from stations or through inline tanks. The routes of inline tanks are determined by their assignments to train trip legs. Then, the assigned inline tanks to each train trip leg are able to perform inline refueling to the corresponding locomotives through the added inline refueling connections between trains and locomotives layers. Movement connections are within each layer and are not shown in the general sketch. The rest of this section explains the different node and arc types that are introduced to capture different operations in fuel management. Moreover, *refueling* refers to direct refueling at the fuel stations which we distinguish from inline refueling.

Trip leg nodes:

Trip leg nodes represent scheduled time period of taking a trip. For each trip leg, there is a trip leg node for each locomotive that hauls the train and one node for all inline tanks. An example of a trip node for a locomotive and for inline tanks are shown in Figure 3.3. The solid arcs, called *Tank Arcs* from here on, are the movement connections that join the trip leg node to the nodes representing previous (staying at the origin station during the previous time period) and next (staying at the destination station during the next time period) activities. We will next discuss the nodes that represent staying at the stations for a time period. As is shown in Figure 3.3a, the trip node of a locomotive has only one incoming tank arc and one outgoing tank arc. Moreover, this node has a demand which is equal to

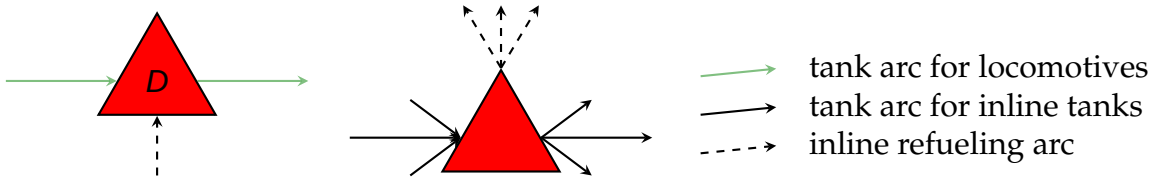


(a) Movement connections for a simple example. A locomotive (or an inline tank) undertakes three activities in a sequence: staying at station s_k during time period $t_{j\ 1}$ (activity j 1), taking a trip from s_k to s_l during time period t_j (activity j), and staying at station s_l during time period t_{j+1} (activity $j+1$). Nodes correspond to activities. Arcs link relevant activities and hence, an arc represents the instant of time when one activity in one period ends and the next activity in the next period begins. Flow on these arcs show the fuel level of the corresponding locomotive (or inline tank) at the time the corresponding activity of the origin node of such an arc ends (or equivalently, at the time the corresponding activity of the destination node of such an arc starts).

(b) Fuel circulation connections. Each layer includes some nodes and movement connections within itself that are not shown in this figure. Fuel circulation connections transfer the fuel between the layers by connecting the relevant nodes in two layers. The relation between trains and inline tanks layers is two-way. When an inline tank is assigned to a train, it travels along the train path. After decoupling that inline tank from the corresponding train, it stays at a station which is represented in the relevant layer of the inline tank as a node.

Figure 3.2: A sketch of the time-space network. In the time-space network, nodes represent activities over a time period. Relevant activities are connected by arcs. There are two types of connections: 1) movement connections that connect activities related to movement of inline tanks and locomotives in the network. A movement connection is not a movement itself but a logical connection between a movement (which is represented by an activity node) and another movement or another activity. 2) Fuel circulation connections that transfer the fuel between different components of the problem by direct refueling at stations or inline refueling through inline tanks.

the fuel consumption of the locomotive during the corresponding trip. However, based on Figure 3.3b, the node of the inline tanks have several incoming and several outgoing tank arcs. That is because only one node is considered for all of the inline tanks for each trip leg. Therefore, the number of the incoming tank arcs and the number of the outgoing tank arcs are equal to the number of the inline tanks. Moreover, the outgoing dashed arcs, which we call as *Inline Refueling Arcs*, represent the inline refueling of the locomotives that haul the corresponding train by the assigned inline tanks. Hence, the number of the inline refueling arcs is equal to the number of locomotives that haul the train on the corresponding trip leg.



(a) A trip leg node for a locomotive. Tank arcs of the locomotives are distinguished from the tank arcs of the inline tanks with the green color as the route of locomotives is a given input and indicate a fixed route. The fuel demand D shows the fuel consumption of the locomotive during the trip.

(b) A trip leg node for the inline tanks. Since inline tanks are attached as one wagon to trains, one trip leg node is considered for all inline tanks to represent the trip of the assigned inline tanks along with the train.

Figure 3.3: A trip leg node which represents the time period of taking a trip leg. Tank arcs represent the start and end time of the corresponding trip leg. Tank arcs are of movement connections type. Therefore, they connect a trip leg node to the corresponding previous (staying in the origin station) and next (staying in the destination station) activities. Inline refueling arcs connect the trip leg node of the inline tanks to the locomotives that haul the corresponding train. The flow on tank arcs represents the fuel level of the locomotive or the inline tank before and after taking the trip. The flow on inline refueling arcs represents the amount of fuel that is transferred from the assigned inline tanks of a train to the corresponding locomotives during the trip.

Station nodes:

Station nodes represent the stay of a locomotive or an inline tank during a time period as an activity. Previous and next activities of a stay can be taking a trip leg or staying at the same station during another time period. The relevant activities are connected by tank arcs. Arriving at (departing from) the corresponding station is indicated by an incoming (outgoing) tank arc, the other side of which is connected to the relevant trip leg node. Being idle is represented by a tank arc that connects two station nodes of the same station with different time periods. The flow on tanks arcs indicate the fuel level of the corresponding locomotive or inline tank at the time. A station node is split into two successive nodes. Refueling can be done only at the first node. The second node is the time period that the inline tank or the locomotive dwell at the station, and its duration is less than the required time of refueling. The refueling time is assumed to be predetermined and fixed. The refueling of the locomotive or the inline tank during its stay at the station is shown by the *Refueling Arcs* as the dotted arrow in Figure 3.4, which is connected to the refueling-allowed node. The other side of the refueling arcs is connected to the node that represent the fuel supply of the corresponding station, which we will discuss next. The flow on the refueling arc states the amount of refueling. The two divided nodes are connected by a linking arc to conserve the fuel flow.

The time period of each station node starts from the moment that the first incoming arc

arrives to the station and continues until the outgoing arcs depart the station. The outgoing arcs depart the station at the same time while the incoming arcs arrive at the station at different times during the corresponding time period of the node. Therefore, the station is divided into two successive nodes based on the start time of the node, the end time of the node, and the required refueling time. Figure 3.4 shows an example of a station node for an inline tank. In this figure, it is assumed that the outgoing arcs a_5 and a_6 depart at time 3:00 PM and the required refueling time is 30 minutes. Arc a_0 indicate that the inline tank has been idle at the station at the previous time period. Moreover, arcs a_1, a_2, and a_3 arrive to the station at time 1:00 PM, 2:00 PM, and 2:45 PM, respectively. Therefore, the time period of the node is from 1:00 until 3:00. The first node – in which refueling is allowed – is from 1:00 to 2:30, and the second node represents the remaining time. Since arcs a_0, a_1 and a_2 arrives before 2:30, they are connected to the refueling allowed node. While, the arc a_3, arriving at 2:45, is connected to the refueling disallowed node, which means that if this route is selected for the inline tank, we cannot refuel it at this station. Moreover, arc a_4 connects the two nodes of the station. Station nodes of the locomotives are simpler because the route of the locomotives is predetermined. Therefore, there is only one incoming tank arc, one outgoing tank arc, and possibly a refueling arc.

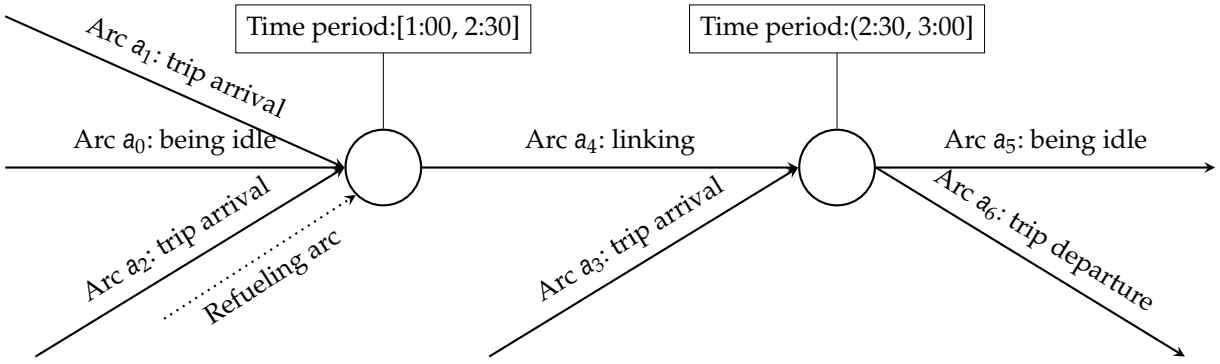


Figure 3.4: A station node for an inline tank. Tank arcs (solid arrows) represent the movement of the inline tank that can be arriving from a trip leg, departing to take a trip leg, and being idle at the station at the previous/next time period. The other side of tanks arcs that are related to a trip leg, are connected to the relevant trip leg nodes. Tank arcs representing being idle are connected to relevant station nodes. Flow on tank arcs indicate the fuel level at the time. Refueling arcs represent a refueling operation for the inline tank during its stay at the station. The other side of the refueling arc is connected to the node that represents the fuel supply of the corresponding station. Flow on the refueling arc indicate the amount of the refueling.

Fuel supply nodes:

All fuel flowing in the time-space network stems from one node, the *total fuel supply* node. The supply of this node is equal to the total fuel consumption of all locomotives during the time horizon. The fuel flows into the network through intermediate nodes, *station supply*

nodes. Each station supply node represents the fuel supply at a station. Therefore, the flow on the arc between the total fuel supply node and each station supply node indicates the total fuel purchased at the corresponding station during the time horizon. Moreover, such arcs capture the decisions on opening/closing the fuel stations. These arcs are called *Station Supply Arcs* from here on and shown by double-lined arrows in Figure 3.5. Refueling arcs connect the fuel supply nodes to the relevant station nodes of the locomotives and inline tanks, which represents a refueling operation for the corresponding locomotive or inline tank at the respective station. The connection of supply nodes is illustrated in Figure 3.5.

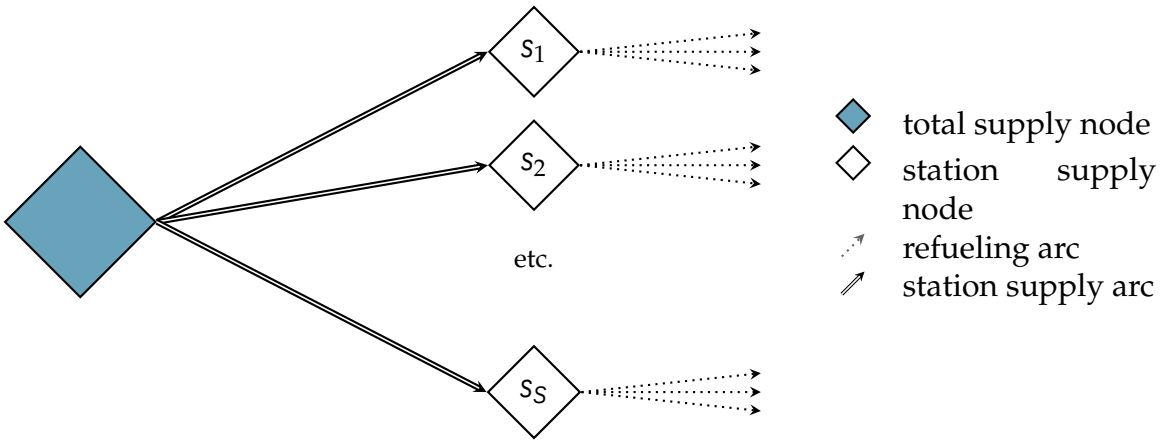


Figure 3.5: Fuel supply nodes. All fuel in the network stems from the total supply node and flows to station supply nodes. A station supply node represents the fuel supply at a station. The flow on station supply arcs indicate the total fuel purchased at the corresponding stations during the time horizon. Refueling arcs represent the refueling operations at the stations. Therefore, the other side of refueling arcs is connected to the corresponding stations nodes of the locomotives and inline tanks.

Inline refueling arcs:

After assigning the inline tanks to a trip leg of a train, it is required to determine which locomotives (of the locomotives that haul the train) are refueled by the assigned inline tanks. Suppose that three locomotives haul a train during a trip. In Figure 3.6, the bottom triangle represents the corresponding trip leg node for the inline tanks, and the upper triangles represent the trip node for the corresponding locomotives. The trip nodes of the locomotives and trip node of the inline tanks are connected through the dashed arcs, which we call *Inline Refueling Arcs* from here on. The inline refueling arcs determine which locomotives are refueled by the inline tanks as well as the amount of inline refueling.

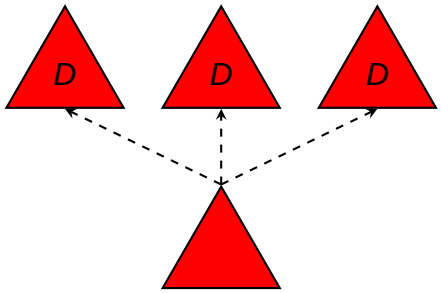


Figure 3.6: Connection of inline refueling arcs. This represents the inline refueling connections for a trip leg that is hauled by three locomotives (upper triangle nodes with the fuel demand of D). The bottom triangle node represents the same trip leg for inline tanks. Trip leg nodes of locomotives and inline tanks are connected by dashed arcs that represent inline refueling. The flow on these arcs indicates the amount of inline refueling that each locomotive receives during the trip leg. These nodes have incoming and outgoing tank arcs as explained before, which are not shown here.

Network of the problem:

Figure 3.7 illustrates the time-space network for a small example, consisting of 3 stations, 2 locomotives l_1 and l_2, and one inline tank i_1. Three trips legs, represented by colored triangle trip leg nodes, are taken by these two locomotives. The red trip leg is done by locomotives l_1 and l_2 collaboratively. Each layer of the network includes nodes and arcs related to a component. Refueling arcs (dotted arrows) and inline refueling arcs (dashed arrows) connect the relevant nodes of two layers. The connection between the inline tank i_1 layer and trip leg nodes layer represent the assignment of this inline tank to the corresponding train of each trip leg. For each extra inline tank becoming available, the same layer as the inline tank i_1 layer will be added to the time-space network. However, there is no need to add another similar layer to trip leg nodes layer. This is because all assigned inline tanks to a train are attached to a train as one wagon with a shared pumping system.

The problem represented by the time-space network can be viewed as two interrelated network design and network flow problems. Network design problem selects a subset of arcs to determine routes of inline tanks, refueling scheduling decisions, stations opening/closing decisions, and inline refueling decisions. Locomotive routes are given input. Therefore, the arcs that determine their routes are fixed by the input and distinguished from other arcs by the green color in figures. Once a subset of arcs is selected, the network flow problem determines the circulation of the fuel flow in the network which translates to fuel plans of the locomotives, inline tanks, and fuel stations. These two problems are shown in Figure 3.8 for the illustrated instance of Figure 3.7. Refueling arcs in this figure originate from the fuel supply layer that is not shown here. The design problem in Figure 3.8a selects a subset of arcs. For the locomotives, the problem is to select which refueling operations to be operated (by dotted arrows) and when to receive inline refueling (by dashed arrows).

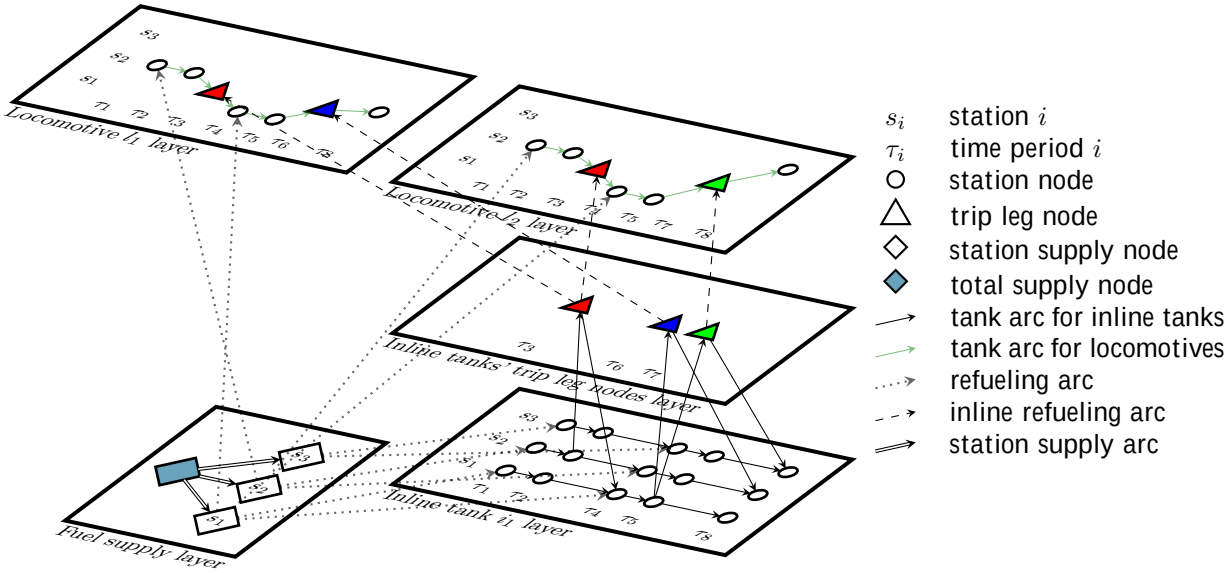
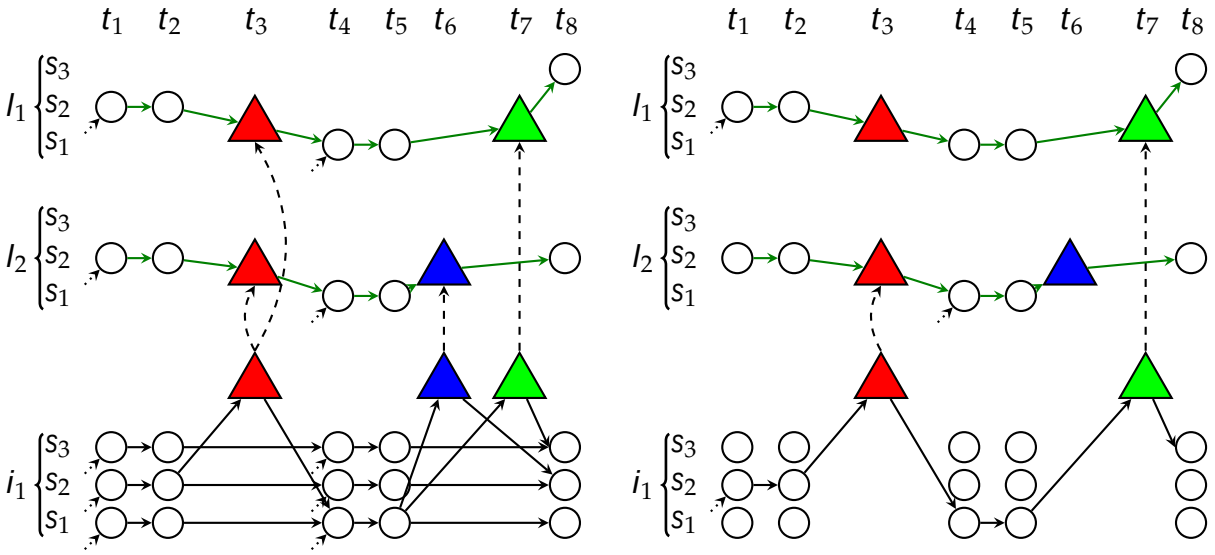


Figure 3.7: Network representation of an instance of the problem with 2 locomotives, 1 inline tank, 3 stations, and 3 train trip legs. Station nodes represent the time period an entity (locomotive or inline tank) stays at a station. Trip leg nodes represent the time period of taking a trip leg. The demand is nonzero only for trip leg nodes of the locomotives and is equal to their fuel consumption during the corresponding trip leg. The supply is nonzero only for the total supply node and is equal to the total fuel consumption of all locomotives.

For the inline tank i_1, the problem is to select the refueling operations, which train trip legs to be assigned to and operate inline refueling. An example of the selected arcs is shown in Figure 3.8b. Based the selected arcs, locomotive l_1 is refueled at station s_2 during time period t_1 and receives fuel from the assigned inline tanks during its second trip. Locomotive l_2 receive fuel from the inline tanks during its first trip and is refueled at station s_1 during time period t_4. Inline tank i_1 travels from station s_2 to station s_1 and then from station s_1 to station s_3 along with the corresponding trains it is assigned to. During these trips, it delivers fuel to locomotives l_2 and l_1, respectively. Similarly, if a station is decided to be open during the time horizon, its corresponding supply arc (shown in Figure 3.7) is included in the selected arcs. The network flow problem determines the fuel flow on the selected arcs to satisfy the demand of locomotives. Next, we explain how the time-space network incorporates the cyclicity requirements.

Cyclic arcs of locomotives:

The cyclic plan requires that for each locomotive starting from a station, a locomotive of the same type (with the same capacity) returns to that station at the end of the time horizon. The returned locomotive might be a different locomotive. However, its fuel level at the end of time horizon must be same as the initial fuel level of the locomotive that has started from



(a) Network of the design problem

(b) Network of the flow problem

Figure 3.8: The problem as two interrelated network problems: a network design and a network flow problem. This figure corresponds to the illustrated example in Figure 3.7 with fuel supply layer ignored. The design problem is to select a subset of arcs that translates to yes/no decisions, such as an assignment or performing a refueling operation. The flow problem is to determine the fuel flow on the selected arcs by the design problem.

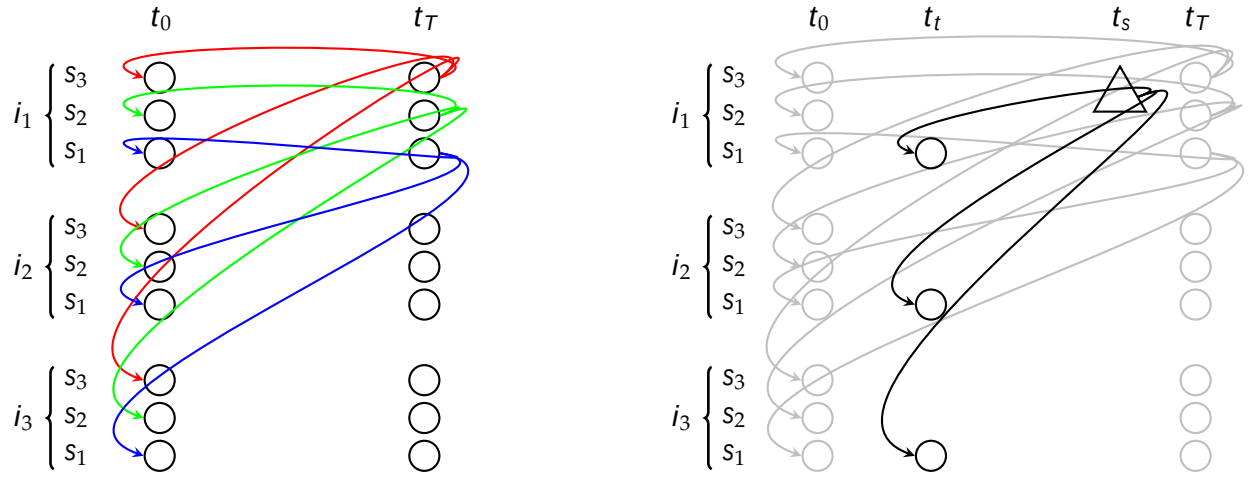
there.Suppose that the fuel level of locomotive l at the end of time horizon must be same as the fuel level of locomotive l^0 at the start of time horizon. This implies that the last node of locomotive l must be linked to the initial node of locomotive l^0 by a tank arc. Such arcs for all locomotives can handle the cyclic flow of the fuel in the locomotives tanks.

Cyclic arcs of inline tanks:

Since the refueling plan must be cyclic, the fuel level of the inline tanks should be also cyclic. On the other hand, the final location of the inline tanks might not be same as their initial location. Therefore, similar to locomotives, the fuel level of an inline tank at a station at the end of the time horizon should be equal to the the fuel level of the inline tank that has been at that station initially. For the locomotives, we could determine the pairs of the locomotives that are related to cyclic fuel level. However, since the inline tank routes are not predetermined, these pairs for inline tanks should be determined by the mathematical model. In order to incorporate the cyclic flow of the tanks, it is necessary to connect the final node of a station of a tank to its pair with an arc. However, since the final location and also the pairs of the inline tanks are not predetermined, all the possible combinations should be connected. The final node of an inline tank, corresponding to station s, should be connected to the initial nodes of all inline tanks that correspond to the station s. An

example is illustrated in Figure 3.9a with three stations and three inline tanks. In this figure, only the outgoing cyclic arcs of inline tank i_1 are shown. However, the cyclic arcs for the other inline tanks would be the same. Moreover, arcs with the same color are related to the same station.

In addition to the cyclic arcs between the station nodes, the cyclic arcs between a trip leg node which arrives at its destination after the end time of time horizon and the station nodes should be considered. Suppose that the first time period of the time horizon is t_0. Moreover, the time horizon ends at time T, and a trip leg arrives at its destination at time $T + t$. In such case, the outgoing arcs from the trip leg node should be connected to the corresponding station node of time period t_t instead of time period t_{T+t}. Although the number of outgoing tank arcs from the trip leg node is same as other trip leg nodes, we consider these arcs as cyclic arcs since the continuation of the route is in the next cycle and can be done by another inline tank. An example is illustrated in Figure 3.9b. The aforementioned two arc types are called as *Cyclic Arcs* from here on.



(a) Cyclic arcs from the station nodes. The last and the first nodes corresponding to the same stations are connected to capture the cyclicity of fuel plans. Cyclic arcs are shown only for inline tank i_1. However, they are similarly connected for other inline tanks.

(b) Cyclic arcs from a trip leg node. The trip leg, represented by the triangle, is a trip that happens at time period t_s. However, it arrives at s_1 at time $T + t$ which is after the time horizon end. Therefore, it is connected to stations nodes representing time period t_t to capture the cyclicity of fuel plans.

Figure 3.9: Cyclic arcs for inline tanks. In these figures, it is assumed that the first and the last time periods of the time horizon are t_0 and t_T, respectively. Other nodes and arcs within the horizon are not shown and are as shown in previous figures.

3.4.2 Mixed-Integer Program

In this section, we present a mixed integer program for GFMIR. This is based on the network definitions provided in Section 3.4.1. Given the locomotive routes and train schedules, fuel consumption of trip legs, fuel prices at different stations, and capacities of inline and locomotives tanks, this model gives the optimal fuel station locations, fuel plans of the locomotives and the inline tanks, and the routes and the assignment of the inline tanks. The following notation is adopted for the modeling. The capacity of the arcs, corresponding to inline tank, is equal to the inline tank capacity. Similarly, the arcs that are related to locomotives have the capacity equal to their tanks capacity. The station opening arcs has a capacity of the total fuel consumption of the scheduled trains.

Sets and indices

S	Set of stations,
I	Set of inline tanks,
T	Set of train trip legs,
V	Set of nodes of the network,
V^{SI}	Set of station nodes that are related to inline tanks,
V^{TI}	Set of trip leg nodes that are related to inline tanks,
V^{QI}	Set of inline trip leg nodes which have outgoing cyclic arcs,
V^{S}	Set of refueling-allowed nodes of stations,
V^{T}	Set of train trip leg nodes,
A	Set of arcs of the network,
$\hat{A}_v$	Set of outgoing arcs of node v,
$\check{A}_v$	Set of incoming arcs of node v,
$\hat{A}^K_v$	Set of outgoing tank arcs of node v,
$\check{A}^K_v$	Set of incoming tank arcs of node v,
A^Q_i	Set of cyclic arcs that terminate to a station node of inline tank i,
A^D	Set of design arcs,
A^C_v	Set of outgoing inline refueling arcs of node v which connect inline tanks to locomotives,
A^R_s	Set of refueling arcs that correspond to station s,
$\check{a}^K_{iv}$	Incoming tank arc to node v which is related to inline tank i,
$\hat{a}^K_{iv}$	Outgoing tank arc from node v which is related to inline tank i,
a^R_v	Refueling arc of node v,
a^O_s	Arc that corresponds to opening station s during the time horizon.

Parameters

C_s^O Fixed cost of operating station s during the time horizon ($ per entire time horizon per station),
C_v^H Stop cost at the corresponding station of node v ($ per refueling stop),
C_v^F Fuel price at the corresponding station of node v ($ per unit of fuel),
P_a Capacity of the fuel flow on arc a (unit of fuel),
D_v Fuel demand of node v (unit of fuel),
S_v Fuel supply of node v (unit of fuel).

Decision variables

u_a The amount of fuel on arc a (unit of fuel);
x_a 1, if arc a is used, 0 otherwise;

Set of design arcs A^D includes all arcs for which a binary decision variable x_a is defined. The arcs related to the routes of the locomotives are not included in this set because they are predetermined. The objective function (3.1) minimizes the total refueling costs and contains three parts. The first term is the total fuel purchasing costs. The second term is related to the stop costs, and the last part states the total fixed costs of operating the stations.

$$Min \quad \sum_{v \in V^S} C_v^F u_{a_v^R} + \sum_{v \in V^S} C_v^H x_{a_v^R} + \sum_{s \in S} C_s^O x_{a_s^O} \tag{3.1}$$

The constraints set is composed of three different parts. The first part, is related to the network design, is as follows. This part of the model includes only the binary variables.

$$\sum_{a \in \hat{A}_v^K} x_a = \sum_{a \in \check{A}_v^K} x_a \qquad \forall v \in V^{SI} \cup V^{QI} \tag{3.2}$$

$$x_{\hat{a}_{iv}^K} = x_{\check{a}_{iv}^K} \qquad \forall v \in V^{TI} \cap V^{QI}, i \in I \tag{3.3}$$

$$\sum_{a \in A_i^Q} x_a = 1 \qquad \forall i \in I \tag{3.4}$$

$$x_a \quad \sum_{a' \in \check{A}_v^K} x_{a'} \qquad \forall v \in V^{TI}, a \in A_v^C \tag{3.5}$$

$$x_a \quad x_{a_s^O} \qquad \forall s \in S, a \in A_s^R \tag{3.6}$$

$$x_a \in \{0,1\} \qquad \forall a \in A^D \tag{3.7}$$

Constraints (3.2) to (3.4) assure the flow conservation and the cyclic plan of the inline tanks. Particularly, Equation (3.2) assures that if an inline tank enters the station node v, it will leave the node. Similarly, Equation (3.3) ensures the flow conservation of the tanks in the

trip nodes of the inline tanks. Note that the trip nodes that have outgoing cyclic arcs have been incorporated in Constraint (3.2) since at these stations it is not required to find the route of the tanks and only the flow conservation must be respected. Equation (3.4) assures that each tank is assigned to exactly one station for the next cycle. Constraint (3.5) determines which locomotives use the assigned inline tanks. The locomotive can use fuel of the inline tank, only if the tank is assigned to the corresponding train. Finally, Constraint (3.6) states that we are allowed to refuel only in the stations that are open during the time horizon. Constraint (3.7) defines the binary decision variables.

The second part of the mathematical formulation is related to the flow continuous variables and the flow conservation constraint. The set of constraints are as follows:

$$\sum_{a2\hat{A}_v} u_a = \sum_{a2\check{A}_v} u_a \quad D_v + S_v \qquad 8v\ 2\ V \tag{3.8}$$

$$0 \quad u_a \quad P_a \qquad 8a\ 2\ A \tag{3.9}$$

Equation (3.8) represents the flow conservation constraints for all nodes. This constraint is applied for all types of the nodes. Each node has its own incoming and outgoing arc types. For instance, an inline trip leg node has several incoming and outgoing tank arcs and several outgoing inline refueling arcs. While, a locomotive trip leg node has an incoming tank arc, an outgoing tank arc, an incoming inline refueling arc, and a demand. Moreover, the node demand D_v is nonzero for only the locomotives trip leg nodes, and the node supply S_v is nonzero only for the total supply node and is equal to the total fuel consumption of all trains. The decision variable u_a is bounded in Constraint (3.9).

The third and the last part of the model contains the capacity constraint (3.10) that links the binary and the continuous variables, which assures that flow can take a value greater than zero only if the corresponding arc is selected, and it is less than its capacity.

$$u_a \quad P_a x_a \qquad 8a\ 2\ A^D \tag{3.10}$$

3.5 Extensions: The Locomotive Fuel Management Problem with Inline Refueling

The *GFMIR* model proposed in Section 3.4 presents a general version of the problem which considers the routing and cyclic scheduling constrains of using inline tanks as well as generic restrictions of the refueling operation. However, from an operational point of view, the solution of the general version of the problem might not be practicable in real railroad applications, since it ignores some operational issues. In this section, some operational

issues in the railroad industry are stated. These issues lead to a number of additional constraints that need to be added to the model described above. The *Locomotive Fuel Management problem with Inline Refueling*, referred to as *LFMIR*, consists of the GFMIR together with the additional constraints presented in this section.

3.5.1 Maximum Number of Assigned Inline Tanks to a Train

In practice, the total number of inline tanks assigned to a train must not be greater than a specified number M. In real railroad cases, at most two inline tanks can be connected to trains $M = 2$. Thus, Constraint (3.11) should be added to the formulation. This constraint ensures that at most two tanks are assigned to a train trip leg.

$$\sum_{a \in \check{A}^K_v} x_a \quad M \qquad \forall v \in V^{TI} \tag{3.11}$$

3.5.2 Maximum Number of the Locomotives Refueled Inline

Inline tanks are connected to a train by being mounted on a wagon. The train may be hauled by one or several locomotives. One practical constraint is that the total number of the locomotives that can be connected to the wagon, which inline tanks are mounted on, is limited to N. Constraint (3.12) incorporates this restriction. In real railroad cases, at most four locomotives can be connected to and refueled by the assigned inline tanks.

$$\sum_{a \in A^C_v} x_a \quad N \qquad \forall v \in V^{TI} \tag{3.12}$$

3.5.3 Fuel Transfer Between the Tanks

In practice, no fuel can be transferred between inline tanks. However, this happens in the optimal solution of GFMIR. Thus, Constraint (3.13) should be added to the formulation. This ensures that the fuel level of the inline tank is not increased after making the trip which prevents the fuel transfer between the tanks.

$$u_{a^K_{iv}} \quad u_{\grave{a}^K_{iv}} \qquad \forall v \in V^{TI}, i \in I \tag{3.13}$$

Adding Constraint (3.13) is not enough to prevent the fuel transfer because the indices of the inline tanks might be changed at the trip nodes that have outgoing cyclic arcs. Therefore, Constraint (3.13) is not valid for such nodes. In order to solve this issue, the network representation should be modified as follows. Instead of connecting the cyclic arcs from a

trip node to station nodes, an intermediate node per each inline tank should be added (see Figure 3.10).

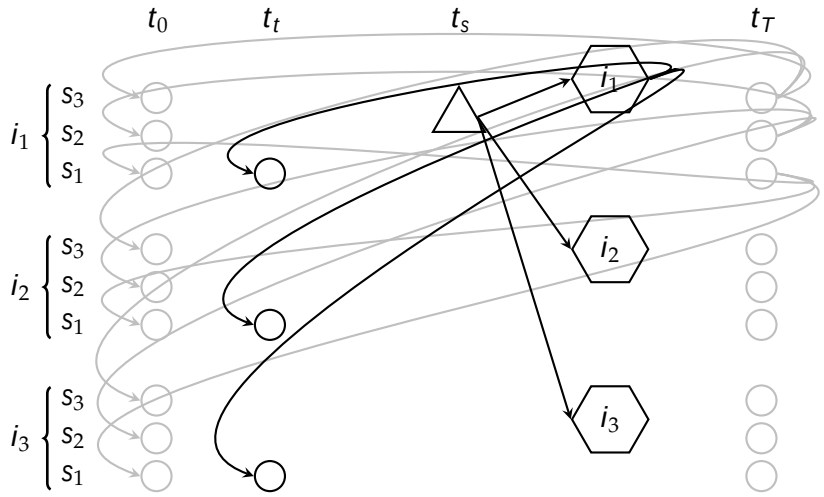


Figure 3.10: Intermediate nodes for cyclic arcs from a trip leg node. By network configuration as Figure 3.9b, outgoing cyclic arcs from trip leg nodes are not related to a specific inline tank. Hence, fuel level increase cannot be tracked on such arcs. Intermediate hexagon nodes in this figure are added to have outgoing arcs from the trip leg node, each corresponding to a specific inline tank. Then, the outgoing arcs from the intermediate nodes can be viewed as cyclic arcs. In this figure, outgoing arcs for intermediate node of inline tank i_1 are shown. However, cyclic arcs of other inline tanks similarly connect the corresponding intermediate node to station nodes of station s_1 at time period t_t.

In Figure 3.10, a trip leg node, which arrives to its destination after time horizon, is shown. Instead of connecting the outgoing arcs of the trip leg node to the stations, the outgoing arc of each inline tank is connected to its corresponding intermediate node (hexagon nodes). Subsequently, since it is possible that fuel level of each inline tank be equal to fuel level of another inline tank at next cycle, there are $j \mid j$ outgoing arcs from each intermediate node which connect it to the corresponding station node of all tanks. Figure 3.10 illustrates the intermediate nodes for the trip node of Figure 3.9b and its corresponding arcs. Finally, this trip leg node will be treated same as other trip nodes. Moreover, Constraint (3.2) should be applied for the intermediate nodes, and the arcs between the intermediate nodes and stations are of cyclic arc type.

3.5.4 Balanced Fuel Level of Inline Tanks

If two tanks are assigned to a train at the same time, the difference of their fuel levels after the trip must be within a specified tolerance. This is required to have balanced load in the

inline tanks to prevent potential derailments. Moreover, the pumping system consumes fuel from the tanks in a manner that their fuel levels remain balanced. Let s be the allowed tolerance of the fuel levels. Constraint (3.14) ensures that if two tanks are assigned to a trip leg at the same time, their final fuel level will be balanced within the tolerance. Moreover, if $s = 0$, the condition $(i \neq j)$ should be changed to $(i < j)$.

$$u_{\hat{a}^K_{iv}} \quad u_{\hat{a}^K_{jv}} \quad P_{\hat{a}^K_{iv}}(2 \quad x_{\hat{a}^K_{iv}} \quad x_{\hat{a}^K_{jv}}) + s \qquad 8v \in V^{TI}, i, j \in I j (i \neq j) \tag{3.14}$$

3.6 An Australian Case Study

In this section, the proposed model is applied to an Australian case study and analysed in terms of the effectiveness of the inline refuelling wagons as a means to saving costs. In the next section this data set and a set of standard data sets from the USA are used to test the computational effectiveness of solving these instances with a MIP solver. The Australian instance is based on a real schedule of a fleet of trains. The goal is to evaluate the adoption of the inline refueling technology on the ongoing operation of the respective railroad company. The Australian rail network is an example of a sparse network with long distances between the origins and destinations. Both GFMIR and LFMIR models and their restricted versions are studied in this section. The summary of the considered problems in this section is shown in Table 3.1. Furthermore, some managerial insights and sensitivity analyses are provided for the case studies. The focus of the computational experiments in this chapter is purely on the new element introduced in this thesis, the use of inline refueling. All computational experiments are run on a cluster with 4 Xeon-E5-2667-v3 cores and 32 GB RAM using CPLEX 12.8.0 via a Python API.

Table 3.1: Summary of the considered problems in the computational experiments

Problem	Description
GFMIR	A variation of the INFORMS Competition Problem (2010) to which the planning of inline tanks is added
RGFMIR	The restricted problem of GFMIR in which the decisions related to selecting the fuel stations locations and the decisions related to scheduling the refueling operations are ignored (defined in Corollary 3.3.4 as RGFMIR2)
LFMIR	GFMIR with the extensions introduced in Section 3.5
RLFMIR	RGFMIR with the extensions introduced in Section 3.5

The Australian case study consists of 115 schedules of trains which travel between 8 stations during one week. Each train schedule between an origin and a destination is

composed of one or several trip legs that pass through the intermediate stations. The considered case includes 218 trip legs. Fuel prices at stations ranges from \$1.0742 to \$1.3674 per liter. Trains are hauled by one or several locomotives. There are a total of 133 locomotives. In order to have a comparison, a *Basic Strategy*, which mimics the current practice of the company, is defined. In the Basic Strategy, locomotives are refueled at each station with just enough fuel to reach the next station on their trip. Employing Basic Strategy on the trains and locomotives schedules of the Australia case results in a weekly total cost of \$2,895,616.

In this section, first, proposed models are compared on the Australia case. Then, the optimal size of the inline tank fleet is investigated. Based on the optimal size of the inline tank fleet, sensitivity analyses and managerial insights are provided.

3.6.1 Evaluating GFMIR, LFMIR, and the Restricted Models

We evaluate the proposed models on the Australia case by considering different inline tank fleet sizes. In particular, we show that the rail models provide feasible solutions for the real application in the railroad industry at the little expense of complexity in comparison with the general models. The summary of the results are presented in Table 3.2. Computing time is limited to 12 hours for all experiments. In this table, the number of variables, binaries, constraints, and nonzeros are reported based on the reduced MIP of CPLEX.

Table 3.2: Evaluating different models on the Australia case

Fleet size	Model	Total costs (\$)	Optimality gap (%)	Columns	Binaries	Rows	nonzeros	Root relaxation time (sec)
0	RGFMIR & RLFMIR	2,771,871	0.00	630	0	300	1,259	0.00
	GFMIR & LFMIR	2,929,409	0.00	2,138	843	1,327	4,398	0.01
5	RGFMIR	2,754,200	0.03	11,366	4,234	10,060	36,805	0.49
	RLFMIR	2,754,226	0.03	11,382	4,234	16,026	58,038	0.41
	GFMIR	2,863,528	0.27	13,824	6,312	14,206	46,141	0.42
	LFMIR	2,864,657	0.33	13,939	6,312	20,187	67,298	0.93
10	RGFMIR	2,747,222	0.01	22,986	9,454	19,462	79,701	0.72
	RLFMIR	2,747,858	0.02	23,212	9,454	41,971	164,558	1.49
	GFMIR	2,842,364	0.97	26,634	12,622	25,783	93,384	1.19
	LFMIR	2,845,254	1.19	26,859	12,622	48,312	178,178	1.50

The solutions in Table 3.2 are not optimal for any of the models with different inline tank fleet sizes except when there are no inline tanks. For these cases, optimal solutions are obtained in less than 1 second. As is shown in Table 3.2, using RLFMIR and LFMIR instead of RGFMIR and GFMIR brings trivial additional costs and complexity. On the other hand, the solutions of RGFMIR and GFMIR include several locomotive-refueling infeasibilities. For instance, fuel is transferred between inline tanks in the solution of RGFMIR and GFMIR

several times. This cannot happen in the real locomotive fuel management. Moreover, optimality gaps of the obtained solutions by CPLEX for RLFMIR are small. Therefore, RLFMIR is used in our analysis to provide business insights. All the results in the rest of this section are based on the solutions with a small optimality gap (at most 0.05%).

3.6.2 Optimizing the Size of the Inline Tank Fleet

The railroad company that this case emanates from is in the process of trialling this new technology with an intention of phasing in the adoption of this technology over the next few years. Therefore, in this section, we first demonstrate that the proposed model results in significant cost-saving in comparison with the 'Basic Strategy'. Furthermore, marginal weekly savings of inline tanks are provided, which assists decision-makers to determine the inline tank fleet size. As mentioned previously, RLFMIR is employed for analyzing.

Table 3.3 provides weekly total costs, potential weekly cost-saving, and net cost-saving for different strategies. Net weekly cost-saving is obtained by subtracting the weekly amortized cost of the employed inline tanks from the weekly savings. The Basic Strategy resembles the current practice of the company. The other strategies are employing RLFMIR with the different inline tank fleet sizes. Based on Table 3.3, with no inline tanks, RLFMIR brings significant cost-savings. Moreover, adding inline tanks still contribute to additional savings. In particular, with fleet size of 10, it brings $\$147,758$ weekly cost-saving ($\$144,488$ net weekly cost-saving), which $\$24,014$ ($\$20,294$) of it is direct consequence of using 10 inline tanks through RLFMIR.

Table 3.3: Total fuel costs per different inline tank fleet sizes and potential cost-savings. Net weekly cost-savings are obtained by subtracting the weekly amortized cost of the employed inline tanks from the weekly cost-savings.

Strategy	Fleet size	Weekly total fuel costs	Weekly cost-saving	Net weekly cost-saving
Basic Strategy	-	$2,895,616	-	-
RLFMIR	0	$2,771,871	$123,744	$123,744
	5	$2,754,226	$141,390	$139,755
	10	$2,747,858	$147,758	$144,488
	15	$2,747,071	$148,545	$143,640

To determine the number of inline tanks to purchase, we compute the net marginal revenue of adding each inline tank. Considering the weekly amortized cost of each inline tank as $327, we repeatedly solve RLFMIR by increasing the fleet size from 0 to 15 inline tanks and report the net marginal weekly cost-saving at each step. Figure 3.11 shows the net marginal revenue of adding the w^{th} inline tank. As shown in the Figure 3.11, the optimal fleet size is 10, as employing more inline tanks does not contribute to actual savings. Any

actual capital investment decision should of course be tested on a variety of potential future schedules, however the results give an indication that the use of multiple inline tanks for the types of operations seen in Australia with large distances between stations.

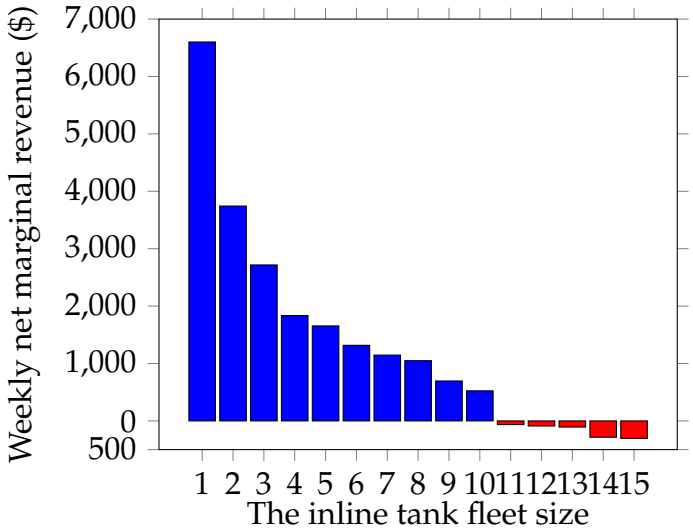


Figure 3.11: The diminishing marginal revenue of the inline tanks

3.6.3 Safety Inventory

A common practice in transportation companies is to always preserve a safety inventory of fuel to avoid running out of fuel due to the unpredictable increases in the fuel consumption. Studying the effects of safety level is analogous to sensitivity analysis on the tank capacity of locomotives. In this section, we examine different scenarios in which 0, 5, 15, and 20 percentage of the locomotives' tanks are considered as safety level. Figure 3.12 shows the total weekly fuel costs per different safety inventories where each line represents different inline tank fleet size. According to the results, apart from the fact that RLFMIR decreases the costs at the different safety levels, it reduces the cost of imposing a robustness constraint in the form of safety inventory. As the safety level increases, the total fuel costs increases. However, with more inline tanks, the slope of cost increase reduces. For the fleet size of 0, the weekly total costs increase by $19,191 when the safety level increases from 0% to 20% of tank capacity. With 10 available inline tanks, the total costs increase by $3,377 over the same increase in the safety level. This is because inline tanks increase the flexibility of refueling operations, which can benefit the company in situations where there is a restriction such as safety fuel level and smaller locomotives' tank capacity.

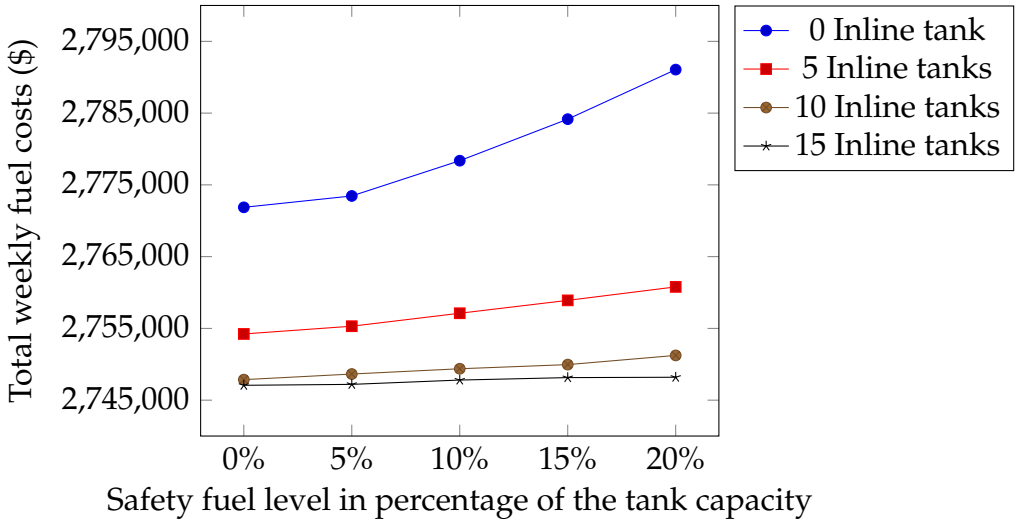


Figure 3.12: Total fuel costs per different safety levels considering different fleet sizes

3.6.4 Critical Paths

The case of Australia includes 218 trip legs, however, they repeat on 16 paths. From the transportation cost point of view, each path has a different importance since the fuel prices vary at the stations. Therefore, providing the cost incurred by an increase in the fuel consumption on the paths can benefit the locomotive drivers and the operators to identify critical paths and exert fuel-efficient behaviors to avoid consumption increase on the critical paths. Each constraint in (3.8) that is related to a locomotive trip node corresponds to the fuel consumption during a trip leg. Hence the dual variables of such constraints represent the cost of one liter fuel consumption increase on the corresponding path. In this section, we consider two scenarios where there are no inline tanks and 10 inline tanks available. RLFMIR with no inline tanks available does not include any binary variables and the dual variables can be obtained easily. However, with 10 inline tanks, it is a MIP and the dual variables cannot be obtained as the case of the fleet size of 0. In order to acquire the dual variables for RLFMIR with 10 inline tanks, we fix the binary variables based on the final solution to obtain a linear program (LP). However, the dual variables of the obtained LP are not the dual variables of the original MIP. We claim that they are an upper bound for the dual variables of the original MIP. This is because the problem stays feasible after one liter fuel consumption increase on a path. Therefore, the obtained LP is equivalent to the problem of fuel planning while considering a fixed plan for the inline tanks. Therefore, the dual variable of the corresponding constraint of the obtained LP represents the shadow price of one liter fuel consumption increase on the corresponding path while a fixed plan is considered for the inline tanks. However, the total costs might be decreased additionally by

improving inline tanks' plans. Hence, the actual cost of one liter fuel consumption increase on a trip leg is at most equal to the dual variable of the corresponding constraint of the optimal solution of the obtained LP.

The cost of fuel consumption increase of all trips for both fleet sizes is determined as mentioned, and the average cost of one liter fuel consumption increase on each path is reported in Figure 3.13. For the fleet size of 0, the four paths with the highest costs, P5,

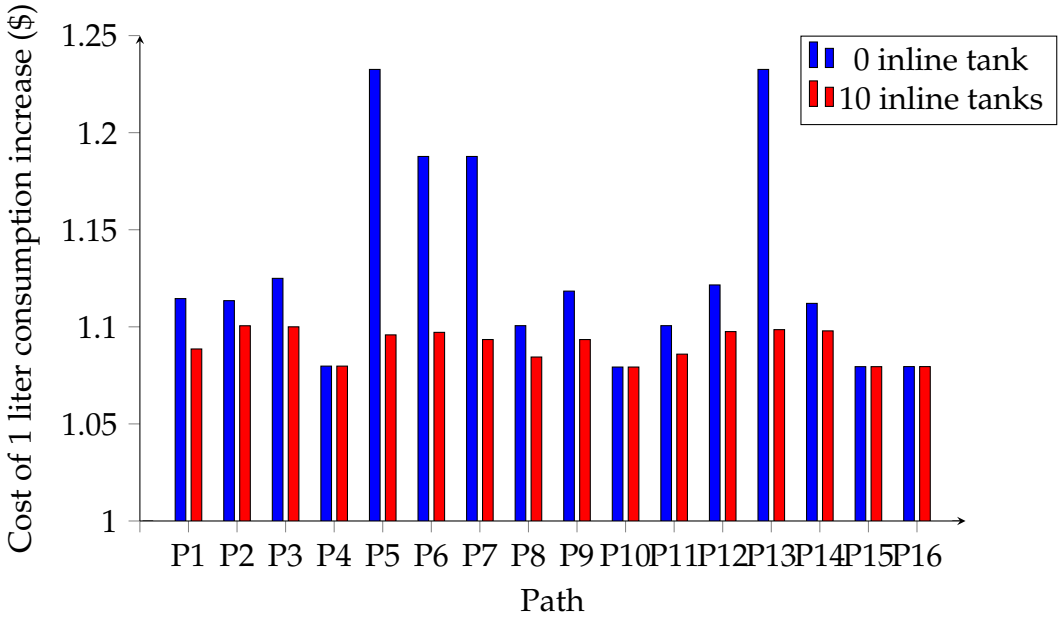


Figure 3.13: Average cost of 1 liter fuel consumption increase for different paths

P6, P7, and P13, are related to the paths that originate from or terminate to the station *Cook*, which for fuel is the most expensive station. However, the fuel price at the origin and destination is not the only decisive factor since the least-costly path is the path P10, which corresponds to path between *Sydney* and *Melbourne* while neither of them is the cheapest station. In fact, the fuel prices at the neighbor stations of the origin and the destination of a path also affect it to be a critical path.

As is shown in Figure 3.13, RLFMIR with the fleet size of 10 reduces the cost of the consumption increase on 12 paths. For other 4 paths, P4, P10, P15, and P16, the cost of the consumption increase is not changed by adding 10 inline tanks. This is because inline refueling is not performed on these paths during the time horizon according to the solution of RLFMIR with 10 inline tanks. Table 3.4 presents the number of the assignments of inline tanks to the trips on each path. Although inline tanks are assigned once to a trip on path P16, it is not used for inline refueling, but only to relocate the tank. According to Table 3.4, no inline tank is assigned to trips on path P14. However, the cost of the consumption increase

on this path with 10 inline tanks is reduced in comparison with no inline tanks. This is because inline tanks are utilized on neighbor stations as we mentioned before. The cost reduction is particularly significant for the four most-costly paths of 0 fleet size scenario. This is because we have to refuel locomotives at Cook, the most expensive station, based on results of RLFMIR with fleet size of 0. However, results of RLFMIR with fleet size of 10 suggest all locomotives can pass the Cook station without the need to refuel. The improvement on the four most-costly paths are 11.09% on P5, 10.87% on P13, 7.94% on P7, and 7.63% on P6. On the average, RLFMIR with fleet size of 10 improves the cost of consumption increase 3.56% in comparison with the fleet size of 0. These results imply that in addition to cost-savings, RLFMIR is able to improve the robustness of the railroad companies against the unpredictable events.

Table 3.4: The number of inline tanks assignment to paths during time horizon

Path	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number of assignments	3	4	1	0	21	11	21	24	4	0	24	4	11	0	0	1

3.7 Computational Performance

In this section we focus on the computational effectiveness of obtaining solutions to this problem using an integer programming solver. The second data set we use in this section is from the INFORMS Problem Solving Competition (2010) that is focused on the locomotive fuel management problem without inline refueling. Unlike the Australian instance, the INFORMS data set is based on a simplification of the real operations to test the ability of the model to scale-up. Particularly the INFORMS case study that contains larger instances is very challenging. Since the MIP of the large instances proves intractable for CPLEX, we propose a heuristic to tackle such instances.

The case study of INFORMS includes 2996 train schedules that includes 5264 trip legs. The trains are hauled by 214 identical locomotives between 73 stations during a two-week time horizon. It is assumed each train is hauled by exactly one locomotive. The fuel prices vary between $2.90 and $3.56 per gallon. The train schedules are stated in days. However, this time unit is not useful to plan inline tanks since inline tanks may be assigned to trains that are running at the same time on a day. Hence, the time unit must be transformed to an appropriate time unit (hours in this case).

In order to fill in the missing time information, there are restrictions to ensure that the transformed arrival and departure times are consistent with the original times and also the trips' precedences are respected. However, the choice of times affects the number of

the opportunities where an inline tank can be switched between two trains, which subsequently has an impact on the total potential cost-savings. Inline tank switching is possible whenever a train arrives to a station before the next train departs the same station. Thus the selection of times affects the switch opportunity between two trains that arrive at and depart from the same station on the same day. The other opportunities for switching tanks are implied by the original times in days. Since each locomotive's trips are interconnected, considering a switch opportunity between two particular trips might destroy the switch opportunities for the subsequent or previous trips. Therefore, an optimization model is employed to select the times. We consider two extreme cases in which the switch opportunities are minimized and are maximized which are called *Minimal Connectivity* and *Maximal Connectivity*, respectively. The number of possible switches, reported as *Connectivity* of an instance, includes the possibility of switching an inline tank between two trains on the same day, whether they are hauled by the same or the different locomotives.

For the INFORMS data set, CPLEX is not able to provide satisfactory solutions in reasonable computing time even for RLFMIR. Therefore, we divide the original INFORMS data set into eight separate instances based on the locomotives included in each. We separately evaluate the benefits of the proposed models on the divided instances of the INFORMS case.

Similar to the Australia case, RLFMIR is considered for the INFORMS case to provide economic analyses. The details of the instances and the corresponding results of RLFMIR are reported in Table 3.5. The optimal fleet size of 10 for these instances is obtained by a similar approach as described for the Australia case in Section 3.6. In the first part of the table, the details of the instances are described. The row *Trip legs* shows the total number of the trip legs of the trains of the instances. The row *Stations* presents the number of the distinct stations that all locomotives pass during the time horizon. The average of the fuel prices at corresponding distinct stations to each instance is reported in the row *Fuel price average*. The row *Total consumption* presents the total fuel consumption of the locomotives during the time horizon in gallons. We generate both minimal and maximal connectivity cases for each instance and the number of connectivity of each case is reported in the row *Connectivity*. The second part of Table 3.5 reports the total costs of RLFMIR by considering both connectivity cases. When there is no inline tank available, the connectivity does not affect the solution. Therefore, for fleet size of 0, only one case is reported. The results of RLFMIR with no inline tanks are optimal while the solutions of RLFMIR with fleet size of 10 for all instances are not optimal but with less than 0.52% optimality gaps in 4 hours of computing time.

As is shown in Table 3.5, significant cost savings are achievable by using RLFMIR with 10 inline tanks. This demonstrates the effectiveness of RLFMIR on the different instances.

On average, RLFMIR with fleet size of 10 contributes $32,243 of fortnightly cost-savings in comparison with the fleet size of 0, which implies 98.5% rate of return on investing on 10 inline tanks. Furthermore, the least cost saving is $15,218 which related to the minimal connectivity case of the instance 4. However, even for this instance the rate of return is 45.4%. These benefits are in comparison with RLFMIR with 0 tank available, which is the current modeling tool in the literature. It is obvious that the minimal case always has higher total costs in comparison with the maximal case of the same instance since there are more opportunities to utilize the inline tanks in the maximal case.

Table 3.5: Evaluating RLFMIR on instances of INFORMS

Instance		1	2	3	4	5	6	7	8
Locomotives		27	27	27	27	27	27	27	25
Trip legs		1,071	903	518	574	476	448	658	616
Stations		37	23	20	22	20	14	28	25
Fuel price average		3.159	3.143	3.103	3.095	3.096	3.110	3.196	3.107
Total consumption		715,939	614,999	346,430	402,829	302,379	317,030	478,828	417,088
Connectivity	Min	1,987	2,177	487	913	593	539	890	824
	Max	2,805	3,292	1,102	1,595	1,157	1,488	1,297	1,258
0 tanks		2,156,015	1,849,770	1,031,585	1,186,023	909,403	988,833	1,480,914	1,260,578
10 tanks	Min	2,119,094	1,805,035	1,012,456	1,170,805	891,625	943,922	1,454,347	1,231,754
	Max	2,101,406	1,797,314	1,005,648	1,169,801	890,322	940,133	1,446,420	1,230,272

Figure 3.14 depicts the percentage of the improvement of RLFMIR with fleet size of 10 in comparison with the fleet size of 0 on the instances of Table 3.5 with respect to the cost of one unit of consumed fuel in the optimal solution of RLFMIR with fleet size of 0. The cost of one unit of consumed fuel is obtained by dividing the total costs in the optimal solution of RLFMIR by the total fuel consumption. The label of each point represents the corresponding instance. Apart from the connectivity, other characteristics of an instance affect the potential cost-savings of using inline tanks. As is shown in Figure 3.14, except from the instance 7, there is an implied relation between the percentage of cost improvement and the cost of one unit of consumed fuel of RLFMIR with fleet size of 0. The effectiveness of employing inline tanks, particularly increases as the cost of one unit fuel consumption in the optimal solution of RLFMIR with fleet size of 0 increases. The average of the fuel prices of the corresponding stations of the instance 7 is higher than the other instances. Hence, there are less opportunities to buy fuel from the inexpensive stations to tanker and transfer by the inline tanks.

3.7.1 A Simple and Efficient Heuristic

To apply the LFMIR model to large industrial cases, practitioners need to obtain good-quality solutions in short computing times. This accelerates scenario analysis which bene-

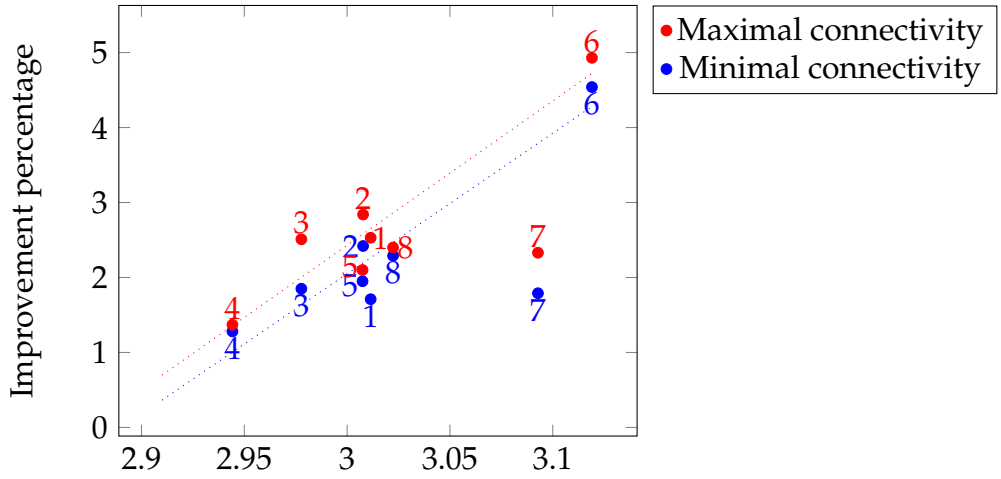


Figure 3.14: Improvement percentage of RLFMIR with fleet size of 10 in comparison with the fleet size of 0 on the instances of Table 3.5. The improvement percentage is shown with respect to the average cost of one unit of consumed fuel in the optimal solution of RLFMIR with fleet size of 0. The average cost is obtained by dividing the optimal cost of RLFMIR with 0 inline tank by the total fuel consumption.

fits the decision makers and also facilitates the further extensions of the LFMIR models to include additional practical assumptions. Particularly, since solving industrial cases using exact methods usually takes more than 10 hours, it is unrealistic to be able to solve more complicated versions of the problem or to try multiple scenarios without the use of heuristics. Hence, the development of efficient heuristic algorithms plays an essential role for this class of problems.

Numerical experiments we carry out in this section show a surprising trend in the solution by CPLEX as the size of the problem grows. As the number of available inline tank increases, the optimal cost must decrease. However, because the complexity of the problem also increases, CPLEX fails to find these better solutions, and in fact the quality of the solutions found by CPLEX degrades. Therefore we have developed a heuristic to tackle the large instances. The main computational difficulties arise from the time-consuming LP relaxations of the large instances, and, we suspect, from the symmetry between inline tanks.

Although the problem is strongly NP-hard, CPLEX is able to provide close-to-optimal solutions in a reasonable time for two variants of the problem even for the large instances: (a) RLFMIR with one inline tank available, and (b) LFMIR with no inline tank available. For instance, for LFMIR with 0 tanks, a solution with 0.06% optimality gap is found within 10 minutes. This solution is the same as the best-known solution for the INFORMS Problem Solving Competition (2010). The heuristic we propose iteratively solves one of these two

variants at each step.

As a baseline, we first implement a **Greedy Algorithm** (**GA**). At every iteration, GA myopically optimizes the route and fuel plan of one inline tank by solving a RLFMIR with one inline tank available. Its route and fuel plan are fixed in subsequent iterations, and the demand of the locomotives that receive fuel from this inline tank is updated accordingly. GA terminates when all inline tanks are routed. Finally, if we wish to further optimize the location and refueling scheduling decisions, we can solve LFMIR with fixed routes. This method performs quite poorly in some cases, hence we will not analyze in detail.

Our contribution is a **Semi-Greedy Algorithm** (**SGA**), that similarly optimizes the decisions of one inline tank at each step. However, after optimizing the route and fuel plan of an inline tank, in subsequent iterations only its route is fixed, but not its fuel plans. So, at iteration i, the routes of tanks $0, \ldots i \quad 1$ are fixed, the route of tank i is variable, and all fuel plans are variable. The rationale for this design decision is that the fuel plans are modeled with continuous variables, while the routes require discrete variables. As a result, the size of the MIP model increases as the algorithm progresses, but the difficulty of solving it with CPLEX does not change significantly as the number of discrete variables is constant at every iteration. Similar to GA, LFMIR with fixed routes for inline tanks is solved to optimize the location and refueling scheduling decisions. SGA is a form of diving heuristic (Berthold, 2006) in which branching is made to fix the path of one inline tank at a time.

GA and SGA are implemented in Python 3.6 with CPLEX 12.8.0. The computing time for each single-tank RLFMIR is limited to 5 minutes. In our experiments, CPLEX is not able to reach optimality within this time limit, but the solutions have small optimality gaps (less than 0.5%). If we additionally attempt to polish those solutions with LFMIR, this last step is limited to 10 minutes. The results are compared with the solutions provided for the MIP of RLFMIR and LFMIR by CPLEX in 12 hours of computing time limit. The summary of results is presented in Table 3.6 and Figure 3.15.

Three data sets are considered here: (1) the Australia case, (2) the complete INFORMS data set with the maximal connectivity (with connectivity of 46,954), and (3) the complete INFORMS data set with the minimal connectivity (with connectivity of 23,655). In Table 3.6, *Gap* shows the optimality gap, obtained as $\frac{Solution \quad LowerBound(LB)}{Solution} \quad 100$. Column *Imp.* shows the relative improvement of SGA over CPLEX (in %). Therefore, positive (negative) values imply that the solution by SGA is better (worse) than the solution by CPLEX. Computing times are given in seconds in the column *Time*. For all instances provided in Table 3.6, CPLEX and also all steps of the SGA reach the time limit.

The Australia case is a smaller data set, thus CPLEX is able to provide good-quality solutions for RLFMIR even with 20 inline tanks available. However, the optimality gap for

Table 3.6: Comparison between the results of CPLEX and SGA for RLFMIR and LFMIR over different instances

Data set	Problem	Fleet size	CPLEX			SGA			
			Solution ($)	Gap (%)	Time	Solution ($)	Gap (%)	Imp. (%)	Time
Australia	RLFMIR	5	2,754,226	0.03%	43,200	2,754,899	0.06%	-0.02%	1,500
		10	2,747,858	0.02%	43,200	2,748,019	0.03%	-0.01%	3,000
		15	2,747,071	0.01%	43,200	2,747,239	0.02%	-0.01%	4,500
		20	2,747,024	0.01%	43,200	2,746,881	0.01%	0.01%	6,000
	LFMIR	5	2,864,657	0.33%	43,200	2,889,426	1.19%	-0.86%	2,100
		10	2,845,254	1.19%	43,200	2,854,441	1.50%	-0.32%	3,600
		15	2,841,341	1.43%	43,200	2,837,235	1.29%	0.14%	5,100
		20	2,879,781	3.05%	43,200	2,829,420	1.32%	1.75%	6,600
INFORMS-Maximal connectivity	RLFMIR	5	10,651,524	0.16%	43,200	10,646,390	0.11%	0.05%	1,500
		10	11,318,609	6.71%	43,200	10,577,810	0.18%	6.54%	3,000
		15	11,318,609	7.06%	43,200	10,545,277	0.24%	6.83%	4,500
		20	11,318,609	7.22%	43,200	10,526,525	0.24%	7.00%	6,000
	LFMIR	5	11,400,377	4.10%	43,200	11,117,041	1.65%	2.49%	2,100
		10	11,777,571	8.42%	43,200	11,016,056	2.09%	6.47%	3,600
		15	11,777,126	8.98%	43,200	10,934,527	1.97%	7.15%	5,100
		20	11,781,455	9.32%	43,200	10,879,023	1.80%	7.66%	6,600
INFORMS-Minimal connectivity	RLFMIR	5	10,789,407	0.12%	43,200	10,785,550	0.08%	0.04%	1,500
		10	11,318,609	5.30%	43,200	10,741,822	0.21%	5.10%	3,000
		15	11,318,609	5.67%	43,200	10,706,063	0.28%	5.41%	4,500
		20	11,318,609	5.99%	43,200	10,685,521	0.42%	5.59%	6,000
	LFMIR	5	11,381,321	2.63%	43,200	11,321,142	2.11%	0.53%	2,100
		10	11,758,345	6.19%	43,200	11,281,141	2.22%	4.06%	3,600
		15	11,758,345	6.73%	43,200	11,224,272	2.29%	4.54%	5,100
		20	11,784,929	7.38%	43,200	11,204,890	2.59%	4.92%	6,600

LFMIR increases with the increase in the fleet size. The best solution costs found by CPLEX, GA, and SGA for varying fleet sizes in the Australia case is presented in Figures 3.15a and 3.15b. We know that the optimal cost decreases as the number of available inline tanks increases, and the solutions found by CPLEX and SGA for RLFMIR follow this trend, except for LFMIR, where the solutions found by CPLEX worsen for 15 tanks and more. On the other hand, GA and SGA continue to provide better solutions as the fleet size increases. Note that SGA is also significantly faster than CPLEX, as shown in Table 3.6.

Looking now at the larger INFORMS data set, SGA always provides better solutions than CPLEX, and much more efficiently. The optimality gap of the solutions by CPLEX for both RLFMIR and LFMIR is more than 5% for both connectivity cases, except for the cases with 5 inline tanks. However, SGA is able to provide solutions with small optimality gap (at most 0.42%) for RLFMIR for all fleet sizes tested. Moreover, the solutions by SGA for LFMIR are on average 4.73% better in comparison with the solutions by CPLEX. Importantly, the best solutions (w.r.t. cost) found by CPLEX are for 5 inline tanks. This means that CPLEX's

heuristics are unable to detect and exploit the fact that a solution with 5 tanks can readily be extended to arbitrarily many tanks at a cost no higher.

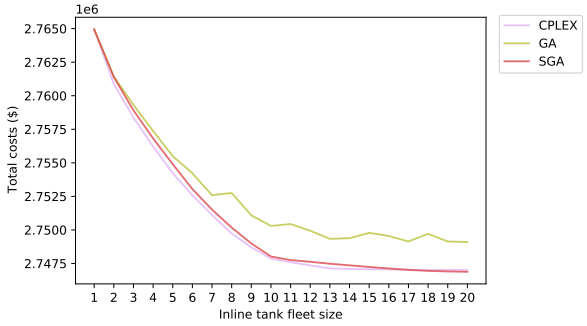
This phenomenon is illustrated in Figures 3.15c to 3.15f. The solution costs found by GA, SGA and CPLEX are plotted, together with two other values that help interpret the performance of CPLEX. The first one is *Basic strategy*, corresponding to heuristic fuel planning as explained in Section 3.6. In this heuristic, at each stop, locomotives are refueled with just enough fuel to reach the next station on their trip. The second one, reported as *no-tank*, is the total cost without inline tanks, corresponding to the optimal fuel purchases found using the best modeling approach currently available in the literature. Notice that for RLFMIR with more than 5 inline tanks, CPLEX almost always produces a solution with the same cost as the Basic Strategy. In contrast, GA and SGA keep finding solution with smaller costs as the fleet size increases, and SGA always produces better solution than GA. The conclusion of our numerical experiments is that SGA is currently the most time- and cost-efficient approach for large instances such as the INFORMS ones.

3.8 Conclusions and Future Research

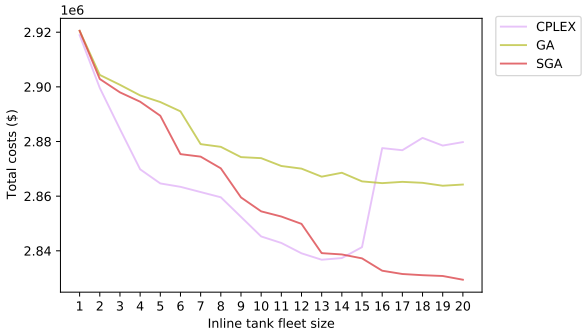
This thesis introduces a new class of models to study and solve existing fuel management problems in railways with the option of inline refueling. First, the problem and also its restricted version are proved to be strongly NP-Hard. The restricted problem includes only the decisions related to inline tanks and ignores the location and refueling scheduling decisions. We prove that the restricted problem is strongly NP-Hard even when only one inline tank is available. We propose a MIP model based on a time-space network with generic refueling constraints. As the focus of this chapter is on rail transport, we propose extensions to capture the operational constraints of inline refueling in the railroad application. The model can be utilized to determine fuel plans of locomotives and inline tanks, location of fuel stations, and assignment of inline tanks.

We computationally test the effectiveness of the proposed models on two real case studies, from Australia and the USA, respectively. The results demonstrate the effectiveness of employing the models and deliver significant cost-savings. The Australian case study is used to evaluate the impact of adoption of the inline tanks on the operations of the railroad companies. Based on the potential cost-savings of RLFMIR and the price of inline tanks, we were able to determine the optimal size of the inline tank fleet. Therefore, apart from fuel plans and inline tanks management, the model may be used for the decision to invest in an inline tank fleet.

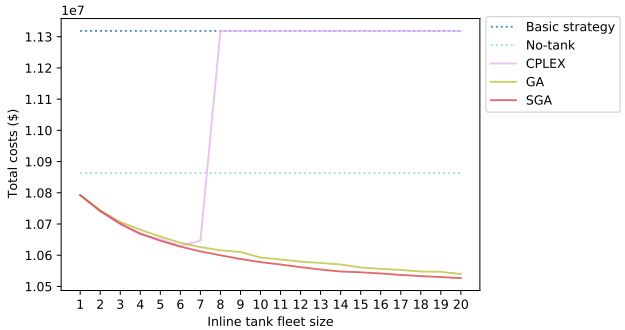
Furthermore, we assess the use of inline refueling to handle unpredictable events. We show that RLFMIR allows fuel safety levels to be increased at a lower cost. Using dual



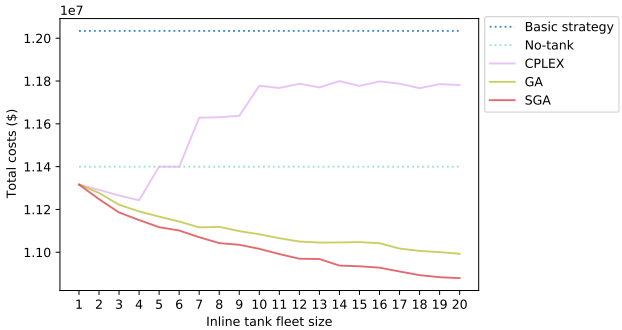
(a) Data set: Australia, Problem: RLFMIR



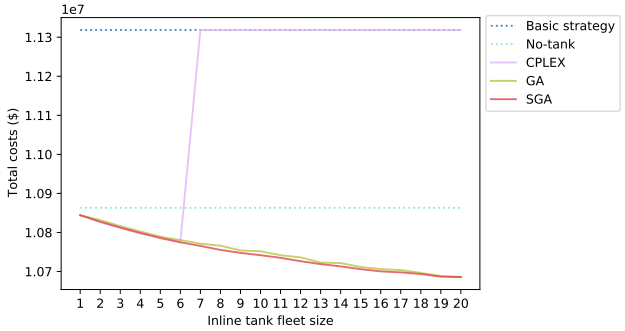
(b) Data set: Australia, Problem: LFMIR



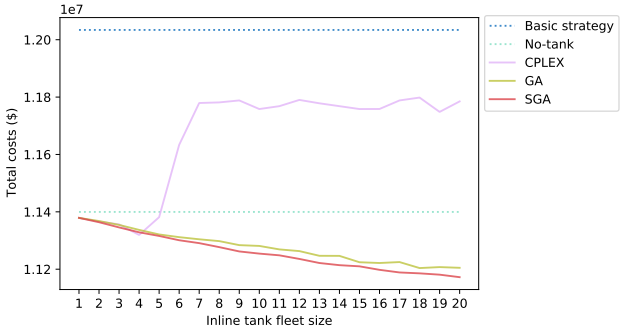
(c) Data set: INFORMS (Maximal connectivity), Problem: RLFMIR



(d) Data set: INFORMS (Maximal connectivity), Problem: LFMIR



(e) Data set: INFORMS (Minimal connectivity), Problem: RLFMIR



(f) Data set: INFORMS (Minimal connectivity), Problem: LFMIR

Figure 3.15: Performance of CPLEX, GA, and SGA over different instances of RLFMIR and LFMIR per different fleet sizes

variables of the model, we provide the shadow prices of fuel consumption increase on the paths and identify the critical paths. We indicate that the fuel prices at the origins and the destinations are not the only factor that affect the shadow price. Moreover, we show that using inline tanks along with RLFMIR increases the robustness to uncertainties in fuel consumption. In order to verify the advantages of the proposed models, we apply them to an instance from an INFORMS problem solving competition. We show that our proposed models are able to contribute to additional savings in comparison with optimization models

that were developed to solve these instances. Different instances are derived from the INFORMS data set and it is shown that for all instances, investing in inline tanks and using the fuel plans given by our model (RLFMIR) results in significant cost savings. Moreover, we identify the cases with more potential savings for employing the inline tanks. RLFMIR contributes to more savings in situations where the cost of one unit of consumed fuel is high in the optimal solution of RLFMIR with the fleet size of 0. Furthermore, we demonstrate the complexities of solving the large instances by one commercial solver. The general-purpose solver fails to assist practitioners to perform scenario analyses and planning since the solution by CPLEX implies an unexpected increasing trend as more resource (inline tank) becomes available. We resolve this issue by proposing a semi-greedy heuristic that is able to provide good quality solutions for all instances.

Future research on this topic could consider extensions of the models, uncertainty, and applications to other modes of transportation. One possible direction is to integrate the fuel management problem with other planning processes of rail businesses. In particular, fuel management of the locomotives is closely connected to train scheduling, since the weight and the speed of the trains determine the fuel consumption. Moreover, fuel management determines if a schedule is feasible regarding the fuel requirements. Hence, an integrated problem could bring additional cost-savings and also avoids infeasible plans. Another direction is to consider uncertainty in the data such as fuel prices, which can be valuable for strategic or tactical decisions. To determine the inline tank fleet size, we assume that fuel prices and train schedules do not change during the life of the inline tanks. However, train schedules change in smaller time periods. Moreover, although companies often hedge fuel prices with contracts, this is again for much shorter periods than the lifetime of inline tanks. Stochastic programming would therefore benefit the proposed models to determine the fleet size efficiently. Since the fuel plan is determined for one or two weeks, it is reasonable to assume that fuel price is deterministic and constant for this period. However, the fuel consumption during trip changes because of the weather condition, the drivers' behavior, and other unpredictable events. Hence, considering fuel consumption as a random variable would results in more robust fuel plans. Models and algorithms presented in this chapter provide a basis for modeling and solving the problems with such uncertain problems. For instance, the GFMIR and LFMIR models can be used to determine recourse actions in a two-stage stochastic optimization model that decides for strategic decisions such as location of long-term fuel stations and/or inline tanks fleet size. The proposed models and algorithms can be directly used to handle uncertain fuel price and consumption. Considering the average fuel price in the input data reflects the fuel price uncertainty in the optimization model as minimizing expected costs is the same as minimizing LFMIR/GFMIR using the mean fuel prices. As we show in Section 3.6.3, LFMIR provides practical fuel plans to combat

unpredictable fuel consumption increases with a small cost by considering safety inventory. The third direction is adapting the proposed model for other modes of transportation in which there are similar auxiliary fuel reservoirs same as inline tanks. One particular application could be hydrogen-powered locomotives for which hydrogen gas tanks can be switched between locomotives likewise the inline tanks (Marin et al., 2010).

Computational results in this chapter show that the B&B algorithm implemented in commercial solvers performs weakly. This is particularly because of the long computing time of the corresponding LP relaxations of the large instances. The LP relaxation size and computing time increases linearly in the number of available inline tanks. The same trend appears when solving the large instances of the multicommodity network flow problems. This motivates the next chapter of the thesis, which proposes new aggregation schemes for multicommodity network flow problems to handle large instances. In Chapter 5, we apply such schemes to an abstract version of the LFMIR problem introduced in this chapter.

CHAPTER 4

Partial Aggregations for Multicommodity Network Flow Problems

4.1 Overview

This chapter proposes novel aggregation techniques for the multicommodity network flow problems which have extensive applications in optimization problems, particularly in transportation. In Section 4.2, we begin by introducing a new representation of commodities as *dispersions* which allow us to construct *partial aggregations*. These concepts are then applied to the multicommodity capacitated fixed-charge network design problem, whose disaggregated formulation (DA) and fully-aggregated formulation (FA) are presented in Section 2.3.1, to obtain a *partially-aggregated* formulation (PA). Section 4.3 improves the PA formulations by three approaches, equality and inequality tightening constrains and a heuristic to judiciously construct the partial aggregations. Section 4.4 studies the polyhedra of the LP relaxations of the proposed formulations and compares them with the existing formulations in the literature. In Section 4.5, we perform an extensive computational study on a set of benchmark instances to empirically compare the LP relaxations and also the impact of different formulations on the performance of the MIP algorithms.

This chapter contains four main contributions:

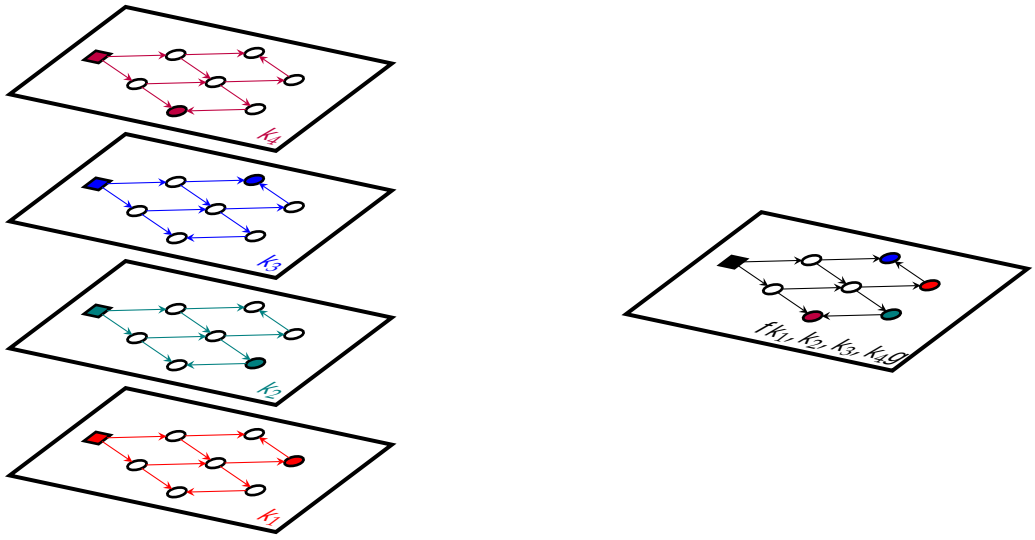
1. It introduces new commodity representations for the multicommodity network flow problems that make the partial aggregations possible.

2. It proposes a base partially-aggregated formulations for the multicommodity capacitated fixed-charge network design problem and improves it by three approaches.
3. It studies the polyhedra of the LP relaxations of the proposed formulations and their relation with the polyhedra of the LP relaxation of two existing formulations in the literature.
4. It empirically investigates the LP relaxations of the proposed formulations and the performance of the MIP algorithms over the proposed formulations for a set of benchmark instances.

4.2 Commodity Definitions and Aggregation Levels

Recall that in the input, each commodity $k \in K$ is defined by three attributes; an origin o^k, a destination s^k, and a demand of d^k. The representation of commodities in the DA formulation is congruous with the definition of commodities in the input. In particular, there is a one-to-one mapping between commodities of the input and commodities in the DA formulation. However, this need not be the case, as illustrated in the FA formulation, in which a "*commodity*" is an aggregation of the input commodities that share the same origin.

Alternative commodity definitions actually correspond to different network structures that represent the problem. This results in various sizes of the MIP model that encodes the problem since the nodes and arcs of the network correspond to constraints and variables of the MIP model. These MIPs have the same optimal integer solution but different LP relaxations. In the DA formulation, the flow conservation constraints (2.15b) describe the routes of each commodity on the graph G independently from other commodities. This means that the DA formulation essentially considers a hypothetical network *layer* for each commodity. This hypothetical layer is a copy of the original graph G on which the demand of its corresponding commodity flow. On the other hand, in the FA formulation, all layers related to the aggregated commodities are merged into a single layer on which the total demand of the aggregated commodities flow. Figure 4.1 shows the network representations of the DA and FA formulations for an instance with 4 commodities. In this instance, the original graph G consists of 8 nodes and 10 arcs, and all commodities are originated from the square node. Figure 4.1a shows the network representation of the DA formulation. In this figure, the opaque nodes represent the origin and destination of the corresponding commodity of each layer. As there are 4 commodities, network representation of the DA formulation includes 4 layers. These layers are merged into one layer by the FA formulation as shown in Figure 4.1b. This layer includes an origin (square) node for the aggregated commodities and four opaque (circle) nodes as destinations of the aggregated commodities.



(a) Implied network by the DA formulation

(b) Implied network by the FA formulation

Figure 4.1: Network representation of the DA and FA formulations for an instance with 4 commodities, all originating from the square node in the figure, on a graph with 8 nodes and 10 arcs.

In this section, we introduce new commodity definitions that give a control on the aggregation level that could range from the disaggregated approach to the fully aggregated approach. This enables us to develop a spectrum of formulations that fill in the gap between the DA and the FA formulations in terms of the trade-off between the computational difficulty and bound quality of the LP relaxation of a formulation. The intention of our approach is to allow groups of commodities to start in an aggregated form at a shared origin and to be disaggregated as they move towards their individual destination. Therefore, the set of commodities included in a same group varies over the network. For this purpose we define a *dispersion* of commodities. A dispersion is defined as a set of commodities and specifies how each of its commodities is aggregated on each arc.

Definition 1 (dispersion)**.** A dispersion b of commodities on a directed graph $G = (N, A)$ is defined by the following information:

1. A set $K_b \quad K$ of commodities that share the common origin $o_b \; 2 \; N$.
2. The set of destination nodes $S_b \quad N$ $(S_b = fs^k : k \; 2 \; K_b g)$.
3. On each arc $(i, j) \; 2 \; A$, a partition of K_b into a subset K_b^{ij} of commodities aggregated on that arc and its complement D_b^{ij} of commodities which are disaggregated on that arc.

A dispersion is essentially similar to a fully-aggregated commodity but with disaggregation of some commodities on some arcs. Suppose a commodity is disaggregated from the

group on the arc (i, j). Such disaggregation is represented by adding an extra arc with the same origin and destination as the arc (i, j) but exclusive to the flow of the disaggregated commodity. Therefore, the network representation of a dispersion is similar to the network representation of a fully-aggregated commodity but with some extra arcs dedicated to the flow of the separated/disaggregated commodities, and there is one network layer for a dispersion. Figure 4.2 shows an example dispersion based on the fully-aggregated commodity shown in Figure 4.1b. In this figure, extra arcs that represent the flow of the disaggregated commodities are shown in colors. We will discuss the selection of such arcs for the commodities in Section 4.3.3. Black arcs correspond to the flow of the commodities that are not disaggregated from the group on that arc.

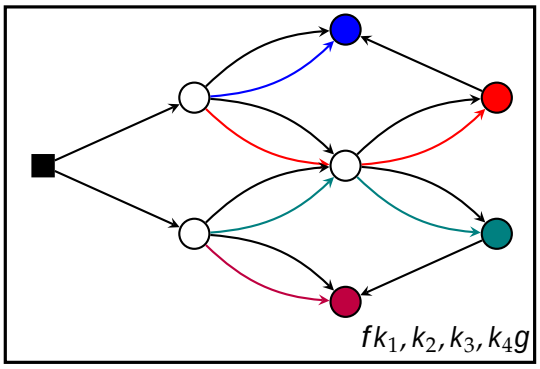


Figure 4.2: An example dispersion based on the aggregated commodities of Figure 4.1b. In this figure, colored arcs represent the disaggregation of the corresponding commodity from the group on that arc.

To represent an instance of the problem by dispersions, we have to define a set of dispersions in which there is at least one dispersion that includes each commodity. This ensures that all possible flow assignments of each commodity to all arcs of the network are covered. We next define partial aggregations in which there is a unique dispersion for each commodity.

Definition 2 (partial aggregation)**.** A partial aggregation for the MCND with $G = (N, A)$, original commodities K and demand d^k, consists of a set B of dispersions, where for every commodity $k \ 2 \ K$ there exists a unique $b \ 2 \ B$ such that $k \ 2 \ K_b$.

As defined in Definitions 1 and 2, partial aggregations must satisfy three conditions: (a) aggregated commodities must always share a common origin, (b) each commodity belongs to a single dispersion, and (c) a commodity is either disaggregated on an arc or it is grouped with others of the group (that are also not disaggregated on this arc). An alternative definition is to consider dispersions based on same destinations or a mixed approach in which some dispersions include commodities with a same origin and others include commodities with a same destination. It should be noted that these definitions

could of course be extended in a natural way to allow different types of dispersions and aggregations.

Conventional disaggregation and full aggregation are special cases of partial aggregations:

DA The original, fully disaggregated version (as a special partial aggregation case) is defined by B, where $|B| = |K|$, each dispersion $b \in B$ includes only one commodity $K_b = \{k_b\}$, and $\bigcup_{b \in B} K_b = K$. Therefore, $S_b = \{i \in N \mid s_i^{k_b} = 1\}$, $|S_b| = 1$ for all $b \in B$, and $K_b^{ij} = K$ & $D_b^{ij} = \{k_b\} : \forall b \in B, (i,j) \in A$.

FA Let $\tilde{N}$ be the set of nodes from which at least one commodity originates. The fully aggregated version is defined by B, where $|B| = |\tilde{N}|$ and $\bigcup_{b \in B} \{o_b\} = \tilde{N}$. Therefore, $S_b = \{i \in N \mid \exists k \in K : s_i^k = 1, o^k = o_b\}$ for all $b \in B$, and $K_b^{ij} = S_b$ & $D_b^{ij} = K : \forall b \in B, (i,j) \in A$.

The network implied by DA has one complete replica of graph G for each commodity $k \in K$, which results in a network with $|K|$ layers. The network implied by FA has one complete replica of graph G for each dispersion $n \in \tilde{N}$, which results in a network with $|\tilde{N}|$ layers. In both cases, each layer is an exact copy of the graph G. However, the network layer that represents a dispersion includes some extra arcs that correspond to the disaggregation of commodities on those arcs. In particular, such a layer includes: (a) a complete copy of the nodes of graph G, (b) one copy of arc $(i,j) \in A$ for which $K_b^{ij} \neq K$, and (c) one copy of arc $(i,j) \in A$ for each $k \in D_b^{ij}$ to represent disaggregated arcs. An example network layer corresponding to a dispersion is shown in Figure 4.2. Here we define a network layer that represent a dispersion as *dispersion layer*.

Definition 3 (dispersion layer)**.** A dispersion layer that represents a dispersion b includes a set of nodes $C_b := N$ and a set of arcs A_b^{ij} for each $(i,j) \in A$. Each arc $a \in A_b^{ij}$ corresponds to a commodity set of the set $G_b^{ij} = K_b^{ij} \setminus \{\{k\} \mid k \in D_b^{ij}\}$ by a one-to-one mapping. Each set $D \in G_b^{ij}$ implies that in the dispersion layer corresponding to dispersion b there is a copy of arc (i,j) that corresponds to the commodity set D.

A partial aggregation consists of a set of dispersions. Similarly, the network implied by a partial aggregation, called *partial aggregation network*, is a set of dispersion layers, where each layer corresponds to a dispersion $b \in B$. A partial aggregation network that corresponds to to the partial aggregation B includes $|B|$ layers. A partial aggregation network has $|B||N|$ nodes and $\sum_{b \in B} \sum_{(i,j) \in A} |G_b^{ij}|$ arcs. The DA and FA networks have respectively $|K||N|$ and $|\tilde{N}||N|$ nodes and $|K||A|$ and $|\tilde{N}||A|$ arcs.

To have a formulation incorporating a partial aggregation and its corresponding partial aggregation network, we can now modify the DA formulation (2.15a)–(2.15f) by

1. Summing (2.15b) over $k \in K_b$ for all $b \in B$, $i \in C_b$, resulting in $|B||N|$ constraints.
2. Summing (2.15d) over $k \in D$ for all $(i,j) \in A, b \in B, D \in G_b^{ij}$, resulting in between $|A||B|$ and $|A||K|$ constraints.
3. Replacing any sum over variables x_{ij}^k in the constraints of the resulting model by a new variable $x_{ij}^D = \sum_{k \in D} x_{ij}^k$ that corresponds to aggregated flow of a set of commodities $D \in G_b^{ij}$ for some $(i,j) \in A$ and $b \in B$.

This gives a new formulation that in general has both fewer constraints and fewer variables than DA.

Partially-Aggregated Formulation (PA):

$$\min \quad \sum_{(i,j)\in A} c_{ij} \sum_{b\in B} \sum_{D\in G_b^{ij}} x_{ij}^D + \sum_{(i,j)\in A} f_{ij} y_{ij} \tag{4.1a}$$

$$s.t: \quad \sum_{j\in N_i^+} \sum_{D\in G_b^{ij}} x_{ij}^D - \sum_{j\in N_i} \sum_{D\in G_b^{ji}} x_{ji}^D = \sum_{k\in K_b} (o_i^k - s_i^k) d^k \qquad \forall b \in B,\ i \in C_b \tag{4.1b}$$

$$\sum_{b\in B} \sum_{D\in G_b^{ij}} x_{ij}^D \le u_{ij} y_{ij} \qquad \forall (i,j) \in A \tag{4.1c}$$

$$x_{ij}^D \le \left(\sum_{k\in D} d^k \right) y_{ij} \qquad \forall (i,j) \in A,\ b \in B,\ D \in G_b^{ij} \tag{4.1d}$$

$$x_{ij}^D \ge 0 \qquad \forall (i,j) \in A,\ b \in B,\ D \in G_b^{ij} \tag{4.1e}$$

$$y_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{4.1f}$$

In the PA formulation, equation (4.1b) conserves the aggregated flow of commodities in a same dispersion at each node of the network, whether they are disaggregated from the group or not. Note that if $\sum_{k\in D} d^k > u_{ij}$ for $D \in G_b^{ij}$, as normally happens in the FA formulation, then the SI constraints (4.1d) become redundant.

4.3 Improving the Partially-Aggregated Formulation

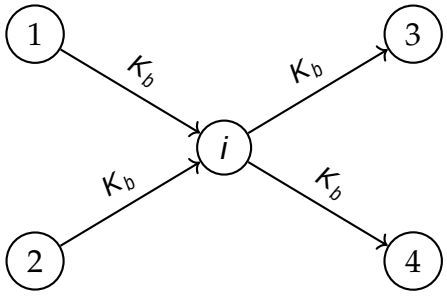
While (partial) aggregation makes the MIP model smaller, it weakens the formulation in terms of the LP relaxation bound. To combat this we employ two strategies that are discussed in this section, namely (a) adding tightening constraints and (b) making judicious choices about which commodities to aggregate/disaggregate for each arc. If the coefficient

of variable y_{ij} in the corresponding SI (4.1d) is large enough, the structure of a dispersion layer allows a commodity to flow on arc (i, j) on the aggregated arc even if this commodity has been disaggregated on arc (i, j). The tightening constraints we introduce in this section prevent the commodities from flowing on such arcs. We employ two approaches for this. The first approach in Section 4.3.1 adds a type of inequalities that are similar to flow conservation constraints. The second approach in Section 4.3.2 modifies the partial aggregation network by adding some artificial nodes and arcs that translates to more constraints and variables. However, the extra constraints can be expressed as equality flow conservation constraints, which both tighten the bound and, in practice, speed-up the computation of the LP. In Section 4.3.3, we propose a heuristic that uses a K-shortest paths algorithm to construct dispersions of the commodities and form a partial aggregation that provides a good trade-off between model size and LP bound. Here we are particularly interested in dispersions where the disaggregated arcs for each commodity form a connected network leading to the destination node of that commodity.

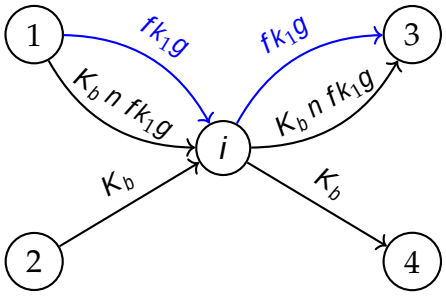
4.3.1 Partially-Aggregated Formulation with Inequality Tightening Constraints

We start with an example that shows a fractional solution of the LP relaxation of the PA formulation, and propose valid inequalities that can cut it. The example is shown for the incoming and outgoing flow of the node i in Figure 4.3a. This node is part of the larger network of the problem, but the other parts are not shown as this node is enough to illustrate the example. This node has two incoming arcs from nodes 1 and 2, and two outgoing arcs to nodes 3 and 4. Consider the dispersion b of commodities $K_b = \{k_1, k_2, k_3\}$. Origin and destination nodes of commodities of set K_b are not shown in this figure. However, in our examples we consider feasible solutions that flow of some commodities of this set pass the node i as an intermediate node. In this figure, we assume that all commodities of K_b are grouped together on these arcs, therefore $G_b^{1i} = G_b^{2i} = G_b^{i3} = G_b^{i4} = K_b$ and there is no disaggregated arc. As mentioned in Section 4.2, each arc of the network corresponds to a commodity set. In Figure 4.3, the corresponding commodity set to each arc is labeled on it. In Figure 4.3a, commodities are grouped on all arcs shown, and hence, all arcs correspond to the commodity set K_b. Now, suppose that we disaggregate the commodity k_1 on arcs $(1, i)$ and $(i, 3)$. Such disaggregation requires the network to have two additional arcs, shown in blue in Figure 4.3b.

As the flow conservation constraint (4.1b) considers all commodity sets G_b^{ij} related to a dispersion on an arc in one equation. Therefore, the flow of the disaggregated commodity k may flow on the arc that is related to the commodity set K_b^{ij} even if $k \notin K_b^{ij}$, in case the



(a) Node i in case of aggregating all commodities K_b on the arcs related to this node



(b) Node i in case of disaggregating the commodity k_1 on arcs $(1,i)$ and $(i,3)$

Figure 4.3: An example of fully aggregated and partial aggregation network. The example shows node i in the dispersion layer b with two incoming and two outgoing arcs. Each arc is labeled with its corresponding commodity set.

coefficient for variable y_{ij} in the corresponding SI constraint (4.1d) of K_b^{ij} is large enough, which degrades the LP bound. For instance, in Figure 4.3b the flow of commodity k_1 can arrive to node i by the blue arc $(1,i)$ and leave it by the black arc $(i,3)$. However, the related commodity set of the black arc $(i,3)$ does not include commodity k_1. Let $d^{k_1} = 1$. Consider Example 4.3.1 which is a feasible flow for the PA formulation over node i.

Example 4.3.1. *This is an example feasible solution for the PA formulation for the flow on incoming and outgoing arcs of node i in Figure 4.3b: $x_{1i}^{fk_1g} = x_{i3}^{K_b \cap fk_1g} = 1$ and flow on other incoming and outgoing arcs of node i equals to zero.*

Although Example 4.3.1 is a feasible solution for the PA, we want to prevent such flows as this flow arrives at node i by an arc dedicated to k_1. However, it leaves node i by an arc related to a commodity set that does not include k_1. To prevent such flow assignments we add two inequalities (4.1g) and (4.1h).

$$\sum_{j \in N_i^+} \sum_{D \in G_b^{ij} : k \in D} x_{ij}^D \;-\; \sum_{j \in N_i} \sum_{D \in G_b^{ji} : D = fkg} x_{ji}^D \;\le\; (o_i^k - s_i^k) d^k \quad \forall b \in B,\ k \in K_b,\ i \in C_b \tag{4.1g}$$

$$\sum_{j \in N_i^+} \sum_{D \in G_b^{ij} : D = fkg} x_{ij}^D \;-\; \sum_{j \in N_i} \sum_{D \in G_b^{ji} : k \in D} x_{ji}^D \;\ge\; (o_i^k - s_i^k) d^k \quad \forall b \in B,\ k \in K_b,\ i \in C_b \tag{4.1h}$$

Intuitively, Constraint (4.1g), called *forward labeling inequality*, states that any flow that arrives at node i labelled as commodity k (i.e. with a x_{ij}^D variable where $D = fkg$) must leave either as commodity k or as an aggregated commodity that includes k. Example 4.3.1 violates this constraint, as in this case, the left-hand side of this of constraint is 1 while its right-hand side is zero. Constraint (4.1h), called *backward labeling inequality*, creates the symmetric requirement on arriving arcs for an individual commodity k leaving node i. A

feasible solution of the PA formulation for this instance as $x_{1i}^{K_b \setminus \{k_1\}} = x_{i3}^{\{k_1\}} = 1$ and the flow on other incoming and outgoing arcs as zero violates Constraint (4.1h). In the rest of the thesis, we consider (4.1a)-(4.1h) as the *partially aggregated formulation with forward and backward labeling inequalities*, and refer to it as **PAi**.

4.3.2 Partially-Aggregated Formulation with Equality Tightening Constraints

In this section, we present another approach to tackle the issue raised in Section 4.3.1 for the PA formulation. In this approach, instead of adding inequalities, we modify the partial aggregation network to differentiate the incoming flow to a node labeled as a disaggregated commodity from the aggregated incoming flow. Consider the node i of the dispersion layer b with some incoming and outgoing arcs (aggregated and disaggregated). We add some artificial nodes and arcs in this node to be able to differentiate the flow of disaggregated commodities. As this approach adds extra nodes and arcs to the partial aggregation network, its corresponding MIP would be larger than the MIP of PAi for a same partial aggregation B. However, its merit is that the added constraints are equality flow conservation constraints which keeps the multicommodity network structure of the problem.

As formulated, there is one flow conservation constraint per dispersion layer and per node. This means that a node of a dispersion layer does not differentiate the incoming and outgoing flow by the disaggregated arcs from the incoming and outgoing flow by the aggregated arcs, which implies that the flow of one commodity can be routed on arcs that do not represent that commodity. Here we propose to decompose every node into a small gadget that will track which commodity can arrive and leave on which arc, and restrict the possibilities of incorrect routing.

As an example, consider the node i in the dispersion layer b shown in Figure 4.4a. Dispersion b includes 4 commodities as $K_b = \{k_1, k_2, k_3, k_4\}$. Node i is part of a larger network, but other nodes are not shown as this node is enough to show the examples we intend. Moreover, origin and destination nodes of commodities of set K_b are not shown in this figure. Node i in the original graph G has two incoming arcs $(1, i)$ and $(2, i)$ and two outgoing arcs $(i, 3)$ and $(i, 4)$. Commodity k_1 is disaggregated from the group on arcs $(1, i)$ and $(2, i)$. Commodity k_2 is disaggregated from the group on arcs $(2, i)$, $(i, 3)$, and $(i, 4)$. The original node i in the partial aggregation network is shown in Figure 4.4a. One possible incorrect flow route for this example is a flow arriving at node i by the red arc between nodes 2 and i which corresponds to the commodity k_2, but leaving the node by the black arc between nodes i and 3 which does not include commodity k_2 in its corresponding set.

The gadget we propose to tackle the raised issue considers the node i as a larger node

that itself includes a set of some artificial nodes, where each of such nodes correspond to a specific commodity set. We define the group of artificial nodes for each node and their connection as in fact a network modification that adds artificial nodes and arcs inside a node of a dispersion layer. For each node i and dispersion layer b, we create three groups of artificial nodes:

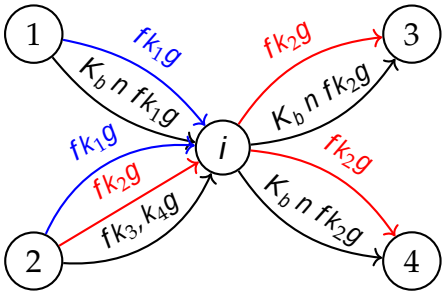
- in-flow aggregated nodes: one node for each distinct commodity set that incoming aggregated arcs of node i represent. For node i in Figure 4.4a, we add two in-flow aggregated nodes, one corresponds to the commodity set $K_b \cap \{k_1\}$ and the other one corresponds to the commodity set $\{k_3, k_4\}$.
- intermediate nodes: a node for each commodity $k \in K_b$ that has at least one incident disaggregated arc to node i and one node for the rest of commodities of dispersion b. For node i in Figure 4.4a, we add three intermediate nodes. Two nodes are added because two commodities, k_1 and k_2, have incident disaggregated arcs to node i. The third node is added to represent the rest of commodities of set K_b, i.e. $\{k_3, k_4\}$.
- out-flow aggregated nodes: one node for each distinct commodity set that outgoing aggregated arcs of node i represent. For node i in Figure 4.4a, we add one out-flow aggregated node as both outgoing aggregated arcs correspond to the same commodity set, i.e. $K_b \cap k_2$.

Then, there is an artificial arc from any in-flow aggregated node to any intermediate node if and only if they represent at least one common commodity. Similarly, there is an artificial arc from any intermediate node to any out-flow aggregated node if and only if they represent at least one common commodity. Such artificial arcs facilitate the modeling as they describe the relation of artificial nodes added. They have no flow cost or opening cost, hence their flow are not present in the objective function and there is no associated binary variable as they are always open. Original arcs of the partial aggregation network remain same. However, they must be connected to the relevant artificial node. Disaggregated arcs connect to the intermediate node of the corresponding disaggregated commodity. Incoming (outgoing) aggregated arcs connect to the corresponding in-flow (out-flow) aggregated node. Node i in Figure 4.4a is modified and illustrated as the larger rectangular node in Figure 4.4b, where artificial nodes and arcs are shown by dashed circles and arrows. Moreover, each group of artificial nodes inside the node i is shown in a dotted ellipse for clarity.

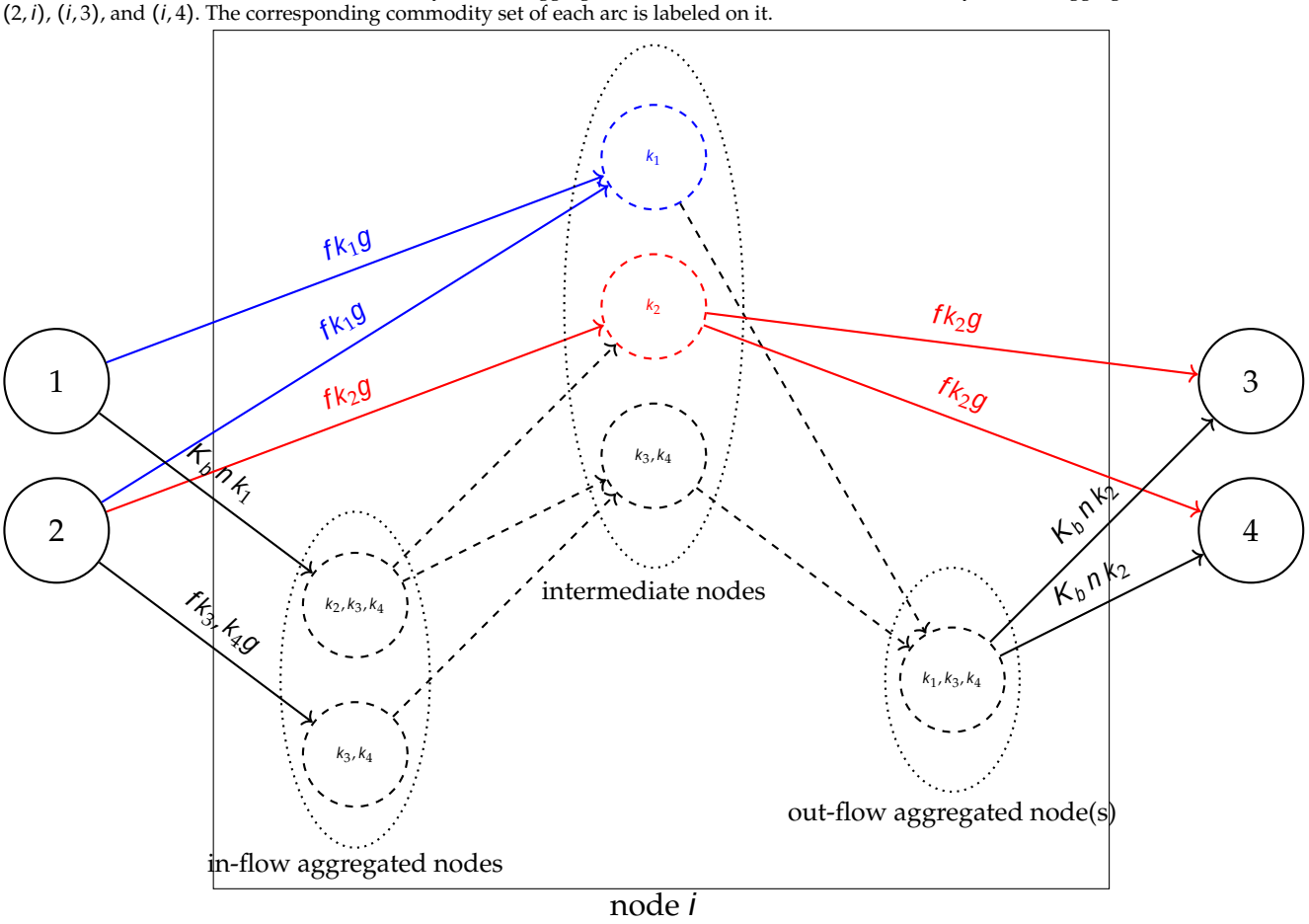
Specifically, consider the following sets for each $b \in B$ and $i \in C_b$.

L_b^i Set of commodities of dispersion b that have at least one incident disaggregated arc to node i. For node i in Figure 4.4, we have $L_b^i = \{k_1, k_2\}$.

M_b^i Set of commodity sets that each added intermediate node for node i of dispersion b represents. For node i in Figure 4.4, we have $M_b^i = \{\{k_1\}, \{k_2\}, \{k_3, k_4\}\}$.



(a) Node i of dispersion layer b. Node i in the original graph G has two incoming arcs and two outgoing arcs. Dispersion b includes commodities $K_b = \{k_1, k_2, k_3, k_4\}$. Commodity k_1 is disaggregated on arcs $(1, i)$ and $(2, i)$. Commodity k_2 is disaggregated on arcs $(2, i)$, $(i, 3)$, and $(i, 4)$. The corresponding commodity set of each arc is labeled on it.



(b) Modified node i as the larger rectangular node with added artificial nodes and arcs shown by dashed circles and arrows.

Figure 4.4: An illustrative example for network modification for the partially-aggregated formulation with equality tightening constraints

$\check{T}_b^i$ Set of distinct commodity sets that each incoming aggregated arc of node i corresponds to. For node i of Figure 4.4, we have $\check{T}_b^i = \{\{k_2, k_3, k_4\}, \{k_3, k_4\}\}$.

$\hat{T}_b^i$ Set of distinct commodity sets that each outgoing aggregated arc of node i corresponds to. For node i of Figure 4.4, we have $\hat{T}_b^i = \{\{k_1, k_3, k_4\}\}$.

Formally, these sets for the node i of the dispersion layer b are obtained by equations (4.2a)-(4.2d). Artificial nodes and arcs are added inside the node i of the dispersion layer b only if $L_b^i \neq \emptyset$.

$$L_b^i = \{k \in K_b \mid (j,i) \in N : k \in D_b^{ji} \text{ or } (i,j) \in N_i^+ : k \in D_b^{ij}\} \tag{4.2a}$$

$$M_b^i = \{\{k\} \mid k \in L_b^i\} \cup \{K_b \setminus L_b^i\} \tag{4.2b}$$

$$\check{T}_b^i = \{C_b^{ik} \quad K_b \mid k = 1,\ldots,|\check{T}_b^i|, C_b^{ik} \neq \emptyset\}$$
$$\text{such that: } \bigcup_{C \in \check{T}_b^i} C = \bigcup_{j \in N_i} \{K_b^{ji}\} \text{ and } C \setminus D = \emptyset \ \forall C \neq D \in \check{T}_b^i \tag{4.2c}$$

$$\hat{T}_b^i = \{C_b^{ik} \quad K_b \mid k = 1,\ldots,|\hat{T}_b^i|, C_b^{ik} \neq \emptyset\}$$
$$\text{such that: } \bigcup_{C \in \hat{T}_b^i} C = \bigcup_{j \in N_i^+} \{K_b^{ij}\} \text{ and } C \setminus D = \emptyset \ \forall C \neq D \in \hat{T}_b^i \tag{4.2d}$$

Consider Example 4.3.2 which is a feasible flow for the PA formulation over node i. Similar to Example 4.3.1, we want to prevent such flows as Example 4.3.2.

Example 4.3.2. *This is an example feasible solution for the PA formulation for the flow on incoming and outgoing arcs of node i in Figure 4.4: $x_{2i}^{\{k_3,k_4\}} = x_{i3}^{\{k_2\}} = 1$ and flow on other incoming and outgoing arcs of node i equals to zero.*

By having such modification on the partial aggregation network, it is enough to add flow conservation constraints for the added artificial nodes to tackle the issue raised in Section 4.3.1. The modified network structure labels a commodity that arrives at a node by a disaggregated arc and thus excludes flow on an aggregated arc that does not include that specific commodity. Specifically, equations (4.3a)-(4.3c) defines such flow conservation constraints. Here, we define the flow on artificial arcs by z variables as constraints (4.3d) and (4.3e).

$$\left(\sum_{j \in N_i^+ : D \setminus D_b^{ij} \neq \emptyset} x_{ij}^D \quad - \sum_{j \in N_i : D \setminus D_b^{ji} \neq \emptyset} x_{ji}^D\right) + \left(\sum_{C \in \hat{T}_b^i : D \setminus C \neq \emptyset} z_{DC}^{ib} \quad - \sum_{C \in \check{T}_b^i : D \setminus C \neq \emptyset} z_{CD}^{ib}\right) = \sum_{k \in D} (o_i^k - s_i^k) d^k$$
$$\forall b \in B,\ i \in C_b,\ D \in M_b^i : L_b^i \neq \emptyset \tag{4.3a}$$

$$\sum_{D \in M_b^i : C \setminus D \neq \emptyset} z_{CD}^{ib} \quad - \sum_{j \in N_i : C = K_b^{ji}} x_{ji}^C = 0 \qquad \forall b \in B,\ i \in C_b,\ C \in \check{T}_b^i : L_b^i \neq \emptyset \tag{4.3b}$$

$$\sum_{j \in N_i^+ : C = K_b^{ij}} x_{ij}^C \quad \sum_{D \in M_b^i : C \setminus D \neq \emptyset} z_{DC}^{ib} = 0 \qquad \forall b \in B,\ i \in C_b,\ C \in \hat{T}_b^i : L_b^i \neq \emptyset \quad (4.3c)$$

$$z_{CD}^{ib} \geq 0 \qquad \forall b \in B,\ i \in C_b,\ C \in \check{T}_b^i,\ D \in M_b^i : C \setminus D \neq \emptyset \ \&\ L_b^i \neq \emptyset \quad (4.3d)$$

$$z_{DC}^{ib} \geq 0 \qquad \forall b \in B,\ i \in C_b,\ C \in \hat{T}_b^i,\ D \in M_b^i : C \setminus D \neq \emptyset \ \&\ L_b^i \neq \emptyset \quad (4.3e)$$

Equation (4.3a) represents the flow conservation constraint for artificial intermediate nodes. The first part determines the net outgoing flow by the disaggregated arcs. The second part determines the net outgoing flow by the artificial arcs. The total outgoing flow must respect the corresponding demand at the node i for the corresponding commodity set that an intermediate node represents. Equation (4.3b) shows the flow conservation constraint for artificial in-flow aggregated nodes. The outgoing flow for such arcs is on artificial arcs and the incoming flow is on original aggregated arcs of the partial aggregation network. Equation (4.3c) shows the flow conservation constraint for artificial out-flow aggregated nodes. The outgoing flow for such arcs is on original aggregated arcs of the partial aggregation network and the incoming flow is on artificial arcs. Decision variable z is defined for the flow on an artificial arc where ever an artificial arc is added based on constraints (4.3d) and (4.3e). Constraints (4.3a)-(4.3e) are added if node i of dispersion b is required to be modified, i.e. $L_b^i \neq \emptyset$. In the rest of the thesis, we consider (4.1a)-(4.1f) and (4.3a)-(4.3e) as the *partially aggregated formulation with equality tightening constraints*, and refer to it as **PAe**. Example 4.3.2 is not feasible for Constraints (4.3a)-(4.3e) as those flow conservation constraints for the modified node i do not link the flows $x_{2i}^{\{k_3,k_4\}}$ and $x_{i3}^{\{k_2\}}$.

4.3.3 K-Shortest Path Aggregations

The choices to cluster commodities can significantly affect the LP bound of a partially-aggregated formulation. In this section, we present a heuristic that employs a K-shortest path algorithm to effectively cluster the commodities. Based on Definition 1, a dispersion aggregates a set of commodities, but it disaggregates each commodity from the group on a subset of arcs. Therefore, to construct a dispersion of a set of commodities, it is enough to determine a subset of arcs that each commodity of the considered set is disaggregated on. We call the subset of arcs on which the commodity k is disaggregated from the corresponding group as *critical arcs* and represent it by A^k. Given A^k for all $k \in K_b$, sets K_b^{ij} and D_b^{ij} of dispersion b of the commodity set K_b are obtained as $D_b^{ij} = \{k \in K_b \mid (i,j) \in A^k\}$ and $K_b^{ij} = K_b \cap D_b^{ij}$ for all $(i,j) \in A$.

We determine the set of critical arcs of commodity k by solving a K-shortest path problem from its origin o^k to its destination s^k considering the surrogate cost of $\tilde{c}_{ij} = c_{ij} + \frac{f_{ij}}{u_{ij}}$ for

all arcs $(i, j)\ 2\ A$. Disaggregating commodity k on arc (i, j) translates to keeping the corresponding SI of the DA formulation in the partially-aggregated formulation. The surrogate cost $\tilde{c}_{ij}$ incorporates the variable cost, fixed cost, and capacity of an arc simultaneously, and results in keeping impactful SIs that tighten the partially-aggregated formulation. Such surrogate cost has been also utilized in other solution algorithms for MCND. Gendron et al. (2018) uses $\tilde{c}_{ij}$ as the initial values for their slope scaling heuristic. Yaghini et al. (2013) uses $\tilde{c}_{ij}$ to determine the set of variables that exist in the initial step of their column generation based heuristic. We have also examined two other cost structures as $c_{ij} + \frac{f_{ij}}{d^k}$ and $c_{ij} + \frac{f_{ij}}{\min_{k2K_b} d^k}$. However, $\tilde{c}_{ij}$ appears to be significantly more promising, and therefore other cost structures are not considered in the rest of this chapter. The details of the heuristic is explained in Algorithm 1. This algorithm essentially considers the arcs that construct K shortest paths from origin to destination of the commodity k as the set of its critical arcs A^k, and forms the dispersions and the partial aggregation accordingly. We use the algorithm by Yen (1971) for finding K shortest paths between an origin and a destination.

Algorithm 1: K-shortest path aggregation

Input: $G = (N, A)$, set of commodities K, and the number of shortest paths considered K
Output: A partial aggregation B which is a set of dispersions
for $b\ 2\ N$ **do**
  $K_b = fk\ 2\ K\ j\ o^k = bg$
  for $k\ 2\ K_b$ **do**
    A^k = Set of arcs included in K shortest paths from origin o^k to destination s^k considering the cost $\tilde{c}_{ij}$ for each arc
  end
  for $(i, j)\ 2\ A$ **do**
    $D_b^{ij} = fk\ 2\ K_b\ j\ (i, j)\ 2\ A^k g$
    $K_b^{ij} = K_b\ n\ D_b^{ij}$
  end
  Add the dispersion b to the set B
end

4.4 Polyhedral Analysis

This section compares the polyhedra of the LP relaxations of the DA, PAe, PAi, PA, and FA formulations. We consider the LP relaxations of these formulation which are obtained

by relaxing constraints (2.15f) and (4.1f) to $0 \quad y_{ij} \quad 1$. In the each of Theorems 4.4.1-4.4.4 we compare the strength of the LP relaxation of two formulations. To prove one of these two is stronger than the other, we show that there always exists a mapping that transforms an arbitrary feasible solution of the LP relaxation of the stronger formulation to a feasible solution for the LP relaxation of the weaker formulation. However, there exists fractional solutions to the weaker formulation that are not feasible for the stronger formulation.

Theorem 4.4.1. *The DA formulation is stronger than the PAe formulaion.*

Proof. Let $(\bar{x}^k_{ij}, \bar{y}_{ij})$ be an arbitrary solution of the LP relaxation of the DA formulation defined by (2.15b)-(2.15f). The following mapping transforms this solution to the solution $(x^D_{ij}, y_{ij}, z^{ib}_{CD})$ for the LP relaxation of PAe formulation defined by (4.1b)-(4.1f) and (4.3a)-(4.3e).

$$x^D_{ij} := \sum_{k2D} \bar{x}^k_{ij} \qquad 8\,(i,j)\ 2\ A,\ b\ 2\ B,\ D\ 2\ G^{ij}_b \tag{4.4a}$$

$$y_{ij} = \bar{y}_{ij} \qquad 8\,(i,j)\ 2\ A \tag{4.4b}$$

$$z^{ib}_{CD} = \sum_{j2N_i} \sum_{k2C \setminus D} \bar{x}^k_{ji} \qquad 8b\ 2\ B,\ i\ 2\ C_b,\ C\ 2\ \check{T}^i_b,\ D\ 2\ M^i_b : C \setminus D \neq \text{Æ} \ \&\ L^i_b \neq \text{Æ} \tag{4.4c}$$

$$z^{ib}_{DC} = \sum_{j2N^+_i} \sum_{k2D \setminus C} \bar{x}^k_{ij} \qquad 8b\ 2\ B,\ i\ 2\ C_b,\ C\ 2\ \hat{T}^i_b,\ D\ 2\ M^i_b : C \setminus D \neq \text{Æ} \ \&\ L^i_b \neq \text{Æ} \tag{4.4d}$$

We now prove that the transformed solution $(x^D_{ij}, y_{ij}, z^{ib}_{CD})$ obtained by mapping (4.4a)-(4.4d) is feasible for LP relaxation of PAe. For a node related to the dispersion layer b, the flow conservation constraint (4.1b) of PAe is just a summation of flow conversation constraints (2.15b) of DA over $k\ 2\ K_b$. The capacity constraint (4.1c) is feasible over the transformed solution since $\sum_{b2B} \sum_{D2G^{ij}_b} x^D_{ij} = \sum_{k2K} \bar{x}^k_{ij} \quad u_{ij}\bar{y}_{ij} = u_{ij}y_{ij}\ 8(i,j)\ 2\ A$. Strong inequality (4.1d) is just a summation of SIs (2.15d) over $k\ 2\ D$, and therefore feasible for the transformed solution.

We then show that the transformed solution is feasible for constraints (4.3a)-(4.3c). Constraint (4.3a) defines a flow conservation constraint for each artificial intermediate node (set M^i_b) that is added to the network. Based on equations (4.4c) and (4.4d), Constraint (4.3a) in fact conserves the flow of commodities of set D at node i. Therefore, it is just a summation of Constraint (2.15b) over the commodities of the corresponding commodity set D.

Since the set M^i_b partition the commodity set K_b, in Constraint (4.3b) we have:

$$\sum_{D2M^i_b : C \setminus D \neq \text{Æ}} z^{ib}_{CD} = \sum_{j2N_i} \sum_{k2C} \bar{x}^k_{ji}.$$

On other hand, $\sum_{j \in N_i : C = K_b^{ji}} x_{ji}^C = \sum_{j \in N_i} \sum_{k \in C} \bar{x}_{ji}^k$ based on equation (4.4a). Therefore, Constraint (4.3b) is satisfied by the transformed solution. In a similar way, Constraint (4.3c) can be proved to be satisfied by the transformed solution.

Now we show an example where a solution of the PAe LP relaxation is not feasible for the DA LP relaxation. Consider an example that the commodity k_1 is aggregated with some other commodities into the set C on arc (i,j). Moreover, $u_{ij} > \sum_{k \in C} d^k$. A feasible solution for PAe could be flowing only the demand of k_1 on arc (i,j) that translates to $x_{ij}^C = d^{k_1}$. In this case, a feasible value for y_{ij} is $\frac{d^{k_1}}{\sum_{k \in C} d^k}$ based on the corresponding SI and capacity constraint. However, this solution ($\bar{x}_{ij}^{k_1} = d^{k_1}, \bar{y}_{ij} = \frac{d^{k_1}}{\sum_{k \in C} d^k}$) is not feasible for the LP relaxation of DA as it violates the corresponding SI of the commodity k_1 on arc (i,j).

Therefore, the DA formulation is stronger than the PAe formulation because of two reasons: first, there always exists a mapping that transforms an arbitrary solution of the LP relaxation of the DA formulation to a solution for the LP relaxation of the PAe formulation. Second, there exists feasible solutions for the LP relaxation of the PAe formulation that are not projectable to the LP relaxation of the DA formulation. □

Theorem 4.4.2. *The PAe formulation is stronger than the PAi formulaion, where both formulations are based on a same partial aggregation B.*

Proof. Let $(\bar{x}_{ij}^D, \bar{y}_{ij}, \bar{z}_{CD}^{ib})$ be an arbitrary solution of the LP relaxation of the PAe formulation defined by (4.1b)-(4.1f) and (4.3a)-(4.3e). A feasible solution (x_{ij}^D, y_{ij}) for the LP relaxation of PAi formulation defined by (4.1b)-(4.1f), (4.1g), and (4.1h) can be obtained by the mapping $x_{ij}^D := \bar{x}_{ij}^D$ and $y_{ij} := \bar{y}_{ij}$.

Constraints (4.1b)-(4.1f) are common in both formulations. Therefore, we just prove that a solution by LP relaxation of PAe also satisfies constraints (4.1g) and (4.1h). Consider the commodity k and its corresponding dispersion b over node i where $\{k\} \in M_b^i$. Therefore, as in a partial aggregation the commodity k on the arc (i,j) is either aggregated with some other commodities or disaggregated, i.e. only one of two statements ($k \in K_b^{ij}$) $\exists! D \in G_b^{ij} : k \in D$) or ($k \in D_b^{ij}$) $\exists! D \in G_b^{ij} : D = \{k\}$) is correct for all $(i,j) \in A$, we have:

$$\sum_{j \in N_i^+ : D \setminus D_b^{ij} \neq \emptyset} x_{ij}^D = \sum_{j \in N_i^+} \sum_{D \in G_b^{ij} : D = \{k\}} x_{ij}^D \tag{4.5a}$$

$$\sum_{j \in N_i : D \setminus D_b^{ji} \neq \emptyset} x_{ji}^D = \sum_{j \in N_i} \sum_{D \in G_b^{ji} : D = \{k\}} x_{ji}^D \tag{4.5b}$$

Furthermore, based on equation (4.2d) and considering $D = \{k\}$, we have $\{C \in \hat{T}_b^i :$

$D \setminus C \in \{\emptyset\} = \{K_b^{ij} : \forall j \in N_i^+ \ j \neq k \in K_b^{ij}\}$. Moreover, equation (4.3c) implies that $z_{DC}^{ib} \geq \sum_{j \in N_i^+ : C = K_b^{ij}} x_{ij}^C$ for all $C \in \hat{T}_b^i$. Therefore, we have:

$$\sum_{C \in \hat{T}_b^i : D \setminus C \in \{\emptyset\}} z_{DC}^{ib} \geq \sum_{j \in N_i^+ : k \in K_b^{ij}} x_{ij}^{K_b^{ij}} \tag{4.5c}$$

From Definition 3 we have $G_b^{ij} = K_b^{ij} \cup \{\{k\} \mid k \in D_b^{ij}\}$. Moreover, $z_{CD}^{ib} \geq 0$. Having these in mind and also statements (4.5a)-(4.5c), flow conservation constraint (4.3a) of PAe implies the inequality (4.1g) of PAi. Therefore, any arbitrary solution of PAe satisfies Constraint (4.1g) of the PAi formulation. Similar approach can prove that the feasibility of solutions of PAe for Constraint (4.1h) of PAi. Clearly, when $\{k\} \in M_b^i$, constraints (4.1g) and (4.1h) are redundant regarding flow conservation constraint (4.1b).

Now we show an example where a solution of the PAi LP relaxation that is not feasible for the PAe LP relaxation. Consider an example that dispersion b of the commodity set $K_b = \{k_1, k_2, k_3, k_4\}$ over node i, where $N_i = \{1, 2\}$ and $N_i^+ = \{3, 4\}$. All commodities of K_b are disaggregated on arcs $(1, i)$ and $(i, 3)$, i.e. $K_b^{1i} = K_b^{i3} = \emptyset$ and $D_b^{1i} = D_b^{i3} = K_b$. However, they are all aggregated on two other arcs, i.e. $K_b^{2i} = K_b^{i4} = K_b$ and $D_b^{2i} = D_b^{i4} = \emptyset$. A feasible solution by PAi for this example is ($x_{1i}^{\{k_1\}} = x_{1i}^{\{k_2\}} = 1$, $x_{1i}^{\{k_3\}} = x_{1i}^{\{k_4\}} = 0$, $x_{i3}^{\{k_1\}} = x_{i3}^{\{k_2\}} = 0$, $x_{i3}^{\{k_3\}} = x_{i3}^{\{k_4\}} = 1$, $x_{2i}^{K_b} = x_{i4}^{K_b} = 1$). However, this solution is infeasible for the additional flow conservation constraints (4.3a)-(4.3c) of the PAe formulation as shown in the modified node i in Figure 4.5. In this figure, the flow on each is arc is labeled on it inside the parenthesis. Considering a flow conservation constraint for each internal node, the solution shown on the arcs is not feasible for PAe LP relaxation. Therefore, the PAe formulation is stronger than the PAi formulation. □

Theorem 4.4.3. *The PAi formulation is stronger than the PA formulation, where both formulations are based on a same partial aggregation B.*

Proof. PA formulation could be obtained from PAi by dropping constraints(4.1g) and (4.1h). Therefore, any feasible solution of PAi is also feasible for PA formulation.

Now we show an example where a solution of the PA LP relaxation that is not feasible for the PAi LP relaxation. Consider node i in Figure 4.3b and its corresponding partial aggregation. A feasible solution for PA LP relaxation over this partial aggregation over node i is $x_{1i}^{K_b \cap \{k_1\}} = d^{k_1}$ and $x_{i3}^{\{k_1\}} = d^{k_1}$. However, this solution violates the corresponding Constraint 4.1g in the PAi LP relaxation.

Therefore, the PAi formulation is stronger than the PA formulation. □

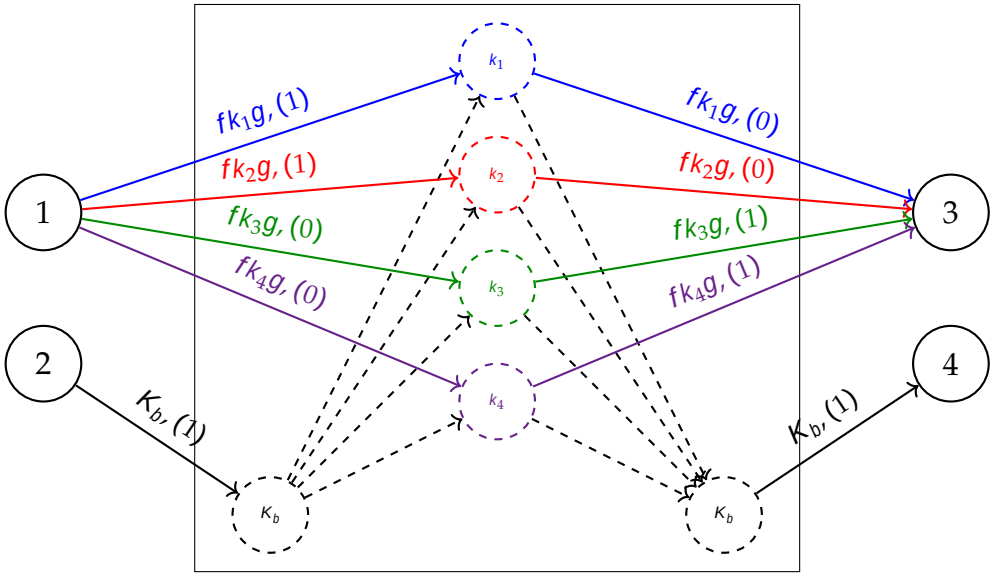


Figure 4.5: An example solution of PAi LP relaxation that is not feasible for PAe LP relaxation. This figure shows the modified node i for PAe, where $N_i = \{1,2\}$ and $N_i^+ = \{3,4\}$. The dispersion b (including commodity set $K_b = \{k_1, k_2, k_3, k_4\}$) of a partial aggregation states that $K_b^{1i} = K_b^{i3} = \emptyset$, $D_b^{1i} = D_b^{i3} = K_b$, $K_b^{2i} = K_b^{i4} = K_b$, and $D_b^{2i} = D_b^{i4} = \emptyset$. A feasible solution of the PAi LP relaxation for the flow of arcs is shown as the numbers in parenthesis above each arc. However, this solution is infeasible for PAe because of the flow conservation constrains for the internal nodes.

Theorem 4.4.4. *The PA formulation is stronger than the FA formulation.*

Proof. Let $(\bar{x}_{ij}^D, \bar{y}_{ij})$ be an arbitrary solution of the LP relaxation of the PA formulation for an arbitrary partial aggregation B^p. The following mapping transforms this solution to the solution (x_{ij}^D, y_{ij}) for the LP relaxation of FA formulation based on a full aggregation $B^f = \tilde{N}$ as explained in Section 4.2.

$$x_{ij}^{K_n} := \sum_{b \in B^p : o_b = n} \sum_{D \in G_b^{ij}} \bar{x}_{ij}^D \qquad \forall (i,j) \in A,\ n \in B^f \tag{4.6a}$$

$$y_{ij} = \bar{y}_{ij} \qquad \forall (i,j) \in A \tag{4.6b}$$

The solution obtained by mappings (4.6a) and (4.6b) is feasible for the FA formulation. The flow conservation constraint of FA is a summation of flow conservation constraints of dispersion with the same origin. The capacity constraint in both cases is the same since the flow of all commodities are included in one inequality for each arc. Similar to the flow conservation constraints, SIs of the FA formulation are summation of the SIs of relevant dispersions with the same origin. Hence, an arbitrary solution of PA formulation is always

feasible for the FA formulation as the FA formulation is obtained by aggregating some of the PA constraints.

Now we show an example where a solution of the DA LP relaxation that is not feasible for the PA LP relaxation. Consider an example of a partial aggregation that two commodities k_1 and k_2 are not aggregated on the arc (i, j). Moreover, $d^{k_1} + d^{k_2} > u_{ij}$, but $d^{k_1} < u_{ij}$ and $d^{k_2} < u_{ij}$. As in the FA formulation these commodities are aggregated together, a feasible solution is $x_{ij}^{fk_1,k_2g} = d^{k_1}$ and $y_{ij} = \frac{d^{k_1}}{u_{ij}}$. However, this solution is not feasible for the PA formulation based on its SIs for the arc (i, j).

Therefore, the PA formulation is stronger than the FA formulation. □

Theorems 4.4.1-4.4.4 imply a hierarchy of formulations in terms of LP relaxation strength. In this hierarchy, the DA formulation is the strongest formulation and the FA formulation is the weakest. Partially-aggregated formulations lie in between these two extreme cases. Moreover, as each commodity is included in one unique dispersion, we have $\sum_{b2B} \sum_{D2G_b^{ij}} x_{ij}^D = \sum_{k2K} x_{ij}^k$ which shows that the objective function of the partially-aggregated formulations is equivalent to the objective function of the DA formulation. Therefore, we can conclude Corollary 4.4.1.

Corollary 4.4.1. *PA, PAi, and PAe formulations are valid bounds for MCND.*

4.5 Computational Results

In this section we perform two sets of computational experiments to evaluate the proposed formulations. The first set of experiments investigates the LP relaxations of these formulations in terms of tightness and computing time. The second set of experiments compares the mixed-integer programming algorithms over the formulations. All computational experiments are run on a cluster with 4 Xeon-Gold-6150 cores and 8 GB RAM using CPLEX 12.10.0 via a Python API as LP and MIP solver. Default settings are always selected unless otherwise specified.

Computational experiments are conducted on 196 publicly available instances for MCND, called *Canad* instances. These instances were generated by Crainic et al. (2001) and used as benchmark instances by subsequent papers to evaluate different solution algorithms and relaxations. These instances cover a diverse range of number of commodities, capacity tightness, network density, and significance of fixed cost over the flow cost. The details of the instances can be found in Crainic et al. (2001). These instances are divided into three classes. Class *C* includes 31 large instances with dense networks and many commodities. Class *C+* includes 12 instances with sparser networks and few commodities. Class *R*

includes 153 small and medium size instances with dense networks but a smaller ratio of the number of commodities per number of nodes in comparison with the C class.

Table 4.1 gives a summary of instances considered. In three instances of the C+ class, each node is the origin of at most one commodity. Therefore, we ignore these three instances as all formulations are the same for such cases. As the formulations mainly affect the computing time of the LP relaxations, we distinguish a set of instances as *long* instances. A long instance is an instance that the computing time of its DA LP relaxation is longer than 10 seconds by the solver.

Table 4.1: Summary of the considered instances

Class	**C**	**C+**	**R**
#instances	31	9	153
#long instances	11	0	7
#nodes	{20,30}	{25,100}	{10,20}
#arcs	{230,300,520,700}	{100,400}	{35,60,85,120,220,320}
#commodities	{40,100,200,400}	{10,30}	{10,25,40,50,100,200}
Arc density	70%	11%	63%
Commodity density	31%	2%	31%
Commodity/node ratio	7.3	0.6	4.3
FA reduction	78%	24%	67%

The first part of Table 4.1 shows the number of total and long instances of each class. It also describes the possible values for the number of nodes, arcs, and commodities of instances in each class. The second part reports the average value of some characteristics of instances in each group. *Arc density* expresses how many of the possible arcs exist in an instance as $\frac{|A|}{|N|(|N|-1)}$. *Commodity density* presents the same notion for the commodities as $\frac{|K|}{|N|(|N|-1)}$. The *Commodity/node ratio* is the average number of commodities that originate from a node. *FA reduction* states the average reduction of the number of commodities in percentage across all instances by transforming the DA formulation ($|K|$ commodities) to the FA formulation ($|\tilde{N}|$ commodities). FA reduction gives an indication of possible size reduction for an instance by a commodity aggregation scheme. As the numbers in Table 4.1 suggest, there is more opportunity on average to reduce the size of the instances of C and C+ classes by commodity aggregations.

As we use Algorithm 1 in the rest of the thesis to obtain partially-aggregated formulations, we refer to PAi and PAe as PAi-K and PAe-K to develop and compare different formulations with respect to the number of shortest paths (K) used to construct dispersion of commodities.

4.5.1 Experimental Evaluation of the LP Relaxations

In this section we evaluate the LP relaxation of the proposed formulations. LP relaxations are obtained by relaxing Constraint (2.15f) to $0 \quad y_{ij} \quad 1$. First, we investigate the LP relaxations in terms of solution time, size, and LP bound by solving the LP model by CPLEX LP solver. Afterwards, the performance of the CPLEX cutting plane algorithm at the root node over the formulations is studied. In both cases, results indicate that formulations based on partial aggregation are on a Pareto frontier for the trade-off between the LP bound strength and the LP solution time. In our tests, the DA and FA formulation, which can be seen as extreme versions of both PAi-K and PAe-K, were compared against various forms of partial aggregation. These partial aggregations were created using the heuristic Algorithm 1 using the K shortest paths solutions for $K \; 2 \; f1,\ldots,5,10g$, giving rise to 12 partially aggregated formulations PAe-1,. . .,PAe-10 and PAi-1,. . .,PAi-10.

Figures 4.6-4.8 presents the results for solving the LP relaxations using CPLEX with default settings. In these figures, the LP bound of fully- and partially-aggregated formulations are compared with the LP bound of the DA formulation.

Bound loss measures the percentage of the LP optimal value decrease in a formulation with respect to the LP optimal value of the DA formulation. Figure 4.6 compares the bound loss of the partially-aggregated formulations against the bound loss of the fully-aggregated formulation over all instances. The y-axis value of a point shows the bound loss of the corresponding formulation over an instance, whereas the x-axis value of this point represents the bound loss of the FA formulation over the corresponding instance. The results demonstrate the effectiveness of partially-aggregation formulations as their bound loss is significantly less than the bound loss of the FA formulation, even when considering just one shortest path to construct the partial aggregations.

Figures 4.7 and 4.8 indicate the trade-off between the bound loss and the LP size/computing time. *Size reduction* is quantified over both dimensions, rows and columns, as the percentage change in the multiplication of the number of variables and the number of constraints of a formulation in comparison with the DA formulation. Similarly, *time reduction* is with respect to the computing time of the DA LP relaxation. These figures indicate that the size and solution time reduction of the partially-aggregated formulations are a substantial fraction of those achieved by the FA formulation. The time reduction is particularly significant over long instances. The average computing time of DA LP relaxation over long instances is 61.71 seconds. For instance, using the PAe-5 formulation reduces the average computing time by 85.6% to 8.92 seconds in the expense of 1.2% bound loss on average. On the other hand, the FA formulation reduces the computing time by 99.5% but with a significant bound loss (17.3% on average).

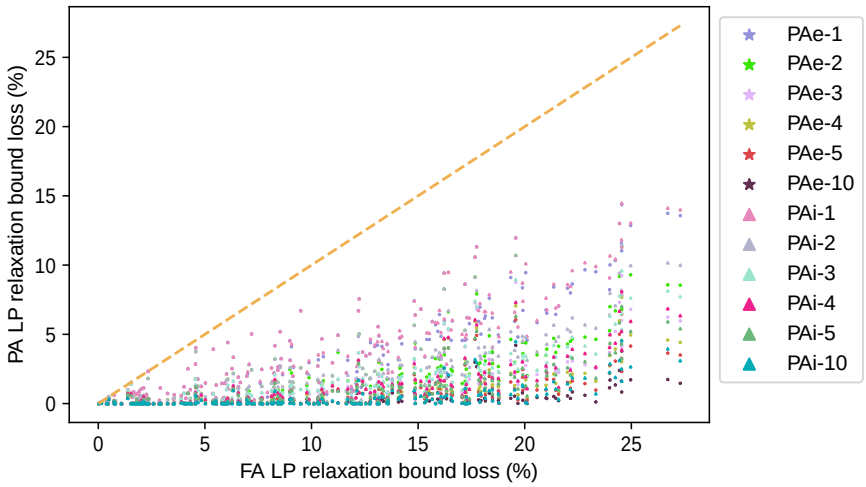


Figure 4.6: LP bound loss of partially-aggregated formulations versus LP bound loss of the FA formulation, considering the DA formulation as the base

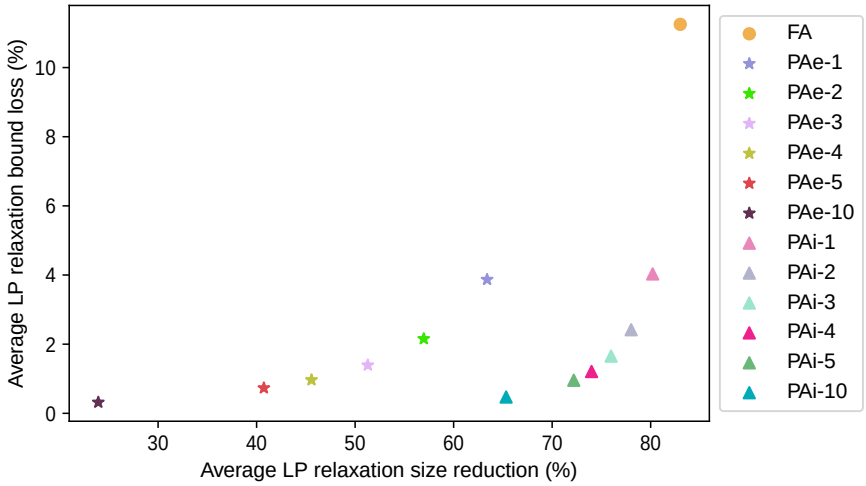


Figure 4.7: The trade-off between the LP bound loss and size for aggregated formulations considering the DA formulation as the base. Size of a formulation is considered as the multiplication of the number of variables and the number of constraints it includes.

Figure 4.9 depicts the average scaled number of variables, flow conservation constraints, and strong inequalities for all formulations considering the dimensions of the DA formulation as the base. Only the partial formulations constructed using 5 shortest paths are shown in this figure. However, the trend can be inferred for other values. Moreover, Constraints (4.1g) and (4.1h) are considered as flow conservation constraints for PAi-5. Partial formulations include more flow conservation constraints, particularly PAe-5. However, the number of strong inequalities and the number of variables are reduced leading

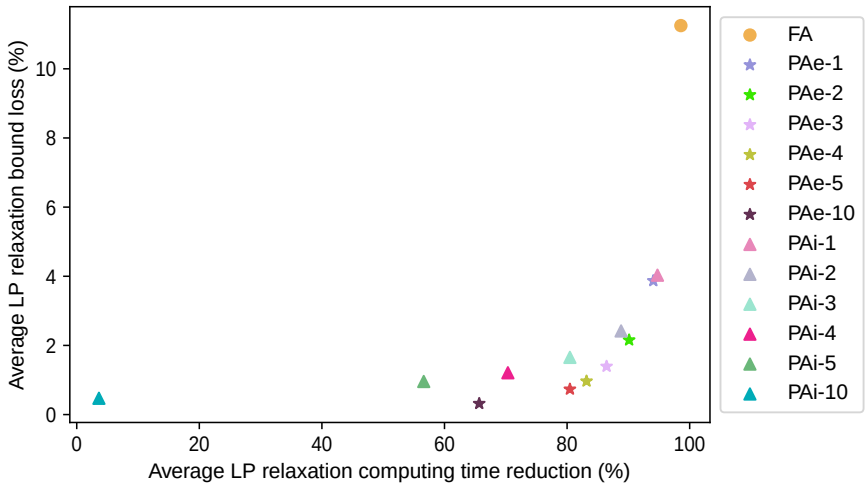


Figure 4.8: The trade-off between the LP bound loss and computing time for aggregated formulations considering the DA formulation as the base

to faster LP solve times. Mostly, strong inequalities make a difference in a formulation in terms of the LB bound and computing time. In fact, without any SI, the LP bound of all formulations are the same, and their computing time are in the same order. The number of SIs in the partial formulations (with the ratio of 0.41) is slightly more than the number of SIs in the FA formulation (with the ratio of 0.34). This implies an average of 59% reduction in the number of SIs by partial formulations and 66% reduction by the FA formulation. However, the slightly more SIs leads to significant error bound reduction as shown earlier.

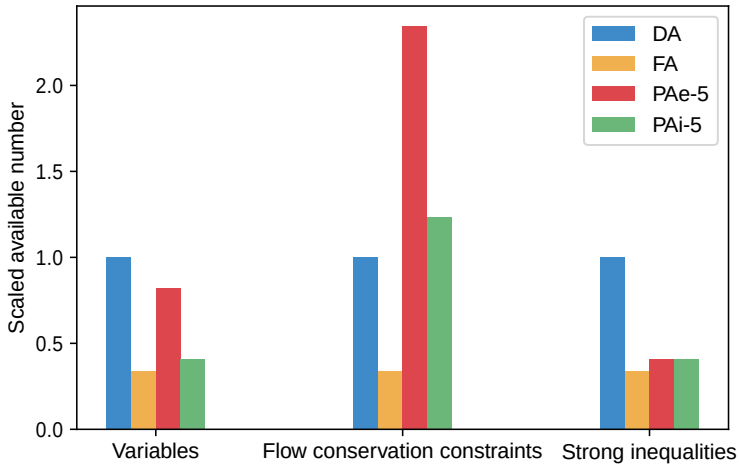


Figure 4.9: Structure of different formulations in terms of scaled number of variables and constraints, considering the dimensions of the DA formulation as the base

Because of the artificial nodes and arcs, the size of the PAe-K formulation is generally

larger than the PAi-K formulation. However, the larger size does not translate to longer computing time as seen in Figure 4.8. The shorter computing time of PAe-K despite a larger size is due to different structures of PAe-K and PAi-K. The PAe-K formulation, in contrast to PAi-K, keeps the multicommodity network flow structure as it adds only equality flow conservation constraint. Another characteristic that affects the LP solution times is the density of nonzeros in the formulations. Figure 4.10 illustrates the average nonzero density of each formulation with respect to the average solution time over all instances. Nonzero density is obtained by dividing the number of nonzeros by the size of the LP as the number of constraints multiplied by the number of constraints. In general, the nonzero density of PAe-K is less than the nonzero density of PAi-K which benefits the LP solution time.

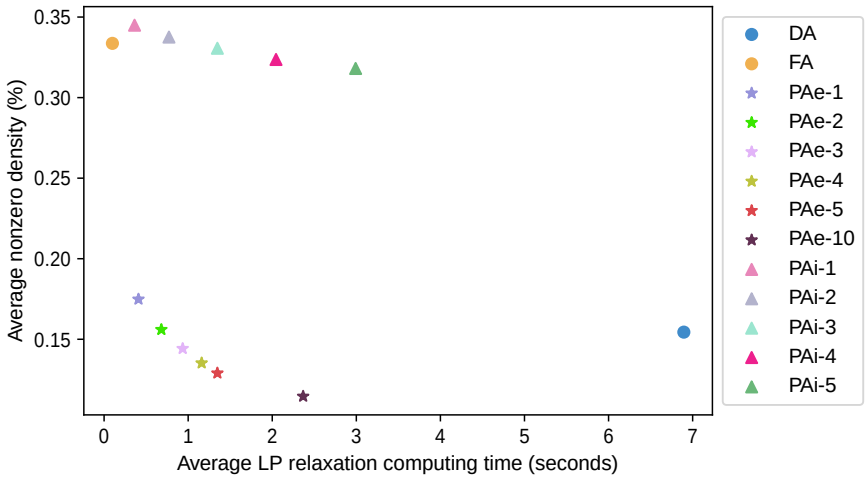


Figure 4.10: LP nonzeros density versus LP solution time

The strength of the lower bound is significantly impacted by the addition of cuts during the MIP solution process. To assess the effect of this, Table 4.2 presents the summary results of the CPLEX cutting plane algorithm over all formulations. The results are obtained by restricting the CPLEX to 0 node of B&B tree. Moreover, the best-known solutions are given as the initial incumbent to reduce the effect of heuristics on the computing times. Bound losses are with respect to the DA LP relaxation. Therefore, negative bound losses in the cutting plane algorithm column implies that the LP bound is improved in comparison with the DA LP relaxation. Based on Table 4.2, the significant bound loss of FA LP relaxation is compensated by the cutting plane algorithm. The DA formulation gives the best bound by using the cutting plane algorithm, however, by a remarkably higher computational effort. It is interesting that the partially- and fully-aggregated formulations, by using the cutting plane algorithm, are able to provide better bounds in comparison with the DA LP relaxation in almost half computing time. Figure 4.11 displays the data from Table 4.2 in a graphical

form by showing the average scaled LP bound for each formulation, considering the LP relaxation of DA formulation as the base. As Figure 4.11 shows that in both cases, the LP bound and the LP bound by the cutting plane algorithm, there are partial formulations that are on the Pareto frontier for the trade-off between the LP bound and the computing time. The cutting plane algorithm is significantly faster over aggregated formulations in comparison with the DA formulation.

Table 4.2: Effect of formulations on the performance of the CPLEX cutting plane algorithm

Formulation	LP relaxation		Cutting plane algorithm		
	Time	Bound loss	Time	Bound loss	#Cuts
DA	6.9	0.00%	19.8	-1.28%	56
FA	0.1	11.25%	3.2	-0.89%	289
PAe-1	0.4	3.87%	2.8	-0.68%	355
PAe-2	0.7	2.16%	3.1	-0.66%	279
PAe-3	0.9	1.39%	3.3	-0.76%	230
PAe-4	1.2	0.97%	3.9	-0.86%	202
PAe-5	1.3	0.73%	4.3	-0.93%	177
PAe-10	2.4	0.32%	6.2	-1.08%	120
PAi-1	0.4	4.03%	2.8	-0.83%	242
PAi-2	0.8	2.42%	3.4	-0.88%	213
PAi-3	1.3	1.65%	4.2	-0.94%	194
PAi-4	2.0	1.21%	5.0	-0.95%	178
PAi-5	3.0	0.96%	5.8	-1.01%	166
PAi-10	6.6	0.47%	10.3	-1.12%	134

4.5.2 Solving the MIP Model

In this section we investigate the performance of the mixed-integer programming algorithms over the proposed formulations. In particular, we examine the CPLEX MIP solver with default settings, with cutting plane algorithm disabled, and using automatic Benders decomposition algorithm to demonstrate the impact of partial aggregations on various MIP algorithms. The tightness and computing time of a formulation's LP relaxation impact the performance of the B&B algorithm. Additionally, the CPLEX heuristics' and the cutting plane algorithm's performance and the strength of Benders cuts vary over the proposed formulations. We consider the FA and the DA formulations as existing models in the literature. Among the partial formulations, we employ PAe-5 and PAi-5 as they possess a satisfactory trade-off between the LP bound and computing time.

The computing time for each instance is limited to 1 hour. In Tables 4.3-4.7, instances are divided into three groups: *easy*, *medium*, and *difficult* instances. Easy instances are in-

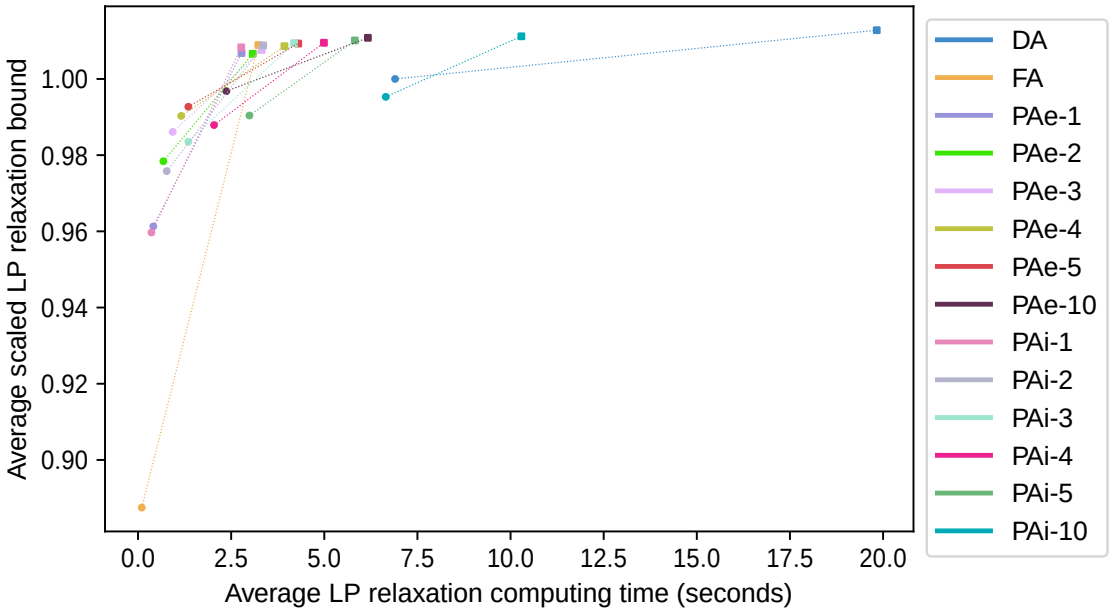


Figure 4.11: Average scaled LP bounds of the formulations with respect to their average computing time. Circle nodes correspond to the LP relaxation, and square nodes correspond to the LP bound by the cutting plane algorithm. The DA LP relaxation is considered as the base for scaling.

stances that all formulations have been able to prove optimality for within the time limit. Medium instances are instances for which some but not all formulations have proved optimality for. The other instances are classified as difficult. Note that the division of instances into the three groups and identifying the best solutions are based on the comparisons among the formulations under the corresponding settings of each table separately. Therefore, the difficulty classification of instances and the best solutions vary across tables. All numbers are reported as averages across instances in each group. The numbers in the parenthesis show the number of instances for each group. The notation used in the tables are as follows: *Time*: solution time, *#n*: the number of explored nodes in the B&B tree, *#B*: the number of instances for which the best solution is obtained by the corresponding formulation, *#O*: the number of instances where the corresponding formulation has been able to prove optimality within the time limit, *B%*: gap from the best solution, *G%*: optimality gap for the instances where their optimality has not been proven in the time limit.

Table 4.3 reports the average performance of the CPLEX default MIP solver over the formulations per each instance class and overall. According to these results, the FA formulation outperforms other formulations over all instance groups and classes, in terms of proving the optimality and finding good-quality, feasible solutions. Analysis of the CPLEX logs shows that the heuristics perform better over the FA formulation. Moreover, although

the FA formulation suffers from a weak LP bound, the cutting plane algorithm resolves this issue. These results are very surprising given that the latest study by Chouman et al. (2017) showed that the DA formulation outperforms the FA formulation. The DA formulation proves the optimality of easy instances in slightly shorter times than the partial formulations. However, the partial formulations perform better over the difficult instances, particularly to obtain good-quality solutions in comparison with the DA formulation. Moreover, this better performance is intenser for the large instances with many commodities, the C class. As the trend for long instances is similar, the results are not reported separately.

Table 4.3: Performance of the MIP solver with default setting over different formulations for all instances

Instance type	DA			FA			PAe-5			PAi-5		
	Time		#n	Time		#n	Time		#n	Time		#n
Easy - C (12)	423.8		5,043	**310.8**		15,449	364.2		5977	545.9		11518
Easy - C+ (8)	219.6		1,846	**207.7**		1,881	298.4		2,151	223.9		1,759
Easy - R (131)	119.3		1,843	**87.0**		4,891	149.2		3,443	151.1		3,703
Easy - all (151)	*148.8*		*2,097*	*111.1*		*5,570*	*174.2*		*3,576*	*186.3*		*4,221*
	#B	#O	G%	#B	#O	G%	#B	#O	G%	#B	#O	G%
Medium - C (3)	**3**	**1**	**0.22%**	**3**	**1**	0.47%	**3**	**1**	0.54%	2	0	0.64%
Medium - R (5)	4	**3**	**0.40%**	4	**3**	1.23%	**5**	1	0.71%	4	1	0.93%
Medium - all (8)	*7*	*4*	*0.31%*	*7*	*4*	*0.84%*	*8*	*2*	*0.65%*	*6*	*1*	*0.81%*
	#B	B%	G%	#B	B%	G%	#B	B%	G%	#B	B%	G%
Difficult - C (16)	3	2.40%	3.54%	**11**	**0.02%**	**1.40%**	2	0.21%	1.77%	*7*	0.10%	1.87%
Difficult - C+ (1)	0	1.59%	5.77%	0	0.57%	4.57%	0	1.24%	5.30%	**1**	**0.00%**	**4.10%**
Difficult - R (17)	6	0.21%	2.20%	**9**	**0.06%**	**1.95%**	3	0.17%	2.32%	3	0.14%	2.40%
Difficult- all (34)	*9*	*1.28%*	*2.94%*	*20*	*0.06%*	*1.76%*	*5*	*0.22%*	*2.15%*	*11*	*0.12%*	*2.20%*

As Table 4.3 shows, the cutting plane algorithm plays a significant role in the performance of the MIP solver over a formulation. Therefore, to investigate the impact of the trade-off between the LP bound and computing time on the B&B tree, we disable the CPLEX cutting plane algorithm. Based on Table 4.4, the partial formulations outperforms the DA and FA formulations in general. PAi-5 performs better over easy instances, while PAe-5 provides smaller optimality gaps over medium and difficult instances. Although heuristics are still able to provide good-quality solutions for the FA formulation, the optimality gaps are significantly poorer due to its weak LP bound. Table 4.5 presents the results over long instances. The superiority of the partial formulations is more significant over the large instances with many commodities (C class) and also long instances.

Benders decomposition algorithms have been successfully applied to MCND and its extensions as an effective solution algorithm (Costa, 2005; Costa et al., 2007; Zetina et al., 2019). In the next set of experiments, we evaluate the performance of the Benders

Table 4.4: Performance of MIP solver with cutting plane algorithm disabled over different formulations for all instances

Instance type	DA			FA			PAe-5			PAi-5		
	Time		#n	Time		#n	Time		#n	Time		#n
Easy - C (7)	1.2		1,115	341.1		1,017,368	**1.1**		989	1.4		1564
Easy - C+ (6)	**1.9**		2,176	2.2		4,309	2.8		2,403	2.3		2,402
Easy - R (102)	41.7		2,326	98.3		242,590	35.0		2,718	**21.4**		3,918
Easy - all (115)	*37.2*		*2,244*	*108.0*		*277,318*	*31.3*		*2,596*	*19.1*		*3,695*
	#B	#O	G%	#B	#O	G%	#B	#O	G%	#B	#O	G%
Medium - C (4)	**4**	**4**	–	**4**	0	10.22%	**4**	**4**	–	**4**	2	0.83%
Medium - R (33)	32	32	0.60%	31	0	7.53%	**33**	27	1.23%	**33**	24	2.53%
Medium - all (37)	*36*	*36*	*0.60%*	*35*	*0*	*7.83%*	*37*	*31*	*1.23%*	*37*	*26*	*2.22%*
	#B	B%	G%	#B	B%	G%	#B	B%	G%	#B	B%	G%
Difficult - C (20)	7	1.91%	2.97%	8	0.14%	12.56%	9	0.09%	**1.60%**	**11**	**0.07%**	2.35%
Difficult - C+ (3)	2	0.10%	**3.10%**	**3**	**0.00%**	3.96%	2	0.34%	3.96%	2	0.29%	3.87%
Difficult - R (18)	6	0.17%	**2.28%**	7	0.11%	10.86%	6	0.12%	2.83%	**10**	**0.07%**	3.53%
Difficult - all (41)	*15*	*1.02%*	*2.67%*	*18*	*0.12%*	*11.18%*	*17*	*0.13%*	*2.32%*	*23*	*0.08%*	*2.98%*

Table 4.5: Performance of MIP solver with cutting plane algorithm disabled over different formulations for long instances

Instance type	DA			FA			PAe-5			PAi-5		
	#B	#O	G%	#B	#O	G%	#B	#O	G%	#B	#O	G%
Medium - R (2)	1	**1**	**0.60%**	**2**	0	10.46%	**2**	0	1.09%	**2**	**1**	4.59%
	#B	B%	G%	#B	B%	G%	#B	B%	G%	#B	B%	G%
Difficult - C (11)	3	3.35%	4.27%	3	0.19%	13.20%	**5**	0.12%	**1.73%**	4	**0.05%**	2.46%
Difficult - R (5)	0	0.30%	**2.66%**	1	0.15%	12.14%	1	0.12%	2.82%	**3**	**0.11%**	3.76%
Difficult - all (16)	*3*	*2.40%*	*3.77%*	*4*	*0.18%*	*12.87%*	*6*	*0.12%*	*2.07%*	*7*	*0.07%*	*2.87%*

decomposition algorithm over the proposed formulations. We use the CPLEX automatic Benders decomposition algorithm. Our aim is to provide indicative results on the effects of different level of aggregations on the performance of the Benders algorithm. To solve the problem of course specialized Benders algorithms would be more efficient. All settings are left at their default, except the presolve which is disabled. According to our experiments, the presolve destroys the special structure of the formulations and increases the LP solution time. Moreover, we add single-node cut-set constraints for the source and sink nodes to assist the feasibility of the master problem as it starts with only the constraint set (2.15f). The added constraints are as (4.7a) and (4.7b). These constraints improve the performance of the Benders algorithm over all formulations on average.

$$\sum_{j \in N_i^+} u_{ij} y_{ij} \quad \sum_{k \in K : o^k = i} d^k \qquad \forall i \in N \tag{4.7a}$$

$$\sum_{j \in N_i} u_{ji} y_{ji} \qquad \sum_{k \in K : s^k = i} d^k \qquad \forall i \in N \tag{4.7b}$$

Table 4.6 displays the summary results for the performance of the Benders algorithm over all formulations. The DA formulation outperforms other formulations on average. However, the PAe-5 formulation has still the best performance on the difficult instances of C class in terms of the quality of the solutions and the optimality gaps. The trend differs over the long instances as Table 4.7. Over these instances, the PAe-5 performs the best in terms of all of the criteria. Moreover, the PAi-5 outperforms the DA and FA formulations. This particularly emphasizes the effectiveness of the formulations based on partial aggregations as their impact is more significant over long instances to reduce the LP computing time. The results indicate that the subproblems of the partial formulations benefits from shorter computing times, yet generate strong Benders cuts.

Table 4.6: Performance of the Benders algorithm over different formulations with added single-node cut-set constraints (4.7a) and (4.7b) for all instances

Instance type	DA			FA			PAe-5			PAi-5		
	Time		#n	Time		#n	Time		#n	Time		#n
Easy - C+ (5)	**137.7**		210,528	540.4		539,608	191.3		255,806	433.6		375,894
Easy - R (75)	**28.5**		11,424	130.3		115,893	39.6		21,475	39.3		17,111
Easy - all (80)	*35.3*		*23,868*	*156.0*		*142,375*	*49.1*		*36,121*	*63.9*		*39,535*
	#B	#O	G%	#B	#O	G%	#B	#O	G%	#B	#O	G%
Medium - C (8)	**8**	**8**	–	1	0	5.42%	7	7	0.27%	7	7	0.57%
Medium - C+ (1)	**1**	**1**	–	1	0	0.43%	**1**	**1**	–	**1**	**1**	–
Medium - R (29)	**29**	**29**	–	0	0	9.52%	26	24	1.66%	24	23	1.52%
Medium - all (36)	*38*	*38*	–	*2*	*0*	*8.42%*	*34*	*32*	*1.43%*	*32*	*31*	*1.39%*
	#B	B%	O%	#B	B%	O%	#B	B%	O%	#B	B%	O%
Difficult - C (23)	**13**	3.67%	6.54%	0	5.69%	23.59%	4	**0.73%**	**5.43%**	6	1.03%	6.33%
Difficult - C+ (3)	**1**	**0.14%**	**11.13%**	**1**	6.89%	16.74%	0	5.06%	14.52%	**1**	6.68%	15.71%
Difficult - R (49)	**28**	**0.63%**	**5.61%**	0	4.70%	21.69%	16	0.90%	7.36%	7	1.32%	8.19%
Difficult - all (75)	*42*	*1.54%*	*6.11%*	*1*	*5.09%*	*22.08%*	*20*	*1.01%*	*7.06%*	*14*	*1.45%*	*7.92%*

Table 4.7: Performance of the Benders algorithm over different formulations with added single-node cut-set constraints (4.7a) and (4.7b) for long instances

Instance type	DA			FA			PAe-5			PAi-5		
	#B	B%	O%	#B	B%	O%	#B	B%	O%	#B	B%	O%
Difficult - C (11)	4	7.66%	9.84%	0	6.05%	23.81%	2	**0.51%**	**5.67%**	**5**	0.60%	6.28%
Difficult - R (7)	0	2.58%	8.89%	0	6.29%	27.89%	**5**	**0.58%**	**8.38%**	2	2.53%	10.93%
Difficult - all (18)	*4*	*5.68%*	*9.47%*	*0*	*6.14%*	*25.40%*	*7*	*0.54%*	*6.72%*	*7*	*1.35%*	*8.09%*

The summary of the results on solving the MIP model is presented as performance profiles in Figure 4.12. Such performance profiles are proposed by Dolan and Moré (2002)

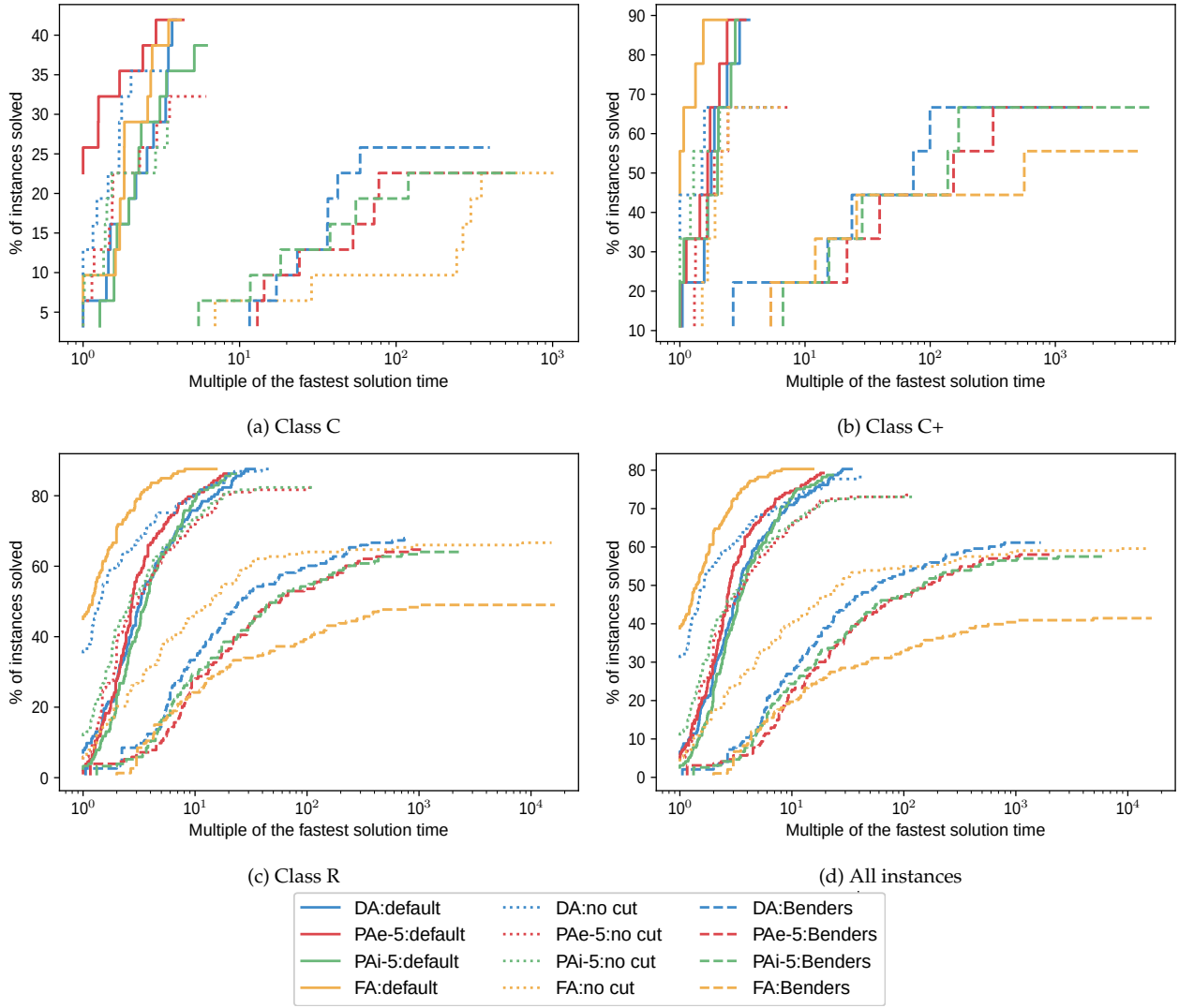


Figure 4.12: Performance profile of the formulations based on different aggregation schemes with respect to solution time (log scale)

to evaluate and compare optimization solvers. A performance profile shows the percentage of instances solved to the optimality within the multiple time of the fastest solution time of each instance. Performance profiles are reported per instance class and overall. Generally, the default setting performs better as expected. For the class C of instances, the PAe-5 formulation performs the best. This formulation proves the most percentage of these instances in less time in comparison with other formulations. Moreover, it solves more instances in the fastest time in comparison with other formulations. In other words, PAe-5 using CPLEX with the default settings is able to prove the optimality of almost 50% of instances, which their optimality is proven in the time limit, in the fastest time. This is particularly inter-

esting since this class of instances include large instances with many commodities, and partial aggregation is effective on such instances. The trend for other instance classes and average indicate the superiority of the FA formulation by using CPLEX with the default settings as discussed before. Benders decomposition and FA with cutting plane algorithm disabled performs poorly in comparison with other formulations. However, our goal for Benders decomposition experiments is to study the impacts of aggregation schemes on its performance and not to compare against state-of-the-art technology. While the cutting plane algorithm significantly improves the performance of the FA formulation, the DA formulation with cutting plane algorithm disabled performs better than the DA with default settings. In general, in shorter computing times, the DA with cutting plane algorithm disabled perform better than the PAe-5 with the default setting. However, the performance profile of the PAe-5 with the default setting surpasses the DA with cutting plane algorithm disabled as the time passes.

4.6 Conclusions and Future Research

This chapter introduces new commodity representations for multicommodity network flow problems. Such commodity representations are utilized to aggregate constraints and variables in the MIP model partially instead of the conventional full aggregation. We apply the proposed partial aggregations to the multicommodity fixed-charge network design problem to show the implications of this approach. However, they are applicable to any other problem that can be formulated with a multicommodity network flow subproblem, and for which there exist an alternative fully-aggregated formulation. We propose two variants of MIP formulations based on this partial aggregation concept and a heuristic algorithm for constructing partial aggregations of various sizes. These MIPs are proven to provide valid lower bounds for MCND which are tighter than the fully aggregated formulation. The proposed MIPs and their LP relaxations are evaluated empirically through an extensive computational study on benchmark instances. The results indicate that the partial formulations have significantly shorter computing times in comparison with the DA formulation with only small loss in the LP bound in contrast to the FA formulation. The proposed formulations have dimensions in between the dimensions of the DA and FA formulations. Furthermore, the proposed models form a Pareto frontier for the trade-off between the LP bound and computing time, whether only the LP relaxation is solved or a cutting plane algorithm is employed. We investigate the performance of the mixed-integer programming algorithms over the formulations. The PAe-5 formulation is particularly effective for the large instances with many commodities and long instances. In addition, the proper trade-off between the LP bound and computing time benefits the partial formulations in the B&B

algorithm. The partial formulations are also beneficial for long instances using a Benders decomposition algorithm since the reduced sub-problems have shorter computing times yet generate effective cuts.

We propose three possible directions for future research. First, partially-aggregated formulations can be employed to develop specialized solution algorithms. Particularly, our early experiments show that the Benders decomposition algorithm performs well over the partial formulations for the long instances and large instances with many commodities. The second direction is to develop valid cuts for the partial formulations. The DA and FA formulations have been extensively studied to develop efficient cutting plane algorithms. Similarly, cutting plane algorithms would be beneficial to solvers using partial aggregations. The third direction is to apply this partial aggregation approach to other multicommodity network flow problems and study the implications. The extensions are particularly straightforward for the problems where there is no commodity-specific attribute over the underlying network. As an example of such research directions, next chapter demonstrates that how utilizing partial aggregations leads to the development of an efficient solution algorithm for a problem with the multicommodity network flow substructure. The problem studied is an abstract version of the LFMIR problem.

CHAPTER 5

From Partial Aggregations to Solution Algorithms

5.1 Overview

This chapter demonstrates that partial aggregations can be used to develop efficient algorithms for large network optimization problems with a multicommodity network flow substructure. It applies the concept of partial aggregations introduced in Chapter 4 to a variant of the locomotive fuel management problem (LFMIR) defined in Chapter 3. In Section 5.2, we define an abstracted version of LFMIR, called Inline Fuel Delivery problem (*IFD*),which retains the main complexities arising from incorporating the inline tank planning in the fuel management problem. We prove that this problem is also NP-hard in Section 5.3. IFD is modelled based on a multicommodity network flow representation in Section 5.4. We propose an aggregated formulation for IFD and improve it in Section 5.5. However, unlike the MCND problem, when applying aggregations to this problem, one may need to deal with the cases that the solution is infeasible. We discuss such cases in Section 5.6. In Section 5.7, we propose partial aggregations and formulations for IFD and employ them to develop an exact solution algorithm for the IFD problem. Computational experiments in Section 5.8 confirm the efficiency of the proposed algorithm in providing good-quality upper and lower bounds in short computing times in comparison with solving the disaggregated MIP model by a MIP solver.

The approach presented in this chapter provides an example of the use of partial aggregations in an algorithmic approach to:

1. Provide guides to generate good-quality upper bounds when the solution of the (partially-) aggregated formulation is infeasible for the original problem, and
2. Employ the concept of partial aggregations to develop efficient, specialized solution algorithms.

5.2 The Inline Fuel Delivery Problem

This section introduces an abstracted version of the LFMIR problem, called the inline fuel delivery problem, *IFD*. Our motivation is to define a problem that retains the main complexity driving factor of LFMIR while it is easy to understand how the proposed algorithm performs and tackles this complexity. The main complexity factor of LFMIR is the huge number of switch opportunities for inline tanks between the locomotives which expands the solution space combinatorially. We design the IFD problem in a way that includes this feature as we will show. Moreover, the operational details are dropped to study the performance of the proposed algorithm for tackling this feature.

IFD seeks to minimize the total fuel purchasing costs. The main restriction is that sufficient fuel must be supplied to all locomotives. The given information includes the set of stations S, the set of locomotives L, and the set of inline tanks W. Each locomotive $\ell \in L$ makes a single trip from its origin station o_ℓ to its destination station d_ℓ, which requires D_ℓ gallons/litres of fuel, which can be purchased from its origin station at price C_ℓ (\$ per unit of fuel), or (in part) provided by an inline tank. The departure time a_ℓ and the arrival time w_ℓ of the trip of each locomotive are given and fixed. At the beginning of the time horizon, each inline tank $w \in W$ starts from the initial station s_w and can be refueled only once at this initial station at price C_w (\$ per unit of fuel). The fuel capacity of each inline tanks is limited to P.

The required fuel of a locomotive can be supplied directly from its origin station, from the assigned inline tanks to it during the trip, or any combination of both. The inline tanks assigned to a locomotive travel along with it from the origin station of the locomotive to its destination. Fuel cannot be transferred between inline tanks at the stations or during the trips. The number of inline tanks per locomotive is not limited. Moreover, it is not required that the plan of inline tanks be cyclic. Given all required information, IFD consists in determining the minimum-cost assignment of inline tanks to locomotives and the fuel plan of the inline tanks. The fuel plan of each inline tank is the initial refueling amount and the amount of fuel delivery by it to the assigned locomotives. Having the fuel delivery

amounts by the inline tanks, one can determine the minimum amount of fuel to purchase for each locomotive at its origin station.

5.3 Complexity of the Problem

In this section, we study the theoretical complexity of the IFD problem. The proof is a reduction from 3-Partition, which decides whether it is possible to partition a multiset of integer numbers into 3 subsets (or k subsets in general) of equal sum. This problem is proven to be weakly NP-hard when $k = 2$ (Karp, 1972) and strongly NP-hard when $k \quad 3$ (Garey and Johnson, 1979).

Theorem 5.3.1. *The IFD problem is weakly NP-hard when there are two inline tanks available and is strongly NP-hard when there are at least three inline tanks available.*

Proof. Consider an instance of the k-way number partition decision problem with the multiset $S = \{a_1, a_2, \ldots, a_n\}$, where $\sum_{i=1}^{n} a_i = kT$. The problem is to answer if it is possible to partition S into k subsets, each with sum of T.

In order to reduce the partition instance, we consider different gadgets in the reduction that each makes a part of the reduced instance and have different characteristics as the following:

- Integer locomotives: there is one locomotive for each number a_i in the multiset S. For each locomotive $i \in \{1, \ldots, n\}$, we have $D_i = a_i$, $C_i = 1$, $a_i = 2i \quad 1$, $w_i = 2i$, $o_i = s_i$, and $d_i = s_{i+1}$. Therefore, the set of integer locomotives L includes n locomotives such that their trip times are mutually exclusive. Moreover, the stations visited by these locomotives, in the order of the indices, make a sequence in the form of $(s_1, s_2, \ldots, s_{n+1})$.
- Decimal locomotives: there are $k \quad 1$ locomotives per each number a_i in the multiset S, each have the same characteristic as their corresponding integer locomotive in the set L, except the fuel consumption/demand which is equal to $1/n$. Formally, there are $k \quad 1$ decimal locomotives for each $i \in L$, where $D_i^j = \frac{1}{n}$, $C_i^j = C_i$, $a_i^j = a_i$, $w_i^j = w_i$, $o_i^j = o_i$, and $d_i^j = d_i$ for each $j \in \{1, \ldots, k \quad 1\}$. Therefore, in total there are $n(k \quad 1)$ locomotives in the set of decimal locomotives $\bar{L}$.
- The free locomotive: there is one locomotive l^0 that makes a trip from the *free* station s_0, in which fuel price is equal to zero. This locomotives leaves s_0 at the time zero and arrives at the station s_1 one time unit later. Formally, we have $D_0 = 0$, $C_0 = 0$, $a_0 = 0$, $w_0 = 1$, $o_0 = s_0$, and $d_0 = s_1$.
- Inline tanks: there are k inline tanks, all initially located at the station s_0 and each has

the capacity of $P = T + \frac{n-1}{n}$.

Considering this reduction, we show that an instance of the k-way number partition decision problem is satisfiable if and only if there is an optimal solution with total cost of zero for its corresponding instance of IFD. Here, we consider each inline tank as a subset that includes corresponding numbers of the "integer locomotives" which receive fuel from that inline tank.

The total fuel consumption of all locomotives in the reduced instance is equal to $kT + k - 1$ and the total capacity of the inline tanks is equal to $kT + k\frac{n-1}{n}$. Since $kT + k\frac{n-1}{n} > kT + k - 1$ and inline tanks can be refueled initially with the cost of zero, there is enough capacity in the available inline tanks to deliver free fuel to all locomotives. In particular, when an instance of the k-way number partition decision problem is satisfiable, we can translate the numbers in each subset to assignments of an inline tank to the corresponding integer locomotives. Moreover, the corresponding inline tank of a subset is assigned to the relevant decimal locomotives, when the corresponding number of an integer locomotive is not included in that subset.

Conversely, assignments of inline tanks to integer locomotives correspond to including the corresponding number in the subset that is represented by each inline tank. These assignments, if having the total cost of zero, are equivalent to a solution of the k-way number partition decision problem because of the two reasons. First, to have the cost of zero, exactly one inline tank must be assigned to each locomotive, whether an integer or a decimal locomotive. Therefore, each number is included in exactly one subset. Second, since one inline tank is assigned to each integer locomotive and the inline tanks capacity is $T + \frac{n-1}{n}$, each inline tank can deliver at most T units of fuel to integer locomotives. Here, all fuel demand is covered by the inline tanks, and hence, each inline tank delivers exactly T units of fuel to integer locomotives, which translates to the sum of T for the corresponding subsets.

In addition, the IFD problem is in the class NP since every solution for this problem can be certified in polynomial time whether all of the demands are satisfied or not. Furthermore, the reduction is polynomial with respect to n and k. Therefore, the IFD problem is weakly NP-hard with 2 inline tanks available and is strongly NP-hard with at least 3 inline tanks available.

□

5.4 Mathematical Modeling

In this section a MIP model is proposed for the IFD problem. We first conceptualize the problem as a time-expanded network. Then, a MIP model, with an underlying multicommodity network flow structure, is proposed.

5.4.1 Time-space network

We propose a time-space network for the IFD problem in this section. As the IFD problem is an abstracted version of the LFMIR problem, its network representation is simpler than the network proposed in Section 3.4.1. The IFD network conceptualizes the movement of inline tanks, which are of two types: (a) staying at a station for a time period, (b) making a trip along with a locomotive.

Nodes of the proposed network correspond to stations at different *event times*, where each event time corresponds to an event (a locomotive arrival or departure) related to that station. Formally, nodes corresponds to pairs (i, t_{ij}), where $i \ 2 \ S$, $t_{ij} \ 2 \ T_i$, and T_i is the set of all event times of the station i and $T_i = f0g \ [\ fa_l : o_l = i, l \ 2 \ Lg \ [\ fw_l : d_l, l \ 2 \ Lg$.

Arcs of the proposed network represent the two types of movements by *idle arcs* and *trip arcs*. An idle arc connects two consecutive nodes of a station, $(i, t_{ij}) \ ! \ (i, t_{ij+1})$, which represents that an inline tank stays at station i during time period $[t_{ij}, t_{ij+1}]$. Each trip by a locomotive is represented by a trip arc. A trip arc that represents the trip of locomotive l connects the node that represents the departure time of l from the origin o_l to the node that represents the arrival time of l to the destination d_l, i.e. $(o_l, a_l) \ ! \ (d_l, w_l)$. Trip arcs have a demand equal to the fuel consumption of the corresponding locomotive. Note that as this network corresponds to trains schedules and also we drop the cyclicity requirement in IFD, the proposed time-space network is an acyclic directed network.

An example of such time-space network is shown in Figure 5.1. The solid black nodes represent the initial location of the inline tanks. This network represents 3 stations (s_1, s_2 and s_3), 4 locomotive trips ($l_1, \ldots, l_4$), and 2 inline tanks (w_1 and w_2).

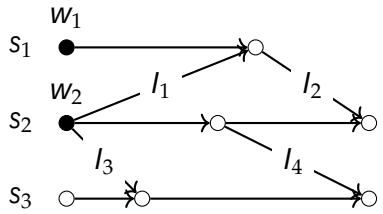


Figure 5.1: The time-space representation of the IFD problem

5.4.2 The MIP model

Fuel cannot be transferred between inline tanks, either at stations or during trips. Therefore, to develop the MIP model, we consider each inline tank as a distinct commodity. This results in a multicommodity type formulation, which in fact replicates the proposed time-space network in Section 5.4.1 for each inline tank. The following notation and decision variables are adopted for the proposed MIP.

Sets and parameters

- A Set of arcs of the time-space network
- $\tilde{A}$ Set of trips arcs of the time-space network
- O_v Set of outgoing arcs of node v in the time-space network
- I_v Set of incoming arcs of node v in the time-space network
- V Set of nodes of the time-space network with at least one outgoing arc
- W Set of available inline tanks
- a_v^w A binary parameter; 1 if node v corresponds to the initial location of the inline tank w, 0 otherwise
- D_a Fuel demand of trip arc a
- C_a Price of direct fuel purchase for demand of trip arc a
- C_w Fuel price at the initial location of inline tank w
- P Inline tank fuel capacity

Decision variables

- x_a^w 1, if inline tank w is assigned to arc a, 0 otherwise
- u_a^w The amount of fuel delivery to arc a by inline tank w
- r_a The amount of direct fuel purchase for the demand of arc a
- r_w The amount of initial refueling of inline tank w

The proposed MultiCommodity Formulation for IFD (IFD-MCF) is as follows.

$$\text{(IFD-MCF)} \quad Min \sum_{w \in W} C_w r_w + \sum_{a \in \tilde{A}} C_a r_a \tag{5.1a}$$

$$s.t: \quad \sum_{a \in O_v} x_a^w - \sum_{a \in I_v} x_a^w = a_v^w \qquad \forall w \in W, v \in V \tag{5.1b}$$

$$u_a^w \geq D_a x_a^w \qquad \forall w \in W, a \in \tilde{A} \tag{5.1c}$$

$$\sum_{w \in W} u_a^w + r_a = D_a \qquad \forall a \in \tilde{A} \tag{5.1d}$$

$$\sum_{a \in \tilde{A}} u_a^w = r_w \qquad \forall w \in W \tag{5.1e}$$

$$0 \leq r_w \leq P \qquad \forall w \in W \tag{5.1f}$$

$$0 \leq r_a \leq D_a \qquad \forall a \in \tilde{A} \tag{5.1g}$$

$$0 \leq u_a^w \leq D_a \qquad \forall w \in W, a \in \tilde{A} \tag{5.1h}$$

$$x_a^w \in \{0,1\} \qquad \forall w \in W, a \in A \tag{5.1i}$$

The objective function (5.1a) minimizes the total fuel purchasing cost. Constraint (5.1b) assigns a set of feasible arcs to each inline tank. The assignments of an inline tank accordingly determine its route. Constraint (5.1c) ensures that fuel is delivered only to the arcs that the inline tank passes. Constraint (5.1d) guarantees that the fuel demand of all arcs are satisfied. Fuel can be supplied by two sources: the assigned inline tanks ($\sum u_a^w$) and direct purchase from the origin station (r_a). Constraint (5.1e) states that total fuel delivery by an inline tank is equal to its initial refueling amount. Decision variables are defined in (5.1f)–(5.1i).

5.5 Dealing with Large Instances Using Aggregations

IFD-MCF becomes intractable for large instances of IFD, and as we will show, commercial solvers fail to solve or even find good-quality solutions for IFC-MCF over large instances. Aggregation methods may be useful to deal with such large instances. As discussed in Chapter 4, the common approach for commodity aggregation is to aggregate commodities, inline tanks in this case, with the same origin or with the same destination. Here, we prefer the destination-based aggregation as it results in one commodity/dispersion in total. This is because inline tanks do not have a predetermined destination, and hence, we can consider a dummy destination node as the shared destination for all inline tanks.

To obtain the aggregated formulation of IFD-MCF based on the destination-based approach, *AMCF*, we can modify IFD-MCF by:

1. Summing (5.1b) over $w \in W$ for all $v \in V$, decreasing the number of flow conservation constraints by the factor of $|W|$.
2. Summing (5.1c) over $w \in W$ for all $a \in \tilde{A}$, decreasing the number of such linking constraints by the factor of $|W|$.
3. Summing the constraint set (5.1e) into one constraint.

We then replace the common expressions $\sum_{w \in W} x_a^w$ and $\sum_{w \in u_a^w} u_a^w$ with n_a and v_a, respectively, which represent the aggregate variables. The variable n_a shows the number of inline

tanks assigned to the arc a, and the variable v_a determines the total fuel delivery by inline tanks to the arc a. The resulting AMCF model is provided below.

$$\text{(AMCF)} \quad Min \sum_{w \in W} C_w r_w + \sum_{a \in \tilde{A}} C_a r_a \tag{5.2a}$$

$$s.t: \quad \sum_{a \in O_v} n_a - \sum_{a \in I_v} n_a = \sum_{w \in W} a_v^w \qquad \forall v \in V \tag{5.2b}$$

$$v_a \leq D_a n_a \qquad \forall a \in \tilde{A} \tag{5.2c}$$

$$v_a + r_a = D_a \qquad \forall a \in \tilde{A} \tag{5.2d}$$

$$\sum_{a \in \tilde{A}} v_a = \sum_{w \in W} r_w \tag{5.2e}$$

$$0 \leq r_w \leq P \qquad \forall w \in W \tag{5.2f}$$

$$0 \leq r_a \leq D_a \qquad \forall a \in \tilde{A} \tag{5.2g}$$

$$0 \leq v_a \leq D_a \qquad \forall a \in \tilde{A} \tag{5.2h}$$

$$n_a \in \{0, 1, \ldots, |W|\} \qquad \forall w \in W, a \in A \tag{5.2i}$$

The AMCF model reduces the number of variables and the number of constraints of the IFD-MCF model approximately by the factor of $|W|$. However, the AMCF model is weak and fails to provide tight lower bounds for IFD-MCF. A main reason of the weakness of AMCF is that it allows that the fuel of inline tanks be transported to some arcs of the network without transporting the inline tanks, which carry that fuel, to such arcs. We show an example of such "illegal" *fuel teleportations* on a small instance. Consider an instance of IFD, which its corresponding time-space network is shown in Figure 5.2. This example includes 4 trips that correspond to arcs $a_1, \ldots, a_4$. The fuel demand of direct fuel purchase price for corresponding trips of arcs $a_1, \ldots, a_4$ are respectively as: $D_1 = 10$ & $C_1 = 0$, $D_2 = 10$ & $C_2 = 0$, $D_3 = 10$ & $C_3 = 1$, and $D_4 = 10$ & $C_4 = 1$. The fuel demand and direct fuel purchase price for each arc are shown on their labels in Figure 5.2. There are two inline tanks, w_1 and w_2, initially located at the stations s_1 and s_5 with the capacity of $P = 10$. Both inline tanks are initially located at stations with the fuel price of 0. Therefore, they can be initially refueled with the price of 0.

The optimal solution for AMCF over the network of Figure 5.2 is as: $n_{a_1} = n_{a_2} = n_{a_3} = n_{a_4} = 1, r_{w_1} = 10$, $r_{w_2} = 10$, $v_{a_3} = 10$, $v_{a_4} = 10$, $r_{a_1} = 10$, $r_{a_2} = 10$ and other variables equal to zero with the total cost of \$0. However, this solution is infeasible for the IFD problem. It implies that the fuel in inline tank w_1 is delivered to the arc a_4 through a fuel teleportation to this arc while it is impossible for this inline tank to reach the arc a_4

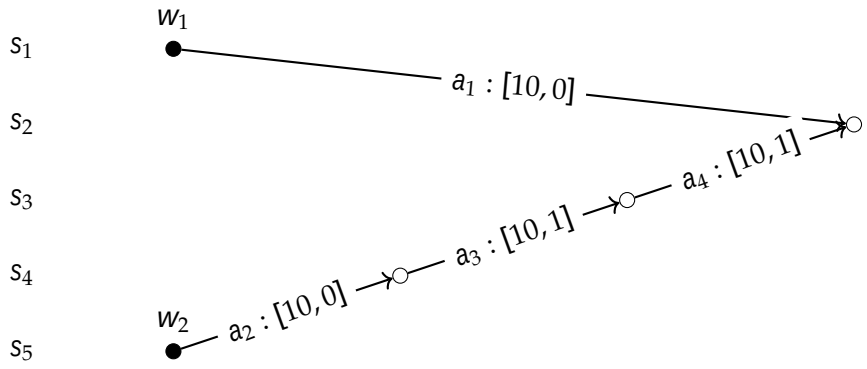


Figure 5.2: An example time-space network for IFD, in which the AMCF model gives solutions with "illegal" fuel teleportation. In this example, fuel in the inline tank w_1 can be transported to the inline tank w_2 on the arc a_4 based on the optimal solution of AMCF. The label on each arc shows their index and a tuple that its first entry is the demand and the second entry is the direct fuel purchase price for that arc.

based on its initial location and the time-space network.

We can avoid such illegal fuel teleportations in AMCF by considering an origin-based aggregation approach. However, this increases the size of model linearly in the number of stations from which inline tanks originate. Here, we propose an alternative approach to avoid illegal fuel teleportations. We add new variables and constraints to trace the total fuel flow in the inline tanks throughout the network. This is in addition to tracing of the number of inline tanks that the AMCF model does by the flow conservation constraints (5.2b). By this approach, the size of model does not increase significantly, yet avoids illegal fuel teleportations. Let f_a be the decision variable that determines the total fuel in the inline tanks on the arc a. The aggregated model that incorporates the inline tanks fuel tracing, called *ANF* in the rest of thesis, is provided below.

$$\text{(ANF)} \quad Min \sum_{w \in W} C_w r_w + \sum_{a \in \tilde{A}} C_a r_a \tag{5.3a}$$

$$s.t: \quad \sum_{a \in O_v} n_a - \sum_{a \in I_v} n_a = \sum_{w \in W} a_v^w \qquad \forall v \in V \tag{5.3b}$$

$$f_a \leq P n_a \qquad \forall a \in A \tag{5.3c}$$

$$\sum_{a \in O_v} f_a - \sum_{a \in I_v} f_a + \sum_{a \in \tilde{I}_v} v_a = \sum_{w \in W} a_v^w r_w \qquad \forall v \in V \tag{5.3d}$$

$$v_a \leq f_a \qquad \forall a \in \tilde{A} \tag{5.3e}$$

$$v_a + r_a = D_a \qquad \forall a \in \tilde{A} \tag{5.3f}$$

$$n_a \in \{0, 1, \ldots, |W|\} \qquad \forall a \in A \tag{5.3g}$$

$$0 \quad f_a \quad jWjP \qquad 8a\ 2\ A \qquad (5.3h)$$

$$0 \quad v_a, r_a \quad D_a \qquad 8a\ 2\ \tilde{A} \qquad (5.3i)$$

$$0 \quad r_w \quad P \qquad 8w\ 2\ W \qquad (5.3j)$$

Objective function (5.3a) minimizes the total fuel purchasing costs same as the objective function (5.1a). Flow conservation constraint (5.3b) determines the total number of inline tanks assigned to each arc of the network. Constraint (5.3c) limits the total fuel in the inline tanks on an arc to the total capacity of the available inline tanks on that arc. Constraint (5.3h) conserves the fuel flow in the inline tanks. This constraint also incorporates the loss of fuel flow that is delivered to locomotives. Constraint (5.3e) states that the total amount of fuel delivery to an arc is limited to the available fuel in the inline tanks on that arc. Constraint (5.3f) ensures the demand satisfaction, either by delivery by inline tanks (v_a) or by direct fuel purchase from the origin station (r_a). Decision variables are defined and bounded in constraints (5.3g)-(5.3j).

The only dimension of ANF that grows with an increase in the number of available inline tanks, is the number of r_w variables, as the constraints and other variables are stated per each node or each arc of the time-space network once. Therefore, the ANF model is stronger than the AMCF model with a little computational expense, and we employ the ANF model as the aggregated model for IFD in the next sections.

5.6 Infeasibilities Arise by (Full) Aggregation

Although aggregations significantly reduce the model size for the IFD problem, unlike the MCND problem, the solution of the aggregated model is not always feasible for the original problem. The infeasibilities arise because of the possibility of *fuel transfer* between inline tanks in the AFN solution. Here, we distinguish between the fuel transfer and the fuel teleportation that is discussed in Section 5.5. Fuel transfer occurs when two inline tanks on the same arc of the network transfer fuel between each other, while, fuel teleportation is the transportation of the fuel of an inline tank to an arc of the network that is not visited by that inline tank. In this section, we discuss the cases in which fuel transfer may occur.

IFD-MCF considers each inline tank as a distinct commodity. Therefore, commodity aggregation in this case translates to aggregating inline tanks. Aggregating a group of inline tanks on an arc of the time-space network implies that: (a) the capacity of aggregated inline tanks is accumulated as one inline tank with a larger capacity, and (b) the fuel in the aggregated inline tanks are combined into the larger inline tank. These implications allow the fuel transfer between aggregated inline tanks, which is prohibited in the original IFD

problem.

As an example, consider the time-space network shown in Figure 5.3. In this example, the fuel price at the station s_1 is 0, while it is equal to 1 at other stations. There are 6 trips in total that correspond to arcs $a_1, \ldots, a_6$, and the arc a_7 is an idle arc. The fuel demand and direct fuel purchase price for the corresponding trips of arcs $a_1, \ldots, a_6$ are respectively as: $D_1 = 10$ & $C_1 = 0$, $D_2 = 10$ & $C_2 = 0$, $D_3 = 10$ & $C_3 = 1$, $D_4 = 10$ & $C_4 = 1$, $D_5 = 15$ & $C_5 = 1$, and $D_6 = 5$ & $C_6 = 1$. The fuel demand and direct fuel purchase price for each arc are shown on their labels in Figure 5.3. There are two inline tanks, w_1 and w_2, both initially located at the station s_1 with the capacity of $P = 20$.

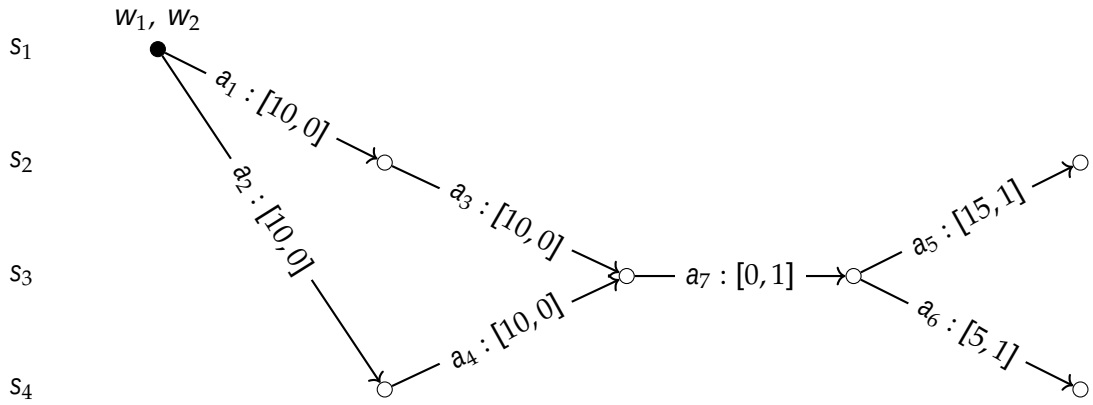


Figure 5.3: An example time-space network for IFD, in which aggregating inline tanks allows fuel transfer between inline tanks. In this example, if we aggregate inline tanks w_1 and w_2 on arc a_7, they can transfer fuel on this arc to reduce the total fuel purchasing costs further. The label on each arc shows their index and a tuple that its first entry is the demand and the second entry is the direct fuel purchase price for that arc.

An optimal solution for IFD-MCF over the network of Figure 5.3 is as: $r_{w_1} = 20$, $r_{w_2} = 20$, $u_{a_3}^{w_1} = 10$, $u_{a_5}^{w_1} = 10$, $u_{a_4}^{w_2} = 10$, $u_{a_6}^{w_2} = 5$, $r_{a_1} = 10$, $r_{a_2} = 10$, $r_{a_5} = 5$, and other variables equal to zero with the total cost of \$5. This means that when inline tanks w_1 and w_2 arrive at the origin node of arc a_7, the fuel level of each of them is equal to 10. If we aggregate these two inline tanks on arc a_7, they can transfer fuel on this arc to have a fuel level of 15 for the inline tank w_1 and a fuel level of 5 for the inline tanks w_2. Therefore, the optimal solution in the aggregated case changes to $u_{a_5}^{w_1} = 15$ and $r_{a_5} = 0$ with the total cost of \$0.

5.7 Partial Aggregations Shine

The possibility of fuel transfer between the aggregated inline tanks may result in infeasible solutions for the IFD problem. Therefore, aggregated formulations cannot be directly used to obtain upper bounds for the original problem. Moreover, the disaggregated formulation,

i.e. the IFD-MCF model, is intractable for large IFD instances. These indicate that the partial aggregation may be useful for IFD as formulations based on partial aggregations usually offer a trade-off for the advantages of the disaggregated and the fully-aggregated models. In this section, we propose partial aggregation schemes for IFD and utilize them to develop a solution algorithm. Section 5.7.1 proposes tailored partial aggregations for IFD based on the characteristics of the underlying time-space network. Section 5.7.2 modifies the ANF formulation to incorporate such partial aggregations. Section 5.7.3 develops a solution algorithm using the proposed partial aggregation scheme and formulation.

5.7.1 Constructing Partial Aggregations

Recall from Chapter 4 that a partial aggregation of commodities consists of a set of dispersions, where for every commodity there is a dispersion that includes that commodity. Here, we adopt the destination-based aggregation and consider the inline tanks as commodities. Therefore, the partial aggregation B for IFD in fact includes only one dispersion b. As per Definition 1 in Section 4.2, we define the dispersion b for an instance of IFD as:

1. Set of commodities $K_b = W$ that share a common destination.
2. The set of destination nodes N_b includes only a dummy node as the destination for all inline tanks.
3. On each arc $a \in A$, the set of W is partitioned into a subset K_b^a of inline tanks aggregated on that arc and its complement D_b^a which are disaggregated on that arc.

We now discuss the method to construct the dispersion b, which is in fact determining the subsets K_b^a and D_b^a for each arc $a \in A$. To construct the partial aggregations, instead of the shortest path approach for the MCND problem, we propose a Time-Based Aggregation approach (TBA) here as the time plays an important role in the underlying network of IFD. Intuitively, TBA disaggregates the inline tanks on incident arcs of their reachable nodes within a *disaggregation time period* $[0, t]$ and keeps them aggregated on others arcs to construct a partial aggregation. Let n_w be the node of the time-space network that represents the initial location of the inline tank w, i.e. $n_w = (s_w, 0)$. Formally, TBA disaggregates an inline tank w on all incoming and outgoing arcs of the node $i = (s_i, t_i)$ where $0 \quad t_i \quad t$ and there is a path from the node n_w to the node i.

Suppose that inline tank w is disaggregated on a set of arcs A^w based on TBA over the disaggregation time period $[0, t]$. The trip arcs in the set A^w are shown by the set $\tilde{A}^w$. Since TBA disaggregates the inline tanks based on the corresponding times of the nodes, it partitions the nodes of the network into three subsets for each inline tank:

1. Set of *disaggregated nodes* V^w, where the inline tank w is disaggregated on their incom-

ing and also their outgoing arcs, i.e. $V^w = \{i = (s_i, t_i) \in V | t_i \geq t$ & there is a path from n_w to $i\}$.

2. Set of *semi-disaggregated nodes* $\bar{V}^w$, where the inline tank w is disaggregated only on their incoming arcs, i.e. $\bar{V}^w = \{i = (s_i, t_i) \in V | t_i > t$ & $\exists a \in I_i$ where origin of $a \in V^w\}$.
3. Set of *aggregated nodes* which includes other nodes than the disaggregated and the semi-disaggregated nodes, i.e. $V \setminus \{V^w \cup \bar{V}^w\}$.

For any partial aggregation by TBA, $n_w \in V^w$ for all inline tanks $w \in W$. Therefore, each inline tank first travels between its disaggregated nodes and then exits its set of disaggregated nodes by traversing to one of its semi-disaggregated or aggregated nodes. Since the time-space network of IFD is an acyclic directed network, an inline tank cannot return to set of its disaggregated nodes once it exits them. Hence, to incorporate the TBA partial aggregations and develop a partially-aggregated formulations, we are not required to add the tightening constraints introduced in Chapter 4.

5.7.2 Partially-Aggregated Formulations

To incorporate a TBA partial aggregation, it is enough to create new variables that correspond to the disaggregated arcs A^w of each inline tank $w \in W$, and add and modify the relevant constraints. We define following parameters to modify the AFN model.

b_v^w A binary parameter; 1 if $v \in V^w$, 0 otherwise,
g_a^w A binary parameter; 1 if $a \in A^w$, 0 otherwise.

The steps to modify the AFN model to incorporate a partial aggregation obtained by TBA for the disaggregation time period $[0, t]$ includes:

1. Create new decision variables defined as the constraint set 5.4 for each $w \in W$. The purpose of these variables is the same as the purpose of n_a, f_a, and v_a variables, but for individual inline tanks.

$$x_a^w \in \{0, 1\} \qquad \forall a \in A^w \tag{5.4a}$$
$$0 \leq f_a^w \leq P \qquad \forall a \in A^w \tag{5.4b}$$
$$0 \leq v_a^w \leq D_a \qquad \forall a \in \tilde{A}^w \tag{5.4c}$$

2. Modify the flow conservation constraint (5.3b) to the equation (5.5). This ensures that inline tanks visit the aggregated or semi-aggregated nodes after they exit their

disaggregated nodes.

$$\sum_{a2O_v} n_a \quad \sum_{a2I_v} n_a \quad \sum_{w2W:v2\bar{V}^w} \sum_{a2I_v} x_a^w = \sum_{w2W} a_v^w(1 \quad b_v^w) \qquad 8v\ 2\ V \tag{5.5}$$

3. Modify the flow conservation constraint (5.3d) to the equation (5.6). This ensures that the fuel of inline tanks flow to the aggregated or semi-aggregated nodes after they exit their disaggregated nodes.

$$\sum_{a2O_v} f_a \quad \sum_{a2I_v} f_a + \sum_{a2\tilde{I}_v} v_a \quad \sum_{w2W:v2\bar{V}^w} \sum_{a2I_v} f_a^w + \sum_{w2W:v2\bar{V}^w} \sum_{a2\tilde{I}_v} v_a^w = \sum_{w2W} a_v^w(1 \quad b_v^w) r_w \qquad 8v\ 2\ V \tag{5.6}$$

4. Modify the demand satisfaction constraint (5.3f) to the equation (5.7). This incorporates the fuel delivery of individual inline tanks to their disaggregated arcs.

$$\left(\sum_{w2W:a2\tilde{A}^w} v_a^w \right) + v_a + r_a = D_a \qquad 8a\ 2\ \tilde{A} \tag{5.7}$$

5. Add the required flow conversation and linking constraints for the new variables as the constraint set (5.8) to ensure feasible (partial) paths and flows for inline tanks within their disaggregated nodes.

$$\sum_{a2O_v} x_a^w \quad \sum_{a2I_v} x_a^w = a_v^w b_v^w \qquad 8w\ 2\ W, v\ 2\ V^w \tag{5.8a}$$

$$f_a^w \quad Pn_a^w \qquad 8w\ 2\ W, a\ 2\ A^w \tag{5.8b}$$

$$\sum_{a2O_v} f_a^w \quad \sum_{a2I_v} f_a^w + \sum_{a2\tilde{I}_v} v_a^w = a_v^w b_v^w r_w \qquad 8w\ 2\ W, v\ 2\ V^w \tag{5.8c}$$

$$v_a^w \quad f_a^w \qquad 8w\ 2\ W, a\ 2\ \tilde{A}^w \tag{5.8d}$$

The fully-aggregated model (5.3) does not provide information for the path of individual inline tanks and only gives the total number of inline tanks on each arc through n_a variables. However, as constraints (5.5) and (5.8a) indicate, the partially-aggregated formulation provides information for the path of individual arcs on their disaggregated arcs by x_a^w variables. These information gives the path of inline tanks over a subset of network, and we call them *partial paths*. Such partial paths provides feasible solutions for the original problem over a subset of the network.

5.7.3 The Partial Aggregation-Based Algorithm (PAA)

This section develops a partial aggregation-based algorithm for the IFD problem. The PAA algorithm consists of four successive steps that are iterated until the termination condition is met.

In the first step, the PAA algorithm solves an aggregated model (ANF) which gives a lower bound on IFD and also a guide for finding upper bounds. The ANF model that PAA solves is the fully aggregated model (5.3) at the first iteration. However, as it progresses it modifies the ANF model to incorporate the partial aggregations that it constructs in the fourth step. The *guide* that this step provides is the total number of inline tanks assigned to each arc and possibly partial paths provided by the partially-aggregated models.

In the second step, inline tanks are greedily routed, one by one, based on the guide provided in the first step. Here, a fast algorithm to route the inline tanks is useful as we have a guide to route them. Our initial experiments show that the computing time of IFD-MCF instances with one inline tank available is fast and usually less than 0.5 seconds. Therefore, we employ the GA algorithm introduced in Section 3.7.1 but with using the provided guide by the first step. Recall that at every iteration, GA myopically optimizes the route and fuel plan of one inline tank by solving a IFD-MCF model with one inline tank available. An inline tank is routed based on the available partial path for it and the number of inline tanks on the arcs provided by the guide. The route and fuel plan of that inline tank are fixed in subsequent iterations, and the demand of the locomotives that receive fuel from this inline tank as well as the number of inline tanks on the arcs in the guide are updated accordingly. GA terminates when all inline tanks are routed.

Once routes of inline tanks are determined, fuel deliveries of all inline tanks are simultaneously optimized by solving an easy LP problem in the third step, which provides a feasible solution to IFD.

The fourth step constructs a partial aggregation by TBA for the disaggregation time period $[0, t]$. The size of the disaggregation time period t is equal to t percent of the time horizon at the first iteration, and increases by t percent of the time horizon at each subsequent iteration, where t is a user-defined parameter. Then, the ANF model is modified based on such a partial aggregation. An overview of PAA is given in Algorithm 2. The time horizon of an IFD instance is shown by T and is equal to $\max_{l \in L} w_l$.

As the PAA algorithm progresses, the disaggregation time period size t for constructing partial aggregations increases, which expands the span of the disaggregated arcs and nodes. This implies that the ANF model becomes closer to the original disaggregated model as t increases at each iteration. Therefore, unless the time limit is reached, the PAA algorithm is an exact algorithm that is able to prove the optimality of the original IFD problem.

Algorithm 2: The PAA algorithm

Input: $W, G = (A, V)$, the optimality gap tolerance e, *Timelimit*, t, and T
Output: Routes and fuel plans for all inline tanks and locomotives
Initialization: Develop the fully-aggreagated ANF model as (5.3), and $t = t \quad T$
while *Timelimit is not reached and* $\frac{UB \quad LB}{UB} > e$ **do**
  Step 1: Solve ANF to obtain a lower bound, LB, and the guide for routing inline tanks
  Step 2: Route inline tanks greedily based on the guide obtained in Step 1
  Step 3: Optimize the fuel delivery of inline tanks simultaneously to obtain an upper bound, UB
  Step 4: Construct a partial aggregation by TBA for the disaggregation time period $[0, t]$ and modify the ANF model accordingly, and update $t := t + t \quad T$
end

However, we demonstrate in the next section that, in practice, only a few iterations are sufficient to achieve a lower bound that is equal to the optimal value of the LP relaxation of IFD-MCF. As ANF gets modified by the progress of PAA, its lower bound may increase and also the obtained partial paths becomes longer, which means the guide provides more accurate information for the greedy algorithm to route inline tanks.

5.8 Computational Results

In this section, we perform a set of computational experiments to evaluate the efficiency of the PAA algorithm in providing good-quality lower and upper bounds. We use the dataset from the INFORMS Problem Solving Competition (2010), which is described in Section 3.7. Recall that this dataset is a large instance of the refueling problem that includes 2996 train schedules with 5264 trip legs. As in the IFD problem, it is assumed that each locomotive makes a single trip, we consider that each trip leg is made by a distinct locomotive, resulting in a total of 5264 locomotives with a single trip. Fuel prices range from \$2.90 to \$3.56 per gallon, and the total consumption of all locomotives is 3,595,522 gallons. We adjust the fuel prices so they range from \$0.00 to \$0.66 so that the objective function does not include the constant part and hence, the results and optimality gaps distinguish the performance of different solution approaches better. We consider 8 different sizes of the problem where the number of available inline tanks vary from 25 to 200.

We consider two solution approaches for the IFD problem in our experiments, solving the IFD-MCF model by Gurobi and using the PAA algorithm. The PAA algorithm requires two user-specified parameters, the optimality gap tolerance (e) and the percentage of in-

crease in the disaggregation time period in each step (t). We consider a same optimality gap tolerance as the default Gurobi setting, i.e. $e = 0.01\%$, and based on our initial experiments, we set the parameter t to 4%. We choose a small tolerance gap because of two reasons: first, our goal is to propose an exact method. Second, the objective function reports the weekly costs, and hence, even a slight reduction in the objective function translates to annual savings worth thousands of dollars. All computational experiments are run on a cluster with 4 Xeon-Gold-6150 cores and 16 GB RAM using Gurobi 9.1 via a Python API as the LP and the MIP solver. Default Gurobi settings are always selected. The PAA algorithm is also implemented in Python 3.6. A time limit of 12 hours (43,200 seconds) is considered for both approaches.

Table 5.1 gives a summary of the performance of these two approaches over 8 instances. For each approach, the best upper bound (UB), the best lower bound (LB), the optimality gap (Gap=$\frac{\text{UB} \quad \text{LB}}{\text{UB}} \quad 100$), and the total computing time in seconds (Time) are reported. For all instances, the lower bound obtained by the PAA algorithm is the same as the lower bound by the IFD-MCF model. Note that these lower bounds are the same as the optimal value of the LP relaxation of the IFD-MCF model for all instances. For instances with 25, 50, and 75 inline tanks available, the PAA algorithm proves the optimality significantly faster. Although for instances with 100 and 125 inline tanks available, the solution by the PAA algorithm has a slightly larger optimality gap, the solutions are close to optimality. Moreover, the PAA algorithm finds such solutions in significantly shorter times as we will show later in this section. The performance of Gurobi is particularly weak over instances with 150 and more inline tanks available, where it fails to provide any feasible solution in 12 hours. However, the PAA algorithm provides solutions with almost 1% gap within the time limit. In summary, these results demonstrate the efficiency of the PAA algorithm, which is mainly because it solves the fully- and partially-aggregated formulations which require shorter computing times. As the results suggest, solving the aggregated formulations gives tight lower bounds and also reliable guides to find good-quality upper bounds.

The PAA algorithm is not only efficient in proving optimality, but also it provides good-quality lower and upper bounds in short computing times as shown in Table 5.2. This table reports the upper and lower bounds and the gap between those, obtained in the first iteration of the PAA algorithm. The time that PAA takes to find these bounds is reported in the column *Time*. For comparison, the computing time of the IFD-MCF root relaxation for the same instance is shown in the last column. As Table 5.2 shows, the PAA algorithm generates bounds with a small gap in short times, much faster that just solving the root relaxation of IFD-MCF. Note that the gap shown here is between the bounds reported in the table, and the actual optimality gap of the upper bound is even smaller. Moreover, the PAA algorithm improves the first obtained upper and lower bounds for all instances (except

Table 5.1: Comparing the performance of solving the IFD-MCF by the MIP solver and using the PAA algorithm over different instance sizes of the IFD problem.

Size	IFD-MCF				PAA			
	UB	LB	Gap	Time	UB	LB	Gap	Time
25	830,224.9	830,224.9	0.00%	437.1	830,224.9	830,224.9	0.00%	18.8
50	767,318.7	767,284.0	0.00%	2129.7	767,320.8	767,284.0	0.00%	173.5
75	720,559.7	720,496.8	0.01%	25306.2	720,559.7	720,496.8	0.01%	692.1
100	692,179.7	692,179.7	0.00%	28409.5	693,453.4	692,179.7	0.18%	43200
125	665,624.2	665,273.0	0.05%	43200	670,367.2	665,273.0	0.76%	43200
150	–	6461,56.3	–	43200	651,042.5	646,156.3	0.75%	43200
175	–	626,997.6	–	43200	634,009.2	626,997.6	1.11%	43200
200	–	603,428.5	–	43200	610,239.4	603,428.5	1.12%	43200

with 25 inline tanks which its optimality is proved in the first iteration) as the comparison between the corresponding figures in Tables 5.1 and 5.2 indicate. This demonstrates that constructing partial aggregations in the next iterations of the PAA algorithm is effective in both improving the lower bound and providing better guides to generate an upper bound.

Table 5.2: The quality of obtained bounds by the PAA algorithm in the first iteration, and comparing the required time to find those with the computing time of the IFD-MCF root relaxation.

Size	UB	LB	GAP	Time	IFD-MCF root relaxation time
25	830,224.9	830,224.9	0.00%	18.8	416.4
50	768,882.9	767,284.0	0.21%	49.3	1683.2
75	723,852.7	719,561.5	0.59%	61.3	12,777.4
100	694,832.8	690,915.4	0.56%	93.2	12,888.0
125	672,017.1	663,896.8	1.21%	132.9	1588.7
150	654,349.5	644,594.9	1.49%	95.5	2296.3
175	634,057.5	624,839.2	1.45%	104.5	2776.3
200	613,009.7	601,559.2	1.87%	131.2	3216.3

We now demonstrate that progress of the PAA algorithm by increasing the disaggregation time period size improves both lower and upper bounds by showing its performance over an instance. Table 5.3 reports the progress of the PAA algorithm over an instance of IFD with 75 inline tanks. Here, column *Total time* shows the total elapsed time by PAA up to the corresponding iteration. The time taken by each step of PAA at the corresponding iteration of each row is shown in the last four columns of the table. The column $D\%$ reports the percentage of the time-space network on which inline tanks are disaggregated. As the table shows, the algorithm converges when inline tanks are disaggregated on 19.1% of the

network at the iteration 6. Based on the times reported in the last four columns, the time required by all steps remain almost the same except the first step. This is expected because as the algorithm progresses, the number of variables and constraints of the AFN model increases. The convergence of PAA is due to improvement in both the upper bound and the lower bound. The lower bound is improved because the disaggregated period increases as the algorithm progresses and hence, AFN becomes a more accurate estimate of the original problem. The increase in the disaggregated period results in creating more x_a^w variables in AFN that describes a partial path for individual inline tanks. Subsequently, longer partial paths by added x_a^w variables in the AFN model benefits the greedy algorithm in the second step for generating better paths.

Table 5.3: The convergence of the PAA algorithm over an instance of the IFD problem with 75 inline tanks available.

Iter.	D%	Total time	UB	LB	Gap	Time of each step			
						Step 1	Step 2	Step 3	Step 4
1	0.0%	61.3	723,852.7	719,561.5	0.59%	22.0	38.7	0.7	0.0
2	3.8%	129.3	723,707.8	719,561.5	0.57%	25.8	41.7	0.4	0.0
3	7.6%	188.5	723,209.6	719,950.9	0.45%	16.4	42.2	0.6	0.0
4	11.4%	283.6	723,209.6	720,496.8	0.38%	50.0	44.5	0.6	0.0
5	15.3%	481.1	722,064.4	720,496.8	0.22%	152.1	44.9	0.4	0.1
6	19.1%	692.1	720,559.7	720,496.8	0.01%	169.5	41.0	0.4	0.2

To show the effectiveness of the PAA algorithm in providing lower bounds as tight as the IFD-MCF formulation during its progress, we compare the initial lower bound the best lower bound obtained by the PAA algorithm in Table 5.4. The first lower bound and the best lower bound obtained by PAA are shown in the second and third columns, respectively. The improvement from the first lower bound to the best lower bound is shown in the column *Improvement* in percentage. The iteration at which the best lower bound is shown in the column *Iteration*. The column D% expresses the percentage of the network on which inline tanks are disaggregated at such iteration, and the last column reports the computing time of the corresponding AFN model. The best lower bound is achieved at the first iteration with the fully-aggregated formulation for instances with 25 and 50 inline tanks . However, the best lower bound is achieved at the subsequent iterations with the partially-aggregated formulation for larger instances. The best lower bound obtained by PAA for all instances is the same as the optimal value of the LP relaxation of the corresponding IFD-MCF formulation. Note that the optimal value of the LP relaxation of IFD-MCF is usually same as its optimal integer solution as shown for instances in Table 5.1.

Table 5.4: Comparing the first and the best lower bounds obtained by the PAA algorithm.

Size	Initial LB	Best LB	Improvement	Iteration	D%	ANF time
25	830,224.9	830,224.9	0.00%	1	0.0%	8.0
50	767,284.0	767,284.0	0.00%	1	0.0%	23.8
75	719,561.5	720,496.8	0.13%	4	11.4%	50.0
100	690,915.4	692,179.7	0.18%	9	30.5%	4319.7
125	663,896.8	665,273.0	0.21%	6	19.1%	1063.8
150	644,594.9	646,156.3	0.24%	4	11.4%	74.2
175	624,839.2	626,997.6	0.35%	7	22.9%	14,187.4
200	601,559.2	610,239.4	1.44%	5	15.3%	1044.5

5.9 Conclusions and Future Research

The IFD problem includes a multicommodity network flow substructure, and its large instances become intractable for current technology. Therefore, this chapter applies the concept of partial aggregations introduced in Chapter 4 to tackle large instances of IFD. We first define the IFD problem as an abstracted version of the LFMIR problem, which retains the main complexity of incorporating the inline tanks planning in the fuel management of locomotives, i.e. the combinatorial number of switch opportunities of inline tanks between locomotives. We prove that IFD is also NP-hard. We discuss the infeasibilities arise by aggregating inline tanks in this problem. Therefore, the partial aggregations can be used to obtain lower bounds in this case. On the other hand, the solution of the aggregated model provides a good guide to obtain upper bounds for the original problem. Based on these insights, we develop a solution algorithm that utilizes the partial aggregations for this problem. Our computational experiments on a set of standard instances demonstrate the efficiency of the proposed algorithm in providing tight lower bounds and good-quality upper bounds in much shorter times in comparison with just solving the disaggregated MIP model with a MIP solver.

The approach taken in this chapter gives insights on: (a) adopting appropriate methods to construct partial aggregations based on the problem structure. Since the underlying time-space network of the IFD problem is an acyclic graph, we proposed a time-based approach to construct partial aggregation which their corresponding formulations do not require the tightening constraints introduced in Chapter 4. (b) Utilizing the solutions by the partially-aggregated formulations to generate feasible upper bounds for the disaggregated problem, when the aggregated formulation may give infeasible solutions for the original problem. (c) Developing solution algorithms that utilize partial aggregations. As we demonstrated here, such algorithms usually converge with partially-aggregated formulations without the

need for returning to the original disaggregated model. One potential direction to extend this work is to apply this approach to problems with a similar structure, such as the vehicle scheduling problems that are usually modelled on a time-space network. Moreover, the PAA algorithm can be further improved by proposing smarter aggregation schemes that consider other aspects, such as the obtained solutions in the previous iterations, in addition to time considerations. Developing better heuristics to transform the guide provided by the aggregated models to a feasible solution for the original problem may also benefit the PAA algorithm to converge faster. The IFD problem includes the main complexities of the LFMIR problem. Therefore, the PAA algorithm could be extended to the LFMIR problem as another future research direction. As we show, the PAA algorithm is able to tackle the main complexity factor of IFD and subsequently, the LFMIR problem. However, the algorithm would require additional gadgets for operational constraints such as cyclic requirements.

CHAPTER 6

Concluding Remarks

This thesis is motivated by refueling challenges of railroad companies and introduces a new class of fuel management problems to tackle such challenges. The problem is to optimally plan a fleet of inline refueling wagons for a fleet of locomotives and trains during the time horizon. Inline refueling facilitates the fuel tankering for long-haul trips, which in turn reduces direct and indirect fueling costs. As such problems lie in the scope of network optimization, this thesis proposes general methods to handle large instances of multicommodity network flow problems. Such problems are important in the field of network optimization as they apply to a wide range of problems in various applications, including the fuel management problems that are subject of this thesis. Fuel management problems that this thesis studies incorporate the inline refueling planning into the conventional locomotive fuel management problems, which brings opportunities for cost reduction and also further complexity. We define and study the Locomotive Fuel Management with Inline Refueling (LFMIR) in Chapter 3 to address the inline refueling planning. Chapter 4 introduces the concept of partial aggregation for a broad range of network optimization problems that contain a subset of variables and constraints which make up a multicommodity flow component. Chapter 5 applies this concept to a variant of the fuel management problem. The thesis contributes to both theory and practice of network optimization, with results on the business implications of adopting inline refueling as well as new methods to deal with large multicommodity network flow problems.

To study the logistical implications of inline refueling, we conceptualize the LFMIR

problem on a time-space network based on which we develop a MIP model. The proposed MIP model provides fuel plans for locomotives and inline tanks operations and can be also utilized for strategic decisions such as inline tank fleet sizing and locating fuel stations. We apply this model to two case studies from Australia and the USA and show potential cost savings. Furthermore, we investigate the impacts of inline refueling on the ongoing operations of the railroad companies and show that employing inline refueling: (a) reduces the cost of imposing the safety fuel inventory as a robustness constraint, and (b) reduces the marginal cost of fuel consumption increase on different paths. Based on the parameters of the given datasets, we identify the cases for which inline refueling leads to greater savings.

Chapter 3 shows both the theoretical and empirical complexity of the LFMIR problem. We prove that the LFMIR is strongly NP-hard even in its simplest form and with only one inline tank available. Moreover, computational experiments on large instances demonstrate the difficulty of solving the problem in practice. As the number of available inline tanks increases, CPLEX, surprisingly, produces worse solutions. Therefore, we develop the SGA algorithm that generates good-quality solutions for large instances in a reasonable time.

Computational experiments show that as the number of available inline tanks increases, even solving the LP relaxation of the LFMIR MIP model is significantly time-consuming. This is interesting as the same trend appears in solving multicommodity network flow problems, and here, we can consider inline tanks as commodities. Therefore, we introduce new commodity representations, *dispersions*, for network optimization problems with a multicommodity flow substructure. The introduction of dispersions allows us to propose novel aggregation schemes, *partial aggregations*, that aggregate the commodities over a subset of the network instead of conventional full aggregation or disaggregation of commodities over the entire network. The LP relaxation of disaggregated formulations usually provides tight bounds but with a long computing time. On the other hand, the LP relaxation of fully-aggregated formulations has a short computing time but gives loose bounds. Partial aggregations provide formulations with short LP relaxation computing time and yet tight bounds.

We apply the partial aggregations to Multicommodity fixed-charge Capacitated Network Design problem (MCND), which is an archetypal network optimization problem with a multicommodity flow substructure. We develop a heuristic to construct partial aggregations and propose two MIP models for MCND. We compare these new partially-aggregated formulations with the existing formulations in the literature and prove that they are valid bounds for MCND. An extensive computational study on a set of benchmark instances demonstrates that the partially-aggregated formulations are on the Pareto frontier for the trade-off between LP relaxation computing time and bound tightness. This offers a high level of control over this trade-off and can be leveraged in MIP algorithms. We show that

the partially-aggregated formulations benefit the MIP algorithms, particularly B&B and Benders decomposition, over large instances of MCND with many commodities.

Chapter 5 applies the concept of partial aggregation to an abstracted version of the LFMIR problem. We define the Inline Fuel Delivery problem (IFD) which retains the main complexities arising from inline refueling considerations and prove that this problem is also strongly NP-hard. We discuss the infeasibilities that emerge by aggregating inline tanks in this problem. Therefore, partially aggregated formulations for this problem cannot directly generate feasible upper bounds, but still provide tight lower bounds. We employ the partially-aggregated formulations for this problem to develop the partial aggregation-based algorithm (PAA). This algorithm uses the solution of the partially-aggregated formulations as a lower bound and also as a guide to generate feasible upper bounds. Moreover, it iteratively expands the scope of the corresponding partial aggregation until it converges to the optimality. Computational experiments verify the efficiency of the PAA algorithm in providing tight lower bounds and good-quality upper bounds in comparison with directly solving the disaggregated formulation for this problem.

We outline two main directions for future research: extending the LFMIR problem to include additional considerations and further development on the concept of partial aggregation. The first direction expands the practical aspects of the research work of this thesis, and the latter develops on the theoretical works.

We propose the following potential future research avenues to extend the locomotive fuel management problem studied in this thesis:

- The LFMIR problem ignores the uncertainty in the parameters of the problem. However, some parameters of the problem, such as the fuel consumption and the fuel prices, involve randomness. Randomness of some parameters such as the fuel consumption can be handled with the proposed models, for instance, by introducing a fuel safety level for the locomotives to avoid infeasible solutions due to the fuel consumption increase. In the case of existence of more uncertain parameters, addressing such parameters in the LFMIR may result in more practical fuel plans.
- The refueling operations of rail companies are closely related to other business procedures, particularly train scheduling. The integration of locomotive fuel management and train scheduling problems benefits the rail companies to further reduce the operational costs.
- The LFMIR problem can be applied to other modes of transportation that involve similar refueling options as inline refueling. One example of such transportation modes are the hydrogen-powered locomotives on which hydrogen gas tanks can be switched. The operational restrictions of such transportation modes may require

modifications in the proposed models and methods proposed in this thesis.

Multicommodity network flow problem is a classical network optimization problem. Moreover, it arises in a wide variety of applications in transportation, logistics, telecommunication, and energy systems. Therefore, there are many promising research directions to utilize and apply the concept of partial aggregation, including:

- As our computational experiments showed, partially-aggregated formulations benefit the MIP algorithms, especially over large instances with many commodities. Therefore, such formulations can be used to develop specialized solution algorithms. Computational experiments showed the remarkable performance of the partially-aggregated formulations when the Benders decomposition algorithm is employed. Hence, development of a tailored Benders decomposition algorithm that unifies these formulations and their offered trade-off between the LP relaxation computational difficulty and bound tightness is a potential future research direction.
- The disaggregated and the fully-aggregated formulations for MCND have been extensively studied in the literature and several efficient cutting planes algorithms are developed for these formulations. Moreover, as we showed in Chapter 4, such algorithms significantly improve the performance of the B&B algorithm over those formulations. Although some of the proposed cutting plane algorithms are applicable to the partially-aggregated formulations, proposing new valid cuts for these formulations can further improve the performance of the B&B algorithm.
- In this thesis, we construct partial aggregations in advance of using the MIP algorithms, and they are fixed throughout the progress of the corresponding algorithm. Another approach is to adaptively construct partial aggregations based on the information obtained during the algorithm progress. One such example is to construct partial aggregations based on the fractional values of integer variables in the nodes of the B&B tree.
- Partial aggregation can be applied to other problems with a multicommodity network flow substructure. The application is particularly straightforward for the cases in which there is no commodity-specific attribute over the underlying network.

The PAA algorithm is a typical example of a solution algorithm that utilizes partial aggregations. We outline two future research directions for this algorithm:

- The PAA algorithm can be further improved in two ways: (a) using better heuristics to transform the guide provided by the aggregated formulation to a feasible solution, and (b) construing smarter partial aggregations that use the information obtained during the progress of PAA.
- The PAA algorithm can be applied to problems with a similar structure, which re-

quires a tailored constructive algorithm for partial aggregations and possibly a heuristic to generate or transform solutions.

While some of the future research suggested here might be interesting, this thesis has laid the groundwork for all of this by defining a new class of optimization problems in the context of railways' fuel management and developing a novel modeling approach based on the partial aggregation that opens up new algorithmic options for many other types of network problems.

Bibliography

[1] Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (2014). *Network Flows: theory, algorithms, and applications*. Pearson.

[2] Andrews, M., Antonakopoulos, S., and Zhang, L. (2010). Minimum-cost network design with (dis)economies of scale. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 585–592.

[3] Anjos, M. F., Gendron, B., and Joyce-Moniz, M. (2020). Increasing electric vehicle adoption through the optimal deployment of fast-charging stations for local and long-distance travel. *European Journal of Operational Research*, 285(1):263 – 278.

[4] Appelgren, L. H. (1969). A column generation algorithm for a ship scheduling problem. *Transportation Science*, 3(1):53–68.

[5] Asciano (2017). 2017 Annual Report. Technical report, Asacanio Limited, Melbourne, Victoria.

[6] Assad, A. A. (1978). Multicommodity network flows—a survey. *Networks*, 8(1):37–91.

[7] Avella, P., Mattia, S., and Sassano, A. (2007). Metric inequalities and the network loading problem. *Discrete Optimization*, 4(1):103 – 114. Mixed Integer Programming.

[8] Barnhart, C., Hane, C. A., Johnson, E. L., and Sigismondi, G. (1994). A column generation and partitioning approach for multi-commodity flow problems. *Telecommunication Systems*, 3(3):239–258.

[9] Barnhart, C., Hane, C. A., and Vance, P. H. (2000). Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Operations Research*, 48(2):318–326.

[10] Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., and Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329.

[11] Belieres, S., Hewitt, M., Jozefowiez, N., and Semet, F. (2021). Meta partial benders decomposition for the logistics service network design problem. *European Journal of Operational Research*.

[12] Belieres, S., Hewitt, M., Jozefowiez, N., Semet, F., and Van Woensel, T. (2020). A benders decomposition-based approach for logistics service network design. *European Journal of Operational Research*, 286(2):523–537.

[13] Belotti, P., Malucelli, F., and Brunetta, L. (2007). Multicommodity network design with discrete node costs. *Networks*, 49(1):90–99.

[14] Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252.

[15] Berman, O., Larson, R. C., and Fouska, N. (1992). Optimal Location of Discretionary Service Facilities. *Transportation Science*, 26(3):201–211.

[16] Berthold, T. (2006). *Primal Heuristics for Mixed Integer Programs*. PhD thesis, Technische Universitat Berlin.

[17] Bertsimas, D. and Tsitsiklis, J. N. (1997). *Introduction to linear optimization*. Athena Scientific.

[18] Besbes, O. and Savin, S. (2009). Going Bunkers: The Joint Route Selection and Refueling Problem. *Manufacturing & Service Operations Management*, 11(4):694–711.

[19] Bienstock, D., Chopra, S., Günlük, O., and Tsai, C.-Y. (1998). Minimum cost capacity installation for multicommodity network flows. *Mathematical Programming*, 81(2):177–199.

[20] Bienstock, D. and Günlük, O. (1995). Computational experience with a difficult mixedinteger multicommodity flow problem. *Mathematical Programming*, 68(1-3):213–237.

[21] Bienstock, D. and Günlük, O. (1996). Capacitated network design—polyhedral structure and computation. *INFORMS Journal on Computing*, 8(3):243–259.

[22] Bland, R. G. (1977). New finite pivoting rules for the simplex method. *Mathematics of Operations Research*, 2(2):103–107.

[23] BNSF Railway (2018). Annual Report 2017 on Form 10-K. Technical report, BNSF Railway Company, Fort Worth, Texas.

[24] Boland, N., Dumitrescu, I., Froyland, G., and Gleixner, A. M. (2009). Lp-based disaggregation approaches to solving the open pit mining production scheduling problem with block processing selectivity. *Computers & Operations Research*, 36(4):1064 – 1089.

[25] Borgwardt, K. H. (1982). The average number of pivot steps required by the simplex-method is polynomial. *Zeitschrift für Operations Research*, 26(1):157–177.

[26] Bush, B. A. (2006). *Analysis of Fuel Consumption for an Aircraft Deployment with Multiple Aerial Refuelings*. PhD thesis, North Carolina State University.

[27] Chiraphadhanakul, V. and Figueroa, C. (2010). 2010 RAS problem solving competition: A locomotive refueling problem. Technical report, Operations Research Center, Massachusetts Institute of Technology, Cambridge.

[28] Chouman, M., Crainic, T. G., and Gendron, B. (2017). Commodity representations and cut-set-based inequalities for multicommodity capacitated fixed-charge network design. *Transportation Science*, 51(2):650–667.

[29] Chung, S. H. and Kwon, C. (2015). Multi-period planning for electric car charging station locations: A case of Korean expressways. *European Journal of Operational Research*, 242(2):677–687.

[30] Conforti, M., Cornuejols, G., and Zambelli, G. (2014). *Integer programming*. Springer.

[31] Costa, A. M. (2005). A survey on benders decomposition applied to fixed-charge network design problems. *Computers & Operations Research*, 32(6):1429 – 1450.

[32] Costa, A. M., Cordeau, J.-F., and Gendron, B. (2007). Benders, metric and cutset inequalities for multi-

commodity capacitated network design. *Computational Optimization and Applications*, 42(3):371–392.

[33] Crainic, T. G. (2000). Service network design in freight transportation. *European Journal of Operational Research*, 122(2):272–288.

[34] Crainic, T. G., Frangioni, A., and Gendron, B. (2001). Bundle-based relaxation methods for multicommodity capacitated fixed charge network design. *Discrete Applied Mathematics*, 112(1):73 – 99. Combinatorial Optimization Symposium, Selected Papers.

[35] Crainic, T. G., Gendron, B., and Hernu, G. (2004). A slope scaling/lagrangean perturbation heuristic with long-term memory for multicommodity capacitated fixed-charge network design. *Journal of Heuristics*, 10(5):525–545.

[36] Croxton, K. L., Gendron, B., and Magnanti, T. L. (2007). Variable disaggregation in network flow problems with piecewise linear costs. *Operations Research*, 55(1):146–157.

[37] Danna, E., Rothberg, E., and Pape, C. L. (2004). Exploring relaxation induced neighborhoods to improve mip solutions. *Mathematical Programming*, 102(1):71–90.

[38] Dantzig, G. B. (1963). *Linear programming and extensions*. Princeton University Press.

[39] Dantzig, G. B. and Wolfe, P. (1960). Decomposition principle for linear programs. *Operations Research*, 8(1):101–111.

[40] Darnell, D. W. and Loflin, C. (1977). National Airlines Fuel Management and Allocation Model. *Interfaces*, 7(2):1–16.

[41] De, A., Choudhary, A., Turkay, M., and K. Tiwari, M. (2019). Bunkering policies for a fuel bunker management problem for liner shipping networks. *European Journal of Operational Research*.

[42] Desaulniers, G., Desrosiers, J., loachim, I., Solomon, M. M., Soumis, F., and Villeneuve, D. (1998). *A Unified Framework for Deterministic Time Constrained Vehicle Routing and Crew Scheduling Problems*, pages 57–93. Springer US, Boston, MA.

[43] Desrosiers, J., Dumas, Y., Solomon, M. M., and Soumis, F. (1995). Chapter 2 time constrained routing and scheduling. In *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*, pages 35–139. Elsevier.

[44] Desrosiers, J. and Lübbecke, M. E. (2005). *A Primer in Column Generation*, pages 1–32. Springer US, Boston, MA.

[45] Dolan, E. D. and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213.

[46] Dorigo, M., Birattari, M., and Stutzle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1(4):28–39.

[47] Edmonds, J. and Karp, R. M. (1972). Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2):248–264.

[48] Elhallaoui, I., Metrane, A., Soumis, F., and Desaulniers, G. (2008). Multi-phase dynamic constraint aggregation for set partitioning type problems. *Mathematical Programming*, 123(2):345–370.

[49] Ernst, A. T. and Krishnamoorthy, M. (1996). Efficient algorithms for the uncapacitated single allocation p-hub median problem. *Location Science*, 4(3):139–154. Hub Location.

[50] Errico, F., Crainic, T. G., Malucelli, F., and Nonato, M. (2017). A benders decomposition approach for the symmetric tsp with generalized latency arising in the design of semiflexible transit systems. *Transportation Science*, 51(2):706–722.

[51] Evans, J. R. (1983). A network decomposition/aggregation procedure for a class of multicommodity transportation problems. *Networks*, 13(2):197–205.

[52] Ferdowsi, F., Reza, H., and Maleki, S. (2018). Air refueling tanker allocation based on a multi-objective zero-one integer programming model. *Operational Research*, pages 1–26.

[53] Fischetti, M. and Fischetti, M. (2016). *Matheuristics*, pages 1–33. Springer International Publishing, Cham.

[54] Fischetti, M. and Lodi, A. (2003). Local branching. *Mathematical Programming*, 98(1-3):23–47.

[55] Fontaine, P. and Minner, S. (2018). Benders decomposition for the hazmat transport network design problem. *European Journal of Operational Research*, 267(3):996–1002.

[56] Ford, L. R. and Fulkerson, D. R. (1956). Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404.

[57] Ford, L. R. and Fulkerson, D. R. (1958). A suggested computation for maximal multi-commodity network flows. *Management Science*, 5:97–101.

[58] Francis, R. L., Lowe, T. J., Rayco, M. B., and Tamir, A. (2008). Aggregation error for location models: survey and analysis. *Annals of Operations Research*, 167(1):171–208.

[59] Fregnani, J. A. T. G., Müller, C., and Correia, A. R. (2013). A fuel tankering model applied to a domestic airline network. *Journal of Advanced Transportation*, 47(4):386–398.

[60] Gallo, G. and Pallottino, S. (1988). Shortest path algorithms. *Annals of Operations Research*, 13(1):1–79.

[61] García-Martínez, C., Rodriguez, F. J., and Lozano, M. (2018). Genetic algorithms. *Handbook of Heuristics*, page 431–464.

[62] Garey, M. R. and Johnson, D. S. (1979). *Computers And Intractability A Guide To The Theory Of Np-Completeness*. Freeman & Company, New York.

[63] Gendron, B., Crainic, T. G., and Frangioni, A. (1999). *Multicommodity Capacitated Network Design*, pages 1–19. Springer US, Boston, MA.

[64] Gendron, B., Hanafi, S., and Todosijević, R. (2018). Matheuristics based on iterative linear programming and slope scaling for multicommodity capacitated fixed charge network design. *European Journal of Operational Research*, 268(1):70 – 81.

[65] Gendron, B., Scutellà, M. G., Garroppo, R. G., Nencioni, G., and Tavanti, L. (2016). A branch-and-benders-cut method for nonlinear power design in green wireless local area networks. *European Journal of Operational Research*, 255(1):151–162.

[66] Geoffrion, A. M. and Graves, G. W. (1974). Multicommodity distribution system design by benders decomposition. *Management Science*, 20(5):822–844.

[67] Gilmore, P. C. and Gomory, R. E. (1961). A linear programming approach to the cutting-stock problem. *Operations Research*, 9(6):849–859.

[68] Gomory, R. E. (2010). *Outline of an Algorithm for Integer Solutions to Linear Programs and An Algorithm for the Mixed Integer Problem*, pages 77–103. Springer Berlin Heidelberg, Berlin, Heidelberg.

[69] Guedes, P. C. and Borenstein, D. (2018). Real-time multi-depot vehicle type rescheduling problem. *Transportation Research Part B: Methodological*, 108:217 – 234.

[70] Hane, C. A., Barnhart, C., Johnson, E. L., Marsten, R. E., Nemhauser, G. L., and Sigismondi, G. (1995). The fleet assignment problem: Solving a large-scale integer program. *Mathematical Programming*, 70(1-3):211–232.

[71] Hodgson, M. J. (1990). A Flow-Capturing Location-Allocation Model. *Geographical Analysis*, 22(3):270–279.

[72] Hosseini, M., MirHassani, S. A., and Hooshmand, F. (2017). Deviation-flow refueling location problem with capacitated facilities: Model and algorithm. *Transportation Research Part D: Transport and Environment*, 54(March 2014):269–281.

[73] Hu, J., You, S., and Østergaard, J. (2011). Optimal charging schedule of an electric vehicle fleet. In *2011 International Universities' Power Engineering Conference*, pages 5–10, Soest, Germany.

[74] Hubert, T., Guo, C., Mouton, C. A., and Powers, J. D. (2015). Tankering Fuel on U.S. Air Force Transport Aircraft An Assessment of Cost Savings. Technical report, RAND Corporation, Santa Monica,California.

[75] INFORMS Problem Solving Competition (2010). Railway Applications Section (RAS) of INFORMS. Institute for Operations Research and the Management Sciences, Hanover, MD.

[76] Jeihoonian, M., Kazemi Zanjani, M., and Gendreau, M. (2016). Accelerating benders decomposition for closed-loop supply chain network design: Case of used durable products with different quality levels. *European Journal of Operational Research*, 251(3):830–845.

[77] Kannon, T. E., Nurre, S. G., Lunday, B. J., and Hill, R. R. (2015). The aircraft routing problem with refueling. *Optimization Letters*, 9(8):1609–1624.

[78] Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395.

[79] Karp, R. M. (1972). *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA.

[80] Kazemi, A., Ernst, A. T., Krishnamoorthy, M., and Le Bodic, P. (2021a). Locomotive fuel management with inline refueling. *European Journal of Operational Research*, 293(3):1077–1096.

[81] Kazemi, A., Le Bodic, P., Ernst, A. T., and Krishnamoorthy, M. (2021b). New partial aggregations for multicommodity network flow problems: An application to the fixed-charge network design problem. *Computers & Operations Research*, 136:105505.

[82] Kheraie, A. Z. and Mahmassani, H. S. (2012). Leveraging Fuel Cost Differences in Aircraft Routing by Considering Fuel Ferrying Strategies. *Transportation Research Record: Journal of the Transportation Research Board*, 2300(1):139–146.

[83] Kim, J. G. and Kuby, M. (2012). The deviation-flow refueling location model for optimizing a network of refueling stations. *International Journal of Hydrogen Energy*, 37(6):5406–5420.

[84] Kirkpatrick, S. (1984). Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, 34(5-6):975–986.

[85] Kliewer, N., Mellouli, T., and Suhl, L. (2006). A time–space network based exact optimization model for multi-depot bus scheduling. *European Journal of Operational Research*, 175(3):1616 – 1627.

[86] Kuby, M. and Lim, S. (2005). The flow-refueling location problem for alternative-fuel vehicles. *Socio-*

Economic Planning Sciences, 39(2):125–145.

[87] Kulkarni, S., Krishnamoorthy, M., Ranade, A., Ernst, A. T., and Patil, R. (2018). A new formulation and a column generation-based heuristic for the multiple depot vehicle scheduling problem. *Transportation Research Part B: Methodological*, 118:457–487.

[88] Kumar, V. P. and Bierlaire, M. (2015). Optimizing Fueling Decisions for Locomotives in Railroad Networks. *Transportation Science*, 49(1):149–159.

[89] Land, A. H. and Doig, A. G. (1960). An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520.

[90] Lemke, C. E. (1954). The dual method of solving the linear programming problem. *Naval Research Logistics Quarterly*, 1(1):36–47.

[91] Li, X., Chien, C. F., Yang, L., and Gao, Z. (2014). The train fueling cost minimization problem with fuzzy fuel prices. *Flexible Services and Manufacturing Journal*, 26(1-2):249–267.

[92] Litvinchev, I. and Tsurkov, V. (2003). *Iterative Aggregation-Decomposition in Optimization Problems*, pages 61–161. Springer US, Boston, MA.

[93] Lübbecke, M. E. and Desrosiers, J. (2005). Selected topics in column generation. *Operations Research*, 53(6):1007–1023.

[94] Magnanti, T. L. and Wong, R. T. (1981). Accelerating benders decomposition: Algorithmic enhancement and model selection criteria. *Operations Research*, 29(3):464–484.

[95] Magnanti, T. L. and Wong, R. T. (1984). Network design and transportation planning: Models and algorithms. *Transportation Science*, 18(1):1–55.

[96] Mamer, J. W. and McBride, R. D. (2000). A decomposition-based pricing procedure for large-scale linear programs: An application to the linear multicommodity flow problem. *Management Science*, 46(5):693–709.

[97] Marin, G., Naterer, G., and Gabriel, K. (2010). Rail transportation by hydrogen vs. electrification – case study for ontario canada, i: Propulsion and storage. *International Journal of Hydrogen Energy*, 35(12):6084 – 6096.

[98] Meng, Q., Wang, S., and Lee, C. Y. (2015). A tailored branch-and-price approach for a joint tramp ship routing and bunkering problem. *Transportation Research Part B: Methodological*, 72:1–19.

[99] MirHassani, S. A. and Ebrazi, R. (2013). A Flexible Reformulation of the Refueling Station Location Problem. *Transportation Science*, 47(4):617–628.

[100] Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.

[101] Mokhtar, H., Krishnamoorthy, M., and Ernst, A. T. (2019). The 2-allocation p-hub median problem and a modified benders decomposition method for solving hub location problems. *Computers & Operations Research*, 104:375–393.

[102] Moreno, A., Munari, P., and Alem, D. (2019). A branch-and-benders-cut algorithm for the crew scheduling and routing problem in road restoration. *European Journal of Operational Research*, 275(1):16–34.

[103] Nag, B. and Murty, K. G. (2012). Diesel locomotive fueling problem (LFP) in railroad operations. *OPSEARCH*.

[104] Nazari, A., Ernst, A., Dunstall, S., Bryan, B., Connor, J., Nolan, M., and Stock, F. (2015). Combined aggregation and column generation for land-use trade-off optimisation. In Denzer, R., Argent, R. M., Schimak, G., and Hřebíček, J., editors, *Environmental Software Systems. Infrastructures, Services and Applications*, pages 455–466, Cham. Springer International Publishing.

[105] Nemhauser, G. L. and Wolsey, L. A. (1999). *Integer and combinatorial optimization*. John Wiley & Sons.

[106] Nourbakhsh, S. M. and Ouyang, Y. (2010). Optimal fueling strategies for locomotive fleets in railroad networks. *Transportation Research Part B: Methodological*, 44(8-9):1104–1114.

[107] Orlin, J. B. (1997). A polynomial time primal network simplex algorithm for minimum cost flows. *Mathematical Programming*, 78(2):109–129.

[108] Oğuz, M., Bektaş, T., and Bennell, J. A. (2018). Multicommodity flows and benders decomposition for restricted continuous location problems. *European Journal of Operational Research*, 266(3):851–863.

[109] Park, Y. W. and Klabjan, D. (2016). An aggregate and iterative disaggregate algorithm with proven optimality in machine learning. *Machine Learning*, 105(2):199–232.

[110] Philpott, A. B. and Mudaliar, S. K. (1992). Constrained Fuel Tankering on a Transportation Network. In *New Zealand Operational Research 1992 Conference*, pages 271–278, Christchurch, New Zealand.

[111] Pishvaee, M., Razmi, J., and Torabi, S. (2014). An accelerated benders decomposition algorithm for sustainable supply chain network design under uncertainty: A case study of medical needle and syringe supply chain. *Transportation Research Part E: Logistics and Transportation Review*, 67:14–38.

[112] Plum, C. E. M., Jensen, P. N., and Pisinger, D. (2014). Bunker purchasing with contracts. *Maritime Economics and Logistics*, 16(4):418–435.

[113] Raack, C., Koster, A. M., Orlowski, S., and Wessäly, R. (2011). On cut-based inequalities for capacitated network design polyhedra. *Networks*, 57(2):141–156.

[114] Rahmaniani, R., Crainic, T. G., Gendreau, M., and Rei, W. (2017). The benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3):801–817.

[115] Ramsden, E. (2010). RAS Problem Solving Competition 2010. Technical report, Lattice Semiconductor Corporation.

[116] Ranjbar, M. and Kazemi, A. (2018). A generalized variable neighborhood search algorithm for the talent scheduling problem. *Computers & Industrial Engineering*, 126:673–680.

[117] Rardin, R. L. and Wolsey, L. A. (1993). Valid inequalities and projecting the multicommodity extended formulation for uncapacitated fixed charge network flow problems. *European Journal of Operational Research*, 71(1):95 – 109.

[118] Raviv, T. and Kaspi, M. (2012). The locomotive fleet fueling problem. *Operations Research Letters*, 40(1):39–45.

[119] Reinhardt, L. B., Pisinger, D., Sigurd, M. M., and Ahmt, J. (2020). Speed optimizations for liner networks with business constraints. *European Journal of Operational Research*, 285(3):1127 – 1140.

[120] Rodríguez-Martín, I. and José Salazar-González, J. (2010). A local branching heuristic for the capacitated fixed-charge network design problem. *Computers & Operations Research*, 37(3):575 – 581. Hybrid Metaheuristics.

[121] Rogers, D. F., Plante, R. D., Wong, R. T., and Evans, J. R. (1991). Aggregation and disaggregation

techniques and methodology in optimization. *Operations Research*, 39(4):553–582.

[122] Sassi, O. and Oulamara, A. (2017). Electric vehicle scheduling and optimal charging problem: complexity, exact and heuristic approaches. *International Journal of Production Research*, 55(2):519–535.

[123] Schindl, D. and Zufferey, N. (2013). Solution Methods for Fuel Supply of Trains. *INFOR: Information Systems and Operational Research*, 51(1):23–30.

[124] Schwartz, M. and Stern, T. (1980). Routing techniques used in computer communication networks. *IEEE Transactions on Communications*, 28(4):539–552.

[125] Sheng, X., Chew, E. P., and Lee, L. H. (2015). (s, S) policy model for liner shipping refueling and sailing speed optimization problem. *Transportation Research Part E: Logistics and Transportation Review*, 76:76–92.

[126] Sheng, X., Lee, L. H., and Chew, E. P. (2014). Dynamic determination of vessel speed and selection of bunkering ports for liner shipping under stochastic environment. *OR Spectrum*, 36(2):455–480.

[127] Shetty, C. and Taylor, R. W. (1987). Solving large-scale linear programs by aggregation. *Computers & Operations Research*, 14(5):385–393.

[128] Smale, S. (1983). On the average number of steps of the simplex method of linear programming. *Mathematical Programming*, 27(3):241–262.

[129] Spielman, D. A. and Teng, S.-H. (2004). Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM*, 51(3):385–463.

[130] Steinzen, I., Gintner, V., Suhl, L., and Kliewer, N. (2010). A time-space network approach for the integrated vehicle- and crew-scheduling problem with multiple depots. *Transportation Science*, 44(3):367–382.

[131] Stroup, J. S. (1992). A Fuel Management Model for the Airline Industry. *Operations Research*, 40(2):229–237.

[132] Sundström, O. and Binding, C. (2010). Optimization Methods to Plan the Charging of Electric Vehicle Fleets. In *2010 International Conference on Control Communication and Power Engineering*, pages 28–29, Chennai, India.

[133] Suzuki, Y. (2012). A decision support system of vehicle routing and refueling for motor carriers with time-sensitive demands. *Decision Support Systems*, 54(1):758–767.

[134] Suzuki, Y., Montabon, F., and Lu, S. H. (2014). DSS of vehicle refueling: A new enhanced approach with fuel weight considerations. *Decision Support Systems*, 68:15–25.

[135] Sweeney, D. J. and Tatham, R. L. (1976). An improved long-run model for multiple warehouse location. *Management Science*, 22(7):748–758.

[136] Teodorović, D. (1988). Strategy for the purchase of fuel on an airline network. *Transportation Planning and Technology*, 12(1):39–44.

[137] Union Pacific Corporation (2018). Annual Report 2017 on Form 10-K. Technical report, Union Pacific Corporation, Omaha, Nebraska.

[138] Upchurch, C., Kuby, M., and Lim, S. (2009). A model for location of capacitated alternative-fuel stations. *Geographical Analysis*, 41(1):127–148.

[139] Vaidyanathan, B., Ahuja, R. K., and Orlin, J. B. (2008). The Locomotive Routing Problem. *Transportation Science*, 42(4).

[140] Vanderbeck, F. (2000). On dantzig-wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research*, 48(1):111–128.

[141] Vanderbei, R. J. (2014). *Linear Programming Foundations and Extensions*. Springer US.

[142] Vermit, A. K. (2014). *Strategic placement of telemetry units and locomotive fuel planning*. PhD thesis, University of Iowa.

[143] Vilhelmsen, C., Lusby, R., and Larsen, J. (2014). Tramp ship routing and scheduling with integrated bunker optimization. *EURO Journal on Transportation and Logistics*, 3(2):143–175.

[144] Wang, C. and Chen, J. (2017). Strategies of refueling, sailing speed and ship deployment of containerships in the low-carbon background. *Computers and Industrial Engineering*, 114(September):142–150.

[145] Wang, I. L., Wang, Y., and Lin, P. C. (2016). Optimal recharging strategies for electric vehicle fleets with duration constraints. *Transportation Research Part C: Emerging Technologies*, 69:242–254.

[146] Wang, S. and Meng, Q. (2015). Robust bunker management for liner shipping networks. *European Journal of Operational Research*, 243(3):789–797.

[147] Wang, Y.-W. and Lin, C.-C. (2009). Locating road-vehicle refueling stations. *Transportation Research Part E: Logistics and Transportation Review*, 45(5):821–829.

[148] Williams, H. P. (2013). *Model building in mathematical programming*. Wiley.

[149] Wong, R. T. (1984). A dual ascent approach for steiner tree problems on a directed graph. *Mathematical Programming*, 28(3):271–287.

[150] Yaghini, M., Rahbar, M., and Karimi, M. (2013). A hybrid simulated annealing and column generation approach for capacitated multicommodity network design. *Journal of the Operational Research Society*, 64(7):1010–1020.

[151] Yamani, A. (1986). *Analysis of an air transportation system*. PhD thesis, University of Florida.

[152] Yao, Z., Ng, S. H., and Lee, L. H. (2012). A study on bunker fuel management for the shipping liner services. *Computers and Operations Research*, 39(5):1160–1172.

[153] Yen, J. Y. (1971). Finding the k shortest loopless paths in a network. *Management Science*, 17(11):712–716.

[154] Yildiz, B., Arslan, O., and Karasan, O. E. (2016). A branch and price approach for routing and refueling station location model. *European Journal of Operational Research*, 248(3):815 – 826.

[155] Zetina, C. A., Contreras, I., and Cordeau, J.-F. (2019). Exact algorithms based on benders decomposition for multicommodity uncapacitated fixed-charge network design. *Computers & Operations Research*, 111:311 – 324.

[156] Zhen, L., Wang, S., and Zhuge, D. (2017). Dynamic programming for optimal ship refueling decision. *Transportation Research Part E: Logistics and Transportation Review*, 100(December 2015):63–74.

[157] Zipkin, P. H. (1980a). Bounds for row-aggregation in linear programming. *Operations Research*, 28(4):903–916.

[158] Zipkin, P. H. (1980b). Bounds on the effect of aggregating variables in linear programs. *Operations Research*, 28(2):403–418.

[159] Zouein, P. P., Abillama, W. R., and Tohme, E. (2002). A multiple period capacitated inventory model

for airline fuel management: A case study. *Journal of the Operational Research Society*, 53(4):379–386.

[160] Árton P. Dorneles, de Araújo, O. C., and Buriol, L. S. (2017). A column generation approach to high school timetabling modeled as a multicommodity flow problem. *European Journal of Operational Research*, 256(3):685–695.