OPTIMIZATION IN MARITIME INVENTORY ROUTING

A Thesis
Presented to
The Academic Faculty

by

Dimitri J. Papageorgiou

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
The H. Milton Stewart School of Industrial and Systems Engineering

Georgia Institute of Technology
December 2012

OPTIMIZATION IN MARITIME INVENTORY ROUTING

Approved by:

Professor George L. Nemhauser, Co-advisor
School of Industrial and Systems Engineering
Georgia Institute of Technology

Professor Joel Sokol, Co-advisor
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Ahmet Keha
Corporate Strategic Research
ExxonMobil Research and Engineering

Professor Shabbir Ahmed
School of Industrial and Systems Engineering
Georgia Institute of Technology

Professor Alan Erera
School of Industrial and Systems Engineering
Georgia Institute of Technology

Date Approved: 1 November 2012

To my parents, George and Helen, and to my wife, Kathleen.

ACKNOWLEDGEMENTS

Of all people at Georgia Tech, I would like to express my sincerest thanks to George Nemhauser. It was always my goal to work with one of the world's best researchers in optimization, and to have the opportunity to collaborate with him, a "titan in the field" (as noted by his colleagues) who literally wrote "the book" on integer programming, will be one of the most valuable souvenirs from my Ph.D. experience. On top of his roughly 50 years of experience in discrete optimization, his good nature and his almost comic-book-superhero-like passion for optimizing anything he can get his hands on, made my Ph.D. experience all the more enjoyable.

I am very thankful to Joel Sokol for his advice and fresh ideas. His keen eye for spotting improvements in models, algorithms, proofs, and papers was a tremendous asset. Likewise, his ability to construct illustrative examples with the snap of a finger was invaluable. It is no wonder that he has received so many teaching awards.

I would like to thank Martin Savelsbergh for his mentorship, his shared affinity of computational research, and his support of generating new approaches. After he said very earlier on that "We (as researchers) are in the business of generating ideas," I felt a certain freedom to explore ideas even if the probability of success was far from one. I am sorry that we only had two years to work together, but it was a very good opportunity for me all the same.

I owe a great deal of thanks to Shabbir Ahmed for his mentorship. Although not my official advisor, he gave me considerable guidance throughout my tenure at Georgia Tech. I am grateful for his time and collaboration. I thank him for exposing me to submodular functions and showing me their power in tackling certain classes of binary optimization problems.

I thank Alan Erera and Ahmet Keha for serving on my thesis committee. I am grateful to: the National Science Foundation for its financial support; ExxonMobil Research and

Engineering for sponsoring my dissertation; ExxonMobil researchers Ahmet Keha, Myun-Seok Cheon, Jin-Hwa Song, Amr El-Bakry, and Nick Sawaya for fruitful discussions; and my professors Arkadi Nemirovski, Bill Cook, Renato Monteiro, Chip White, and many others.

Last, but not least, I am indebted to my peers for their friendship and support. I list them in random order:

- Joshua Schultz. Special thanks to Josh for several years of empathy, support, motivation, and friendship as we both went through our PhDs at a similar stage in life and found light at the end of the tunnel.
- My student collaborators on the ExxonMobil project: Faram Engineer, Mike Hewitt, Alejandro Toriello, and Chengliang Zhang.
- My classmates who entered with me: Camilo Ortiz, Vinod Cheriyan, George Thiers, Monica Villareal, and Todd Levin.
- The Chilean crowd: Rodolfo Carvajal, Diego Ramirez, and, in particular, Gustavo Angulo for being such a pleasant officemate for three years.
- My Egyptian friends: Aly Megahed and Ahmed Nazeem.
- To those who came before me: Steve Tyber, Kael Stilp, Dan Dadush, Jon Petersen, Juan Pablo Vielma, Fatma Kilinc Karzan, Guanghui Lan, Byungsoo Na, So-Yeon Chun, Helder Inacio, Dan Steffy, Akshay Gupte, Feng Qui, Yaxian Li, Qie He, and many others.
- To those who came after me: Eric Tollefson, Soheil Shayegh, Javad Feizollahi, Wajdi Tekaya, Brian Kues, Mallory Soldner, Murat Yildirim, Pantelis Loupos, Carl Morris, Matt Plumlee, Tim Sprock, and so many more.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The primary aim of this thesis is to develop effective solution techniques for large-scale maritime inventory routing problems that possess a core substructure common in many real-world applications. We use the term "large-scale" to refer to problems whose standard mixed-integer linear programming (MIP) formulations involve tens of thousands of binary decision variables and tens of thousands of constraints and require days to solve on a personal computer. Although a large body of literature already exists for problems combining vehicle routing and inventory control for road-based applications, relatively little work has been published in the realm of maritime logistics. A major contribution of this research is in the advancement of novel methods for tackling problems orders of magnitude larger than most of those considered in the literature.

We first present a detailed description of a particular class of deterministic single product maritime inventory routing problems (MIRPs), which we call MIRPs with inventory tracking at every port. After providing a comprehensive literature survey of this class of MIRPs, we introduce a MIP model for a core maritime inventory routing problem. In addition to being a centerpiece of this thesis, this model is quite general and incorporates assumptions and families of constraints that are most prevalent in practice. We also discuss other modeling features commonly found in the literature and how they can be incorporated into the core model. Next, we offer what appears to be the first unified discussion of some of the most common advanced techniques used for solving these problems. Finally, we present a library, called MIRPLib, of publicly available test problem instances for MIRPs with inventory tracking at every port. Despite a growing interest in combined routing and inventory management problems in a maritime setting, no data sets are publicly available, which represents a significant "barrier to entry" for those interested in related research. Our main goal for MIRPLib is to help maritime inventory routing gain maturity as an important and interesting class of planning problems. As a means to this end, we

(1) make available benchmark instances for a particular class of MIRPs; (2) provide the mixed-integer linear programming community with a set of optimization problem instances from the maritime transportation domain in LP and MPS format; (3) provide a template for other researchers when specifying characteristics of MIRPs arising in other settings. Best known computational results are reported for each instance.

We next present a two-stage decomposition algorithm for the single product maritime inventory routing problem defined above. The problem involves routing vessels, each belonging to a particular vessel class, between loading and discharging ports, each belonging to a particular region. We call our algorithm "Zoom" because it iteratively solves a MIRP by zooming out and then zooming in on the problem. Specifically, in the "zoomed out" phase, we solve a first-stage master problem in which aggregate information about regions and vessel classes is used to route vessels between regions, while only implicitly considering inventory and capacity requirements, berth limits, and other side constraints. In the "zoomed in" phase, we solve a series of second-stage subproblems, one for each region, in which individual vessels are routed through each region and loading and discharge quantities are determined. Our algorithm bears a close resemblance to Benders decomposition for mixed-integer linear optimization except that our second-stage problems are mixed-integer linear programs, not pure linear programs. Not only is our solution approach different from previous methods discussed in the maritime transportation literature, but computational experience shows that our approach is promising.

In our next chapter, we study a maritime inventory routing problem with a long planning horizon of up to 365 periods (days). For instances with many ports and many vessels, MIP solvers often require hours to produce good solutions even when the planning horizon is 90 or 120 periods. Building on the recent successes of approximate dynamic programming (ADP) for road-based applications within the transportation community, we develop an ADP procedure to quickly generate good solutions to these problems within minutes. Our algorithm operates by solving many small subproblems (one for each period) and, in so doing, collecting and learning information about how to produce better solutions. Our algorithm is one of the first of its kind for maritime transportation problems and represents a significant departure from the traditional methods used. In particular, whereas virtually all existing methods are “MIP-centric,” i.e., they rely heavily on a solver to solve a nontrivial MIP in a couple of minutes to generate a good or improving solution, our framework puts the effort on finding suitable value function approximations and places much less responsibility on the solver. Computational results illustrate that with a relatively simple framework, our ADP approach is able to generate good solutions to instances with dozens of vessels and varying time horizons much faster than a commercial solver emphasizing feasibility.

Our final research contribution is a polyhedral study of an optimization problem involving a single time period that was motivated by maritime inventory routing, but is applicable to a more general class of problems outside those in a maritime setting. Numerous planning models within the chemical, petroleum, and process industries involve coordinating the movement of raw materials in a distribution network so that they can be blended into final products. The uncapacitated fixed-charge transportation problem with blending (FCTPwB) studied in this chapter captures a core structure encountered in many of these environments. We model the FCTPwB as a mixed-integer linear program and derive two classes of facets, both exponential in size, for the convex hull of solutions for the problem with a single consumer and show that they can be separated in polynomial time. Furthermore, we prove that in certain situations these classes of facets, along with the continuous relaxation of the original constraints, yield a complete description of the convex hull. Finally, we present a computational study that demonstrates that these classes of facets are effective in reducing the integrality gap and solution time for more general instances of the FCTPwB with arc capacities and multiple consumers.

In terms of overall impact, this thesis makes several important contributions. From a scientific perspective, perhaps our most significant contribution is the philosophy that we introduce for attacking large-scale maritime inventory routing problems. By exploiting aggregation and decomposition, our methods represent a paradigm shift in the way one should approach such problems. From a practical perspective, billions of dollars are spent annually shipping bulk goods all across the globe and our models and algorithms have the potential to improve the decision support systems responsible for the movement of

these goods. Lastly, we hope that this thesis brings greater attention to the maritime transportation sector from the OR and transportation science communities.

CHAPTER I

INTRODUCTION

Despite its lack of prominence within the operations research (OR) and transportation science communities, maritime transportation is an essential component of global trade. Capitalizing on their size, strength, and economies of scale, seafaring vessels are responsible for moving vast quantities of goods all around the globe at costs unmatched by other modes of transportation. Proper management and coordination within this sector is critical to the vitality of countless supply chains, yet the application of OR techniques has remained largely untapped compared to its counterparts in the broader transportation industry. The primary contribution of this thesis is the development of effective solution techniques for an important class of optimization problems arising in the maritime transportation industry. As such, we first introduce maritime transportation, emphasize its role in the international trade, and ultimately make a case for why more effective solution methods are a meaningful contribution.

1.1 Maritime Transportation

The maritime transportation industry forms the backbone of several national economies and plays a critical role in the international competitiveness of many others. Virtually all global supply chains include at least one maritime leg, leading to issues of improved integration and giving rise to intricate logistics problems with complex decisions [79]. Figures 1 and 2 underscore the vital role sea-based transport plays in the global economy. For years, it has been responsible for moving over 70% of all international trade in terms of value. Perhaps more compelling is the fact that it is responsible for moving between 80% and 90% of all international trade in terms of volume. Without question, maritime transportation has a stronghold on moving large quantities of goods between continents [28].

Of the four primary modes of industrial transport - air, rail, sea, and truck - sea-based transportation is arguably the most underrepresented within the OR and transportation

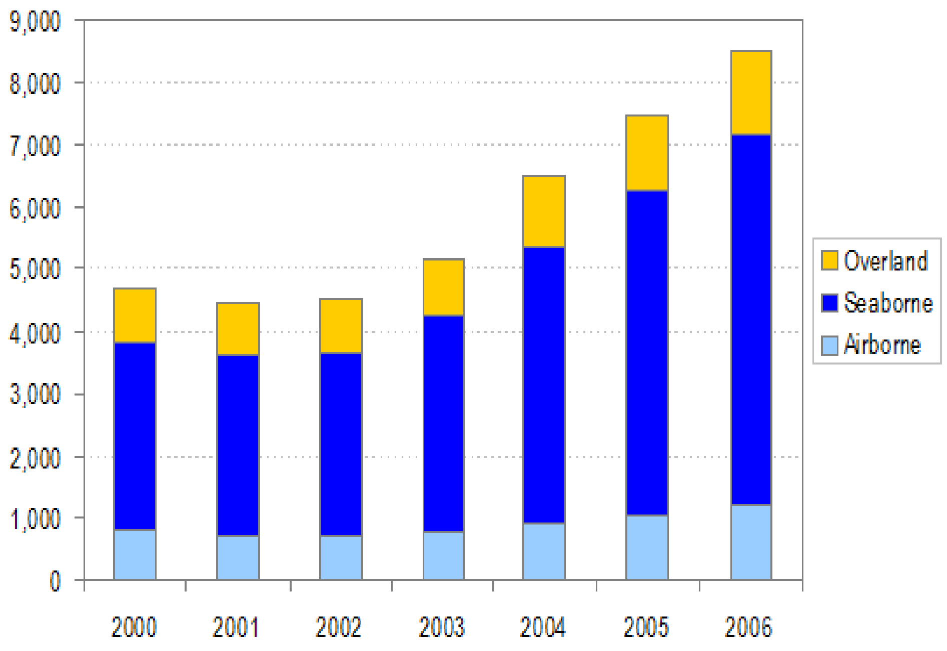


Figure 1: Modal split of international trade in value (US$ billion) 2000-2006. Source: Rodrigue et al. [83] (adapted from GlobalInsight).

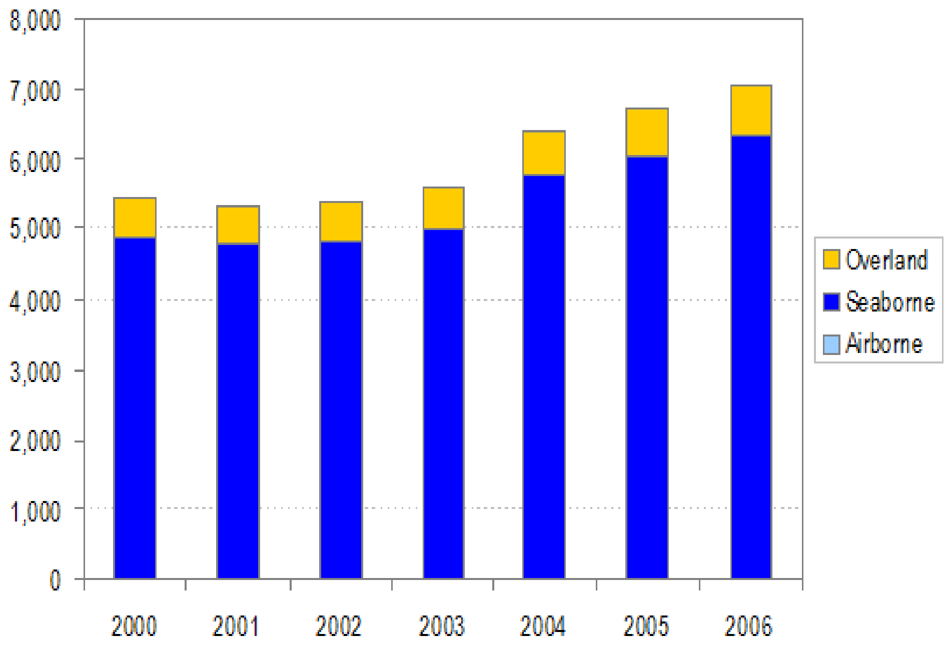


Figure 2: Modal split of international trade in goods (million metric tons) 2000-2006. Source: Rodrigue et al. [83] (adapted from GlobalInsight).

science communities. This fact is somewhat surprising given the richness of the planning problems that arise and the potential economic and environmental impact added efficiency could provide. As Figure 3 shows, transporting freight via water is relatively cost-effective. There are several indicators of this underrepresentation. First, compared to other modes of transportation, there are very few special-interest groups devoted to the maritime industry. For example, the Institute for Operations Research and Management Science (INFORMS) has a section, or special-interest group, for air (the Aviation Application Section) and for rail (the Rail Applications Section), but no such group for maritime applications. Second, there are only a handful of publicly reported OR-based decision support systems being used for maritime applications compared with dozens used in road-based ones. Whereas OR is now ubiquitous in the way airlines develop their schedules, price their itineraries, route their aircraft, and schedule their crew, expert judgment and traditional spreadsheet planning remain prominent tools in the maritime environment. Third, there are no publicly available benchmark instances on which researchers can test their algorithms [29].

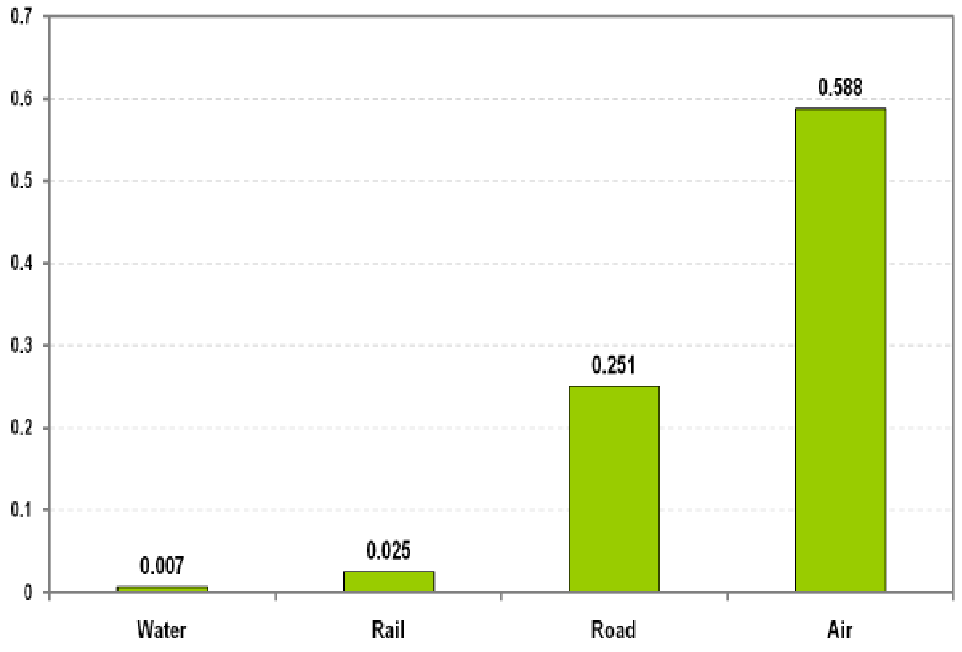


Figure 3: Freight transport costs in cents per ton-mile. Source: Rodrigue et al. [83] (adapted from Ballou [13]).

The three main ingredients of maritime transportation are vessels, ports, and cargos, which we briefly describe in turn. There are numerous types of vessels, from general-purpose cargo vessels to industry-specific ships like liquefied natural gas carriers. Vessels are further classified by their size, expressed in deadweight tonnage (dwt), length, and width. In contrast to trucks in road-based transport, vessels involve major capital investments and high operating costs. For example, approximate costs associated with a 200,000 dwt very large crude carrier (VLCC) include: a construction cost of roughly US $100 million, daily time-charter rates of US $50,000-100,000, daily fuel costs of US $25,000-50,000, and port fees of US $10,000-100,000. Meanwhile, the value of the cargo on-board is roughly US $100 million.

Ports act as transfer points for trade. Because they offer materials handling equipment and facilities for storing and transfering cargo, ports charge vessels a port fee for their services. Idiosynacrasies at a port, such as draft limits, the number of berths, and the amount and type of equipment, influence the number of times a vessel loads or discharges at a particular port in a given time horizon. In this research, port operations are not explicitly modeled as our focus is on the movement of vessels over time. Nonetheless, OR has become integral to port terminal operations [93, 95].

Cargo (or freight) is a set of goods shipped together from a single origin to a single destination. In the vehicle routing literature, it is often referred to as an order. Maritime cargo is conventionally broken down into two categories: break-bulk cargo and bulk cargo. Break-bulk cargo refers to general cargo that is packaged, itemized by containers, and transported by container ships. This cargo tends to have numerous origins, destinations, and clients. Before containerization, economies of scale were difficult to achieve with break-bulk cargo as the loading and unloading process was very labor and time consuming [83]. Bulk cargo, which is our primary focus in this thesis, refers to dry and liquid freight that is not packaged or palletized such as oil, liquid chemicals, coal, iron ore, phosphate, bauxite, and grain. It is the dominant player in maritime shipping. It often requires the use of specialized ships such as oil tankers as well as specialized transshipment and storage facilities. Conventionally, it has a single origin, destination and client and is prone to economies of scale.

Table 1: Development of international seaborne trade, selected years (millions of tons loaded). Source: UNCTAD [105].

Year	**Oil**	**Main bulks**	**Other dry cargo**	**Total (all cargoes)**
1970	1442	448	676	2566
1980	1871	796	1037	3704
1990	1755	968	1285	4008
2000	2163	1288	2533	5984
2006	2698	1836	3166	7700
2007	2747	1957	3330	8034
2008	2742	2059	3428	8229
2009	2642	2094	3122	7858
2010	2752	2333	3323	8408

Services tend to be irregular, except for energy trades, and part of vertically integrated production processes. For more details, see Christiansen et al. [28], Rodrigue et al. [83], and Stopford [96].

In the petro-chemical industry, bulk products are the most prevalent. Energy-related commodities like crude oil, oil products, LNG, and thermal coal account for 44% of seaborne trade by weight [96]. As of 2005, about 2.4 billion tons of petroleum were shipped by maritime transportation, which is roughly 62% of all the petroleum produced [106]. There are roughly 4000 tankers available on the international oil transportation market for this distribution. Meanwhile, transportation costs account for about 5 to 10% of the added value of oil, and they depend on the amounts of oil carried, the origin from which it is extracted, and the destination to which it is being transported. In the global LNG market, some estimate that shipping costs account for 10 to 30% of all costs. Although costs are important, they do not tell the entire story. A well-devised routing schedule for a fleet of vessels may also allow vessels to satisfy demand spikes that occur due to external factors. In this case, a small increase in shipping costs may result in a significant profit.

1.2 Maritime Inventory Routing

Of the nearly 9 billion tons of goods in international seaborne commerce traded in 2011, bulk goods such as coal, crude oil, iron ore, and liquefied natural gas accounted for well over 50% of this quantity and easily represented several hundreds of billions of US dollars in value [105] (see also Table 1). With such figures expected to grow over future decades, effective maritime transportation is paramount. A focal point of this thesis is on a particular maritime transportation planning problem known as the Maritime Inventory Routing Problem (MIRP), which plays an integral role in global bulk shipping.

Inventory routing problems (IRPs) involve the integration and coordination of two components of the logistics value chain: inventory management and vehicle routing. Maritime inventory routing problems are a special class of IRPs that arise in a maritime setting. IRPs have come to prominence because they are an integral component in vendor managed inventory replenishment (VMR), a policy in which a central decision maker coordinates both the inventory and its distribution within a supply chain [22]. The survey paper on combined inventory management and vehicle routing problems by Andersson et al. [8] provides a summary of research on IRPs in road and maritime settings.

A MIRP is best described in terms of its main components: ports and vessels. Each port is classified as a loading port where product is produced and loaded onto vessels or as a discharging port where product is consumed, typically after being discharged from vessels or from an alternative source (e.g., a pipeline). Product can be stored in inventory at both types of ports. Each port typically has: exactly one classification type, "loading" or "discharging"; an inventory capacity that may change over time; a fixed number of berths limiting the number of vessels that can simultaneously load or discharge in a given time period; lower and upper bounds on the amount of product that can be loaded or discharged in a period; and deterministic bounds on the rate of production or consumption. In some settings, rates are only monitored for one side of the supply chain, e.g., only on the production side. When rates are not specified at a port, a set of time windows, each with a minimum quantity that should be loaded or discharged, may be given. Alternatively, there are settings in which contracts with customers (discharging ports) outline monthly demands, or state that a certain amount of product is to be delivered fairly evenly spread throughout the year. Over- and under-deliveries may be accepted, but may incur a penalty.

To transport the product, the planners control or charter a heterogeneous fleet of vessels.

Each vessel belongs to a particular vessel class and has a fixed capacity, a cruising speed, and a travel cost. Vessels make voyages between ports by picking up inventory at one or more ports and delivering inventory to one or more ports. Vessels may partially load and discharge so that two or more ports of the same type (loading or discharging) may be visited in succession. In general, a vessel will fully discharge before loading at another port.

In Chapters 2 and 3, we address a MIRP with precisely the characteristics listed above. Using the nomenclature of Andersson et al. [8], this type of MIRP can be classified as a deterministic, finite-horizon, split-pickup and split-delivery problem. The solution of this planning problem specifies routes, i.e., the sequence and times of ports visited, for each vessel as well as the quantity of product loaded or discharged in each time period by each vessel. In Chapter 4, we study a simplified version of this problem in which there are no split pickups or split deliveries, but instead a much longer time horizon. Such problems are typical encountered in the supply chains for liquefied natural gas.

Having discussed the basic characteristics of a MIRP, we now attempt to distinguish this problem from the class of road-based IRPs, which have received far more attention in the literature. MIRPs possess several noteworthy idiosyncrasies that differentiate them from an IRP typically encountered in road-based applications (see, e.g., [8]). First, the classical IRP assumes that a fleet of vehicles are located at a central depot (a single supplier) and are dispatched to customers to satisfy demand before returning to the depot in the same period. In a maritime setting, the notion of a single central depot is conspicuously absent. Likewise, vessels are typically traveling long distances and around the clock making the time dimension of the problem very important. Second, the planning horizon is typically longer in a maritime setting due to time-consuming port operations and long travel times. On the other hand, with shorter planning horizons, models for road-based applications typically require finer granularity. Third, in traditional IRP models, inventory levels are tracked only at customers, not at the supplier (the depot). Fourth, in a maritime setting, vessels typically visit relatively few (3 or fewer) ports in succession when loading or discharging, whereas traditional IRPs may involve tens of customers to visit with a small quantity (relative to vehicle capacity) being loaded at each visit.

It is also important to distinguish maritime inventory routing problems from a closely related class of problems known as *cargo routing problems*. As discussed in Al-Khayyal and Hwang [5] and Hwang [60], cargo routing problems are mainly constrained by the cargo, which is usually defined by the loading and discharging ports, and by *time windows* for loading and discharging. Inventory routing problems are constrained by inventory requirements such that the inventory level of products at ports should be maintained. In general, cargo routing is performed under more restrictive constraints since the time windows to load and discharge are usually narrow and the quantities to be loaded and discharged are known in advance. In contrast, in a MIRP, the number of calls (i.e., visits) at a given port over the planning horizon, the quantity to be loaded or discharged at each port call, as well as the port pickup and delivery pairings are not specified in the data. Thus, due to the larger solution space, it can be argued that maritime inventory routing is often more challenging computationally than traditional cargo routing.

Even within the class of MIRPs, such problems are typically classified along several axes. The first axis concerns the type of planning: strategic, tactical, and/or operational. In a maritime setting, strategic planning involves decisions over a long time horizon of one to twenty years. Tactical planning usually involves several months, possibly up to a year, of vessel routing and product distribution decisions. Operational planning requires the finest granularity and typically focuses on a planning horizon of several weeks or a few months. The second axis is the type of shipping environment: industrial, tramp, or liner [64, 85]. Industrial operators own or control both the vessels and cargo to be transported, and focus on minimizing their transport costs. Tramp shipping is analogous to a taxi service, as the vessels go after cargoes that become available in the market. Liner shipping, for which there are virtually no MIRP applications in the literature, resembles bus line operations since the vessels follow published itineraries and schedules. In practice, MIRP applications may involve elements from both industrial and tramp shipping (see, e.g., [6]). The third axis distinguishes between deep-sea and short-sea shipping. Deep-sea shipping pertains to intercontinental trips through deep seas in which travel times are much longer than the time required to load and discharge at ports. Short-sea shipping typically refers to short regional trips in which travel times are likely to be shorter than the time requirements at a port, and therefore port operations and service constraints are necessary to adequately model reality.

1.3 Mixed-Integer Linear Programming

A common thread in this thesis is the use of mixed-integer linear programming (MIP for short). In this section, we collect some of the essential concepts that will play a role throughout the thesis. For an extensive treatment of the subject, see Nemhauser and Wolsey [71].

1.3.1 Basic Concepts of Mixed-Integer Linear Programming

A MIP problem is given by

$$
\begin{aligned}
(MIP) \quad \min \ & \mathbf{c}^T\mathbf{x} && && \text{(1a)}\\
\text{s.t.} \ & \mathbf{A}\mathbf{x} \geq \mathbf{b} && && \text{(1b)}\\
& x_j \in \{0,1\}\ , && \forall\, j \in \mathcal{B} && \text{(1c)}\\
& x_j \in \mathbb{Z}_+\ , && \forall\, j \in \mathcal{G} && \text{(1d)}\\
& x_j \geq 0\ , && \forall\, j \in \mathcal{C} && \text{(1e)}
\end{aligned}
$$

where $\mathbf{x}$ is an n-dimensional vector of decision variables, $\mathbb{Z}_+$ is the set of non-negative integers, $\mathcal{B} \subseteq \{1,\ldots,n\}$ is the set of binary variables, $\mathcal{G} \subseteq \{1,\ldots,n\}$ is the set of general nonnegative integer variables, and $\mathcal{C} \subseteq \{1,\ldots,n\}$ is the set of continuous variables. The sets $\mathcal{B}$, $\mathcal{G}$, and $\mathcal{C}$ partition $\{1,\ldots,n\}$. An instance of (1) is specified by the data $(\mathbf{c},\mathbf{A},\mathbf{b})$ where $\mathbf{c} \in \mathbb{Q}^n$ is the cost vector ($\mathbb{Q}$ is the set of rationals), $\mathbf{A} \in \mathbb{Q}^{m\times n}$ is the constraint matrix, and $\mathbf{b} \in \mathbb{Q}^m$ is the right hand side vector. The continuous or linear programming (LP) relaxation of problem (1) is obtained by relaxing the integrality requirements on all non-continuous decision variables. Let X denote the feasible region of (1) and let $\text{conv}(X)$ denote the convex hull of X. We refer to a point $\mathbf{x} \in X$ as an *integer feasible solution*, or simply a *solution*, to problem (1).

Since the class of MIP optimization problems is $\mathcal{NP}$-hard, leading MIP solvers possess a vast array of techniques, which some developers even refer to as “tricks,” for solving MIPs to within some degree of optimality. While these techniques are important, they should not cloud our understanding of the fundamental strategies used in today's solvers. In a nutshell, MIPs are solved via a procedure called "branch-and-cut," which brings together two successful ideas: "branch-and-bound" and "cutting plane" methods.

Branch-and-bound is an implicit enumeration procedure that gradually divides the feasible region into smaller subregions. To implement this divide-and-conquer approach, a search tree is built where each node in the tree corresponds to a particular subregion of the feasible region. The goal is to avoid exhaustive enumeration of the entire solution space by pruning nodes that do not contain or will not lead to an optimal solution. Nodes are pruned for two reasons. First, a node can be pruned by infeasibility if the subregion at that node is empty. Second, a node can be pruned by bound if, after calculating a lower bound on the value of the best possible solution in the subregion at each node, we find that this lower bound exceeds the objective value of a solution we have already found. In either case, we do not need to explore this subregion any further.

Cutting plane methods attempt to strengthen the continuous relaxation of (1) by adding valid inequalities to it in hopes that the resulting formulation more accurately represents conv(X), at least in the region of an optimal solution to (1). A linear inequality $\pi^T\mathbf{x} \geq \pi_0$ is called a *valid inequality* for a set S if it is satisfied by all points in S. By definition, if a valid inequality is added to the LP relaxation for a feasible region S, the LP will still be a relaxation of S. Valid inequalities can be added to the original formulation (1), prior to solving the LP relaxation, by deriving information about the problem from its structure. Alternatively, they can be added after solving the LP relaxation by determining that an integer decision variable takes a fraction value in the LP relaxation. It is in this context that they were called cutting planes (or "cuts") since they cut off the optimal solution to the current LP relaxation, but not any feasible integer solution. Within the MIP community there is agreement that there is no shortage of cuts that can be added to a MIP. Meanwhile, identifying which cuts will bring about the fastest possible search is still an open problem.

When merged together, these two techniques complement one another to form the branch-and-cut procedure, a generic, yet powerful framework for solving MIPs. The algorithm is designed to yield a provably optimal solution by producing one or more solutions whose objective function value is equal to the best known lower bound. At the same time, it can also be modified to give solutions that are provably optimal within any specified tolerance, e.g., 5%, and therefore can be used as an approximation algorithm with a faster running time.

1.3.2 Primal Heuristics for General Purpose MIP Solvers

A primal heuristic is an algorithm that attempts to find a feasible solution to an optimization problem, often with complete disregard for a dual bound or dual information (hence, the qualifier "primal.") Given that a general MIP is $\mathcal{NP}$-hard, such heuristics have the unenviable task of trying to find a good feasible (or sometimes just any feasible) solution to any optimization or feasibility problem that can be formulated as a MIP. Why are primal heuristics so important? A good feasible incumbent solution can help prune sub-optimal branches of the search tree as early as possible and, therefore, reduce the memory and CPU time required to solve the problem. The importance of a good primal solution is reflected in the solution process of the best MIP solvers. Indeed, Figure 4, which depicts a flow chart of the solution process used in SCIP, widely regarded as the best non-commercial general purpose MIP solver in the world, contains four separate calls to "Primal Heuristics" scattered throughout the search procedure. The purpose of this subsection is to familiarize the reader with the state of current MIP solvers and to convey a simple message: Despite the abundance of primal heuristics available for general purpose MIPs, they are typically unable to find good solutions in a reasonable amount of time to the maritime inventory routing problems, when cast as MIPs, studied in this thesis.

State of the art MIP solvers typically possess a rich arsenal of more than 20 different primal heuristics that are dynamically applied before the root node solve, in the root node, and periodically in the nodes of the branch and bound tree. Primal heuristics can be categorized along several dimensions. One dimension is to differentiate strategies based on their goal, function, or purpose: "Start" (or "Starting") heuristics, as they are called in [17],

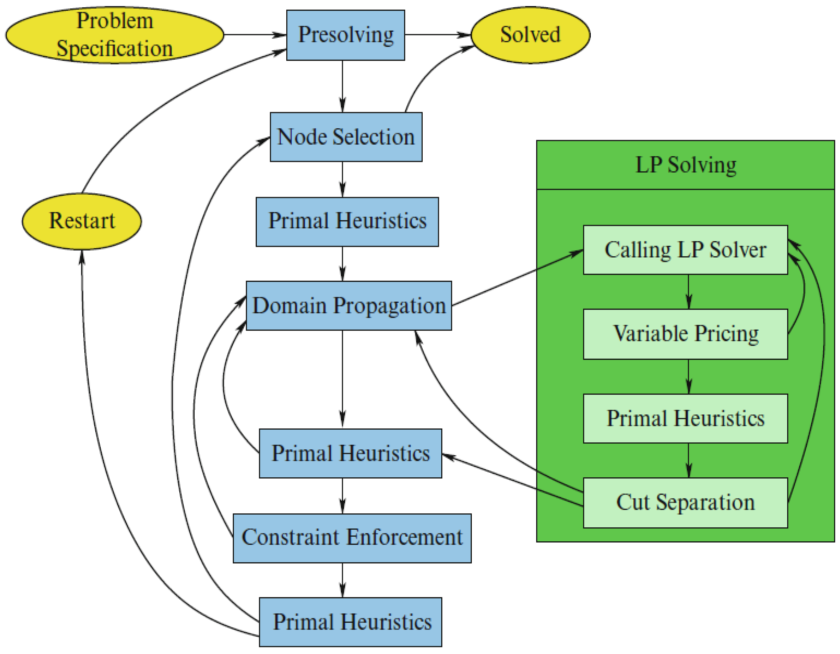


Figure 4: Flow chart of the solution process in SCIP. Source: Achterberg and Lodi [2].

attempt to find an integer feasible solution from scratch; Improvement heuristics seek to improve on a current incumbent solution; Repair heuristics try to repair a partial or slightly infeasible solution.

A second dimension distinguishes between the techniques used to obtain an integer feasible solution. The four most prominent techniques are rounding, diving, pivoting, and "MIPing." Rounding methods round every integer variable which takes a fractional value in an LP feasible vector to an integer value. Every integer variable that is already integral in this LP feasible vector stays unchanged. Diving methods attempt to explore one or more root-leaf paths of the search tree by iteratively bounding or fixing variables of a fractional LP solution to promising values and then re-solving the LP. Pivoting methods work on the LP tableau and use the mechanism of pivoting to move from one LP feasible solution to the next while reducing integer infeasibility. MIPing, a term coined in [39], is a relatively new trend and means "translating into a MIP model." The basic idea is to fix a subset of integer decision variables at an integer value and solve a smaller MIP, called a subMIP, to optimality or up to a given node limit or time limit. Just over a decade ago, such an idea was impractical given the existing technology. Today, solvers are capable of generating good solution to small MIPs in a time comparable to what is would take to solve the LP relaxation. In general, MIPing requires more CPU time than rounding, diving, and pivoting, but also allows a larger neighborhood to be explored and, therefore, may lead to better solutions.

A third dimension is the input given to the heuristic. Let $\mathbf{x}^{\text{mip}}$ be the incumbent solution, i.e., the best feasible solution found thus far, and $\mathbf{x}^{\text{nodeLP}}$ be the solution of the LP relaxation at a node in the search tree. Most Start heuristics take an LP solution $\mathbf{x}^{\text{nodeLP}}$ as input, although some do not even require an LP to be solved, and attempt to transform it into an integer feasible solution. Improvement heuristics often take $\mathbf{x}^{\text{mip}}$ as input along with $\mathbf{x}^{\text{nodeLP}}$ and/or other feasible solutions, and perform some sort of local search to find a better solution. Repair heuristics can take a partial or slightly infeasible solution and try to steer the solution towards a feasible one.

It is beyond the scope of this introduction to describe the dozens of primal heuristics that have been developed for general MIP solvers. Instead, we try to categorize some of the main approaches in Table 2 (Acronyms used: DINS = Distance Induced Neighborhood Search; RINS = Relaxation Induced Neighborhood Search; RENS = Relaxation Enforced Neighborhood Search). For an extensive discussion on this topic, see the Master's theses of Berthold [17] and Christophel [32].

Table 2: Common primal heuristics found in general purpose MIP solvers

Primal Heuristic	**Purpose**	**Technique**	**Input**
Wedelin's heuristic [14]	Starting	Lagrangian	Nothing
Diving	Starting	Diving	$\mathbf{x}^{\text{nodeLP}}$
Feasibility Pump [38]	Starting	Rounding	$\mathbf{x}^{\text{nodeLP}}$
Pivoting [10, 11]	Starting	Pivoting	$\mathbf{x}^{\text{nodeLP}}$
RENS [17]	Starting	MIPing	$\mathbf{x}^{\text{nodeLP}}$
Rounding	Starting	Rounding	$\mathbf{x}^{\text{nodeLP}}$
Objective Feasibility Pump [1]	Starting	Rounding	$\mathbf{x}^{\text{nodeLP}}$
Crossover [17]	Improvement	MIPing	2 or more int. feasible solns
DINS [43]	Improvement	MIPing	$\mathbf{x}^{\text{mip}}$, $\mathbf{x}^{\text{nodeLP}}$
Local Branching [37]	Improvement	MIPing	$\mathbf{x}^{\text{mip}}$, $\mathbf{x}^{\text{nodeLP}}$
RINS [33]	Improvement	MIPing	$\mathbf{x}^{\text{mip}}$, $\mathbf{x}^{\text{nodeLP}}$
Solution Polishing [87]	Improvement	MIPing	2 or more int. feasible solns

CHAPTER II

MIRPLIB: A MARITIME INVENTORY ROUTING SURVEY AND INSTANCE LIBRARY

2.1 Introduction

In this chapter, we study a particular maritime transportation planning problem known as the Maritime Inventory Routing Problem (MIRP), which plays an integral role in global bulk shipping. In recent years, there have been several appeals to create a set of benchmark instances on maritime transportation for the research community. Andersson et al. [8] urge authors, in collaboration with industrial partners, to make their data available along with a full and rich description of the model so that other can reproduce it. Similarly, Christiansen and Fagerholt [26] write "... there are still not any published sets of benchmark problems for maritime transportation problems, while there are numerous in land-based transport." The primary goal of this chapter is to help fill this void by introducing a set (or "library") of benchmark instances for a particular class of single product MIRPs. By doing so, we hope to help maritime inventory routing gain maturity as an important and interesting class of planning problems and to spur the development of better mathematical models and more advanced algorithms. We call this libary MIRPLib in the spirit of other libraries in the OR community such as TSPLib, MIPLib, ORLib, and MineLib, which have been used for the traveling salesman problem, mixed-integer linear programming (MIP), OR, and open-pit mining, respectively.

In order to create the first publicly available library of MIRP instances, we scoped the problem to be interesting and accessible. Christiansen et al. [29] define a MIRP as "a planning problem where an actor has the responsibility for both the inventory management at one or both ends of the maritime transportation legs, and for the ships' routing and scheduling." While this definition is reasonable and concise, it leaves ample room for interpretation and variation. Indeed, just as with the IRP, there is no single well-defined MIRP, but instead many variants that address particular aspects of a specific industrial application [8]. To scope this work, we have opted to study a core model in which inventory levels at all loading and discharging ports must stay within prespecified bounds during every time period throughout the entire planning horizon. We refer to this class of problems as *MIRPs with inventory tracking at every port*. We believe that this class of problems is a suitable starting point for a library since it most closely resembles the traditional concept of VMR in which a central entity is tasked with maintaining inventory levels at all suppliers and customers, while simultaneously managing the distribution of the inventory.

Our emphasis on a core model is in line with what Christiansen and Fagerholt [26] describe as "a need to direct the research on maritime transportation towards more basic research." By focusing on a core model that lies at the intersection of many of the models seen in the literature, we believe that researchers can compare their algorithms in a meaningful way without having to understand a detailed variant of this base model. Meanwhile, this does not discount the importance of rich models. We hope researchers can use this library as a template before making their data available to the community.

The single product MIRP that we study as our core model is best described in terms of its main components: ports and vessels. Each port is classified as a loading port where product is produced and loaded onto vessels or as a discharging port where product is consumed, typically after being discharged from vessels or from an alternative source (e.g., a pipeline). Product can be stored in inventory at both types of ports. Each port has: exactly one classification type, "loading" or "discharging"; a variable inventory capacity; a fixed number of berths limiting the number of vessels that can simultaneously load or discharge in a given time period; lower and upper bounds on the amount of product that can be loaded or discharged in a period; and deterministic, but possibly non-constant, per-period bounds on the rate of production or consumption. If the bounds in a single period coincide, then the rate is fixed. Each discharging port has a deterministic, but possibly non-constant, per-period unit price for the quantity discharged. Port operations, such as time to berth and time to set up equipment for loading or discharging, are not explicitly modeled.

To transport the product, the planners control or charter a fleet of heterogeneous vessels. Each vessel belongs to a particular vessel class and has a fixed capacity, a cruising speed, and a travel cost per km. Vessels make voyages between ports by picking up inventory at one or more ports and delivering inventory to one or more ports. Vessels may partially load and discharge so that two or more ports of the same type (loading or discharging) may be visited in succession. In general, a vessel will fully discharge before loading at another port, but this is not required in the model. A berth is only occupied by a vessel when loading or discharging. Thus, there can be more vessels at a port than there are berths. Using the nomenclature of Andersson et al. [8], this core MIRP model can be classified as a deterministic, finite-horizon, split-pickup and split-delivery problem. The solution of this planning problem specifies routes, i.e., the sequence and times of ports visited, for each vessel as well as the quantity of product loaded or discharged in each time period by each vessel.

To reiterate, in this chapter, we focus exclusively on tactical MIRPs in which inventory levels at all loading and discharging ports must stay within prespecified bounds during every time period throughout the entire planning horizon. MIRPs with explicit time windows constraints are not considered. Of course, there are other interesting types of MIRPs that have been studied. For example, in the liquefied natural gas (LNG) industry, it is sometimes the case (see, e.g., Section 4 in [7]) that a producer is responsible for ensuring that inventory bounds are strictly enforced at a liquefaction plant while fulfilling a set of long-term customer contracts. This problem is a MIRP. However, since inventory level constraints are not stated in every time period for the customers, we do not include it here. Similarly, Christiansen [23] discusses a real-world problem faced by a company that trades ammonia with internal and external customers (ports). Although inventory bounds are explicitly stated for each internal port in every time period, load and discharge amounts with external ports are based on negotiations and are, therefore, specified with time windows. Once again, since inventory constraints are not stated in every time period for all customers, this problem is an extension of the core model presented here. Note that, as long as time window constraints are not included, we allow for lower and upper inventory bounds at some ports to be ignored as this is equivalent to setting these bounds to $-\infty$ and $+\infty$, respectively.

The outline of this chapter is as follows. In Section 2, we review the literature on MIRPs with inventory tracking. In Section 3, we present an arc-flow MIP formulation of a core MIRP along with extensions to handle other features frequently encountered in the literature. In Section 4, we discuss how to use the library. Finally, we provide best known results for the instances currently in the library in Section 5.

2.2 Literature Review

In this section, we present a review of the papers and solution methods for MIRPs with inventory tracking at all ports. A survey of applications, problems, and algorithms in maritime routing and scheduling can be found in Christiansen et al. [29]. Table 6 attempts to categorize the papers discussed below.

As mentioned in the introduction, Christiansen [23] studies a single product MIRP from the ammonia industry. Although the problem that she considers does not satisfy the strict definition of our core MIRP model, it is important to mention this work as it is one of the most cited papers in maritime routing and scheduling, and its model provides the basis of several other models seen in subsequent papers. A company owns both production and consumption facilities and must route a fleet of vessels so that inventory bounds are never breached. Continuous-time arc- and path-flow models are formulated and a branch-and-price algorithm is developed.

Ronen [86] addresses a multi-product MIRP faced by producers of liquid bulk products in which each product must be stored and shipped in separate compartments of a vessel. Vessels are chartered to make voyages that visit a single loading port and a single discharging port, while possibly carrying multiple products. A simple heuristic is suggested. Persson and Göthe-Lundgren [76] also consider a multi-product MIRP for an oil refinery company in Sweden. They formulate both arc- and path-flow models on a time-space network. To solve the problem, they suggest a heuristic that uses column generation and variable fixing within a partial branch-and-bound search.

Extending the model of Christiansen [23], Al-Kayyal and Hwang [5] study an arc-flow model in which multiple liquid bulk products are shipped by a fleet of heterogeneous vessels, each of which has a dedicated compartment for a subset of the products; each compartment is dedicated to the same product throughout the planning horizon. Computational experiments reveal that the time required to solve their model directly using a commercial solver increases exponentially in the number of vessels and time periods considered. Li et al. [66] study a MIP model similar to that of Christiansen [23] and Al-Kayyal and Hwang, but at an operational level with finer granularity. For example, they ensure that inventory bounds are satisfied at every moment in time, rather than just at the beginning and end of each loading and discharging event (or time period in our core model). Like Christiansen [23], their model involves internal and external ports. However, unlike Christiansen, external sites act solely as external suppliers of raw materials that no other site produces, inventory levels at external sites are ignored, and no time windows are specified. Whereas Al-Kayyal and Hwang and Li et al. assume that compartments are dedicated for certain products, i.e., it is not permissible to assign a product to a compartment that has been used previously by other products, Siswanto et al. [91] relax this assumption and study a MIRP with undedicated compartments. Multiple heuristics are applied to generate feasible solutions.

Several case studies also appear in the literature. Dauzère-Pérès et al. [34] describe a case study in VMR involving a Norwegian supplier of calcium carbonate slurry, a product used in paper manufacturing. The supplier is responsible for routing a fleet of heterogeneous vessels and for maintaining sufficient inventory levels of up to sixteen products at ten tank farms in Northern Europe. Ensuring inventory remains within bounds at both the supply point and the tank farms is imperative; moreover, these bounds are rather tight. While vessel voyages are relatively simple (each vessel travels from the supply point to a single tank farm before fully discharging), the decision of which vessel to use and how much of each product to load on the chosen vessel is challenging. A memetic algorithm, a population-based approach that combines local search heuristics with crossover operators, is used to generate solutions within the decision support tool. Note that even though inventory bounds are not enforced at the supplier, we consider this problem to be a MIRP with inventory tracking since the lower and upper inventory bounds at the supply port can be considered $-\infty$ and $+\infty$, respectively. Christiansen et al. [27] present a MIRP encountered by a major cement producer involving bulk ships with multiple compartments that transport multiple non-mixable cement products. While a mathematical programming formulation is not provided, a construction heuristic embedded in a genetic algorithmic framework is used as a solution method. Andersson [6] studies a maritime inventory routing subcomponent of the supply chain of Södra Cell AB, one of the largest producers of pulp in the world. The problem is complicated by the availability of several modes of transportation for distributing the pulp. Along with trucks, trains, and barges, a fleet of long-term time-chartered vessels are used, but additional vessels can also be chartered on the spot market. A path-flow model is formulated and solved using a branch-and-price methodology. Bilgen and Ozkarahan [19] present a MIP model for a multi-product bulk grain blending and shipping problem faced by a company that manages a wheat supply chain. The salient characteristic of their model that differentiates it from other models listed here is the ability to blend multiple products to meet customer demand requirements. Although their routing decisions may be slighly complicated by the presence of split pickups, they include a simplifying assumption that all voyages begun in a period (a month) end in the same period.

Another stream of research emerged from a class of tactical planning problems within vacuum gas oil (VGO) transportation. This class of single product MIRPs is a tramp shipping application involving voyage chartered vessels or spot charters, i.e., vessels that are chartered for a single voyage from a loading region to a discharging region. Furman et al. [41] present a rich arc-flow MIP model embedded in a decision support tool used to aid decision-makers in the routing and inventory management of VGO at ExxonMobil. This case study describes many real-world constraints and techniques for modeling vessels with a complicated cost structure. Driven by a need to generate good solutions quickly to models similar to those described in [41], Song and Furman [92] apply a large neighborhood search to an arc-flow model that extends the ideas introduced in Savelsbergh and Song [89]. In particular, after an initial solution is generated, a local search procedure, akin to a 2-opt procedure, is applied in which the decision variables associated with all but two vessels are fixed and an exact optimization algorithm is called to locally optimize the decisions for these two vessels. This procedure is applied for up to $\binom{|\mathcal{V}|}{2}$ iterations, where $|\mathcal{V}|$ is the number of vessels and vessel pairs are chosen randomly in each iteration. We refer to this type of algorithmic approach as MIP-based local search as a small MIP model is solved during each local search phase. Working off of a simpler problem than the one considered in [41] and [92], Engineer et al. [36] formulate a path-flow model and apply a branch-cut-and-price approach for solving the problem. Three types of valid inequalities are suggested that generalize valid inequalities presented in previous work. Hewitt et al. [57] also attempt to generate good solutions quickly for the instances considered in [36] with branch-and-price guided search (BPGS) [56], a technique that systematically searches restricted neighborhoods of a MIP using information from an extended formulation in the master problem. They consider a much richer set of local search neighborhoods than previously studied and show that, after parallelizing their code on four processors, BPGS is quite effective at finding high-quality solution in 30 minutes for the MIRP instances considered.

Agra et al. [3] study a general MIRP and propose two discrete-time formulations to solve it: an arc-flow formulation and a fixed-charge network flow formulation. They show that the latter formulation is much tighter than the arc-flow formulation. In addition to their alternative formulation, their main contributions are several types of valid inequalities, which can further strengthen the models, and the use of priority branching to accelerate the solution process. All valid inequalities are generated before the branch-and-cut algorithm is launched. Shen et al. [90] devise a Lagrangian relaxation approach to solve a crude oil transportation problem involving chartered vessels and pipelines that are used to transport product from a central supplier to a number of customers. Papageorgiou et al. [75] consider instances involving company owned and long-term chartered vessels and apply a two-stage decomposition algorithm, similar in spirit to Benders decomposition for MIP, in which vessels are first routed between regions and then intra-regional loading/discharging and routing decisions are made. While the first-stage master problem provides useful bounds, an effective construction heuristic to generate good solutions quickly is presented along with extensions to the local search neighborhoods presented in Hewitt et al. [57]. Whereas [75] attempts to find good primal and dual solutions to tactical planning problems of up to 60 periods (days), Papageorgiou et al. [74] focus exclusively on finding good primal solutions to planning problems of up to 360 periods in a short amount of time (i.e., minutes).

Inventory tracking models have also been studied for MIRPs arising in the distribution of LNG, sometimes referred to as LNG-IRPs. Grønhaug and Christiansen [48] are the first to study an LNG-IRP and introduce arc- and path-flow models that also include features idiosyncratic to LNG shipping, e.g., boil-off and cargo tanks. Because larger instances of the arc- and path-flow models are difficult to solve with a commercial solver, Grønhaug et al. [49] introduce a branch-and-price method in which the master problem handles the inventory management and the port capacity constraints, while the subproblems generate the ship route columns. Different accelerating strategies are implemented. Andersson et al. [7] present a path-flow formulation of a planning problem faced by a vertically integrated LNG company. The company is responsible for the inventory management at all liquefaction plants and regasification terminals in addition to the transportation between these plants; no computational experiments are performed. Fodstad et al. [40] study arguably the richest version of an LNG-IRP discussed in the literature as it involves contract management and spot market trading. To solve their LNG-IRP model, Fodstad et al. [40] solve a MIP directly, while Uggen et al. [104] present a fix-and-relax heuristic. Goel et al. [47] study an arc-flow model of a similar LNG-IRP with a single-pickup and single-delivery assumption. They present a construction heuristic and adapt the local search procedure of Song and Furman [92] to generate solutions to instances with 365 time periods. Their main algorithmic contribution is to show how vessel pairs should be chosen to improve solution quality and reduce total solution time.

Table 6 summarizes those papers in the literature whose focus is on modeling or solving a MIRP with inventory tracking at all ports. The headings of Table 6 are: **Aff** refers to the primary affiliation of the authors (BIT = Blekinge Institute of Technology, Sweden; CUPB = Chinese University of Petroleum - Beijing, China; DEU = Dokuz Eylul University, Turkey; Molde = Molde University College, Norway; NUS = National University of Singapore; UMSL = University of Missouri-St Louis, USA; UNSW = University of New South Wales at

The Australian Defense Force Academy); **App** refers to the primary application motivating the paper or the computational instances ("L. bulk" means "liquid bulk"); **Model** notes whether an arc- or path-flow model or both are discussed or used; **Method** refers to the solution method applied (ADP = Approximate Dynamic Programming; BC = Branch-and-Cut; BP = Branch-and-Price; BPC = Branch-Price-and-Cut; BPGS = Branch-and-Price Guided Search; CG = Column Generation; Lagrangian = Lagrangian relaxation); **B** stands for "Branching" and denotes whether any special branching procedures are discussed; **C** stands for "Cuts" or "Constraints" and denotes whether any valid inequalities were derived to improve the model. The final six columns roughly describe the size of the largest instance in each paper, where the size is measured coarsely in terms of the number $|\mathcal{V}|$ of vessels, the number $|\mathcal{J}|$ of ports, the number of loading and discharging ports ($|\mathcal{J}^P|$ and $|\mathcal{J}^C|$, respectively), the number $|\mathcal{T}|$ of time periods, and the number $|\mathcal{K}|$ of products. Note that the largest value reported for each parameter is shown, but there may not be an instance corresponding to the values shown. For example, Grønhaug et al. [49] consider instances with up to 75 time periods, but they do not have an instance with the parameters shown in the table. Not all papers include a computational study.

Table 6 reveals that over two-thirds of the papers on MIRPs with inventory tracking at all ports are affiliated with Norwegian research at the Norwegian University of Science and Technology (NTNU), the Norwegian Foundation for Scientific and Industrial Research (SINTEF), and/or the Norwegian Marine Technology Research Institute (MARINTEK); or with ExxonMobil Research and Company (denoted XOM) and Georgia Tech (denoted GT). We also see that the motivating applications are rather diverse as are the solution techniques to solve the models. It appears that arc-flow models are far more common than path-flow models, which we attribute (at least partially) to the fact, in some computational studies, the number of routes per vessel that are ultimately generated is extremely large since there are no time windows, as there are in cargo routing, to drastically reduce the number of routes to consider. Most instances considered in the literature involve fewer than 5 vessels and no more than 10 ports. (Recall that Table 6 shows only the maximum number of vessels, ports, and time periods considered over all instances.) Roughly one-third of the applications involve multiple products. Other commonalities pertaining to branching strategies (column **B**) and valid inequalities (column **C**) are discussed in Section 3, after we present our core model.

2.3 Our Core Maritime Inventory Routing Problem

In this section, we provide an arc-flow MIP formulation of our core MIRP. This model, or a close variant, has been considered in [3, 41, 48, 75, 92]. The model is a discrete-time model involving an underlying time-space network. Its primary purpose is to identify optimal routing decisions for a fleet of heterogeneous vessels and optimal loading and discharging amounts by each vessel in each time period to ensure that inventory remains within prespecified bounds.

It is worth contrasting this model with other prominent models that appear in the literature. In their introduction to maritime inventory routing, Christiansen and Fagerholt [25] describe a continuous-time arc-flow model for a single product MIRP, which they call a "basic ship inventory routing problem," with constant production and consumption rates. Their model also takes place on a network, but it is quite different from the one presented below. Although arc-flow formulations are more prevalent, path-flow models are also studied (see column **Model** of Table 6 for references). Grønhaug et al. [49] prefer a path-flow model, which they attempt to solve via branch-and-price. As Grønhaug et al. [49] point out: "The advantages of path-based models are that ... intricate and nonlinear constraints and costs can easily be incorporated when generating the paths."

2.3.1 An Arc-Flow Mixed-Integer Linear Programming Model

Some of the sets, parameters, and decision variables introduced below are not used in the standard formulation, but will be used later, so we include them here for ease of reference. Sets are represented using capital letters in a calligraphic font, such as $\mathcal{T}$ and $\mathcal{V}$. Where possible, parameters are denoted with capital letters in italic font or with Greek characters; however, some deviations are made to express constraints more easily, e.g., inventory balance constraints. Decision variables are always lower case.

Indices and sets

$t \in \mathcal{T}$	set of time periods with $T = \lvert\mathcal{T}\rvert$
$v \in \mathcal{V}$	set of vessels
$vc \in \mathcal{VC}$	set of vessel classes
$j \in \mathcal{J}^P$ $(r \in \mathcal{R}^P)$	set of production, a.k.a. loading, ports (regions)
$j \in \mathcal{J}^C$ $(r \in \mathcal{R}^C)$	set of consumption, a.k.a. discharging, ports (regions)
$j \in \mathcal{J}$ $(r \in \mathcal{R})$	set of all ports (regions): $\mathcal{J} = \mathcal{J}^P \cup \mathcal{J}^C$ and $\mathcal{R} = \mathcal{R}^P \cup \mathcal{R}^C$
$n \in \mathcal{N}$	set of regular nodes or port-time pairs: $\mathcal{N} = \{n = (j,t) : j \in \mathcal{J}, t \in \mathcal{T}\}$
$n \in \mathcal{N}_{s,t}$	set of all nodes (including a source node n_s and a sink node n_t)
$a \in \mathcal{A}$	set of all arcs
$a \in \mathcal{A}^v(\mathcal{A}^{vc})$	set of arcs associated with vessel $v \in \mathcal{V}$ (vessel class $vc \in \mathcal{VC}$)
$a \in \mathcal{FS}_n^v(\mathcal{FS}_n^{vc})$	forward star associated with node $n = (j,t) \in \mathcal{N}_{s,t}$ and vessel $v \in \mathcal{V}$ (vessel class $vc \in \mathcal{VC}$)
$a \in \mathcal{RS}_n^v(\mathcal{RS}_n^{vc})$	reverse star associated with node $n = (j,t) \in \mathcal{N}_{s,t}$ and vessel $v \in \mathcal{V}$ (vessel class $vc \in \mathcal{VC}$)

Data

B_j (B_r)	number of berths (berth limit) at port $j \in \mathcal{J}$ (in region $r \in \mathcal{R}$)
C_a^v (C_a^{vc})	cost for vessel v (a vessel in vessel class vc) to traverse arc $a = ((j_1, t_1), (j_2, t_2)) \in \mathcal{A}^v(\mathcal{A}^{vc})$
$d_{j,t}$	number of units produced/consumed at port $j \in \mathcal{J}$ in period $t \in \mathcal{T}$
Δ_j (Δ_r)	an indicator parameter taking value +1 if $j \in \mathcal{J}^P$ $(r \in \mathcal{R}^P)$ and -1 if $j \in \mathcal{J}^C$ $(r \in \mathcal{R}^C)$
$F_{j,t}^{\min}$ $(F_{j,t}^{\max})$	minimum (maximum) amount of product that can be loaded or discharged at port j from a single vessel in a period
Q^v (Q^{vc})	capacity of vessel $v \in \mathcal{V}$ (capacity of a vessel in vessel class vc)
R_n	the unit sales revenue for product discharged at node $n = (j, t)$
$S_{j,t}^{\min}$ $(S_{j,t}^{\max})$	lower bound (capacity) at port $j \in \mathcal{J}$ in time period $t \in \mathcal{T}$
$s_{j,0}$	initial inventory at port $j \in \mathcal{J}$
s_0^v	initial inventory on vessel $v \in \mathcal{V}$

Decision Variables

$d_{j,t}$	(continuous) amount to produce/consume at port $j \in \mathcal{J}$ in period t
f_n^v	(continuous) amount loaded/discharged at port $j \in \mathcal{J}$ in period t from vessel $v \in \mathcal{V}$
$s_{j,t}$	(continuous) number of units of inventory at port $j \in \mathcal{J}$ available at the *end* of period t
s_t^v	(continuous) number of units of inventory on vessel $v \in \mathcal{V}$ available at the *end* of period t
x_a^v	(binary) takes value 1 if vessel $v \in \mathcal{V}$ uses arc a incident to node $n = (j, t) \in \mathcal{N}$
z_n^v	(binary) takes value 1 if vessel $v \in \mathcal{V}$ can load or discharge product at node $n = (j, t) \in \mathcal{N}$

Network

The core model takes place on an underlying time-space network first introduced in Song and Furman [92]. The network has a set $\mathcal{N}_{s,t}$ of nodes and a set $\mathcal{A}$ of directed arcs.

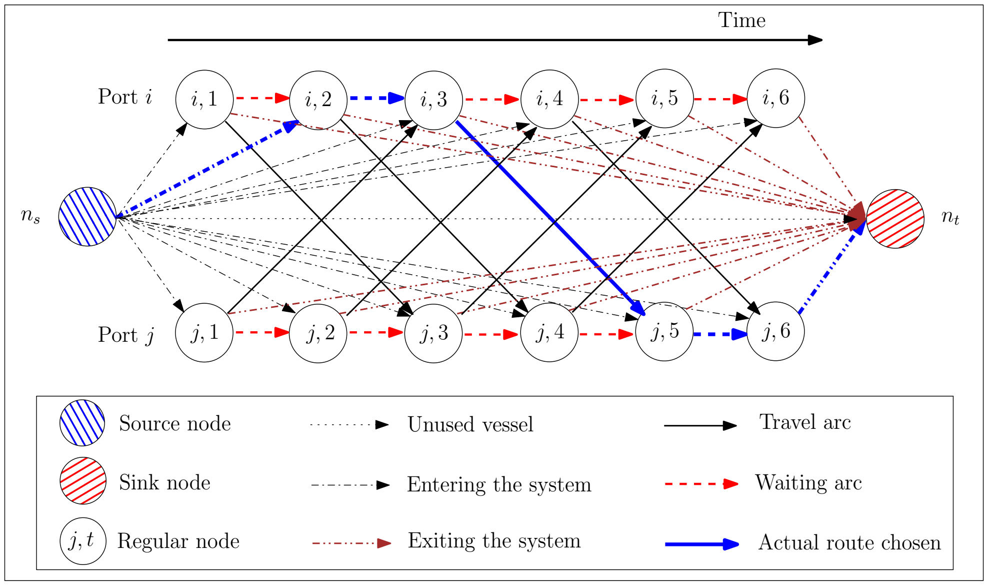


Figure 5: Example of the time-space network structure for a single vessel

The node set is shared by all vessels, while each vessel has its own arc set $\mathcal{A}^v$. The set $\mathcal{N}_{s,t}$ of nodes consists of "regular" nodes or port-time pairs, which represent a potential visit by one or more vessels to port $j \in \mathcal{J}$ in time period $t \in \mathcal{T}$, as well as a source node n_s and a sink node n_t.

Associated with each vessel v is a set $\mathcal{A}^v$ of arcs, which can be subdivided further as shown in Figure 5. An arc from the source to the sink node denotes that the vessel is not used in the solution. A source arc from the source node to a regular node represents the arrival of a vessel to its initial destination. A sink arc from a regular node to the sink node conveys that a vessel is no longer being used and has exited the system. A waiting arc from a port j in time period t to the same port in time period $t+1$ represents that a vessel stays at the same port in two consecutive time periods. Finally, a travel arc from a regular node $n_1 = (j_1, t_1)$ to a regular node $n_2 = (j_2, t_2)$ with $j_1 \neq j_2$ represents travel between two distinct ports. The set of incoming and outgoing arcs associated with vessel $v \in \mathcal{V}$ at node $n \in \mathcal{N}_{s,t}$ are denoted by $\mathcal{RS}_n^v$ (for reverse star) and $\mathcal{FS}_n^v$ (for forward star), respectively.

The network structure affords great flexibility in modeling and embeds a significant amount of data. First, note that the travel duration between two distinct ports on a travel arc is given by the length $(t_2 - t_1)$ of the arc and this duration may be time-dependent, e.g., it may take longer to travel from China to Europe during a particular season. Second, in some applications, all vessels may not be able to visit all ports because of physical restrictions at the port. Such vessel-port incompatibilities can easily be handled in this network by simply not including arcs in the respective sets. For example, if vessel v cannot visit port j, then the sets $\mathcal{FS}_n^v$ and $\mathcal{RS}_n^v$ are empty for all $n = (j,t)$ and $t \in \mathcal{T}$.

Core Model

$$\max \quad \sum_{n \in \mathcal{N}} \sum_{v \in \mathcal{V}} R_n f_n^v - \sum_{v \in \mathcal{V}} \sum_{a \in \mathcal{A}^v} C_a^v x_a^v \tag{2a}$$

$$\text{s.t.} \quad \sum_{a \in \mathcal{FS}_n^v} x_a^v - \sum_{a \in \mathcal{RS}_n^v} x_a^v = \begin{cases} +1 & \text{if } n = n_s \\ -1 & \text{if } n = n_t \\ 0 & \text{if } n \in \mathcal{N} \end{cases}, \quad \forall\, n \in \mathcal{N}_{s,t}, \forall\, v \in \mathcal{V} \tag{2b}$$

$$s_{j,t} = s_{j,t-1} + \Delta_j d_{j,t} - \sum_{v \in \mathcal{V}} \Delta_j f_n^v , \quad \forall\, n = (j,t) \in \mathcal{N} \tag{2c}$$

$$s_t^v = s_{t-1}^v + \sum_{\{n=(j,t) \in \mathcal{N}\}} \Delta_j f_n^v , \quad \forall\, t \in \mathcal{T}, \forall\, v \in \mathcal{V} \tag{2d}$$

$$\sum_{v \in \mathcal{V}} z_n^v \le B_j , \quad \forall\, n = (j,t) \in \mathcal{N} \tag{2e}$$

$$z_n^v \le \sum_{a \in \mathcal{RS}_n^v} x_a^v , \quad \forall\, n = (j,t) \in \mathcal{N}, \forall\, v \in \mathcal{V} \tag{2f}$$

$$F_{j,t}^{\min} z_{j,t}^v \le f_{j,t}^v \le F_{j,t}^{\max} z_{j,t}^v , \quad \forall\, n = (j,t) \in \mathcal{N}, \forall\, v \in \mathcal{V} \tag{2g}$$

$$D_{j,t}^{\min} \le d_{j,t} \le D_{j,t}^{\max} , \quad \forall\, n = (j,t) \in \mathcal{N} \tag{2h}$$

$$S_{j,t}^{\min} \le s_{j,t} \le S_{j,t}^{\max} , \quad \forall\, n = (j,t) \in \mathcal{N} \tag{2i}$$

$$0 \le s_t^v \le Q^v , \quad \forall\, v \in \mathcal{V}, \forall\, t \in \mathcal{T} \tag{2j}$$

$$x_a^v \in \{0,1\} , \quad \forall\, v \in \mathcal{V}, \forall\, a \in \mathcal{A}^v \tag{2k}$$

$$z_n^v \in \{0,1\} , \quad \forall\, n = (j,t) \in \mathcal{N}, \forall\, v \in \mathcal{V} . \tag{2l}$$

The objective function is stated in the form of a profit maximization where revenue is earned at the time product is delivered to a port. However, the objective function appears in many different forms. Some authors prefer to count revenue as being earned at the time product is consumed, e.g., [7,49] replace the terms $\sum_{n \in \mathcal{N}} \sum_{v \in \mathcal{V}} R_n f_n^v$ with $\sum_{n=(j,t) \in \mathcal{N}} R_{j,t} d_{j,t}$.

Other authors, e.g., [3, 5, 36, 104], prefer to omit the revenue component and simply minimize transportation costs and the loading/discharge costs, which include port operations, duties, etc. We assume that these costs are all captured in the parameter C_a^v. The costs incorporated in source and sink arcs can also vary, but we use the calculations provided in Table 3. Still other authors, e.g., [47, 75, 86], include penalty terms for violating inventory bounds at ports. Inventory costs are not included in the objective function because we assume that the shipper owns both the production and consumption sites.

Constraints (2b) require flow balance for every vessel, that is, if a vessel enters node $n \in \mathcal{N}$, it must also exit node $n \in \mathcal{N}$. Constraints (2c) are inventory balance constraints at loading and discharging ports, respectively. Constraints (2d) maintain inventory balance on each vessel. Constraints (2e) limit the number of vessels that can attempt to load/discharge at a port at a given time. Constraints (2f) ensure that a vessel does not attempt to load/discharge at a node unless the vessel is actually at that node. Constraints (2g) state that if a vessel attempts to load/discharge at node n, then the actual amount loaded/discharged is within predetermined port-specific bounds $[F_{j,t}^{\min}, F_{j,t}^{\max}]$. Constraints (2h) ensure that the amount produced or consumed in each period is within prespecified bounds. Constraints (2i) require ending inventory in each time period at each port to be within prespecified bounds.

In a number of models used for tactical or operational planning, the decision variables $d_{j,t}$, which denotes the production and consumption rates at each port over the planning horizon, are deterministic inputs to the model, rather than decision variables. In this case, we have that $D_{j,t}^{\min} = D_{j,t}^{\max}$. Since some strategic models used for vertically integrated supply chains may wish to determine production and consumption rates, e.g., [47, 48], we model $d_{j,t}$ as decision variables.

It is also worth noting some not-so-obvious features and constraints that are not stated (implicitly or explicitly) in the Core Model (2). First, although the notion of a region is not mentioned, in our library of instances, we assume that each port belongs to a prespecified region of the same type, i.e., loading or discharging. Second, it is assumed that if a vessel travels from port i to port j, the vessel will attempt to load/discharge at port j (and, therefore, incurs a port fee). This will always happen in an optimal solution because the data for the instances of interest all satisfy the triangle inequality, i.e., it is cheaper to travel from port a to port c than to travel from a to b and then b to c. Note that the port fee is paid only once. That is, if a vessel attempts to load at port j in period t, remains at port j in period $t + 1$ (but, perhaps, abandons the berth in this period), and then attempts to load again at port j in period $t + 2$, only one port fee is incurred. Third, in a single time period, it may be possible for a vessel to load or discharge more inventory than a port's capacity. For example, suppose a discharging port j consumes 25 units of product per period and has a constant capacity of 250 units. Then, 275 units could be discharged in a single period. This could occur if port j has 0 inventory at the end of period t, i.e., $s_{j,t} = 0$, and a vessel carrying at least 275 units of inventory arrives in period $t + 1$ and discharges 275 units, 25 of which satisfy demand in period $t + 1$ while the remaining 250 units are stored in inventory. This example also shows the limitations of a discrete-time formulation since inventory bounds are only required to be satisfied at the end of each period.

2.3.2 Common Side Constraints and Additional Model Features

2.3.2.1 "Travel at capacity" constraints

Many authors include constraints that require a vessel to travel at capacity from a loading region to a discharging region and empty from a discharging region to a loading region:

$$s_t^v \geq Q^v x_a^v \; , \qquad \forall\, v \in \mathcal{V}, \forall\, a = ((j_1,t),(j_2,t')) \in \mathcal{A}^v : j_1 \in \mathcal{J}^P, j_2 \in \mathcal{J}^C \cup \{n_t\} \tag{3a}$$

$$s_t^v \leq Q^v (1 - x_a^v) \; , \qquad \forall\, v \in \mathcal{V}, \forall\, a = ((j_1,t),(j_2,t')) \in \mathcal{A}^v : j_1 \in \mathcal{J}^C, j_2 \in \mathcal{J}^P \cup \{n_t\} \; . \tag{3b}$$

Although these "travel full" constraints (3a) are usually justified on the basis that vessel capacity is a scarce resource and therefore a vessel's capacity should always be fully utilized when making long voyages, there are applications in which it has been shown that such an assumption may not always be optimal (see, e.g., [40]). On the other hand, in virtually all MIRPs discussed in the literature, vessels fully discharge before reloading. This is in contrast to what occurs in liner shipping where vessels load and discharge containers regularly without ever fully discharging. Finally, note that constraints (3a) and (3b) require vessels to leave the system empty or full.

2.3.2.2 Differentiating among similar solutions

It may be useful to include several features into the Core Model (2) in order to give slight preference to some solutions over what would otherwise be considered almost identical solutions.

When a vessel visits a port, there may be multiple time periods in which it can load or discharge product. In reality, we prefer a vessel to load or discharge as few times as possible to minimize the duration and cost of port operations associated with that vessel. In addition, we prefer a vessel to load or discharge as soon as it arrives at a port, assuming the port has a berth available and enough inventory or capacity to do so. To accommodate these secondary goals without affecting the primary goals of managing inventory and routing vessels, we may choose to associate a negligible cost $t\epsilon_z$ with each binary decision variable $z^v_{j,t}$, where ϵ_z is a small nonnegative parameter representing the cost to load or discharge and t is the time period. If a nonzero ϵ_z parameter is specified, the objective function in the Core Model (2) should include the additional term

$$\sum_{v\in\mathcal{V}}\sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}} -(t\epsilon_z)z^v_{j,t}\ . \tag{4}$$

Note that by using the coefficient $-(t\epsilon_z)$ instead of $-\epsilon_z$, solutions in which a vessel attempts to load or discharge sooner rather than later are preferred.

Because the Core Model (2) is a finite-horizon model, a second useful modeling feature is to give a small value to vessels for “exiting the system” as soon as they are no longer needed. In terms of the Core Model (2), this means that we would like vessels to take a sink arc once it is no longer necessary or profitable for them to engage in other activities. Without such a feature, a solution in which vessel v discharges all of its product at port j in time period t and remains empty at that same port until the end of the planning horizon is valued as highly as a nearly identical solution in which vessel v fully discharges at port j in period t and then exits the system immediately by taking the sink arc from node (j,t) to the sink node. Indeed, we prefer the latter solution since vessel v will be available sooner for service at the start of the next planning horizon. To accomplish this, we introduce a reward ρ per unit time for a vessel that finishes early. This reward is built into the cost of a sink arc, which is discussed in Section 2.4.3, so that the Core Model (2) remains the same.

2.3.2.3 Soft inventory bounds and a simplified spot market representation

In certain strategic and tactical planning models, the inventory bound constraints (2i) at ports may be considered soft, i.e., they should ideally be satisfied, but are permitted to be violated with a penalty. There are several reasons why this “soft” interpretation may be beneficial or necessary from a modeling standpoint. First, the inventory bounds in constraints (2i) may be overly conservative in order to make the solution more robust by preventing ports from running out of inventory or exceeding capacity due to uncertainty in the data. In this case, it may be acceptable to penalize a small bound violation if it is impossible or unfavorable for a vessel to reach the port before the violation occurs. Second, soft constraints may help mitigate unwanted effects of the time discretization used in the planning model. For example, suppose that it takes a vessel 9.5 days to travel from port i to port j, but that a daily time discretization is used requiring the travel time to be modeled as 10 days. Then, while in reality it might be possible for the vessel to arrive a half-day early at port j just in time to prevent a stockout, a daily time discretization may necessitate that a partial stockout take place. Third, in some planning models, it may be interesting to experiment with different fleet compositions in which case the proper mix of vessels may not be available to meet all inventory requirements in every period.

To account for these possibilities, it is convenient to incorporate a simplified spot market representation into the model. Mathematically, let $\alpha_{j,t}$ be a nonnegative decision variable representing the amount of product that port j purchases from (when $j \in \mathcal{J}^C$) or sells to (when $j \in \mathcal{J}^P$) the spot market in time period t. Then, we can re-write the port inventory balance constraints (2c) as

$$s_{j,t} = s_{j,t-1} + \Delta_j \left(d_{j,t} - \sum_{v \in \mathcal{V}} f_n^v - \alpha_{j,t} \right) , \qquad \forall\ n = (j,t) \in \mathcal{N} . \tag{5}$$

Note that with the addition of a simplified spot market, there is no backlogging of inventory. Rather, inventory bounds at ports are satisfied at the end of each time period t. For a concrete example, Goel et al. [47] model a strategic LNG-IRP with inventory tracking at all ports using constraints (5). The objective function in the Core Model (2) should also be amended to penalize the use of the α decision variables.

If $\alpha_{j,t}$ variables are included in the model, then we may include other side constraints as well:

$$\alpha_{j,t} \leq \alpha_{j,t}^{\max} \qquad \forall\, j \in \mathcal{J}, \forall\, t \in \mathcal{T} \tag{6a}$$

$$\sum_{t \in \mathcal{T}} \alpha_{j,t} \leq \alpha_{j}^{\max} \qquad \forall\, j \in \mathcal{J} \ . \tag{6b}$$

Constraints (6a) bound the amount of violation that may occur in a single period by a constant $\alpha_{j,t}^{\max}$. Constraints (6b) limit the amount of cumulative violation that may occur at each port over the entire planning horizon by a constant $\alpha_{j}^{\max}$.

2.3.2.4 Draft limits

The draft of a vessel is the distance between the waterline and the bottom of the vessel and is a function of the load onboard. Draft limit constraints are sometimes necessary to ensure that larger vessels can only enter, reside in, or exit certain harbors if they are not fully loaded [27, 41, 92]. Such constraints may also affect the sequence of port visits made by a vessel. Assuming we can compute the draft associated with a certain inventory level on a vessel, we can write draft limit constraints as

$$s_{t-1}^{v} \leq \text{DRAFT}_{j}^{v,\text{in}} + (Q^{v} - \text{DRAFT}_{j}^{v,\text{in}})(1 - z_{j,t}^{v}) \ , \qquad \forall\, v \in \mathcal{V}, \forall\, j \in \mathcal{J}, \forall\, t \in \mathcal{T} \tag{7a}$$

$$s_{t}^{v} \leq \text{DRAFT}_{j}^{v,\text{out}} + (Q^{v} - \text{DRAFT}_{j}^{v,\text{out}})(1 - z_{j,t}^{v}) \ , \qquad \forall\, v \in \mathcal{V}, \forall\, j \in \mathcal{J}, \forall\, t \in \mathcal{T} \tag{7b}$$

where $\text{DRAFT}_{j}^{v,\text{in}}$ and $\text{DRAFT}_{j}^{v,\text{out}}$ denote the maximum permissible draft for vessel v when entering and exiting port j. These constraints are enforced both before (port inlet) and after (port outlet) loading has been completed and before discharge has begun. Draft limits often apply to port-vessel combinations.

2.3.2.5 Cruising speed as a decision variable

Depending on the age of a vessel, fuel costs typically constitute 15-20% of a vessel's total annual cost [96] and as much as 60% of a vessel's daily cost [85]. To date, virtually all maritime inventory routing research has assumed that vessels travel at a single speed, presumably because most models have been at the strategic or tactical level, not an operational one. Today, with a growing interest to reduce greenhouse gases and to better utilize an existing fleet of vessels, making cruising speed a decision variable in planning models has received attention.

When a vessel is designed, naval architects optimize the hull and power plant to a prescribed design speed [96]. This is the speed typically assumed in planning models. Nevertheless, vessels are capable of traveling at various speeds in order to meet deadlines and to satisfy customer service levels. A common approximation is to assume that fuel consumption is a quadratic or cubic function of a vessel's cruising speed. Consequently, Ronen [85] points out that reducing a vessel's cruising speed by 20% can reduce the daily fuel consumption by 50%.

One way of handling this added flexibility in the Core Model (2) is to include additional inter-port travel arcs into the network. Specifically, in addition to a design speed, vessels have lower and upper limits on the cruising speed that can be achieved. Thus, given a minimum and maximum speed for each vessel, we can compute upper and lower bounds ($\tau_{ij}^{\max}$ and $\tau_{ij}^{\min}$, respectively) on the time required to travel between two distinct ports i and j. Since time is discretized in the Core Model (2), for each time period in the time interval $[\tau_{ij}^{\min}, \tau_{ij}^{\max}]$, we can compute the optimal average cruising speed that minimizes fuel cost while allowing the vessel to arrive in the desired period; the fuel cost associated with this speed is used when computing the cost of this arc. With these additional arcs, the Core Model (2) can be solved "as-is" and an optimal solution will specify the arcs taken by each vessel, and, therefore, the average speed of the vessel between each pair of ports.

2.3.2.6 Multiple products

It is easy to extend the Core Model (2) to handle multiple products. For simplicity, here we discuss a particular setting in which we assume that three parameters are enforced at the aggregate level, rather than on each product: (i) bounds on amounts loaded or discharged at a port, (ii) inventory bounds at ports, and (iii) inventory bounds on vessels. In other words, constraints (2g), (2i), and (2j) still hold without being modified to apply to each specific product. See [5] and [27] for models when dedicated compartments are needed.

Let $\mathcal{K}$ denote the set of products. Let $D^{\min}_{j,t,k}$ and $D^{\max}_{j,t,k}$ denote lower and upper bounds on the amount of product k produced or consumed at port j in period t. Let $R_{j,t,k}$ be the unit sales revenue for product k discharged at port $j \in \mathcal{J}^C$ in period t. We retain all of the decision variables currently in the model, but include additional ones to keep track of product-specific decisions. Namely, define decision variables $d_{j,t,k}$, $f^v_{j,t,k}$, $s_{j,t,k}$, and $s^v_{t,k}$ to correspond to the original variables $d_{j,t}$, $f^v_{j,t}$, $s_{j,t}$, and s^v_t, but now they are specific to product k. Constraints (2c), (2d), and (2h) can be modified to

$$s_{j,t,k} = s_{j,t-1,k} + \Delta_j d_{j,t,k} - \sum_{v\in\mathcal{V}} \Delta_j f^v_{j,t,k} \ , \qquad \forall\ n=(j,t)\in\mathcal{N}, \forall\ k\in\mathcal{K} \tag{8a}$$

$$s^v_{t,k} = s^v_{t-1,k} + \sum_{\{n=(j,t)\in\mathcal{N}\}} \Delta_j f^v_{j,t,k} \ , \qquad \forall\ t\in\mathcal{T}, \forall\ v\in\mathcal{V}, \forall\ k\in\mathcal{K} \tag{8b}$$

$$D^{\min}_{j,t,k} \le d_{j,t,k} \le D^{\max}_{j,t,k} \ , \qquad \forall\ (j,t)\in\mathcal{N}, \forall\ k\in\mathcal{K}\ . \tag{8c}$$

Several constraints to link the variables are needed:

$$f^v_{j,t} = \sum_{k\in\mathcal{K}} f^v_{j,t,k} \ , \quad s_{j,t} = \sum_{k\in\mathcal{K}} s_{j,t,k} \ , \quad s^v_t = \sum_{k\in\mathcal{K}} s^v_{t,k} \ , \qquad \forall\ (j,t)\in\mathcal{N}, \forall\ v\in\mathcal{V}, \forall\ k\in\mathcal{K}\ . \tag{9}$$

Finally, the term $\sum_{n=(j,t)\in\mathcal{N}} \sum_{v\in\mathcal{V}} R_{j,t} f^v_{j,t}$ in the objective function of the Core Model (2) becomes

$$\sum_{n=(j,t)\in\mathcal{N}} \sum_{v\in\mathcal{V}} \sum_{k\in\mathcal{K}} R_{j,t,k} f^v_{j,t,k}\ .$$

2.3.3 Common Modeling Enhancements

Having introduced a core model, our next goal is to summarize two modeling enhancements that can be used to tighten it. These techniques have been used in some form or another in previous papers, but have not been described in a uniform manner or have not been identified as a common strategy. Our aim here is to unify our understanding of these enhancements.

2.3.3.1 Advanced branching techniques

Column **B** of Table 6 lists the authors that have used some form of advanced branching strategy to improve algorithmic efficiency. In its current form, the Core Model (2) contains two types of binary decision variables: x variables representing the flow of a particular vessel along a particular arc and z variables representing an attempt to load or discharge at a particular port and time by a particular vessel. After solving the LP relaxation at a node of the branch-and-cut tree, a MIP solver will branch on a binary decision variable that takes a fractional value in the LP relaxation. Unfortunately, branching on the existing binary decision variables in the model can have little impact due to the symmetry of many solutions. For example, if vessel v is not allowed to travel along an arc beginning in time period t due to a branching decision made by the solver, it may be able to delay starting its travel until the subsequent time period $t+1$. Thus, the solver may just shift a fractional value to different variables in time without ever improving the bound. The motivation behind advanced branching strategies is to overcome this ineffectual branching by branching on more significant decisions.

Priority branching relies on the fact that certain decisions are more influential than others. The problem is that some higher priority decisions may not be explicitly modeled in the existing formulation. Some examples of high-level decisions not modeled as decision variables in the Core Model (2) are: (i) the number of vessels that visit a particular port in a specific time interval [49,75]; (ii) the number of times a particular vessel visits a particular port [3]; and (iii) the number of voyages from a particular loading port/region to a particular discharging port/region in a specific time interval. It can be argued that determining the number of vessels that visit a particular port over the entire planning horizon is more important than knowing the precise times of the visits.

Advanced branching can usually be accomplished in at least two ways. We use item (i) stated above when illustrating these procedures. A first approach, which is straightforward to implement, is to include auxiliary decision variables in the Core Model (2) that represent the number of visits summed over all vessels to port j over the entire planning horizon. All of the leading commercial MIP solvers allow the user to assign a higher branching priority to these variables first so that if one of these variables takes a fractional value in the LP relaxation at a node in the branch-and-cut tree, the solver will choose to branch on one of these variables before all other decision variables taking a fractional value. Alternatively, instead of including additional integer variables into the model, a second approach is to implement a callback that effectively accomplishes the same task, but without increasing the number of decision variables in the model. In this approach, one writes a callback to check if the number of visits to a particular port is fractional. If so, a port is identified and two local cuts are written with respect to the original decision variables. This approach requires more effort from the user, but may yield additional efficiency.

2.3.3.2 Lot-sizing based constraints

The LP relaxation of the Core Model (2) can be weak and often results in many binary variables taking fractional values so as to incur only a fraction of the fixed costs. Using the language of maritime inventory routing, this means that only a fraction of a vessel may travel along an arc and/or only a fractional attempt to load/discharge is made at a port. One way of overcoming this deficiency is to include additional constraints involving the binary variables to ensure, for example, that ports are visited with a minimum and maximum frequency. This can be at least partially accomplished using lot-sizing relaxations based on the standard lot-sizing model which we briefly review here for sake of completeness. Table 6 column **C** lists the authors that have used some form of valid inequalities that can be derived from the standard lot-sizing model.

Consider the standard capacitated lot-sizing set (see, e.g., Pochet and Wolsey [77], whose notation we use here) in which one must decide in what periods to produce an item and how much to produce, given demand data d_t, initial inventory s_0, constant storage capacity

$s^{\max}$, and capacities C_t on production in period t of a finite planning horizon $\mathcal{T}$:

$$s_{t-1} + x_t = d_t + s_t \ , \qquad \forall\, t \in \mathcal{T} \tag{10a}$$

$$0 \le x_t \le C_t y_t \ , \qquad \forall\, t \in \mathcal{T} \tag{10b}$$

$$y_t \in \{0,1\} \ , \qquad \forall\, t \in \mathcal{T} \tag{10c}$$

$$0 \le s_t \le s^{\max} \ , \qquad \forall\, t \in \mathcal{T} \ . \tag{10d}$$

The decision variables are: s_t, the stock (inventory) in period t; x_t, the amount produced in period t; and y_t, a binary decision variable taking value 1 if production takes place in period t and 0 otherwise. For any time interval $[t_1, t_2]$, we can sum over constraints (10a) and apply inequalities (10b) and (10d) to obtain the lot-sizing relaxation

$$s_{t_1-1} + \sum_{u=t_1}^{t_2} C_u y_u \ge d_{[t_1,t_2]} \ , \qquad \forall\, 1 \le t_1 \le t_2 \le T \tag{11a}$$

$$y_t \in \{0,1\} \ , \qquad \forall\, t \in \mathcal{T} \tag{11b}$$

$$0 \le s_t \le s^{\max} \ , \qquad \forall\, t \in \mathcal{T} \ , \tag{11c}$$

where $d_{[t_1,t_2]} = \sum_{u=t_1}^{t_2} d_u$ is the demand in the time interval.

Before describing valid inequalities that can be applied directly to set (11), we give an example of how the capacitated lot-sizing set (10) naturally arises in the Core Model (2). Consider a discharging port $j \in \mathcal{J}^C$ and suppose that $d_{j,t} = D^{\min}_{j,t} = D^{\max}_{j,t}$, $F^{\min}_{j,t} = S^{\min}_{j,t} = 0$, $F^{\max}_{j,t} = F^{\max}_t$ (a constant), and $S^{\max}_{j,t} = S^{\max}$ (a constant) for all $t \in \mathcal{T}$. Let $f_{j,t} = \sum_{v \in \mathcal{V}} f^v_{j,t}$ be the total amount of product discharged in time period t, $z_{j,t} = \sum_{v \in \mathcal{V}} z^v_{j,t}$ be the number of vessels attempting to discharge in time period t, and assume that port j has exactly one berth so that $z_{j,t}$ is binary. Coupling these assumptions with constraints (2c) and (2g), and omitting the subscript j, we obtain the set

$$s_{t-1} + f_t = d_t + s_t, \qquad \forall\, t \in \mathcal{T}$$

$$0 \le f_t \le F^{\max}_t z_t, \qquad \forall\, t \in \mathcal{T}$$

$$z_t \in \{0,1\}, \qquad \forall\, t \in \mathcal{T}$$

$$0 \le s_t \le S^{\max}, \qquad \forall\, t \in \mathcal{T} \ ,$$

which is identical to the capacitated lot-sizing set (10). An analogous set can be derived for each loading port $j \in \mathcal{J}^P$ after making a change of variable commonly used in lot-sizing problems.

From the mixed-binary set (11), it is possible to generate several types of valid inequalities.

- Option 1: After ignoring the upper bound constraint $s_t \leq s^{\max}$ in set (11) and fixing t_1 and t_2, we obtain the continuous 0-1 knapsack set

$$\left\{ (s,y) \in \mathbb{R}^1_+ \times \{0,1\}^n : \sum_{j=1}^{n} a_j \leq b + s \right\} ,$$

  for which several families of strong valid inequalities are known (see, e.g., [77]) . To our knowledge, no attempt at applying valid inequalities for the continuous 0-1 knapsack set has been reported.

- Option 2: Here we describe the most typical valid inequalities that appear in the MIRP literature. Replacing s_t with its upper bound, we obtain the pure binary set

$$\sum_{u=t_1}^{t_2} C_u y_u \geq d_{[t_1,t_2]} - s^{\max}_{t_1-1} , \qquad \forall\, 1 \leq t_1 \leq t_2 \leq T$$

$$y_t \in \{0,1\}, \qquad \forall\, t \in \mathcal{T} ,$$

  where $s^{\max}_0 = s_0$ and $s^{\max}_t = s^{\max}$ for all $t \in \mathcal{T}$. Replacing the coefficients C_u with an upper bound $C^{\max}_{[t_1,t_2]}$ in each interval $[t_1, t_2]$ leads to the set of valid inequalities

$$\sum_{u=t_1}^{t_2} y_u \geq \left\lceil \frac{d_{[t_1,t_2]} - s^{\max}_{t_1-1}}{C^{\max}_{[t_1,t_2]}} \right\rceil, \qquad \forall\, 1 \leq t_1 \leq t_2 \leq T . \tag{12}$$

  In the context of the Core Model (2), these valid inequalities have a nice interpretation: They require a minimum number of attempts to load or discharge at a port during a set of time intervals.

  More generally, one can consider a variety of coefficients $C_{[t_1,t_2]} > 0$ and perform the Chvátal-Gomory procedure to obtain valid inqualities

$$\sum_{u=t_1}^{t_2} \left\lceil \frac{C_u}{C_{[t_1,t_2]}} \right\rceil y_u \geq \left\lceil \frac{d_{[t_1,t_2]} - s^{\max}_{t_1-1}}{C_{[t_1,t_2]}} \right\rceil , \qquad \forall\, 1 \leq t_1 \leq t_2 \leq T . \tag{13}$$

Replacing the coefficients C_u with $F_u^{\max}$, Agra et al. [3] use the different vessel capacities Q^v in place of the coefficients $C_{[t_1,t_2]}$ to generate valid inequalities for their problem prior to invoking the solver.

- Option 3: Whereas constraints (12) and (13) are stated solely in terms of binary variables and were derived from the set (11) by relaxing the continuous decision variables s_t to their upper bounds $s_t^{\max}$, it is also possible to derive a potentially stronger set of valid inequalities by applying another relaxation to the set (11). This approach does *not* relax each s_t variable to its upper bound $s_t^{\max}$, but instead keeps the continuous variables in the model and replaces the coefficients C_u by a constant upper bound C.

  Before describing this relaxation, we need to define the mixing set, the facets of which are useful for capacitated lot-sizing models.. Consider the *mixing set* defined by K inequalities:

$$X_K^{MIX} = \left\{(s, \mathbf{y}) \in \mathbb{R}_+ \times \mathbb{Z}^K : s + y_k \geq b_k, \forall k = 1, \ldots, K\right\} \ . \tag{14}$$

  It is well known that the K simple mixed-integer rounding (SMIR) inequalities

$$s + f_{b_k} y_k \geq f_{b_k} \lceil b_k \rceil \ , \forall k = 1, \ldots, K \ , \tag{15}$$

  where $f_{b_k} = b_k - \lfloor b_k \rfloor$ is the fractional part of b_k, are valid and facet-defining for X_K^{MIX}. However, they do not suffice to give the convex hull of X_K^{MIX} when $K > 1$. In this case, we need the mixing inequalities (see Proposition 8.4 and Theorem 8.5 of [77]).

  Returning to our derivation, consider a variant of set (11), known as the *constant* capacitated lot-sizing relaxation, in which each C_t is replaced by a constant C so that constraints (11a) become

$$s_{t_1-1} + \sum_{u=t_1}^{t_2} C y_u \geq d_{[t_1,t_2]} \ , \qquad \forall \ 1 \leq t_1 \leq t_2 \leq T \ .$$

  Letting $\bar{s}_t = \frac{s_t}{C}$, $\bar{d}_{[t_1,t_2]} = \frac{d_{[t_1,t_2]}}{C}$, and $y_{[t_1,t_2]} = \sum_{u=t_1}^{t_2} y_u$, the constant capacity version of set (11) can be rewritten as

$$\bar{s}_{t_1-1} + y_{[t_1,t_2]} \geq \bar{d}_{[t_1,t_2]} \ , \quad \forall\ 1 \leq t_1 \leq t_2 \leq T \tag{16a}$$

$$0 \leq y_{[t_1,t_2]} - y_{[t_1,t_2-1]} \leq 1 \ , \quad \forall\ t_1 \in \mathcal{T}, \forall\ t_2 = t_1+1,\ldots,T \tag{16b}$$

$$y_{[t,t]} \leq 1 \ , \quad \forall\ t \in \mathcal{T} \tag{16c}$$

$$y_{[t_1,t_2]} \in \mathbb{Z}_+ \ , \quad \forall\ 1 \leq t_1 \leq t_2 \leq T \tag{16d}$$

$$\bar{s}_t \geq 0 \ , \quad \forall\ t \in \mathcal{T} \ . \tag{16e}$$

This is an instance of a mixing set (14) in which additional side constraints appear, but are of the form $\mathbf{By} \leq \mathbf{d}$ with $\mathbf{B}$ the arc-node incidence matrix of a digraph (network) and where $\mathbf{d}$ is an integral vector. Thus, all nontrivial facets of the convex hull of solutions to the constant capacitated lot-sizing relaxation are the mixing inequalities (see p.280 of [77]). Engineer et al. [36] applied a subset of the SMIR inequalities, which they called "port capacity cuts," to a model similar to the Core Model (2) and found that these inequalities tightened the relaxation and improved the bound. To our knowledge, no attempt at applying the mixing inequalities has been made.

Finally, note that if the capacitated lot-sizing set (10) also includes constraints $C_t^{\min} y_t \leq x_t$ for all $t \in \mathcal{T}$, i.e., forcing a minimum amount to produce if production takes place, then applying the same arguments as above, analogous sets and valid inequalities can be derived. For example, analogous to constraints (12), one can bound the maximum number of attempts to load or discharge at a port in different time intervals. Another possible way to generate valid inequalities is to simultaneously consider lot-sizing relaxations involving not just one, but a subset of ports. This approach is used in Papageorgiou et al. [75].

2.4 Using MIRPLib

Instances and results for each instance are maintained at `http://mirplib.scl.gatech.edu/`. The current instances are inspired by characteristics of real-world MIRPs, but do not represent any particular real-world data set. Since our focus is on a core model that lies at the intersection of many real-world models, we believe this choice is justified. We hope to include additional instances and results as they become available.

As mentioned from the outset, our major goals with this library are: (1) to present benchmark instances for a particular class of MIRPs; (2) to provide the mixed-integer linear programming community with a set of optimization problem instances from the maritime transportation domain; (3) to provide a template for other researchers when specifying characteristics of MIRPs arising in other settings; (4) to accelerate the development of advanced algorithms. In this section, we describe the information provided in a data instance and, in so doing, address goals (1), (2), and (3). This information helps explain many of the major details of a maritime inventory routing problem.

Each instance is given in three formats: a "data only" format, LP format, and MPS format. LP and MPS file formats are standard in the MIP computational community as they provide a common format for reading LP and MIP models. Although LP and MPS formats are useful for comparing solver performance, they are somewhat limiting as they impose a specific MIP model on the user, a model which may not be ideal for generating good solutions or bounds. Since other models and techniques may be superior, we also provide instance data in a "data only" format. As shown in Figure 6, each data set consists of four data objects: metadata, port data, vessel class data, and vessel data. When an instance is stated in "data only" format, five files are provided corresponding to each of the aforementioned data objects, but with port data further broken down into loading port data and discharging port data. A discussion of each data object is provided below.

Several notes are needed. The basic unit in which inventory and capacities are measured is kilotons. The basic unit of the objective function coefficients is US \$1000. Note that all data on the MIRPLib website starts indexing from 0, not 1. Consequently, a 60-period instance with 10 ports means that $\mathcal{T} = \{0, \ldots, 59\}$ and $\mathcal{J} = \{0, \ldots, 9\}$. The data type, e.g., `int`, `double`, `string`, etc., associated with each entry in Figure 6 is given on the website.

2.4.1 Metadata

Metadata includes high-level information for an instance. Most of the parameters are self-explanatory, but we describe each for the sake of completeness. In all instances, $\eta = 24$

Metadata	
Number of periods	$\|\mathcal{T}\|$
Number of loading regions	$\|\mathcal{R}^P\|$
Number of discharging regions	$\|\mathcal{R}^C\|$
Number of loading ports	$\{\|\mathcal{J}_r\|\}_{r\in\mathcal{R}^P}$
Number of discharging ports	$\{\|\mathcal{J}_r\|\}_{r\in\mathcal{R}^C}$
Number of vessel classes	$\|\mathcal{VC}\|$
Number of vessels per class	$\{\|\mathcal{V}^{vc}\|\}_{vc}$
Hours per period	η
Spot market price per unit	P^{spot}
Spot market discount factor	γ^{spot}
Attempt cost	ϵ_z
Reward for finishing early	ρ
Constant for single period $\alpha_{j,t}^{\max}$	κ_α^1
Constant for cumulative $\alpha_j^{\max}$	$\kappa_\alpha^{\text{sum}}$

Port data	
Index	j
Type	`type`
Region index	r
x-coordinate	x_j
y-coordinate	y_j
Port fee	π_j
No. of berths	B_j
Max amount	$F_j^{\max}$
Min amount	$F_j^{\min}$
Capacity	$S_j^{\max}$
Initial inventory	$s_{j,0}$
Rate	$\{d_{j,t}\}_t$
Revenue	$\{R_{j,t}\}_t$

Vessel class data	
Index	vc
Capacity	Q^{vc}
Design speed	σ_{kn}^{vc}
Travel cost per km	C_{km}^{vc}
Discount traveling empty	γ^{vc}

Vessel data	
Index	v
Vessel class index	vc
Initial inventory	s_0^v
Initial port	j
First time available	τ_0^v

Figure 6: Data objects

hours per period.

The number of periods $|\mathcal{T}|$ refers to the maximum number of periods for which the instance is defined. That is, a data file may state that $|\mathcal{T}| = 360$ implying that data, e.g., production and consumption rates, are specified for at most 360 periods. We only use the first 45 or 60 periods worth of data to solve a 45- or 60-period problem. However, users may be interested in solving longer horizon problems.

As previously mentioned, ports belong to regions. Thus, in addition to specifying the number of loading regions $|\mathcal{R}^P|$ and discharging regions $|\mathcal{R}^C|$, we also provide the number of ports in each region. In particular, if the number of loading regions $|\mathcal{R}^P|$ is 2, then the data $\{|\mathcal{J}_r|\}_{r\in\mathcal{R}^P}$ following "Number of loading ports" is a list of two positive integers denoting the number of loading ports in each of the two loading regions. The same is true for "Number of discharging ports." The number of vessel classes and vessels per vessel class is listed next. For example, if there are 3 vessel classes, then the number of vessels per vessel class will be a sequence of 3 positive integers, e.g., "2 3 4", implying that there are 2 vessels in the first vessel class, 3 in the second, and 4 in the third.

A spot market price per unit and discount factor are also included. The purpose of the spot market discount factor γ^{spot} is to delay the use of the spot market until the last possible time period in which it is needed. Specifically, if $\alpha_{j,t}$ decision variables are included in the model (see Constraints (5)), then we also include the following additional penalty term $-\sum_j \sum_t P_t \alpha_{j,t}$ into the model, where $P_t = (\gamma^{\text{spot}})^t P^{\text{spot}}$ for all $t \in \mathcal{T}$. Notice that if the spot market discount factor $\gamma^{\text{spot}} \in (0, 1)$ and $P^{\text{spot}} > 0$, then $\{P_t\}_{t \in \mathcal{T}}$ is a decreasing sequence, meaning that it is always cheaper to purchase from the spot market as late as possible. A similar idea was used in Goel et al. [47].

As discussed in Section 2.3.2.2, negligible positive parameters ϵ_z and ρ are included in some instances to give vessels an incentive to load or discharge as few times as possible and to exit the system as soon as it is no longer necessary or profitable for them to engage in service. The precise use of ϵ_z is shown in equation (4). Table 3 shows how ρ is incorporated into the sink arc cost.

The final parameters provided in the metadata are nonnegative parameters κ_α^1 and $\kappa_\alpha^{\text{sum}}$ and are associated with the right hand side values of Constraints (6a) and (6b), discussed in Section 2.3.2.3, when a simplified spot market representation is used. If the decision variables $\alpha_{j,t}$ should be included in the model, then they should include a variable upper bound, i.e., Constraints (6a) should be used with $\alpha_{j,t}^{\max} = \kappa_\alpha^1 d_{j,t}$ for all $j \in \mathcal{J}$ and $t \in \mathcal{T}$. If Constraints (6b) are included in the model, then $\alpha_j^{\max} = \kappa_\alpha^{\text{sum}} d_{j,0}$ for all $j \in \mathcal{J}$. We use $d_{j,0}$ since, for the current instances, it is approximately equal to the average of the $d_{j,t}$ variables over most planning horizons starting at time 0. For example, $\kappa_\alpha^1 = 0.5$ means that, in any period, the amount of product bought from or sold to the spot market is at most one-half of the amount produced/consumed in that period, and $\kappa_\alpha^{\text{sum}} = 1$ implies that the cumulative amount of product bought from or sold to the spot market over the entire planning horizon may not exceed (roughly) the average amount produced/consumed in any one period.

We now describe how to use the constants κ_α^1 and $\kappa_\alpha^{\text{sum}}$. First, if either parameter is not listed, assume it is 0. Second, they work in tandem. The rules, expressed in pseudocode, for using these parameters are: If $\kappa_\alpha^1 \leq 0$, then the decision variables $\alpha_{j,t}$ should not be included in the model; else if $\kappa_\alpha^1 > 0$ and $\kappa_\alpha^{\text{sum}} \leq 0$, then the decision variables $\alpha_{j,t}$ should be included in the model, but Constraints (6b) should not be included; else (i.e., $\kappa_\alpha^1 > 0$ and $\kappa_\alpha^{\text{sum}} > 0$), then the $\alpha_{j,t}$ variables with Constraints (6a) and (6b) should be included in the model.

2.4.2 Port data

Each port is defined by the following information: Each port has an integer index $j \in \{0, \ldots, |\mathcal{J}| - 1\}$ and a type 'Loading' or 'Discharging'. Since ports belong to regions, each port is assigned a region index (an integer $r \in \{0, \ldots, |\mathcal{R}|-1\}$). Each region is classified as a loading region or a discharging region, but not both. All ports within a region have the same classification as the region.

Each port j is given x and y coordinates on a two-dimensional plane. The distance δ_{ij} between two distinct ports i and j is the Euclidean distance between the two ports calculated using the x and y coordinates provided. The travel time between two distinct ports depends on the vessel class and is discussed below.

Each port has a port fee $\pi_j \in \{10, \ldots, 100\}$, which is incurred every time a port is visited, not every time an attempt to load or discharge is made. For example, if a vessel arrives at port j in time period 1, attempts to load in time period 2, waits outside the port in period 3 and then departs for another port, the port fee is incurred just once. On the other hand, if a vessel arrives at port j in period 1, then visits port k in the same region, and then returns to port j before departing from the region, then three port fees are incurred: $\pi_j + \pi_k + \pi_j$. See the travel cost calculation in Table 3. Whereas discharging ports always have a berth limit of $B_j = 1$, loading ports may have multiple berths. In reality, discharging ports may also have multiple berths, but having fewer berths typically results in more challenging instances.

Rather than varying the port inventory capacity $S_{j,t}^{\max}$ and the minimum and maximum loading/discharging amounts per period, $F_j^{\min}$ and $F_j^{\max}$, these values are fixed to a single value throughout the planning horizon. Values for these parameters were chosen so that $F_j^{\min}$ was roughly 50 units at discharging ports, but sometimes higher at loading ports, and $F_j^{\max}$ was almost always greater than one-half of the largest vessel class's capacity. Note that it is possible for $F_j^{\max}$ to be larger than $S_j^{\max}$. At a discharging port with zero inventory in the beginning of a period, this might allow larger vessels to discharge the amount consumed in that period plus the amount $S_j^{\max}$ needed to bring the inventory up to capacity in the end of the period. The minimum inventory level $S_j^{\min}$ at all ports is zero.

Initial inventory levels at each port are given and were selected in connection with the starting position of vessels. Although the production and consumption rates may be constant in some instances, a list $\{d_{j,t}\}_{t\in\mathcal{T}}$ is specified for each port j. Likewise, a list of revenues $\{R_{j,t}\}_{t\in\mathcal{T}}$ is given for each discharging port $j \in \mathcal{J}^C$. For instances with two loading regions, production and consumption rates were chosen so that solutions in which the same subset of discharging regions is served by a single loading region is avoided. In other words, we tried to avoid instances in which the problem could be decomposed with the same vessels always returning to the same loading region.

The port capacity-to-rate ratio $S_j^{\max}/\bar{d}_j$, where $\bar{d}_j$ is the average rate at port j, is one of the factors that determines how tightly constrained an instance is since smaller ratios require a port to be visited more frequently. Production and consumptions rates and capacities at individual ports were generated so that the capacity-to-rate ratios of ports within a region are typically distinct integer values. This makes it less likely to encounter optimal solutions in which two ports are repeatedly visited in a periodic manner. For example, suppose ports i and j belong to the same region, are close to one another, have low inventory capacities relative to that of most vessel classes, and have identical capacity-to-rate ratios, respectively. Then it seems reasonable to expect solutions in which these vessels are visited by a single vessel during each visit to the region. On the other hand, if the capacity-to-rate ratios are distinct, then it seems less likely that these two ports will always be visited by the same vessel during a visit to the region.

2.4.3 Vessel class data

Each vessel belongs to a particular vessel class $vc \in \mathcal{VC}$, which has a fixed capacity Q^{vc}, a design cruising speed σ_{kn}^{vc} in knots, a travel cost C_{km}^{vc} per km, and a discount parameter γ^{vc} for traveling empty. The meaning of each parameter is given below. As previously mentioned, we use the term *vessel class* to refer to vessels with the same aforementioned parameters. Thus, two Panamax vessels may be in different vessel classes if their parameters are different.

The travel time τ_{ij}^{vc} between two distinct ports i and j using a vessel in vessel class vc is calculated as $\tau_{ij}^{vc} = \lceil \eta\delta_{ij}/\sigma_{km/h}^{vc} \rceil$, where δ_{ij} is the Euclidean distance between ports i and j and $\sigma_{km/h}^{vc} = (1.852)\sigma_{kn}^{vc}$ is the design speed in kilometers per hour of a vessel in vessel class vc.

The formulas for calculating arc costs for each vessel class are shown in Table 3. Stopford [96] partitions the cost of operating a vessel into five components: operating costs, e.g., day-to-day crew costs and daily vessel maintenance costs; periodic maintenance costs when a vessel is dry-docked for major repairs; voyage costs, e.g., fuel costs, port fees, and canal dues; capital costs; and cargo-handling cost, e.g., the cost of loading, stowing, and discharging cargo. Here we consider a much simpler cost structure, which does not include period maintenance costs, capital costs, canal dues, or cargo-handling costs. Instead, we assume that the parameter C_{km}^{vc} captures the fuel cost and operating costs per kilometer associated with a nearly full vessel. In addition, we assume that a port fee π_j is incurred if port j is visited, independent of whether or not an attempt to load or discharge at that port is made. Thus, as shown in Table 3, the cost of traveling from a loading port j_1 to a discharging port j_2 is the total fuel and operating costs $C_{km}^{vc}\delta_{j_1,j_2}$ over the entire voyage plus the port fee π_{j_2} at the destination port j_2. Although a vessel may not be near capacity when traveling between two loading or discharging ports, the same calculation is used. On the other hand, since vessels almost always travel empty from a discharging port to a loading port, we assume that a vessel will save fuel on such a voyage; hence, we discount the total fuel and operating costs $C_{km}^{vc}\delta_{j_1,j_2}$ by the factor $(1-\gamma^{vc})$. For example, if $\gamma^{vc} = 0.2$, then the travel component of the arc cost is discounted by 20%.

A vessel cruising at an average speed of 15 knots travels approximately 667 km per day. Assuming an average operating cost of US \$50,000 per day, this implies that the travel cost per km is roughly US \$75. Using these figures as a starting point, we created other vessel classes.

Table 3: Arc cost calculations for each vessel class

Arc type	**Example arc** a	**Cost** C_a^{vc}
Source	$a = (n_s, (j,t))$	π_j
Sink	$a = ((j,t), n_t)$	$-(\lvert\mathcal{T}\rvert - t)\rho$
Waiting	$a = ((j,t),(j,t+1))$	0
Inter-port	$a = ((j_1,t_1),(j_2,t_2))$	$\begin{array}{ll} C_{km}^{vc}\delta_{j_1,j_2}(1-\gamma^{vc}) + \pi_{j_2} & \text{if } j_1 \in \mathcal{J}^C, j_2 \in \mathcal{J}^P \\ C_{km}^{vc}\delta_{j_1,j_2} + \pi_{j_2} & \text{otherwise} \end{array}$

2.4.4 Vessel data

As mentioned above, each vessel $v \in \mathcal{V}$ belongs to a particular vessel class $vc \in \mathcal{VC}$. In addition, a vessel has an initial inventory on board s_0^v. Since these instances involve company owned or long-term time-chartered vessels, the starting port j and the first time τ_0^v the vessel is available to attempt to load or discharge is also specified. Note that $\tau_0^v > 0$ means the vessel is en route to its starting port at the outset of the planning problem. Vessels originating in loading regions initially have zero inventory, while those beginning in discharging regions start full (at capacity).

Voyage chartered vessels are not considered in the current set of instances. However, these instances would not specify the starting port and time available. Instead, we might place bounds on the number of vessels that can be chartered in a given time interval.

2.5 Current Instances and Best Known Results

There are two sets of instances. The first set of instances involves 60-period planning horizons and are studied in Chapter 3 and [75]. The second set, treated in Chapter 4 and [74], involves 360-period planning horizons with a single loading port and multiple discharging ports, each of which belongs to a distinct region. For the results provided, it is assumed that a vessel will only travel from a loading region to a discharging region and vice versa; a vessel will never visit two regions of the same type in succession. Thus, for the second set of instances, vessels only make out-and-back trips from the loading port to a discharging port. All instances are solved as minimization problems, i.e., we minimize the negative of the objective function in the Core Model (2), so dual bounds refer to lower bounds. In all instances, it is assumed that vessels must travel at capacity from a loading port to a discharging port and empty from a discharging port to a loading port; that is, constraints (3a) and (3b) are enforced. Moreover, all instances include soft inventory constraints (6) and constraints (5) in lieu of constraints (2i). Whereas in the first set of instances, the amount of cumulative violation $\alpha_j^{\max}$ allowed is small making feasibility an issue, in the second set, the amount of cumulative violation allowed is infinite making it simple to find feasible solutions. In the second set, revenues $R_{j,t}$ are zero making the goal to minimize travel costs and spot market usage.

Our convention for naming instances is based on the number of loading and discharging regions, the number of ports, the number of vessel classes, and the number of vessels. This convention is best understood with an example. Consider an instance named `LR2_12_DR3_123_VC4_V14c`. `LR2` means that there are two loading regions. `12` means that there is one port in the first loading region and two ports in the second loading region. `DR3` means that there are three discharging regions. `123` means that there is one port in the first discharging region, two in the second, and three in the third. `VC4` means that there are four vessel classes. `V14` means that there are a total of 14 vessels (with at least one vessel belonging to each vessel class). Finally, if a letter is included at the end, this is to distinguish this instance from other instances.

Tables 4 and 5 show the current best known objective function value and bound for each instance. The objective function values in Table 4 were computed using the Zoom algorithm described in [75], while those in Table 5 were computed running Gurobi 5.0 for 24 hours with emphasis on feasibililty and warmstarting the solution procedure with the best solution found using the approach described in [74]. The best bounds, which are currently rather weak and, therefore, open to improvement, were computed running Gurobi 5.0 for 24 hours with default emphasis using the Core Model (2). The solutions corresponding to the best known objective function value are available on the website.

It is worth mentioning two experiments that we conducted on the 60-period instances in Table 4 in order to give potential users a barometer of current techniques. All computations were carried out on a Linux machine with kernel 2.6.18 running on a 64-bit x86 processor equipped with two Intel Xeon E5520 chips, which run at 2.27 GHz, and 48GB of RAM. In the first experiment, we loaded the MPS file for each instance in instance set 1 into Gurobi 5.0 with default emphasis and let the solver work for 24 hours. Gurobi could not find a feasible solution to any instances in this time limit. In a second experiment, we modified the model to allow an unlimited amount of spot market to be purchased (as in instance set 2), but with a high penalty if the cumulative amount of spot market purchased violated the bounds set in the data. This time, Gurobi found truly feasible solutions to five of the instances, while the others had more units from the spot market purchased than is permitted. The purpose of reporting these experiments is to highlight the fact that using the Core Model (2) to generate solutions and bounds to large instances may not be ideal. Indeed, we hope that better models and solution methods are developed in the future.

2.6 Conclusions

This chapter has introduced a core model for maritime inventory routing problems with tracking at every port. A detailed survey of related research is presented and summarized based on common attributes. Several modeling features and extensions are outlined along with a unifying discussion of key structural properties that can be exploited. Finally, we provide the first publicly available set of instances for maritime inventory routing, which we hope will help grow interest in this line of research and be used by many others in the future.

Table 4: Best known results for instances solved with a 60-period planning horizon

Instance	objval	bound
LR1_1_DR1_3_VC1_V7a	-16675	-17847
LR1_1_DR1_4_VC3_V11a	-13257	-15020
LR1_1_DR1_4_VC3_V12a	-11040	-12832
LR1_1_DR1_4_VC3_V12b	-10053	-11287
LR1_1_DR1_4_VC3_V8a	-5191	-6691
LR1_1_DR1_4_VC3_V9a	-7552	-9383
LR1_2_DR1_3_VC2_V6a	-13532	-15841
LR1_2_DR1_3_VC3_V8a	-14652	-17379
LR2_11_DR2_22_VC3_V6a	-12655	-14198
LR2_11_DR2_33_VC4_V11a	-15387	-19565
LR2_11_DR2_33_VC5_V12a	-22730	-25988
LR2_22_DR2_22_VC3_V10a	-32627	-35873
LR2_22_DR3_333_VC4_V14a	-26873	-33503
LR2_22_DR3_333_VC4_V17a	-27000	-33909

Table 5: Best known results for instances solved with a 360-period planning horizon

Instance	objval	bound
LR1_1_DR2_11_VC1_V6a	112022	106901
LR1_1_DR2_11_VC2_V6a	283358	267059
LR1_1_DR2_11_VC3_V7a	184462	120214
LR1_1_DR2_11_VC3_V8a	198409	138735
LR1_1_DR2_11_VC4_V8a	156058	137408
LR1_1_DR2_11_VC5_V8a	216043	120923
LR1_1_DR3_111_VC3_V10b	313870	186956
LR1_1_DR3_111_VC3_V13b	355680	234713
LR1_1_DR3_111_VC3_V16a	498431	267470
LR1_1_DR4_1111_VC3_V15a	274751	248524
LR1_1_DR4_1111_VC3_V15b	371514	277477
LR1_1_DR4_1111_VC5_V17a	268175	247529
LR1_1_DR4_1111_VC5_V17b	357119	249664
LR1_1_DR5_11111_VC5_V25a	402595	362894
LR1_1_DR5_11111_VC5_V25b	526535	376106
LR1_1_DR8_11111111_VC5_V38a	954190	575369
LR1_1_DR8_11111111_VC5_V40a	725759	623686
LR1_1_DR8_11111111_VC5_V40b	895517	629485
LR1_1_DR12_111111111111_VC5_V70a	1139065	982586
LR1_1_DR12_111111111111_VC5_V70b	1384856	980050

Table 6: Summary of relevant MIRP papers

Author	**Aff**	**App**	**Model(s)**	**Method(s)**	**B**	**C**	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{J}\rvert$	$\lvert\mathcal{J}^P\rvert$	$\lvert\mathcal{J}^C\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{K}\rvert$
Agra et al. [3]	NTNU	L. bulk	Arc	BC	✓	✓	5	6	-	-	60	1
Al-Khayyal and Hwang [5]	GT	L. bulk	Arc	Default solver		✓	4	4	-	-	10	3
Andersson [6]	NTNU	Pulp	Path	BP			3+	27	4	23	147	30
Andersson et al. [7]	NTNU	LNG	Path	Model only			-	-	-	-	-	-
Bilgen and Ozkarahan [19]	DEU	Wheat	Path	Default solver			-	6	4	2	3	8
Christiansen [23]	NTNU	Ammonia	Arc, Path	BP			5	16	-	-	36	1
Christiansen and Fagerholt [25]	NTNU	General	Arc	Model only			-	-	-	-	-	-
Christiansen et al. [27]	NTNU	Cement	-	Genetic alg			5	61	12	49	28	11
Christiansen et al. [28]	NTNU	General	Arc, Path	Default solver			-	-	-	-	-	-
Dauzère-Pérès et al. [34]	Molde	Slurry	Arc	Memetic alg			17	11	1	10	84	16
Engineer et al. [36]	GT	VGO	Path	BPC	✓	✓	6	10	6	4	60	1
Fodstad et al. [40]	SINTEF	LNG	Arc	Default solver			8	7	4	6	181	1
Furman et al. [41]	XOM	VGO	Arc	Default solver			-	-	-	-	-	1
Goel et al. [47]	XOM	LNG	Arc	MIP-based LS			69	11	1	10	365	1
Grønhaug and Christiansen [48]	NTNU	LNG	Arc, Path	Default solver			5	6	3	3	60	1
Grønhaug et al. [49]	NTNU	LNG	Path	BP	✓	✓	5	6	3	3	75	1
Hewitt et al. [57]	GT	VGO	Arc	BPGS			6	10	6	4	60	1
Li et al. [66]	NUS	L. bulk	Arc	Default solver			5	8	-	-	80	2
Papageorgiou et al. [75]	GT	L. bulk	Arc	Benders-like	✓	✓	17	13	4	9	60	1
Papageorgiou et al. [74]	GT	L. bulk	Arc	ADP			70	13	1	12	360	1
Persson & Göthe-Lundgren [76]	BIT	Bitumen	Arc, Path	CG; Heuristic	✓	✓	3	18	15	3	42	4
Ronen [86]	UMSL	L. bulk	Arc	Default solver			-	7	2	5	30	5
Shen et al. [90]	CUPB	Crude	Path	Lagrangian			-	11	1	10	12	1
Siswanto et al. [91]	UNSW	L. bulk	Arc	Heuristic			3	4	-	-	15	2
Song and Furman [92]	XOM	VGO	Arc	MIP-based LS	✓	✓	6	8	4	4	60	1
Uggen et al. [104]	SINTEF	LNG	Arc	Fix-and-relax			8	10	-	-	181	1

CHAPTER III

A TWO-STAGE DECOMPOSITION ALGORITHM FOR SINGLE PRODUCT MARITIME INVENTORY ROUTING

3.1 Introduction

In this chapter, we present a two-stage decomposition algorithm for the single product maritime inventory routing problem (MIRP) defined in Chapter 2. The problem involves routing vessels, each belonging to a particular vessel class, between loading and discharging ports, each belonging to a particular region. We call our algorithm "Zoom" because it iteratively solves a MIRP by zooming out and then zooming in on the problem. Specifically, in the "zoomed out" phase, we solve a first-stage master problem in which aggregate information about regions and vessel classes is used to route vessels between regions, while only implicitly considering inventory and capacity requirements, berth limits, and other side constraints. In the "zoomed in" phase, we solve a series of second-stage subproblems, one for each region, in which individual vessels are routed through each region and loading and discharge quantities are determined. Our algorithm bears a close resemblance to Benders decomposition for mixed-integer linear optimization except that our second-stage problems are mixed-integer linear programs, not pure linear programs. Not only is our solution approach different from previous methods discussed in the maritime transportation literature, but computational experience shows that our approach is promising.

By far the most common decomposition approach used for MIRPs is column generation (or branch-and-price) [36, 48, 76]. This fact is not surprising for three reasons. First, given the success of column generation in solving traditional vehicle routing problems, of which inventory routing is a more complicated extension, it is natural to apply similar techniques to MIRPs. Second, virtually all attempts to solve a MIRP using column generation have involved a relatively small number of vessels (at most five or six) which means that the number of pricing problems to solve at each iteration is relatively small. Third, in some settings, complex cost structures associated with vessel voyage costs are easier to compute when considering entire vessel voyages (i.e., using a path-based perspective) rather than when considering the individual legs of the full voyage (i.e., using an arc-based perspective). In most column generation approaches, a restricted master problem selects an optimal set of vessel *voyages* (routes and load/discharge quantities along the routes) from a subset of voyages that satisfy all inventory and routing constraints. In this sense, the master problem attempts to handle all routing and inventory decisions simultaneously. In the pricing subproblem, dual information from the master problem is used to check if there are any other voyages that should be considered in the master problem. In spite of its popularity, Engineer et al. [36] noted that while computational studies have shown that path-based extended formulations for the VRP yield small root node gaps, on the order of 5 to 15 percent, path-based extended formulations for the MIRP can yield very large gaps, often upwards of 100 percent.

While our decomposition algorithm also iteratively solves a master and sequence of subproblems, the philosophy of our decomposition differs in several key ways from existing methods in the literature. First, our master problem is a "route-only" model, meaning that it only attempts to route vessels from region to region while only implicitly considering other details like inventory balance and berth limit constraints. Second, our master problem does not explicitly model the flow (route) of each individual vessel, but instead uses aggregation to route vessels by vessel class. Third, our subproblems decompose by region, not by vessel. In particular, given a solution to the master problem, which specifies the entrance and exit times of vessels in each vessel class within each region, our regional subproblem attempts to handle all regional constraints explicitly and determines voyages (routes and load/discharge times and quantities) for each vessel.

Our reason for decomposing in this fashion was motivated by the fact that, in our setting and many others like it, inter-regional travel times and costs are at least an order of magnitude greater than those within a region. Thus, it seems natural to place a higher priority on inter-regional routing decisions, while subordinating less costly, but nonetheless important, regional decisions to a second stage. Another motivating factor is that, for economic reasons, vessels almost always travel with inventory at capacity from a loading region to a discharging region, and empty from a discharging region to a loading region. Consequently, once we know when a vessel in a particular vessel class is scheduled to arrive and depart from a region, we know how many units must be loaded/discharged and how many periods we have to accomplish this task.

Our algorithm has a similar spirit to classic Benders decomposition for mixed-integer linear programming (MIP), but with at least one major difference. In traditional Benders decomposition, a MIP is decomposed into two stages: the first stage involves solving a pure integer program (excluding an auxiliary continuous decision variable used to represent the objective function value of the second stage subproblem), while the second stage consists solely of continuous decision variables and can therefore be solved using linear programming. In our approach, the first stage is a pure integer program, but the second stage is a MIP.

The outline of this chapter is as follows. In Section 3.2, we describe a multi-start construction heuristic for finding a good feasible solution faster than a commercial solver. In Section 3.3, we describe an exact two-stage decomposition algorithm. Several enhancements are outlined in Section 3.4 followed by a sketch of our complete solution procedure in Section 3.5. Finally, we present computational experiments with our approach in Section 3.6.

Assumptions: Throughout this chapter, we assume that the parameters $F_{j,t}^{\min}$, $F_{j,t}^{\max}$, $S_{j,t}^{\min}$, and $S_{j,t}^{\max}$ do not change over time, i.e., the subscript t can be dropped, and that consumption and production rates are given as data, i.e., $D_{j,t}^{\min} = D_{j,t}^{\max} = d_{j,t}$ for all $j \in \mathcal{J}$ and $t \in \mathcal{T}$.

3.2 A Two-Stage Construction Heuristic

In this section, we describe a two-stage multi-start heuristic for constructing an initial feasible solution to the Core Model (2). This heuristic is effective at generating solutions much faster than a commercial solver and has many similarities with our second approach, but is easier to describe. The heuristic is not guaranteed to find an initial feasible solution, but often finds solutions that are nearly feasible. If the solution produced is infeasible, then local search is performed to attempt to remove infeasibilities. Our local search procedures are described in Section 3.4.1.

The heuristic first generates a solution to an aggregate model in which data is aggregated in two ways. First, port data within each region are aggregated into coarse regional data. This allows us to treat a region as a "super-port" having a production or consumption rate, a capacity, a berth limit, etc, equal to the sum of the corresponding attribute at each individual port in the region. Second, vessel data are aggregated by vessel class so that individual vessel paths are not distinguished by the solution procedure. After aggregation has occurred, vessels are routed from region to region using regional and vessel class data only. Individual ports and vessels are ignored. We call this first-stage aggregate model SystemModel since it is convenient to think of it as the model a system-level manager might solve in order to obtain a coarse solution to a large-scale problem. With a solution to the aggregate model in hand, a sequence of submodels, one for each region, is solved to determine precise routes and loading/discharging decisions for each individual vessel over the entire planning horizon. We call each second-stage submodel RegionalModel as we can think of a regional manager having control over the decisions that affect his particular region.

The motivation for aggregating and decomposing in this manner is due to the following observations. The two largest contributors to the objective function value of our MIRP are inter-regional arc costs and revenues from discharging product at discharging ports. Intra-regional arc costs are typically an order of magnitude less than inter-regional arc costs. Moreover, since prices at ports within a region are highly positively correlated and since we assume that vessels travel at capacity from a loading region to a discharging region, once we know the discharging regions that vessels are scheduled to visit, we know the approximate revenue that will be obtained from those visits. Consequently, it seems natural to give a higher priority to inter-regional routing decisions and only secondary importance to intra-regional decisions.

Before describing SystemModel and RegionalModel, it is important to note that both models are instantiations of the Core Model (2). This means that after writing the code (in an algebraic modeling language or in a programming language) for the Core Model (2) just once, we may create instances of the model multiple times, with different underlying networks and different parameter data, to produce system and regional models for each region. We believe this model re-usability is an attractive practical feature of our approach.

3.2.1 SystemModel

To create SystemModel and to accommodate our aggregations based on region and vessel class, we use a different network from the one described in Section 2.3. The regular nodes are region-time pairs (r, t) for all $r \in \mathcal{R}$ and $t \in \mathcal{T}$, as opposed to port-time pairs, since our goal is determine the movement of vessels from region to region. In addition, the arc set is the union of the arc sets $\mathcal{A}^{vc}$ for each individual vessel class, i.e., $\mathcal{A} = \cup_{vc \in \mathcal{VC}} \mathcal{A}^{vc}$, where the arc sets $\mathcal{A}^{vc}$ are straightforward adaptations of the individual vessel arc sets. Specifically, the arc set $\mathcal{A}^{vc}$ consists of source arcs, sink arcs, inter-regional travel arcs, and waiting arcs within a region. The only intra-regional travel arcs in this network are waiting arcs $((r, t), (r, t+1))$, as shown in Figure 7. Inter-regional travel arcs assume the maximum travel time between regions is required. That is, if τ_{ij} denotes the travel time between two ports i and j, then for each pair of regions r_1 and r_2 of different type (loading and discharging), the inter-regional travel time $\tau_{r_1,r_2} = \max\{\tau_{ij} : i \in r_1, j \in r_2\}$.

We populate the model with data aggregated by region:

$$B_r = \sum_{j \in r} B_j \ , \qquad d_{r,t} = \sum_{j \in r} d_{j,t} \ , \qquad F^{\min}_{r,t} = \min_{j \in r}\{F^{\min}_{j,t}\} \ , \qquad F^{\max}_{r,t} = \sum_{j \in r} F^{\max}_{j,t} \ ,$$

$$s_{r,0} = \sum_{j \in r} s_{j,0} \ , \qquad S^{\min}_{r,t} = \sum_{j \in r} S^{\min}_{j,t} \ , \qquad S^{\max}_{r,t} = \sum_{j \in r} S^{\max}_{j,t} \ , \qquad \forall\ r \in \mathcal{R}, \forall\ t \in \mathcal{T} \ .$$

There is some ambiguity in how the revenues $R_{r,t}$ at nodes should be set. Setting $R_{r,t}$ equal to the average or maximum revenue over all ports in the discharging region are natural choices.

There are three changes to the Core Model (2) that are required:

1. Wherever the index j or v appears, an r or a vc should appear instead. Consequently, all decision variables that are vessel-specific in the Core Model (2) become vessel class-specific in SystemModel, i.e., the variables become $f^{vc}_n, x^{vc}_a, z^{vc}_n$, and $s^{vc}_{r,t}$ (the latter is explained below).

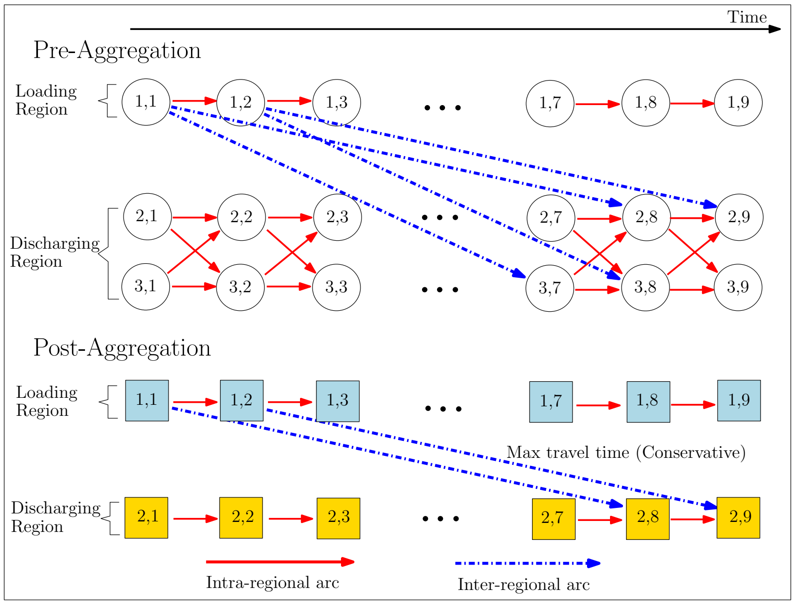


Figure 7: Example of a network before and after aggregation. Circles represent port-time pairs in the original network. Squares represent region-time pairs in the aggregate network. The two ports in the discharging region are aggregated together. No aggregation occurs in the loading region since there is only one loading port.

2. Constraints (2k) become $x_a^{vc} \in \mathbb{Z}_+, \forall\ vc \in \mathcal{VC}, \forall\ a \in \mathcal{A}^{vc}$ to account for the fact that multiple vessels in the same class may travel along the same arc. Constraints (2l) become $z_{r,t}^{vc} \in \mathbb{Z}_+$ with $z_{r,t}^{vc} \leq B_r$, $\forall\ n = (r,t) \in \mathcal{N}, \forall\ vc \in \mathcal{VC}$, since multiple vessels in the same class may simultaneously attempt to load/discharge in the same region.

3. In the Core Model (2), each vessel has its own dedicated arc set $\mathcal{A}^v$. Since a vessel can only be in one location at a time, it was enough to use the decision variable s_t^v to keep track of inventory on each vessel in each time period. When modeling the flow of vessel classes, however, different vessels in the same vessel class are often in different regions at the same time. Thus, we use the continuous decision variable $s_{r,t}^{vc}$ to keep track of the amount of inventory on each vessel class *in each region* in each time period. In addition, inventory balance constraints (2d) for each vessel are replaced by inventory balance constraints for each vessel class,

$$s_{r,t}^{vc} = s_{r,t-1}^{vc} + \Delta_r \left(f_{r,t}^{vc} - \sum_{a \in \mathcal{XS}_{r,t}^{vc,\text{inter}}} Q^{vc} x_a^{vc} \right), \qquad \forall\ r \in \mathcal{R}, \forall\ t \in \mathcal{T}, \forall\ vc \in \mathcal{VC}\ , \quad (17)$$

where $\mathcal{XS}_{r,t}^{vc,\text{inter}}$ is $\mathcal{FS}_{r,t}^{vc,\text{inter}}$ if $r \in \mathcal{R}^P$ and $\mathcal{RS}_{r,t}^{vc,\text{inter}}$ if $r \in \mathcal{R}^C$, and $\mathcal{FS}_{r,t}^{vc,\text{inter}}$ is the set of outgoing arcs from node (r,t) that are inter-regional or sink arcs and $\mathcal{RS}_{r,t}^{vc,\text{inter}}$ is the set of incoming arc to node (r,t) that are inter-regional or source arcs.

Finally, it may be that SystemModel is infeasible because it is impossible to satisfy the inventory bound constraints in all regions. To avoid this situation, analogous to Constraints (5), we may replace regional inventory balance constraints (2c) with the constraints

$$s_{r,t} = s_{r,t-1} + \Delta_r \left(d_{r,t} - \sum_{vc \in \mathcal{VC}} f_n^{vc} - \alpha_{r,t} \right) , \qquad \forall\ n = (r,t) \in \mathcal{N} . \tag{18}$$

There is no bound on the cumulative amount of slack $\sum_t \alpha_{r,t}$ that can be used.

Under this formulation, a solution produced by SystemModel specifies, among other things, routes for all vessel classes from region to region and the *minimum duration* a vessel in each vessel class will remain in each region, all while maintaining inventory balance constraints, inventory bound constraints, and berth limit constraints at an aggregate (regional) level. Note the emphasis on "minimum duration." Since the inter-regional arcs in this model assume the longest port-to-port arc is being taken, once individual vessel routes are determined in the RegionalModel, it may turn out that a vessel will arrive in a region earlier than expected and stay in the region longer than the minimum duration.

Since a solution to SystemModel does not specify routes for individual vessels, only routes for each vessel class, we perform a post-processing step and assign individual vessels from each vessel class to routes based on a simple first-in first-out (FIFO) procedure. That is, if two or more vessels from the same vessel class are routed to a region in overlapping time intervals, then we assign the routes to individual vessels so that the first vessel to enter the region is the first to leave, breaking ties arbitrarily.

We now describe how SystemModel is solved multiple times in order to generate several first-stage solutions. The underlying reason for doing this is that, even if the maximum travel time between regions is used in the first-stage network, SystemModel may still generate solutions that are overly optimistic in which vessels enter a region, immediately load/discharge all of their inventory, and then travel to another region. Of course, for some instances, it may be necessary for vessels to stay multiple periods in a region so that multiple ports can be visited. To allow for this possibility, we add constraints to SystemModel that force vessels in vessel class vc to remain in a region r for a minimum duration of τ_r^{vc} periods. Specifically, we include the constraints

$$\sum_{u=0}^{t+\tau_r^{vc}} \sum_{a \in \mathcal{FS}_{(r,u)}^{vc}} x_a^{vc} \leq \sum_{u=0}^{t} \sum_{a \in \mathcal{RS}_{(r,u)}^{vc}} x_a^{vc} \ , \qquad \forall\ r \in \mathcal{R}, \forall\ t \in \mathcal{T}, \forall\ vc \in \mathcal{VC} \ , \tag{19}$$

which state that the number of vessels in vessel class vc exiting region r by time $t + \tau_r^{vc}$ must not exceed the number of vessels in vessel class vc entering region r by time t. Note that these constants τ_r^{vc} depend on the region and vessel class because larger vessels may need to remain at a single port for multiple periods or visit multiple ports in a region to fully load or discharge, while smaller vessels may only need to visit a single port in one time period.

To obtain multiple first-stage solutions, we can vary the parameter τ_r^{vc} so that vessels are forced to stay in a region for a minimum number of consecutive periods. Let $\tau_r^{vc,\min}$ denote the minimum duration that a vessel in vessel class vc must remain in region r and $\tau_r^{vc,\max}$ denote the largest minimum duration that should be considered (so $\tau_r^{vc,\max} \geq \tau_r^{vc,\min}$). The parameter $\tau_r^{vc,\min}$ can be computed from the data, e.g., if $F_{j,t}^{\max} < Q^{vc}$ for all $j \in r$, then $\tau_r^{vc,\min} \geq 1$. The parameter $\tau_r^{vc,\max}$ is defined by the user. We found that $\tau_r^{vc,\max} = 2$ or 3 works well for the instances we considered. Given these parameters, we first set $\tau_r^{vc} = \tau_r^{vc,\max}$ for each region r and each vessel class vc, and solve SystemModel with minimum duration constraints (19). This produces the most conservative first-stage solution. Then, we simultaneously decrement each τ_r^{vc} by 1, subject to $\tau_r^{vc} \geq \tau_r^{vc,\min}$, and solve SystemModel again so that vessels can potentially make more voyages. We continue this process until $\tau_r^{vc} = \tau_r^{vc,\min}$ for each region r and each vessel class vc. Note that by proceeding in this order, from most conservative to least conservative, we can warm-start the solution process of each iteration with the solution found from the previous iteration.

As a final note, if we assume that the travel times between regions is the *minimum* travel time between any two ports in the two regions, instead of the maximum travel time as above, and set revenues $R_{r,t} = \max\{R_{j,t} : j \in r\}$ for each $r \in \mathcal{R}^C$ and $t \in \mathcal{T}$, then the SystemModel can be used to compute a valid bound on the objective function of the Core Model (2). This bound is easy to compute and almost always better than the bound obtained from solving the Core Model (2) as-is, but is typically not very tight as port specific information is ignored in favor of using regional information.

3.2.2 RegionalModel

Given a first-stage solution, we solve a sequence of second-stage subproblems, one for each region, in order to construct a solution, not necessarily feasible, to the Core Model (2). Throughout this subsection, we assume a fixed region $r \in \mathcal{R}$ is under consideration. Suppose after obtaining a solution to SystemModel and applying the FIFO procedure mentioned above, it is determined that vessel v makes K_r^v visits to region r over the entire planning horizon. Then the purpose of RegionalModel is to determine the route and loading/discharging decisions that vessel v selects during each of its K_r^v visits.

Intra-regional routing and loading/discharging decisions are found by solving an instantiation of the Core Model (2) with three modifications. First, the underlying network involves only those nodes associated with ports in region r and only the source, sink, intra-regional, and waiting arcs in arc set $\mathcal{A}^v$ that are associated with region r. It is a subnetwork of the original network described in Section 2.3.

Second, since a vessel may visit a region multiple times, we allow a vessel to take multiple source and sink arcs in region r over the planning horizon. To accomplish this, we replace the flow balance constraints (2b) for the cases when $n = n_s$ and $n = n_t$, which ensure that exactly one source and one sink arc for each vessel is chosen over the entire planning horizon, with the constraints

$$\sum_{a \in \mathcal{FS}^v_{n_s,k}} x_a^v = 1 \ , \ \forall \ v \in \mathcal{V}, \forall \ k \in K_r^v \tag{20a}$$

$$\sum_{a \in \mathcal{RS}^v_{n_t,k}} x_a^v = 1 \ , \ \forall \ v \in \mathcal{V}, \forall \ k \in K_r^v \ , \tag{20b}$$

where $\mathcal{FS}^v_{n_s,k}$ and $\mathcal{RS}^v_{n_t,k}$ are the sets of source and sink arcs, respectively, associated with vessel v's kth visit to this region. These constraints ensure that exactly one source arc and exactly one sink arc is taken on vessel v's kth visit.

There is some flexibility in how the sets $\mathcal{FS}^v_{n_s,k}$ and $\mathcal{RS}^v_{n_t,k}$ are chosen. We can fix the departure time when a vessel leaves a region or we can fix the arrival time when a vessel enters a region. Either way, because the first-stage solution assumes that the maximum travel time between regions is required, we are guaranteed that any solution produced by the second-stage models will result in a solution that can be made feasible to the full model. We choose to fix the departure time when a vessel leaves a region. Let t^v_k and u^v_k be the first and last time periods that vessel v may enter and exit region r during its kth visit. Thus, for each port $j \in r$, $\mathcal{RS}^v_{n_t,k}$ includes sink arcs $((j, u^v_k), n_t)$ (note that all tail nodes have the same departure time u^v_k) and $\mathcal{FS}^v_{n_s,k}$ includes source arcs $(n_s, (j, t^v_{j,k}))$, where $t^v_{j,k}$ is the latest possible time that vessel v can reach port j on visit k if it leaves the previous region visited at the fixed departure time dictated by the first-stage solution.

Third, for each of vessel v's K^v_r visits, we set $s^v_{t^v_k-1} = 0$ if $r \in \mathcal{R}^P$ and $s^v_{t^v_k-1} = Q^v$ if $r \in \mathcal{R}^C$. During all time periods $t \in \mathcal{T} \setminus \cup_{k=1}^{K^v_r}[t^v_k, \ldots, u^v_k]$ that do not involve a visit, the inventory balance constraints (2d) on vessels are omitted as they are not necessary.

It may be that RegionalModel is infeasible because it is impossible to satisfy the inventory bound constraints at all ports. To avoid this situation, analogous to Constraints (5), we may replace inventory balance constraints (2c) with the constraints

$$s_{j,t} = s_{j,t-1} + \Delta_j \left(d_{j,t} - \sum_{v \in \mathcal{V}} f^v_n - \alpha_{j,t} + \beta_{j,t} \right) , \qquad \forall\ n = (j,t) \in \mathcal{N} . \qquad (21)$$

Here, $\beta_{j,t}$ is a slack variable with a high penalty that takes a positive value if a vessel is forced to load more inventory than a loading port has in inventory or discharge more inventory that a discharging port has capacity for. By including slack variables $\beta_{j,t}$, the solver will not report that the model is infeasible, but instead a solution that uses some positive amount of costly slack.

3.2.3 Summary and Commentary

We now summarize our construction heuristic. Pseudocode is provided in Algorithm 1. Multiple first-stage solutions are generated by solving SystemModel while varying the minimum duration that vessels in each vessel class must stay in each region. Vessels within each vessel class are then assigned to specific routes based on a simple FIFO procedure.

After these solutions are generated, RegionalModel instances are populated, with initial conditions/constraints dictated by the first-stage solution, and solved. A complete solution to the Core Model (2) is now available. Finally, we apply local search on this complete solution to repair it if it is infeasible or improve it otherwise.

A valid criticism of this approach is that there is no feedback loop between the first- and second-stage models. For example, after solving RegionalModel for a particular region, suppose we learn that during a particular visit, a certain vessel (in vessel class vc_1) requires more time in the region to fully load or discharge. It would be instructive if the SystemModel could use this information to amend its first-stage solution. The problem is that, in the aggregate framework devised above, aggregation takes place within each region and within each vessel class. Thus, it is difficult to use vessel-specific information to modify constraints and decision variables that affect all vessels in a vessel class. Continuing the example, if we were to insist that all vessels in vessel class vc_1 must remain an additional period in region r, then this could have an adverse effect in which a majority of vessels remain longer than needed in the region.

We are not arguing that successfully incorporating a feedback loop in this scheme is impossible, but it would take some care. Instead, in the next section, we devise another two-stage procedure that incorporates a feedback loop and systematically makes progress towards a better complete solution.

3.3 A Two-Stage Algorithm with Feedback

In this section, we describe another two-stage procedure that is more exact in nature than our construction heuristic. The purpose of this second approach is to remedy two issues. First, the lower bounds provided by the Core Model (2) are often extremely loose for large instances and our construction heuristic does not attempt to generate better bounds. Second, our construction heuristic does not have a feedback loop in which SystemModel is re-solved after gathering information from the solutions to the regional subproblems. With these two deficiencies in mind, our aim is to devise an algorithm, which continues to use aggregation and decomposition, that iteratively solves first- and second-stage models and

Algorithm 1 Multi-Start Construction Heuristic

```
1: Create an empty list of SystemModel solutions called AggregateSolutionPool.
2: Set $\tau_r^{vc} = \tau_r^{vc,\max}$ for each region $r$ and vessel class $vc$.
3: repeat
4:    Solve SystemModel with minimum duration constraints (19) and modified inventory
      balance constraints (18).
5:    Set $\tau_r^{vc} = \min\{\tau_r^{vc} - 1, \tau_r^{vc,\min}\}$.
6: until no $\tau_r^{vc}$ is decremented
7: for each solution in AggregateSolutionPool do
8:    Perform FIFO procedure to assign vessels to specific routes.
9:    for each region do
10:      Solve RegionalModel with modified inventory balance constraints (21).
11:   end for
12:   Merge the solutions to each RegionalModel into a complete, but possibly infeasible,
      solution to the Core Model (2).
13:   Perform local search on the complete solution to remove infeasibility (penalties) and
      improve the solution.
14: end for
15: return The best solution found.
```

also produces useful bounds.

Throughout this section, we make the following simplifying assumption:

> **"Two-port-with-no-revisits" assumption**: A vessel may visit at most two ports during each visit to a region and, once a vessel leaves a port, it will not return to that port during the same visit in the region.

In other words, routes within a region are simple as they involve at most two ports without any revisits. It is natural to ask: How restrictive is this assumption? In practice, voyages with one or two ports per visit in a region are the most common due to issues of robustness and economies of scale. Planners prefer to design routes with voyages having a limited number of port visits to reduce the impact of the unplanned disruptions on future voyages. Likewise, it is often more economical to have a vessel visit a small number of ports, unlike in the trucking industry where many customers may be served by a single vehicle after it leaves the depot. As a result, there are numerous applications in which at most two ports per visit are considered. Dauzère-Pérès et al. [34] created a decision support tool for the distribution of calcium carbonate slurry in which "ships never unload in more than one port before returning to the processing plant." In a shipment planning problem of bitumen, Persson and Göthe-Lundgren [76] allow for a vessel to discharge at no more than two ports per voyage. In liquefied natural gas (LNG) transportation, the most common practice is to have full load and full discharge where a vessel travels between only one loading port and discharging port in a trip (Andersson et al. [7], Goel et al. [47], Rakke et al. [81]). In the LNG setting studied by [48], multiple port visits in discharging regions are possible, but the number of ports visits are limited to two due to tank restrictions. Other papers in which this restriction is used include [6, 19, 86]. Hennig et al. [54, 55] limit the number of port visits in both loading and discharging regions to three on a crude oil transportation problem and mention that this is a practical limit from both economic and risk reduction perspectives. In summary, while visiting three or more ports is possible in some settings, there are numerous applications when our assumptions are not too stringent. On the other hand, if visits to three or more ports in a region are possible, but rare, our approach may still be of interest as it can find an optimal solution within a large, but restricted solution space of the original solution space.

It is also natural to ask: What happens if this assumption is relaxed? Theoretically, it is not difficult to extend the ideas that we present below. Practically, it is likely that the first-stage model will be much more time consuming to solve and produce weaker bounds.

3.3.1 An Augmented Time-Space Network

Before explaining our first- and second-stage models, we describe an augmented time-space network underlying the models. The fundamental goal behind this augmentation is to decouple inter- and intra-regional routing decisions. To this end, we introduce two sets, denoted $\mathcal{N}'$ and $\mathcal{N}''$, of "customs" nodes. We associate with each original node $n = (j, t) \in \mathcal{N}$ two additional nodes: a customs entrance node $n' = (j', t) \in \mathcal{N}'$ and a customs exit node $n'' = (j'', t) \in \mathcal{N}''$. The purpose of introducing these additional nodes is to keep track of the exact nodes (port-time pairs) used to enter and exit a region (hence, the name "customs" node). All incoming arcs to an entrance node are inter-regional or source arcs and all outgoing arcs are customs arcs. All incoming arcs to an exit node are customs arcs and all outgoing arcs are inter-regional or sink arcs. All inter-regional arcs connected to

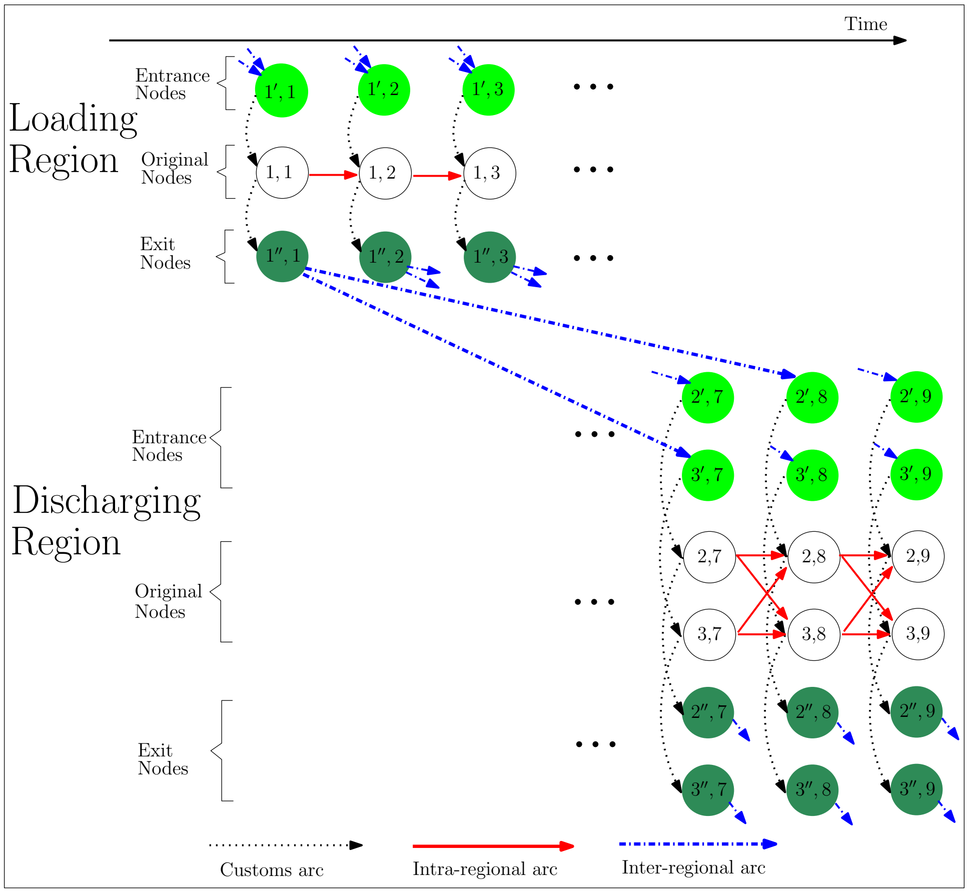


Figure 8: Augmented time-space network

original nodes are removed. An example of an augmented network when there is one loading region with a single port and one discharging region with two ports is shown in Figure 8.

3.3.2 SystemModel-2Port: A Route-Only Master Problem

Our first-stage problem, which we call SystemModel-2Port to distinguish it from SystemModel of the previous section, can again be interpreted as the problem solved by a system-level manager in hopes of getting a coarse solution to the Core Model (2). SystemModel-2Port is a route-only pure integer program that produces as output (i) the number of vessels from each vessel class that travel along each inter-regional arc and (ii) the duration and the number of vessels in each region during each visit. Inventory balance at ports is partially modeled, but only implicitly. Constraints based on well-known lot-sizing relaxations are included to ensure that ports and regions are visited with the correct frequency.

3.3.2.1 Basic elements of SystemModel-2Port

SystemModel-2Port takes place on the augmented time-space network described above, but only considers customs entrance and exit nodes; original nodes are ignored. Since the routes of each vessesl class are modeled, not the individual vessel routes, we define the set $\mathcal{A}^{vc}$ of all arcs associated with vessel class vc and the set $\mathcal{A}^{vc,\text{inter}}$ of all inter-regional arcs associated with vessel class vc. In addition, a set $\mathcal{A}^{vc,ee}$ of *entry-exit arcs* (not to be confused with intra-regional arcs) is required. Whereas intra-regional arcs connect original nodes in a region as shown in Figure 8, entry-exit arcs connect customs entrance nodes to customs exit nodes within a region. Thus, the entry-exit arc $(j_1', t_1), (j_2'', t_2)$ for a particular vessel class corresponds to a vessel that enters the region at port j_1' at time t_1, makes its first visit to the corresponding original port j_1 before traveling to original port j_2 some time before or at t_2, and exits the region from port j_2'' at time t_2. If $j_1 = j_2$, then the vessel remains at the same port for the duration of the visit. Figure 9 depicts a portion of the augmented time-space network used in the first-stage problem. Only arcs for one particular vessel class are shown.

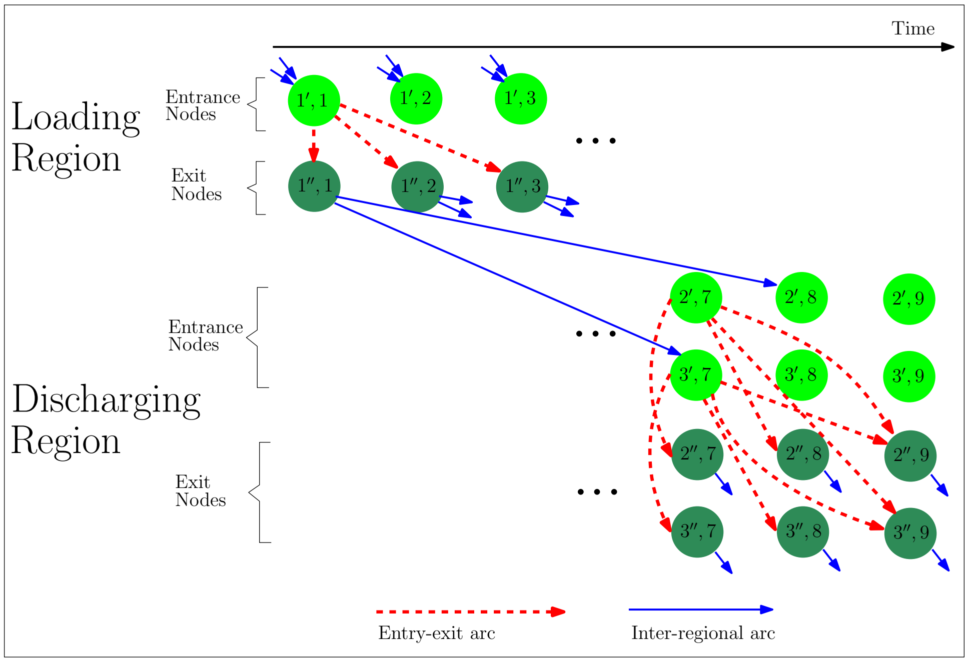


Figure 9: First-stage customs network

Entry-exit arcs exploit information about port capacities, minimum and maximum load/discharge quantities, and vessel class capacities in order to avoid creating vessel routes which will certainly lead to infeasibilities in the second stage. For ease of exposition, assume for the moment that port capacities, production/consumption rates, and travel times are constant over the planning horizon. The idea is straightforward to extend when this is not the case. Let $\nu^{vc}_{j_1,j_2}$ denote the minimum number of periods required for a vessel in vessel class vc to fully load/discharge at ports j_1 and j_2, where $j_1, j_2 \in r$ for some region $r \in \mathcal{R}$, during a single visit to that region. Then, for each pair of ports j_1 and j_2, entry-exit arcs are created with length $\nu^{vc}_{j_1,j_2}$ up to some user-defined parameter. In practice, there is typically a maximum duration that a vessel will remain in a given region, e.g., five or ten days, and this parameter can be used to limit the number of entry-exit arcs that need to be considered.

As an example, suppose that the arc $(3', 7), (3'', 9)$ shown in Figure 9 is the shortest entry-exit arc from port 3 to itself (i.e., ignore arc $(3', 7), (3'', 7)$ and arc $(3', 7), (3'', 8)$). This could occur for the following reason. Suppose that port 3 is a discharging port consuming 40 units of product per period with a capacity $S^{\max}_3 = 210$ and maximum per-period discharge quantity $F^{\max}_3$ of 250 units. Then, the minimum duration required for a vessel with 300 units of capacity to visit only port 3 is three periods since the vessel would have to discharge 250 units in the first period, 40 units in the second period, and 10 units in the third period.

For each entry-exit arc $a = ((j_1, t_1), (j_2, t_2)) \in \mathcal{A}^{vc,ee}$, we introduce an integer decision variable w^{vc}_a to denote the number of vessels in vessel class vc that travel along arc a. In additional, let R_a denote the maximum price at the two ports associated with arc a over all times in the time interval, i.e., $R_a = \max\{R_{j,t} : j \in \{j_1, j_2\}, t \in [t_1, t_2]\}$. Since prices at ports in the same region are assumed to be highly positively correlated and also relatively close to one another, using the maximum value is guaranteed that our lower bound is valid. Since we assume that at most two ports are visited during a particular visit to a region, the cost associated with an entry-exit arc is $C^{vc}_a = C^{vc}_{(j_1,j_2)} - Q^{vc} R_a$, where $C_{(j_1,j_2)}$ is the cost of traveling from port j_1 to j_2 and is assumed to be constant for all time periods. Note that $C_{(j,j)} = 0$ for all $j \in \mathcal{J}$.

In the first-stage model, there are two types of decision variables: x_a^{vc} denoting integer flow on inter-regional arcs $a \in \mathcal{A}^{vc,\text{inter}}$ within vessel class vc, and w_a^{vc} denoting flow on entry-exit arcs $a \in \mathcal{A}^{vc,ee}$ within vessel class vc. The following model can be used to obtain a lower bound on the objective function of a restricted space of the Core Model (2) under the restrictions imposed by the assumptions made at the outset:

SystemModel-2Port

$$\min_{\mathbf{x},\mathbf{w}} \sum_{vc\in\mathcal{VC}} \sum_{a\in\mathcal{A}^{vc,\text{inter}}} C_a^{vc} x_a^{vc} + \sum_{vc\in\mathcal{VC}} \sum_{a\in\mathcal{A}^{vc,ee}} C_a^{vc} w_a^{vc} \tag{22a}$$

$$\text{s.t.} \sum_{a\in\mathcal{FS}_{n_s}^{vc}} x_a^{vc} = |\mathcal{V}^{vc}|, \qquad \forall\ vc \in \mathcal{VC} \tag{22b}$$

$$\sum_{a\in\mathcal{RS}_{n_t}^{vc}} x_a^{vc} = |\mathcal{V}^{vc}|, \qquad \forall\ vc \in \mathcal{VC} \tag{22c}$$

$$\sum_{n''\in\mathcal{N}''} w_{(n',n'')}^{vc} - \sum_{a\in\mathcal{RS}_{n'}^{vc}} x_a^{vc} = 0, \qquad \forall\ vc \in \mathcal{VC}, \forall\ n' = (j',t) \in \mathcal{N}' \tag{22d}$$

$$\sum_{a\in\mathcal{FS}_{n''}^{vc}} x_a^{vc} - \sum_{n'\in\mathcal{N}'} w_{(n',n'')}^{vc} = 0, \qquad \forall\ vc \in \mathcal{VC}, \forall\ n'' = (j'',t) \in \mathcal{N}'' \tag{22e}$$

First-stage berth limit constraints (see Section 3.3.2.2) (22f)

Lot-sizing based covering and packing constraints (see Section 3.3.2.3) (22g)

$$w_a^{vc} \in \mathbb{Z}_+, \qquad \forall\ vc \in \mathcal{VC}, \forall\ a \in \mathcal{A}^{vc,ee} \tag{22h}$$

$$x_a^{vc} \in \mathbb{Z}_+, \qquad \forall\ vc \in \mathcal{VC}, \forall\ a \in \mathcal{A}^{vc,\text{inter}}\ . \tag{22i}$$

The first four sets of constraints ensure flow balance for each vessel classes. Constraints (22f) attempt to ensure that certain berth limit constraints are not violated. Constraints (22g) are covering and packing constraints that ensure that a port or loading region is visited enough times in every time interval. They do not ensure that product is actually loaded/discharged at that time (this will be left to the regional managers to decide). Lot-sizing based covering and packing constraints are discussed below. This model contains no inventory-related constraints.

3.3.2.2 First-Stage Berth Limit Constraints

In the Core Model (2), berth limit constraints at a port are explicitly handled through Constraints (2e). Although SystemModel-2Port (22) does not know when an attempt to load/discharge is actually made, some logical deductions can be made to ensure that certain berth limits are not violated. Initially, we include in Model (22) a first-stage berth limit constraint for each port-time pair by limiting the number of "shortest" entry-exit arcs that can be taken. A "shortest" entry-exit arc $a = ((j_1, t), (j_2, u))$ is simply an arc from j_1 to j_2 whose duration $u - t$ is as small as possible. By definition, a vessel must attempt to load or discharge at time t and time u in order for arc a to be a shortest entry-exit arc (otherwise, the vessel could load or discharge in fewer time periods, e.g, in the interval $[t, \ldots, (u-1)]$). For vessel class vc, let $\bar{\mathcal{FS}}^{vc,ee}_{(j',t)}$ be the set of shortest outgoing entry-exit arcs from node $n' = (j', t)$ and let $\bar{\mathcal{RS}}^{vc,ee}_{(j'',t)}$ be the set of shortest incoming entry-exit arcs to node $n'' = (j'', t)$. Then, the *first-stage berth limit constraint*

$$\sum_{vc \in \mathcal{VC}} \left[\sum_{a \in \bar{\mathcal{FS}}^{vc,ee}_{(j',t)}} w^{vc}_a + \sum_{a \in \bar{\mathcal{RS}}^{vc,ee}_{(j'',t)}} w^{vc}_a \right] \leq B_j, \qquad \forall\, j \in \mathcal{J}, \forall\, t \in \mathcal{T} \ , \tag{23}$$

is valid for the first-stage master problem. Constraint (23) sums over all of the shortest entry-exit arcs incident to the entrance node $n' = (j', t)$ and the exit node $n'' = (j'', t)$. As an example, consider port $j = 1$ at time period $t = 3$ shown in Figure 10 and assume that there is only one vessel class. Suppose that all of the arcs shown in Figure 10 are the shortest entry-exit arcs involving port 1 at time period 3. That is, in order to fully load or discharge, a vessel must remain in the region at least two additional periods if the two ports involved are port 1 and itself or port 1 and port 3; otherwise, if ports 1 and 2 are involved, then a vessel must remain in the region at least one additional period. Then, a valid constraint is: the flow on all arcs shown in Figure 10 must not exceed b_1. If this constraint is violated, then there is no way for all vessels involved to fully load/discharge at port 1 at time period 3.

Additional constraints will be generated dynamically and appended to Model (22).

3.3.2.3 Lot-sizing based covering and packing constraints

Our main tool for ensuring that ports and regions are not under- or overwhelmed by vessels are constraints based on the familiar lot-sizing set. Consider the standard capacitated lot-sizing set (see, e.g., Pochet and Wolsey [77]) in which one must decide in what periods

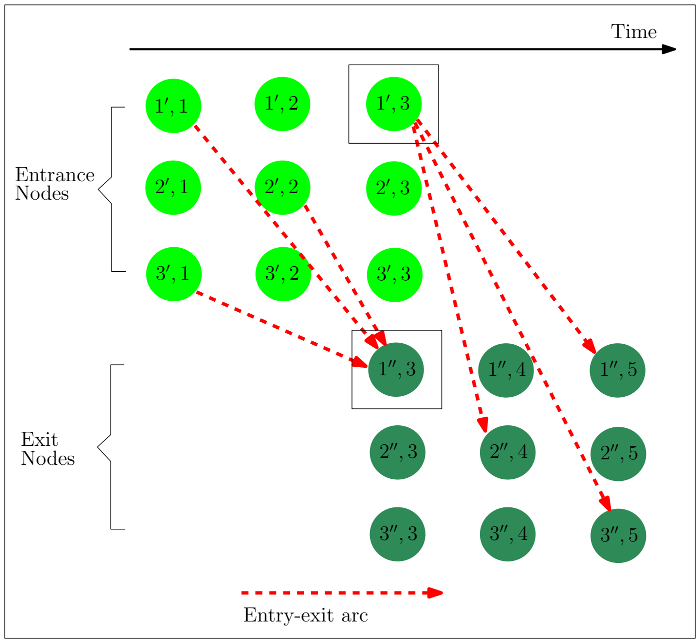


Figure 10: Example of first-stage berth limit constraints

to produce an item and how much to produce, given demand data d_t, initial inventory s_0, constant storage capacity $s^{\max}$, and capacities C_t on production in period t of a finite planning horizon $\mathcal{T}$:

$$s_{t-1} + x_t = d_t + s_t \ , \qquad \forall\ t \in \mathcal{T} \tag{24a}$$

$$0 \le x_t \le C_t y_t \ , \qquad \forall\ t \in \mathcal{T} \tag{24b}$$

$$y_t \in \{0,1\} \ , \qquad \forall\ t \in \mathcal{T} \tag{24c}$$

$$0 \le s_t \le s^{\max} \ , \qquad \forall\ t \in \mathcal{T} \ . \tag{24d}$$

The decision variables are: s_t, the stock (inventory) in period t; x_t, the amount produced in period t; and y_t, a binary decision variable taking value 1 if production takes place in period t and 0 otherwise. For any time interval $[t_1, t_2]$, we can sum over constraints (24a) and apply inequalities (24b) to obtain the relaxation

$$s_{t_1-1} + \sum_{u=t_1}^{t_2} C_u y_u \ge d_{[t_1,t_2]} + s_{t_2} \ , \qquad \forall\ 1 \le t_1 \le t_2 \le T \tag{25a}$$

$$y_t \in \{0,1\} \ , \qquad \forall\ t \in \mathcal{T} \tag{25b}$$

$$0 \le s_t \le s^{\max} \ , \qquad \forall\ t \in \mathcal{T} \ , \tag{25c}$$

where $d_{[t_1,t_2]} = \sum_{u=t_1}^{t_2} d_u$ is the demand in the time interval. Replacing s_{t_1-1} with its upper bound $s_{t_1-1}^{\max}$ and s_{t_2} with its lower bound $s_{t_2}^{\min}$, we obtain a further relaxation

$$\sum_{u=t_1}^{t_2} C_u y_u \geq d_{[t_1,t_2]} + s_{t_2}^{\min} - s_{t_1-1}^{\max} \ , \qquad \forall\ 1 \leq t_1 \leq t_2 \leq T \tag{26a}$$

$$y_t \in \{0,1\} \ , \qquad \forall\ t \in \mathcal{T} \ . \tag{26b}$$

Note that we prefer to include the term $(s_{t_2}^{\min} - s_{t_1-1}^{\max})$ in the right hand side of (26a) instead of $(0 - s^{\max})$ since instance data may reveal that $s_{t_1-1}^{\max} < s^{\max}$ and $s_{t_2}^{\min} > 0$.

Relaxation (26) has an important interpretation that we ultimately exploit. Namely, if we interpret y_t as the decision to turn a production machine on or off, then relaxation (26) has replaced the original lot-sizing set (24) in which stocking, production quantity, and machine on-off decisions are made with a pure binary set with constraints on the number of times the machine must be turned on during every time interval $[t_1, t_2]$. In other words, the original mixed-integer linear set has been relaxed to a pure integer set whose constraints are specified in a purely combinatorial way. Geometrically speaking, our set of interest is the projection of relaxation (25) onto the space of binary variables y.

Finally, note that if the capacitated lot-sizing set (24) also includes constraints $C_t^{\min} y_t \leq x_t$ for all $t \in \mathcal{T}$, i.e., forcing a minimum amount to be produced if production takes place, then applying the same arguments as above, the relaxation

$$\sum_{u=t_1}^{t_2} C_u^{\min} y_u \leq d_{[t_1,t_2]} + s_{t_2}^{\max} - s_{t_1-1}^{\min}, \qquad \forall\ 1 \leq t_1 \leq t_2 \leq T \tag{27a}$$

$$y_t \in \{0,1\}, \qquad \forall\ t \in \mathcal{T} \ , \tag{27b}$$

is valid. Constraints (26a) and (27a) work in tandem to bound the number of times the machine must be turned on over the planning horizon.

Using the ideas above, we now describe how to define what we call lot-sizing based covering and packing constraints. For every time interval $[t_1, t_2]$, these constraints provide lower and upper bounds on the number of vessels (actually, the weighted combination of vessels from each vessel class) that can enter or depart from a subset of ports in a region.

Consider a subset I of ports in the same region. Let $\mathcal{A}_{I,[t_1,t_2]}^{vc,ee}$ be the set of all entry-exit arcs associated with vessels in vessel class vc that "touch" a port in I in the time interval $[t_1, t_2]$. That is, arc $a = ((i,t),(j,u))$ belongs to $\mathcal{A}^{vc,ee}_{I,[t_1,t_2]}$ and "touches" a port in I in $[t_1, t_2]$ if and only if there exists a path $\{(i,t) = (i_1,u_1), \ldots, (i_K,u_K) = (j,u)\}$ in the *original* network described in Section 2.3 such that $i_k \neq i_{k+1}$ for at most one k (i.e., the path satisfies the "two-port-with-no-revisits" assumption) and with a node (i_k, u_k) on the path satisfying $i_k \in I$ and $u_k \in [t_1, t_2]$. In other words, $\mathcal{A}^{vc,ee}_{I,[t_1,t_2]}$ is the set of arcs for which a vessel in vessel class vc has an opportunity to load or discharge at a port in I in the time interval $[t_1, t_2]$. Define $C^{vc,\min}_{I,[t_1,t_2],a}$ and $C^{vc,\max}_{I,[t_1,t_2],a}$ to be the minimum and maximum amount of product that can be loaded/discharged at all ports in I in the time interval $[t_1, t_2]$ by a vessel in vessel class vc if entry-exit arc a is used. Let $d_{I,[t_1,t_2]} = \sum_{j\in I} \sum_{u=t_1}^{t_2} d_{j,u}$ and let $S^{\min}_{I,t}$ and $S^{\max}_{I,t}$ denote the minimum and maximum inventory levels at all ports in I at time t. Finally, let $\mathcal{I}$ be the set of subsets I of ports under consideration.

Then, analogous to Constraints (26a) for the lot-sizing set, we can define subset covering constraints

$$\sum_{vc\in\mathcal{VC}} \sum_{a\in\mathcal{A}^{vc,ee}_{I,[t_1,t_2]}} C^{vc,\max}_{I,[t_1,t_2],a} w^{vc}_a \geq d_{I,[t_1,t_2]} + S^{\min}_{I,t_2} - S^{\max}_{I,t_1-1} \ , \qquad \forall I \in \mathcal{I}, \forall 1 \leq t_1 \leq t_2 \leq T \ . \tag{28}$$

Similarly, analogous to Constraints (27a), we can define subset packing constraints

$$\sum_{vc\in\mathcal{VC}} \sum_{a\in\mathcal{A}^{vc,ee}_{I,[t_1,t_2]}} C^{vc,\min}_{I,[t_1,t_2],a} w^{vc}_a \leq d_{I,[t_1,t_2]} + S^{\max}_{I,t_2} - S^{\min}_{I,t_1-1} \ , \qquad \forall I \in \mathcal{I}, \forall 1 \leq t_1 \leq t_2 \leq T \ . \tag{29}$$

Since the maximum amount that a vessel in vessel class vc can load/discharge in a region is its capacity Q^{vc}, i.e., $C^{vc,\max}_{I,[t_1,t_2],a} \leq Q^{vc}$, one could easily replace the coefficients $C^{vc,\max}_{I,[t_1,t_2],a}$ with Q^{vc} in (28) and still have a valid relaxation, albeit a much weaker one. Instead, by using simple logical arguments, one can exploit the parameters to compute coefficients $C^{vc,\max}_{I,[t_1,t_2],a}$ that are strictly less than Q^{vc}. For example, if we consider a single port $I = \{j\}$ and an entry-exit arc $a = ((i,t),(j,u))$, with $i \neq j$, that touches port j in the time interval of interest, we can assume that at least $F^{\min}_i$ units will be loaded/discharged at port i leaving at most $Q^{vc} - F^{\min}_i$ units to be loaded/discharged at port j.

Example 1. Consider the 3-period horizon instance shown in Figure 11. Customs entrance and exit nodes are shown along with 12 entry-exit arcs. There are two ports in a discharging

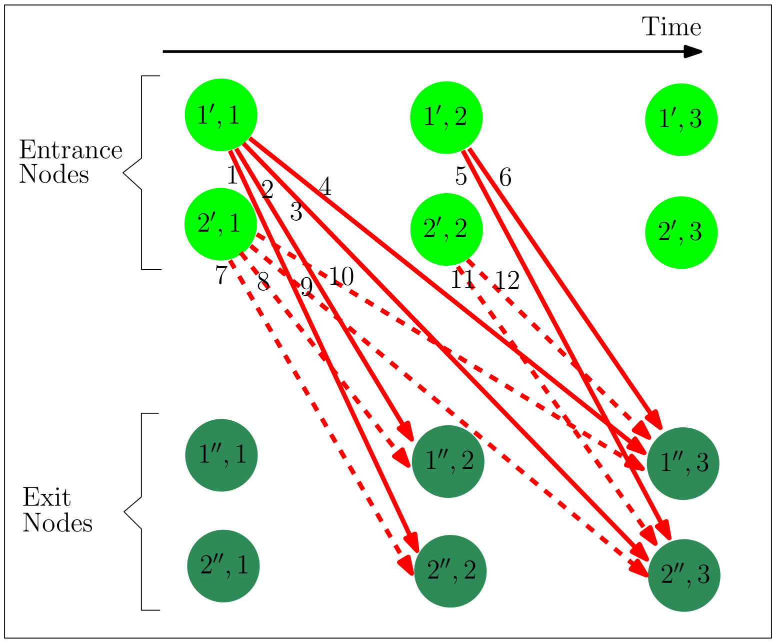


Figure 11: Example of customs network

Table 7: Example of lot-sizing based covering and packing constraints

$I, [t_1, t_2]$	Covering Constraints					
$\{1\}, [1,2]$	$200(w_5 + w_6) + 250(w_1 + w_3 + w_8 + w_{10}) + 300(w_2 + w_4)$	$\geq$	50	$=$	$100 - 50$	(C1)
$\{2\}, [2,2]$	$200(w_1 + w_3 + w_7 + w_9 + w_{10} + w_{11} + w_{12})$	$\geq$	-125	$=$	$75 - (200 - 75)$	(C2)
$\{2\}, [2,3]$	$200(w_1 + w_7) + 250(w_3 + w_5 + w_{10} + w_{12}) + 300(w_9 + w_{11})$	$\geq$	25	$=$	$150 - (200 - 75)$	(C3)
$\{1,2\}, [1,3]$	$300 \sum_{a=1}^{12} w_a$	$\geq$	325	$=$	$375 - 50$	(C4)
	Packing Constraints					
$\{1\}, [1,2]$	$300w_2 + 100(w_1 + w_4 + w_5 + w_6 + w_8) + 50w_3$	$\leq$	350	$=$	$100 + (300 - 50)$	(P1)
$\{2\}, [2,3]$	$300w_{11} + 100(w_1 + w_5 + w_7 + w_9 + w_{12}) + 50w_3$	$\leq$	450	$=$	$150 + 300 - 0$	(P2)
$\{1,2\}, [1,3]$	$300 \sum_{a=1}^{12} w_a$	$\leq$	925	$=$	$375 + 600 - 50$	(P3)

region, each with one berth, and the travel time between ports is one period. Assume that $F_j^{\min} = 50$, $F_j^{\max} = 200$, $S_j^{\max} = 300$ for $j = 1, 2$, and that initial inventories are $s_{1,0} = 50$ and $s_{2,0} = 0$. Suppose there is a single vessel class with capacity $Q^1 = 300$ and note that since $F_j^{\max} < Q^1$ for $j = 1, 2$, which implies that all entry-exit arcs involve at least two periods. Assume that the initial nodes of the vessels are not fixed.

Table 7 lists some of the lot-sizing based constraints that can be derived based on the instance data. We discuss some of the particulars for several of the constraints listed. Constraint (C1): Because $F_1^{\max} = 200$, at most 200 units of inventory can be discharged at port 1 in the time interval $[1, 2]$ if arc 5 or 6 is chosen. Arcs 1, 3, 8, and 10 involve both ports and, because $F_2^{\min} = 50$, at most 250 units can be discharged at port 1 in the time interval $[1, 2]$ if any of these arcs is chosen. Constraint (C2): This constraint is redundant as the right hand side value is below zero. Note, however, that 200 is the coefficient for each variable in the constraint since $F_2^{\max} = 200$. Also note that $S_{\{2\},1}^{\max} = (200 - 75)$ since at most $F_2^{\max} = 200$ units can be discharged in time period 1 and 75 of those units will be consumed by demand in period 1. Constraint (P1): Arcs 1, 5, and 8 involve port 2 and arcs 4 and 6 involve port 1 for some time periods outside of the time interval $[1, 2]$. Using the parameters $F_1^{\max} = F_2^{\max} = 200$ implies that the minimum amount that must be discharged at port 1 in $[1, 2]$ is $300 - 200 = 100$ units. Constraints (C4) and (P3): We call these constraints regional covering and regional packing constraints, respectively, as they apply to all ports in the region.

3.3.3 RegionalModel-2Port: A Constrained Second-Stage Subproblem

The second-stage model, which we call RegionalModel-2Port, to solve our regional subproblems is very similar to RegionalModel with two minor differences. First, a solution to SystemModel-2Port specifies the arrival node (i, t) and departure node (j, u) for each visit. (Recall that a solution to SystemModel specifies the latest possible arrival time and the departure time; arrival and departure ports are left to the regional manager.) Consequently, the sets $\mathcal{FS}_{n_s,k}^v$ and $\mathcal{RS}_{n_t,k}^v$ associated with Constraints (20) become singletons. Second, the "two-port-with-no-revisits" assumption needs to be enforced. This is accomplished by setting to zero all arcs that cannot be on an $(i, t) - (j, u)$ path.

3.3.4 Feedback Loop: Iterating between the Master and Subproblems

Given a first-stage solution to SystemModel-2Port, it may not be possible to find a feasible second-stage solution for each regional subproblem. In this case, each infeasible subproblem must communicate to the master problem a set of cuts that can be used to generate a different first-stage solution. We do this through two types of cuts.

The first type of cut is generated when a first-stage berth limit constraint is violated. This may happen when multiple entry-exit arcs involving the same subset of ports is chosen.

The second type of cut we generate is a so-called *enumeration cut.* When the first-stage solution does not induce a feasible second-stage solution for a particular region, we can always apply an enumeration cut to prevent the first-stage model from generating the same solution for this region. If $x_a^{vc} \in \{0,1\}$ for all $vc \in \mathcal{VC}$ and $a \in \mathcal{A}^{vc}$, then an enumeration cut can be written as

$$\sum_{vc \in \mathcal{VC}} \left[\sum_{a \in \mathcal{A}^{vc} : \hat{x}_a^{vc} = 0} x_a^{vc} + \sum_{a \in \mathcal{A}^{vc} : \hat{x}_a^{vc} = 1} (1 - x_a^{vc}) \right] \geq 1 \tag{30}$$

where $\hat{\mathbf{x}}$ is the current first-stage solution that induces an infeasible second-stage solution. Otherwise, one needs to express this cut using a binary expansion of the integer variables.

Note that we could also attempt to separate lot-sizing based cuts on an as-needed basis, but we prefer to generate them before launching the solver. The main reason for this preference is that, in early testing, we found that checking for violated inequalities, after each new incumbent solution was found, resulted in longer run times than simply including the constraints a priori and letting the presolver eliminate redundant constraints. Separation appears to be time consuming because one needs to check every port and every region in every time interval $[t_1, t_2]$.

3.3.5 Relation to Classical Benders Decomposition

Our approach bears a resemblance to classical Benders decomposition for mixed-integer linear optimization [16]. Consider the MIP

$$\begin{aligned} \min \quad & \mathbf{c}^T\mathbf{x} + \mathbf{d}^T\mathbf{y} && \text{(31a)} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \geq \mathbf{b} && \text{(31b)} \\ & \mathbf{B}\mathbf{x} + \mathbf{C}\mathbf{y} \geq \mathbf{g} && \text{(31c)} \\ & \mathbf{x} \in \mathbb{Z}_+^{n_x}, \mathbf{y} \in \mathbb{R}_+^{n_y} && \text{(31d)} \end{aligned}$$

where all vectors and matrices are of conforming dimension. After projecting problem (31) onto the space defined by the integer variables, Benders decomposition states solving problem (31) is equivalent to solving the so-called full master problem

$$
\begin{aligned}
\min \quad & \mathbf{c}^T\mathbf{x} + \eta && \text{(32a)}\\
\text{s.t.} \quad & \mathbf{A}\mathbf{x} \geq \mathbf{b} && \text{(32b)}\\
& \eta \geq (\mathbf{g} - \mathbf{B}\mathbf{x})^T\mathbf{p} \qquad \forall\, \mathbf{p} \in \mathcal{P} && \text{(32c)}\\
& 0 \geq (\mathbf{g} - \mathbf{B}\mathbf{x})^T\mathbf{r} \qquad \forall\, \mathbf{r} \in \mathcal{R} && \text{(32d)}\\
& \mathbf{x} \in \mathbb{Z}_+^{n_x} && \text{(32e)}
\end{aligned}
$$

where $\mathcal{P}$ and $\mathcal{R}$ denote the set of extreme points and extreme rays, respectively, of the polyhedron $D = \{\mathbf{u} \geq \mathbf{0} : \mathbf{C}^T\mathbf{u} \leq \mathbf{d}\}$. The standard Benders decomposition algorithm solves problem (31) in two stages. In the first stage, problem (32) is solved, except with $\mathcal{P}$ and $\mathcal{R}$ replaced by a subset of extreme points and rays of D, for a first-stage solution $\tilde{\mathbf{x}}$. In the second stage, the LP $\max\{(\mathbf{g} - \mathbf{B}\tilde{\mathbf{x}})^T\mathbf{u} : \mathbf{u} \in D\}$ is solved and an extreme ray is returned if the LP is determined to be unbounded; otherwise an extreme point is returned. This extreme point or ray is then included in the subset or extreme points and rays found thus far and the iterative process continues.

Suppose that $\mathbf{x}$ can be partitioned as $\mathbf{x}^T = (\mathbf{x}_1^T, \mathbf{x}_2^T)$ so that we can rewrite (31) as

$$
\begin{aligned}
\min \quad & \mathbf{c}_1^T\mathbf{x}_1 + \mathbf{c}_2^T\mathbf{x}_2 + \mathbf{d}^T\mathbf{y}\\
\text{s.t.} \quad & \mathbf{A}_1\mathbf{x}_1 + \mathbf{A}_2\mathbf{x}_2 \geq \mathbf{b}\\
& \mathbf{B}_1\mathbf{x}_1 + \mathbf{B}_2\mathbf{x}_2 + \mathbf{C}\mathbf{y} \geq \mathbf{g}\\
& \mathbf{x}_1 \in \mathbb{Z}_+^{n_{x_1}}, \mathbf{x}_2 \in \mathbb{Z}_+^{n_{x_2}}, \mathbf{y} \in \mathbb{R}_+^{n_y}\ .
\end{aligned}
\tag{33}
$$

In our approach, we find it more natural to project problem (33) onto the space defined by the $\mathbf{x}_1$ integer variables only. Assuming prices are constant over time in every region, the objective function value of any second-stage solution is completely determined by the first-stage solution. Thus, solving problem (33) is equivalent to finding a first-stage solution

$\mathbf{x}_1$ that induces a feasible second-stage solution to the MIP

$$\begin{aligned}
\min \quad & \mathbf{c}_2^T \mathbf{x}_2 + \mathbf{d}^T \mathbf{y} \\
\text{s.t.} \quad & \mathbf{A}_2 \mathbf{x}_2 \geq \mathbf{b} - \mathbf{A}_1 \tilde{\mathbf{x}}_1 \\
& \mathbf{B}_2 \mathbf{x}_2 + \mathbf{C}\mathbf{y} \geq \mathbf{g} - \mathbf{B}_1 \tilde{\mathbf{x}}_1 \\
& \mathbf{x}_2 \in \mathbb{Z}_+^{n_{x2}}, \mathbf{y} \in \mathbb{R}_+^{n_y} \ .
\end{aligned} \tag{34}$$

At first glance, projecting the original problem (33) onto the space defined by only a subset of the integer variables does not seem appealing for two reasons. First, the second-stage problem (34) is a MIP, not an LP, and therefore theoretically much harder to solve. It turns out, however, that these MIPs are typically not challenging to solve given the fact that problem (34) also decomposes by region. Second, we are forced to use tools that go beyond standard LP techniques. Namely, LP theory says that if the second-stage LP in standard Benders decomposition is infeasible, then the dual is unbounded and, thus, we can find an extreme ray to include in the restricted master problem. In our decomposition framework, if problem (34) is infeasible, then this could be due to the LP relaxation being infeasible or because no integer solution exists. If the former occurs, we could add a Benders feasibility cut just as in the traditional setting. If the latter occurs, which is more likely, we must add a cut to the restricted master problem that prevents the master problem from generating the same solution again.

3.4 Enhancements: Improving Practical Performance

In this section, we discuss several important techniques that we have found useful to improve the practical performance of the Zoom algorithm. The first improvement leads to primal enhancements, while the last improvement is aimed at improved the dual bound provided by (22).

3.4.1 MIP-Based Local Search

An important and effective enhancement is the use of MIP-based local search, a general method in which a series of smaller/reduced MIPs are solved to locally improve an existing solution to a larger MIP. Although it can be applied to any solution (feasible or infeasible) at any time in the search procedure, we apply it immediately following the construction phase described in Section 3.2 and to each solution stored in a solution pool of SystemModel-2Port (22). Several authors have shown how local search can be used to find high-quality solutions and improve existing solutions. Note that in [92], one loading region, one discharging region, and voyage chartered vessels are considered so that a vessel path involves a single inter-regional trip. In our problem, long-term vessels make multiple inter-regional trips.

The first local search neighborhood that we found to be effective empirically is an extension of the “Fix Supply” and “Fix Demand” neighborhoods proposed in Hewitt et al. [57]. In this neighborhood, all decisions in all regions of a particular type (i.e., loading or discharging) are fixed while the decisions in the remaining regions are selected by the solver. It is effective at optimizing routing and loading/discharging decisions in each region. Moreover, although we do not do it, the regional problems can be solved separately (in parallel). This neighborhood has the advantage that a solver can often solve the MIP-based local search problem to optimality in under 60 seconds for instances involving four ports in a region and a 60-period horizon. It is surprising that this neighborhood is solvable in a reasonable amount of time.

Our algorithm works as follows. We first fix the decisions made in the loading regions and optimize the decisions made in the discharging regions, subject to the constraint that vessels must arrive at the correct fixed starting nodes in the loading regions. For example, if all decisions are fixed in all loading regions, then the path (sequence of port-time pairs) and loading decisions of every vessel when visiting the loading regions is fixed. We prefer to fix loading region decision first because, in our instances, there are typically more ports in a discharging region than in a loading region and, therefore, it seems likely that there are more opportunities for improvement. Next, we fix all decisions made in the discharging regions and optimize the decisions in the loading regions. The search continues iterating between the two regions until no improvements are made. After obtaining a re-optimized solution given a particular fixing in regions of the same type, we attempt to have vessels leave a region as soon as possible. For example, if we find that a vessel has fully discharged by time t but does not leave the region until time $t + 1$, which might happen because the vessel can still arrive at its next loading region when leaving at time $t+1$, then we force the vessel to leave at time t so that in the subsequent solve, when the decisions in the loading regions are re-optimized, the solver has more flexibility.

Algorithm 2 Iterated Fix Supply Fix Demand Local Search

```
Require: A feasible or infeasible solution to the Core Model (2).
 1: repeat
 2:   for fixedRegionType in {Loading,Discharging} do
 3:     for each region of type fixedRegionType do
 4:       Fix all vessel paths and all $z_n^v$ variables in this region to their current value.
 5:     end for
 6:     for each region not of type fixedRegionType do
 7:       Solve RegionalModel with Constraints (20) where $\mathcal{FS}^v_{n_s,k}$ and $\mathcal{RS}^v_{n_t,k}$ are defined
          in the text.
 8:       (Optional) Force vessels to depart from a region in the time period of their last
          attempt to load/discharge.
 9:     end for
10:   end for
11: until no improvement in the objective function value is made or no vessels departs from
      a region in an earlier time period than in the previous iteration
12: return The updated solution.
```

Pseudocode of this procedure is given in Algorithm 2, which we call "Iterated Fix Supply Fix Demand Local Search." To make the algorithm more precise, we explain how the sets $\mathcal{FS}^v_{n_s,k}$ and $\mathcal{RS}^v_{n_t,k}$ in Constraints (20) are modified. We will refer to regions in which decisions are fixed as "fixed regions" and all other regions as "free regions." Recall that the arrival and departure nodes for each vessel in each fixed region are known. For each free region and for each vessel, we define $\mathcal{FS}^v_{n_s,k}$ as the set of source arcs $\{(n_s,(j,t^v_{j,k}))\}$ for visit k, where $t^v_{j,k}$ is the time period in which vessel v would arrive at port j on its kth visit to this free region if it were to depart from the previous region from its fixed node. Similarly, we define $\mathcal{RS}^v_{n_t,k}$ as the set of sink arcs $\{((j,u^v_{j,k}),n_t)\}$ for visit k, where $u^v_{j,k}$ is the time period in which vessel v would need to depart from port j on its kth visit to this free region in order to arrive in the subsequent region at the correct fixed node.

As a final note on this procedure, it is worth mentioning some details about our implementation, since empirically it is superior to what we believe is an easier implementation. Perhaps, the most straightforward implementation is to instantiate the Core Model (2) and then write an iterative procedure that solves reduced instances of the Core Model (2). In this approach, all regional subproblems are solved simultaneously, despite being separable, and, thus, the instances can be relatively large and time consuming. Instead, we solve each regional subproblem separately using RegionalModel and pass pertinent information between regions with a data structure. This approach is parallelizable, but solves relatively quickly in series.

Another local search neighborhood that proved to be useful is similar to the "Fix Time Window" neighborhood proposed in Hewitt et al. [57]. In this neighborhood, a subset S of vessels is selected and the discrete decision variables for all vessels not in S are fixed, i.e., all x_a^v and z_n^v variables are fixed at their current value for $v \in \mathcal{V} \backslash S$. For each vessel in S, a time window is selected and all discrete decision variables outside of the time window are fixed, while all decision variables inside the time window are selected by solving a small MIP. Specifically, if $[t_1, t_2]$ denotes the time window, all routing variables x_a^v that start before or after the time window, i.e., $a = ((j_1, t), (j_2, t'))$ with $t < t_1$ or $t > t_2$, are fixed during the solve. We found that MIP-based local search is particularly well suited for eliminating small infeasibilities in a given solution.

3.4.2 Branching on Auxiliary Decision Variables

SystemModel-2Port (22) has a highly fractional LP relaxation, meaning that even after branching on many variables, the solution to the LP relaxation contains many binary decision variables that take a fractional value. Branching on these arc variables has limited impact because arcs essentially repeat in time, e.g., arcs from port i to j occur in succession until the end of the time horizon. To avoid unproductive branching, we employ a technique commonly used in a column generation in which we include auxiliary integer decision variables so that we can branch on decisions that are likely to have more impact. Specifically, we introduce auxiliary integer decision variables y_r^{vc} and y_j^{vc} to count the number of times vessels in vessel class vc visit region $r \in \mathcal{R}$ and port $j \in \mathcal{J}$ over the entire time horizon. We then assign y_r^{vc} variables the highest branching priority, y_j^{vc} variables the next highest branching priority, and finally the original variables are assigned the solver's default priority.

3.4.3 Integer Knapsack Polytope Constraints

The bound provided by the LP relaxation of SystemModel-2Port is usually good relative to the improvements that are made after branching. As a consequence, it may be interesting for those experimenting with a heuristic to solve only the root node of the branch-and-bound tree and use the bound obtained without going any further. In order to improve the LP relaxation, one can include integer knapsack polytope constraints.

If we isolate a packing constraint (29) in which slack variables are also included (i.e., if a spot market is present), we obtain the set

$$W = \left\{ x \in \{0,1\}^{n_1}, s \in \mathbb{R}_+^{n_k} : \sum_{j=1}^{n_1} \hat{a}_j x_j - \sum_{k=1}^{n_k} s_k \leq \hat{b} \right\} , \tag{35}$$

where the x_j variables correspond to arc variables, the s_k variables correspond to slack variables over an interval of n_k periods, and $\hat{a}_j$ and $\hat{b}$ are data. Solvers like Gurobi and Cplex have specialized routines that attempt to separate inequalities for the continuous knapsack set, which we would have if there were only a single s variable present in (35). We could include auxiliary continuous variables to represent the sum $\sum_{k=1}^{n_k} s_k$, but then we would have to hope that the solver is able to find some helpful cuts for this set while working in a higher dimension. Instead, we make use of the fact that only a small amount of cumulative slack is permitted in our model and replace the sum $\sum_{k=1}^{n_k} s_k$ by its upper bound $s^{\max}$. Setting $b = \hat{b} + s^{\max}$, we obtain the relaxed pure binary knapsack set

$$X = \left\{ x \in \{0,1\}^{n_1} : \sum_{j=1}^{n_1} \hat{a}_j x_j \leq b \right\} . \tag{36}$$

Solvers have extremely efficient routines for performing separation on binary knapsack sets. However, in order for us to take advantage of this, for every packing constraint that we include in the model, we would also have to include a constraint of the form (36), set the solver's parameter for generating knapsack cuts to "aggressive" and again hope that the solver is able to generate helpful cuts. Instead of depending on the solver to generate useful cuts for these single-row binary knapsack constraints, we go one step further and attempt to generate facets of low-dimensional integer knapsack sets. To do this, we collect all binary variables x_j with the same coefficient $\hat{a}_j$ and we create a temporary integer decision variable y_i. We say "temporary" because these variables are not included in the final model; their only purpose is for preprocessing. Assume this aggregation produces n_2 such integer variables y_i. This gives rise to an integer knapsack set

$$Y = \left\{ y \in \mathbb{Z}^{n_2} : \sum_{j=1}^{n_2} a_j y_j \leq b \right\} \ . \tag{37}$$

When n_2 is small, e.g. $n_2 \leq 10$, it is possible to obtain the facets of the integer knapsack polytope (37) by calling PORTA [31].

There are at least two options when applying facets of the integer knapsack polytope: (Option 1) append some or all of them to the initial formulation in a preprocessing step; (Option 2) append them within a branch-and-cut framework in which separation is performed at a subset of nodes in the branch-and-cut tree; or (Option 3) a compromise approach in which the cuts are generated in a preprocessing step, added to a cut pool, and then added on an as-needed basis. We have opted for the latter approach so that the number of rows in the initial constraint matrix is kept small.

3.5 Summary of Our Approach

Having defined all of the ingredients, we summarize in pseudocode our algorithmic approach in Algorithm 3.

Observations:

1. Step 4: For larger instances, warm-starting SystemModel-2Port can save hundreds of seconds in presolve time and an additional hundreds of seconds in solving the root LP.

2. We refer to Steps 5-21 as *Zoom*. This procedure is implemented using a LazyConstraintCallback as is typically required for a Benders-like strategy.

3. Steps 15-20: Since we are searching for the best solution to the Core Model (2), which does not impose the two-port-with-no-revisits assumption, we solve a relaxed version of RegionalModel-2Port in hopes of finding a feasible solution to the Core Model (2). We warm-start the solution process for each relaxed model in Step 16 with the solution

Algorithm 3 Complete Solution Procedure

```
1: Create an empty list of solutions SolutionPool.
2: Apply the two-stage multi-start construction heuristic of Section 3.2 to generate a list
   LIST of (possibly infeasible) solutions to the Core Model (2).
3: Perform local search on each solution in LIST to remove any infeasibilities and/or find
   an improving solution.
4: Warm-start SystemModel-2Port (22) with the best feasible solution found thus far.
5: for each integer feasible solution found do
6:   for each regional subproblem do
7:     Solve RegionalModel-2Port
8:     if RegionalModel-2Port is infeasible then
9:       Add a first-stage berth limit cut or enumeration cut.
10:    end if
11:  end for
12:  if all regional subproblems are feasible then
13:    A new incumbent solution has been found.
14:  end if
15:  for each regional subproblem do
16:    Solve a relaxed version of RegionalModel-2Port with no two-port-with-no-revisits
       constraints.
17:  end for
18:  if all regional subproblems are feasible then
19:    Store this solution in SolutionPool.
20:  end if
21: end for
22: Perform local search on each solution in SolutionPool to find an improving solution.
23: return The best solution found and the bound provided by SystemModel-2Port (22).
```

obtained from the restricted model in Step 8, so that the additional CPU time for this search is often a couple of seconds.

4. Step 22: We could perform local search after each new solution to the Core Model (2) is found, but we chose to use local search as a last step in the spirit of a solution polishing procedure.

3.6 Computational Experiments

All computations were carried out on a Linux machine with kernel 2.6.18 running on a 64-bit x86 processor equipped with two Intel Xeon E5520 chips, which run at 2.27 GHz, and 48GB of RAM. The LP and MIP solvers of Gurobi 5.0 were used. All algorithms were coded in Python and run on a single thread.

All models were solved with the default optimality tolerance of 0.01%. In the construction heuristic, SystemModel was given a time limit of 300 seconds (each time it was called in Step 4 of Algorithm 1) and was solved with emphasis on feasibility.

3.6.1 Experiments with the Core Model

We conducted several experiments with the Core Model (2) to understand its strengths and limitations. These experiments are described in Section 2.5. The main finding was that, within a 24-hour time limit, the Core Model (2) could not find feasible solutions to any of the instances and produced bounds that were virtually useless.

We also performed an experiment to answer the question: Does a solver perform better (i.e., produce feasible or higher quality feasible solutions) when the Core Model (2) is modified so that the solution space is smaller and only includes solutions that satisfy the two-port-with-no-revisits assumption? Since our approach provides a bound on the restricted solution space due to the two-port-with-no-revisits assumption, we would like to know what happens when these same restrictions are incorporated into the Core Model (2). Unfortunately, the results are worse. To reduce the solution space of the Core Model (2), we included additional constraints, which were implemented in two ways. First, we included explicit constraints in the Core Model (2) so as not to violate the two-port-with-no-revisits assumption. The additional constraints led to an even larger model and ultimately bogged down the solution process even further, producing bounds that were worse (in a 24-hour time limit) than when these constraints had not been included at all. Second, we attempted to include these constraints in a lazy fashion through a LazyConstraintCallback. In this approach, additional constraints are only generated on an as-needed basis. The problem with this approach is that in order to use lazy constraints, one must disable dual reductions, which are used in the preprocessing phase, and so the resulting presolved model is larger. Solving this larger presolved model along with checking for lazy constraint violations also resulted in worse performance than the vanilla approach.

3.6.2 Main computational results

Tables 8 and 9 show the main results of our approach for instances with planning horizons of 45 and 60 periods, respectively. The relative and absolute gaps are computed as $(z^{\text{Best}} - z^{LB})/z^{\text{Best}} * 100\%$ and $(z^{\text{Best}} - z^{LB})$, respectively, where z^{Best} is the objective function value of the best known solution and z^{LB} is the lower bound provided by SystemModel-2Port. An asterisk appears next to relGap and absGap as a reminder that the lower bound is with respect to the two-port-with-no-revisits assumption. Nevertheless, *in all of the best known solutions to all instances, the two-port-with-no-revisits assumption is never violated.*

We begin with some general comments about the algorithm. As one would expect, going from 45 periods to 60 periods leads to greater computational challenges as reflected in the gaps. For instances with one loading and one discharging region, our algorithm is quite effective and produces small relative and absolute gaps. For instances with multiple loading and multiple discharging regions, our algorithm has greater difficulty proving optimality (with respect to the "two-port-with-no-revisits" assumption). A partial explanation for this is that, in our instances with multiple loading and discharging regions, many two-port visits are typically required. SystemModel-2Port, which provides the lower (dual) bound, favors splitting vessels, e.g., sending one-half of a vessel to port 1 in region 1 and one-half of the same vessel to port 2 in the same region, in its node LP relaxations to avoid intra-regional travel costs. An example is shown in the Example 2.

Example 2. The purpose of this example is to illustrate how SystemModel-2Port (22) prefers to split vessels in its LP relaxations, leading to highly fractional solutions. There are two vessel classes with capacity 300 and 250, respectively, each with two vessels. The starting nodes for each vessel are given. Source and sink arcs are not shown. Figure 12 shows an optimal solution to this small instance. Meanwhile, Figure 13 shows the solution to the root LP relaxation. Numbers next to arcs denote the fractional amount of flow along each arc. The most important observation is that, since integral flows of vessels in vessel classes does not need to be obeyed in the LP relaxation, vessels are split to avoid avoid intra-regional arc costs. Vessels are also split across time. For example, in an optimal solution, the vessel starting in the discharging region at time 3 makes a two-port visit (it starts at the customs entrance node for discharging port 1 and then terminate at the customs exit node for discharging port 2). On the other hand, in the LP relaxation, this same vessel is split over time and only makes a single-port visit.

Despite some relative gaps above 5%, we believe that the results are quite promising. For a relative comparison, Hewitt et al. [57] use a branch-and-price guided local search technique to find solutions to challenging 60-period instances presented in Engineer et al. [36]. This class of problems is different from ours and, therefore, a direct comparison is difficult, but we would argue that our instances are as complex as theirs. Their algorithm runs for 30 minutes on four processors, which is roughly two hours of serial computation. They produce very good results, but they make no attempt at providing a bound. Our approach is successful at simultaneously finding good solutions and good bounds in just over two hours.

The columns labeled CH+LS denote the construction heuristic combined with local search. The results indicate that this combined method performs reasonably well, producing solutions that are close to the best known solutions. For most of the instances with a single loading region, the best solution to SystemModel was found within 100 seconds, while the remaining time was spent improving the lower bound. In short, we believe that less time could be spent solving SystemModel for these instances with a single loading region without significantly changing the solution found by the construction heuristic.

Our solution polishing procedure calls our local search heuristics. It is given a total

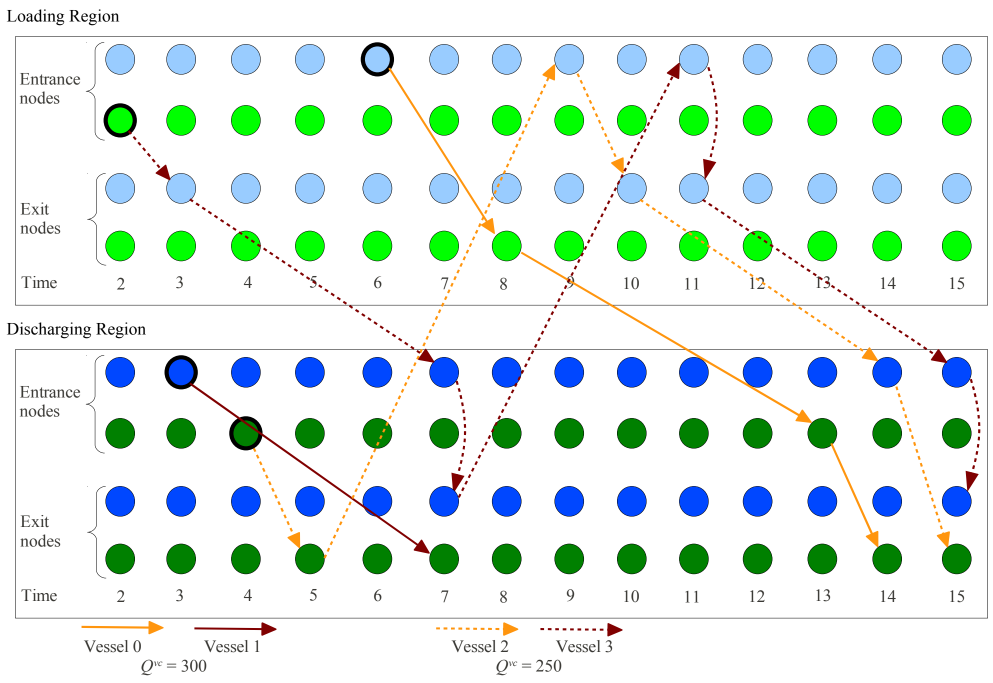


Figure 12: The optimal solution to an instance with 16 periods, 1 loading region having 2 ports, 1 discharging region having 2 ports, 2 vessel classes with 2 vessels in each class.

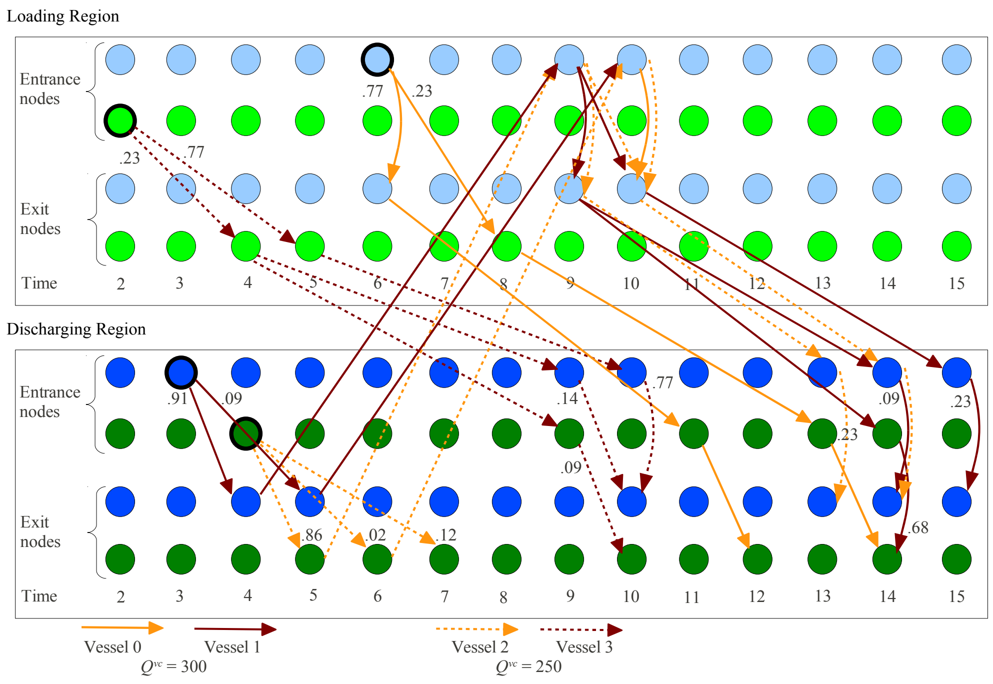


Figure 13: The solution to the LP relaxation of the instance shown in Figure 12.

time limit of 300 seconds since its main purpose is to polish the solution, e.g., force vessels to leave the system sooner and remove any wasted trips, not to significantly improve the solution. In other words, solution polishing was used to “beautify” the solution and was not responsible for any appreciable improvements after calling the Zoom algorithm.

3.6.3 First-stage results

Tables 10 and 11 provide more detail related to SystemModel-2Port (22) by comparing its performance under its default formulation, when auxiliary variables are included to allow for enhanced branching (see Section 3.4.2), and when constraints from the integer knapsack polytope are included (see Section 3.4.3). These tables show the number of first-stage berth limit cuts (**B**) and enumeration cuts (**E**) that are generated; the quality of the root LP relaxation (the value of the LP solution obtained after all processing to the root node of the search tree is completed); and the final bound provided in a two-hour time limit. Again, the final bound (**Final LB***) is denoted with an asterisk as a reminder that the “two-port-with-no-revisits” assumption is in effect. For each instance, SystemModel-2Port (22) was warm-started with the same inital solution found by the construction heuristic with the objective function value reported in Tables 8 and 9.

One might think that the root LP objective function values would be the same for the **Default** and **Integer knapsack cuts** approaches, but this is not the case. The initial LP value should be the same, since the initial models are identical and, therefore, should undergo the same presolve sequence. The processing done at the root node can be rather different, which, indeed, is the case as the final root LP values do not agree. It is surprising that the integer knapsack constraints (added via a cut table) yield inferior objective function values for the 45-period instances, but superior values for the 60-period instances. In general, 20 to 200 integer knapsack cuts are added to the model.

The tables indicate that, despite the increase in the number of variables in the model, including auxiliary decision variables for enhanced branching often improves the value of the root LP relaxation and almost always produces the best final bound. In general, few first-stage berth limit cuts are generated simply because “collisions” at ports are infrequent

Table 8: Main algorithmic results for instances with a 45-period horizon

	CPU Time (s)				Objval				
Instance	**CH+LS**	**Zoom**	**Polish**	**Total**	**CH+LS**	**Zoom**	**LB***	**relGap* (%)**	**absGap***
LR1_1_DR1_3_VC1_V7a	27	132	18	177	-13271	-13272	-13273	0.01	1
LR1_1_DR1_4_VC3_V11a	1032	7200	300	8532	-10650	-11239	-11289	0.44	50
LR1_1_DR1_4_VC3_V12b	487	7200	300	7987	-10112	-10732	-10739	0.06	7
LR1_1_DR1_4_VC3_V12c	550	927	300	1777	-9013	-9069	-9073	0.05	5
LR1_1_DR1_4_VC3_V8a	481	3828	300	4609	-4945	-5106	-5176	1.37	68
LR1_1_DR1_4_VC3_V9a	191	267	54	512	-6730	-6891	-6959	0.98	68
LR1_2_DR1_3_VC2_V6a	473	5190	300	5963	-11062	-11134	-11146	0.10	11
LR1_2_DR1_3_VC3_V8a	134	7200	300	7634	-11747	-12010	-12012	0.02	2
LR2_11_DR2_22_VC3_V6a	649	7200	300	8149	-9571	-9718	-9779	0.63	61
LR2_11_DR2_33_VC4_V11a	1413	7200	300	8913	-13095	-14017	-15137	7.99	1120
LR2_11_DR2_33_VC5_V12a	1434	7200	300	8934	-16666	-18423	-19092	3.63	669
LR2_22_DR2_22_VC3_V10a	1226	7200	300	8726	-23929	-24789	-25576	3.18	787
LR2_22_DR3_333_VC4_V14a	1906	7200	300	9406	-20065	-21952	-23967	9.18	2015
LR2_22_DR3_333_VC4_V17a	2027	7200	300	9527	-19879	-21713	-23553	8.47	1840

Table 9: Main algorithmic results for instances with a 60-period horizon

	CPU Time (s)				Objval				
Instance	**CH+LS**	**Zoom**	**Polish**	**Total**	**CH+LS**	**Zoom**	**LB***	**relGap* (%)**	**absGap***
`LR1_1_DR1_3_VC1_V7a`	386	28	30	444	-16588	-16675	-16676	0.01	2
`LR1_1_DR1_4_VC3_V11a`	779	6060	300	7139	-11407	-13257	-13383	0.94	125
`LR1_1_DR1_4_VC3_V12b`	487	7200	300	7987	-10120	-11040	-11269	2.08	229
`LR1_1_DR1_4_VC3_V12c`	1210	7200	300	8710	-8833	-10053	-10085	0.33	33
`LR1_1_DR1_4_VC3_V8a`	187	7200	300	7687	-4398	-5191	-5628	8.43	437
`LR1_1_DR1_4_VC3_V9a`	785	7200	300	8285	-6479	-7552	-7696	1.91	144
`LR1_2_DR1_3_VC2_V6a`	800	7200	300	8300	-12659	-13631	-13810	1.31	178
`LR1_2_DR1_3_VC3_V8a`	531	7200	300	8031	-12930	-14652	-14931	1.90	278
`LR2_11_DR2_22_VC3_V6a`	904	7200	300	8404	-12581	-12655	-13351	5.50	696
`LR2_11_DR2_33_VC4_V11a`	1443	7200	300	8943	-14852	-15387	-17008	10.53	2701
`LR2_11_DR2_33_VC5_V12a`	1503	7200	300	9003	-19646	-22730	-24246	6.67	1517
`LR2_22_DR2_22_VC3_V10a`	1303	7200	300	8803	-30876	-32627	-34167	4.72	1540
`LR2_22_DR3_333_VC4_V14a`	2042	7200	300	9542	-24239	-26873	-29945	11.43	3072
`LR2_22_DR3_333_VC4_V17a`	2115	7200	300	9615	-24166	-27000	-30248	12.03	3248

due to the spacing of the vessels in good feasible solutions. On the other hand, for some instances, many enumeration cuts were generated, implying that SystemModel-2Port was generating routings that were causing infeasibilities in the regional subproblems. For the larger instances, the bound improvements after the LP relaxation were minor.

Another useful feature of our approach is that the bounds provided by the LP relaxation of SystemModel-2Port are, with only a few exceptions (see instance `LR2_11_DR2_22_VC3_V6a`), relatively close to the final bound produced after running branch-and-cut for an extended period of time and significantly better than those produced by the Core Model. Although SystemModel-2Port is useful in generating new and better solutions, one could also use it only for computing a bound. For example, one could implement a simple parallel framework in which one or more processors are dedicated to computing a primal solution and another is aimed at a dual solution.

Several additional experiments were performed, the results of which we summarize here. We experimented with simultaneously using auxiliary variable branching and integer knapsack constraints, but this did not result in any appreciable improvements. We also gave SystemModel-2Port (22) a five-hour time limit and the additional improvements to the bounds were minor. This suggests that more powerful cuts or model restructuring may be needed.

3.7 Conclusions

In this chapter, we introduced a two-stage algorithm for solving single product MIRPs with a planning horizon of up to 60 periods. These split-pickup and split-delivery problems are very challenging computationally, thus, it is not surprising that our approach calls upon many different techniques to produce good solutions and useful bounds. Our approach uses both aggregation and decomposition to simplify the problem into smaller, more manageable subcomponents. It also borrows well-known results from the lot-sizing literature to provide bounds. Computational results show that our approach is promising.

Another salient feature of our approach is the fact that the decomposition lends itself to parallelization. In both the construction heuristic and the two-stage approach with

Table 10: Comparison using SystemModel-2Port with enhancements for instances with a 45-period horizon. **B** = Berth limit cuts; **E** = Enumeration cuts.

	Default				Auxiliary variable branching				Integer knapsack cuts			
	Cuts		Bounds		Cuts		Bounds		Cuts		Bounds	
Instance	**B**	**E**	**RootLP**	**Final LB***	**B**	**E**	**RootLP**	**Final LB***	**B**	**E**	**RootLP**	**Final LB***
LR1_1_DR1_3_VC1_V7a	0	0	-13368	**-13273**	0	0	-13368	**-13273**	0	0	-13368	**-13273**
LR1_1_DR1_4_VC3_V11a	0	104	-11736	-11619	0	17	-11815	-11583	0	17	-11815	**-11289**
LR1_1_DR1_4_VC3_V12b	26	463	-11226	-11119	34	553	-10835	**-10739**	10	364	-11324	-11138
LR1_1_DR1_4_VC3_V12c	9	40	-9096	**-9073**	0	10	-9086	**-9073**	0	23	-9130	**-9073**
LR1_1_DR1_4_VC3_V8a	17	420	-5234	-5199	29	402	-5227	**-5174**	5	253	-5476	**-5174**
LR1_1_DR1_4_VC3_V9a	0	75	-7369	-7120	0	73	-7113	**-6959**	0	169	-7411	-7336
LR1_2_DR1_3_VC2_V6a	0	1127	-11218	-11147	0	1248	-11218	**-11146**	0	691	-11291	-11149
LR1_2_DR1_3_VC3_V8a	0	16	-12123	-12037	0	4	-12110	-12059	0	12	-12171	**-12012**
LR2_11_DR2_22_VC3_V6a	0	68	-10896	-10133	0	481	-10879	**-9779**	0	82	-11036	-10435
LR2_11_DR2_33_VC4_V11a	2	19	-15254	-15210	1	21	-15286	-15175	1	16	-15482	**-15137**
LR2_11_DR2_33_VC5_V12a	29	674	-19258	**-19092**	22	578	-19188	-19107	0	35	-19491	-19145
LR2_22_DR2_22_VC3_V10a	0	36	-25876	-25791	0	99	-25729	**-25576**	0	69	-26278	-25587
LR2_22_DR3_333_VC4_V14a	0	11	-24427	-24367	0	5	-24597	**-23967**	0	13	-24597	-24518
LR2_22_DR3_333_VC4_V17a	0	201	-23739	-23721	0	28	-23740	**-23553**	0	42	-23832	-23554

Table 11: Comparison using SystemModel-2Port with enhancements for instances with a 60-period horizon. **B** = Berth limit cuts; **E** = Enumeration cuts.

	Default				**Auxiliary variable branching**				**Integer knapsack cuts**			
	Cuts		**Bounds**		**Cuts**		**Bounds**		**Cuts**		**Bounds**	
Instance	**B**	**E**	**RootLP**	**Final LB***	**B**	**E**	**RootLP**	**Final LB***	**B**	**E**	**RootLP**	**Final LB***
LR1_1_DR1_3_VC1_V7a	0	0	-16792	**-16676**	0	0	-16683	**-16676**	0	0	-16792	**-16676**
LR1_1_DR1_4_VC3_V11a	0	40	-13666	-13470	0	63	-13460	**-13383**	3	616	-13566	-13490
LR1_1_DR1_4_VC3_V12b	1	46	-11599	-11485	8	66	-11598	**-11269**	78	363	-11535	-11480
LR1_1_DR1_4_VC3_V12c	0	17	-10541	-10472	0	14	-10311	**-10085**	0	20	-10492	-10473
LR1_1_DR1_4_VC3_V8a	0	76	-5735	-5670	11	39	-5723	**-5628**	24	147	-5719	-5685
LR1_1_DR1_4_VC3_V9a	1	44	-8128	-8055	2	10	-7801	**-7696**	0	29	-8089	-8057
LR1_2_DR1_3_VC2_V6a	0	54	-14108	-14000	0	15	-14063	-13963	0	39	-14057	**-13810**
LR1_2_DR1_3_VC3_V8a	0	7	-15343	-15214	0	4	-15019	**-14931**	0	9	-15030	-15003
LR2_11_DR2_22_VC3_V6a	0	12	-14283	-13908	0	7	-14032	**-13351**	0	8	-14077	-13943
LR2_11_DR2_33_VC4_V11a	0	4	-17491	-17115	0	5	-17346	**-17008**	0	6	-17342	-17083
LR2_11_DR2_33_VC5_V12a	0	5	-24513	-24494	0	12	-24354	**-24246**	0	7	-24491	-24455
LR2_22_DR2_22_VC3_V10a	0	16	-34305	-34181	0	8	-34250	**-34167**	0	24	-34193	-34175
LR2_22_DR3_333_VC4_V14a	0	1	-30000	**-29931**	0	1	-30000	-29975	0	1	-30011	-29961
LR2_22_DR3_333_VC4_V17a	0	6	-30247	30228	0	7	-30247	**-30227**	0	11	30261	-30250

feedback, the regional subproblems can be solved independently. It would be interesting to explore the computational gains from a parallel implementation of our algorithms.

CHAPTER IV

QUICKLY FINDING GOOD SOLUTIONS TO LONG-HORIZON MARITIME INVENTORY ROUTING PROBLEMS

4.1 Introduction

In this chapter, we study a maritime inventory routing problem (MIRP) with a long planning horizon of up to 365 periods (days). For instances with many ports and many vessels, mixed-integer linear programming (MIP) solvers often require hours to produce good solutions even when the planning horizon is 90 or 120 periods. Building on the recent successes of approximate dynamic programming (ADP) for road-based applications within the transportation community, we develop an ADP procedure to quickly generate good solutions to these problems within minutes. Our algorithm operates by solving many small subproblems (one for each time period) and, in so doing, collecting and learning information about how to produce better solutions. Our algorithm is one of the first of its kind for maritime transportation problems and represents a significant departure from the traditional methods used. In particular, whereas virtually all existing methods are "MIP-centric," i.e., they rely heavily on a solver to tackle a nontrivial MIP to generate a good or improving solution in a couple of minutes, our framework puts the effort on finding suitable value function approximations and places much less responsibility on the solver. Computational results illustrate that with a relatively simple framework, our ADP approach is able to generate good solutions to instances with dozens of vessels and varying time horizons much faster than a commercial solver emphasizing feasibility.

The problem that we consider in this chapter is a simplification of our core problem defined in Chapter 2 and resembles that of Goel et al. [47]. It assumes that there is exactly one port within each region (consequently, we will use the terms "port" and "region" interchangeably); port capacity always exceeds vessel capacity, i.e., $S^{\max}_{j,t} \geq \max\{Q^{vc} : vc \in \mathcal{VC}\}$; and vessels can fully load or discharge in a single period, i.e., $F^{\max}_{j} \geq \max\{Q^{vc} : vc \in$

$\mathcal{VC}\}$. These assumptions allow vessels to load or discharge in the same period in which they leave a port so that loading and discharging decisions do not need to be explicity modeled. Instead, if an inter-regional travel arc or sink arc is taken, we assume that a vessel fully loads or discharges immediately before traveling.

It is important to address the question of why a solution with such fine-grained detail and for such a long planning horizon is even needed. The central reason is due to risk and lack of liquidity for certain commodities. Liquefied natural gas (LNG) is a case in point. Historically, LNG has been a highly illiquid commodity. As a consequence, LNG buyers have come to expect specific long-term plans, called "annual delivery plans," that specify exactly when they will be receiving cargoes so that they can plan for their operations based on a contractually bound delivery plan. Essentially, they try to avoid situations where they do not receive a delivery from their contracted seller and where they cannot purchase the required LNG in an illiquid market. That said, in practice, delivery schedules are updated at regular intervals, e.g., monthly, based on how the schedules and market unfold. The buyer and seller typically work together to adjust their schedules based on how the uncertainty reveals itself. Even after negotiations occur, an updated annual delivery plan with the same granularity of detail must be generated.

Although there is uncertainty over such a long time horizon, in this chapter we study a deterministic version of the problem. Such a model might arise if one considers a restricted set of solutions, obtained by using conservative inventory bounds at ports (see Section 2.3.2.3 of Chapter 2) or pessimistic travel times between ports. Alternatively, one could view a deterministic instance as a single scenario in a stochastic programming framework. Either way, obtaining good solutions quickly to deterministic problems remains a challenge.

The primary contribution of this chapter is the development of an ADP algorithm for long-horizon maritime inventory routing planning problems. There are two main reasons why we chose to explore an ADP framework. First, ADP has a proven track record of generating high-quality solutions to dynamic resource allocation problems, of which dynamic fleet management is a special case [102]. Second, it has the ability to accomodate stochasticity without drastic changes to the framework or implementation. Despite the fact that we only consider deterministic problems in this chapter, being able to adapt the framework developed here to stochastic variants of the underlying deterministic problem are of great interest.

4.2 Literature Review

In this section, we briefly survey some of the relevant research on maritime transportation and ADP methods.

4.2.1 Maritime Applications

From an application perspective, this chapter is most concerned with inventory routing problems arising in the LNG industry, which are known as LNG-IRPs. Recall from Chapter 2 that a MIRP can be defined as "a planning problem where an actor has the responsibility for both the inventory management at one or both ends of the maritime transportation legs, and for the ships' routing and scheduling" [29]. Using this definition, previous approaches applied to LNG-IRPs can be divided into two groups based on whether the actor has control of both the production and consumption ports, or just one of the two. Rakke et al. [80, 81], Stålhane et al. [94], and Halvorsen-Weare and Fagerholt [53] treat the case when the actor only has control of production by attempting to generate annual delivery plans for the world's largest LNG producer. The producer has to fulfill a set of long-term customer contracts. Each contract either outlines monthly demands, or states that a certain amount of LNG is to be delivered fairly evenly spread throughout the year to a given consumption port. Over- and under-deliveries are accepted, but incur a penalty. In contrast, there are also LNG-IRPs that arise for vertically integrated companies who have control of both the production and consumption side of the supply chain [40, 47–49]. In some applications, the opportunity to sell LNG in the spot market using short-term contracts is also present.

Several solution methods for the case when the actor only has control of production have been investigated. Rakke et al. [81] propose a rolling horizon heuristic in which a sequence of overlapping MIP subproblems are solved. Each subproblem involves at most 3 months of data and consists of a one-month "central period" and a "forecasting period" of at most two months. Once a best solution is found (either by optimality or within a time limit), all decisions variables in the central period are fixed at their respective values and the process "rolls forward" to the next subproblem. Stålhane et al. [94] propose a construction and improvement heuristic that creates scheduled voyages based on the availability of vessels and product while keeping inventory feasible. Halvorsen-Weare and Fagerholt [53] study a simplied version of the LNG-IRP problem where cargoes for each long-term contract are pre-generated with defined time windows, and the fleet of ships can be divided into disjoint groups. The problem is decomposed into a routing subproblem and a scheduling master problem where berth, inventory and scheduling decisions are handled in the master problem, while routing decisions are dealt with in the subproblem. Unlike branch-and-price, the subproblems are solved only once. Most recently, Rakke et al. [80] developed a branch-price-and-cut approach that relies on delivery patterns at the customers.

Solution techniques for the case of a vertically integrated company were presented in Chapter 2 as this setting falls under the umbrella of the Core Model (2), which considers MIRPs with inventory tracking at every port. Grønhaug et al. [49] introduce a branch-and-price method in which the master problem handles the inventory management and the port capacity constraints, while the subproblems generate the ship route columns. Fodstad et al. [40] solve a MIP directly while Uggen et al. [104] present a fix-and-relax heuristic. Goel et al. [47] present a simple construction heuristic and adapt the local search procedure of Song and Furman [92] to generate solutions to instances with 365 time periods. Their model seeks to minimize penalities and does not consider travel costs.

4.2.2 Approximate Dynamic Programming

Over the past few decades, approximate dynamic programming has emerged as a powerful tool for certain classes of multistage stochastic dynamic problems. The monographs by Bertsekas and Tsitsiklis [18] and Sutton and Barto [98] provide an introduction and solid foundation to this field. At the same time, they are mainly directed at researchers in computer science and electrical engineering, they use a language more commonly found in control theory and artificial intelligence, and they often make implicit assumptions that make it difficult to transfer the techniques to certain problem classes within operations research. It was only in the last decade or so that ADP was successfully applied to truly large-scale applications arising in the transportation and logistics community. Powell [78] and his associates are largely responsible for this achievement. Despite its accomplishments and continued growth, all of the aforementioned authors affirm that successful implementations of ADP methods still require considerable intuition into the structure of a problem.

Our work builds on the ideas presented by Powell and his associates in the context of stochastic dynamic resource allocation problems. These problems involve the assignment of a set of reusable resources to tasks that occur over time. The arrival process of the tasks is known only through a probability distribution. The assignment of a resource to a task produces a reward, removes the task from the system, and modifies the state of the resource. Often, different types of resources can be used to cover a task, and covering a task with different types of resources may yield different rewards. Dynamic fleet management problems are a special case in this problem class. When modeled as MIPs, these problems take place on a time-space network involving location-time pairs. Service requests (demands for service) from location i to location j appear over time (randomly, in the stochastic setting) and profit is earned by assigning vehicles of different types to fulfill these service requests. Myopically choosing the vehicle type that maximizes the immediate profit is often not best over a longer horizon. Empty repositioning is also a key issue.

Our point of departure is the class of the dynamic fleet management problems studied in [45, 46, 100–102]. In Godfrey and Powell [45], a stochastic dynamic fleet management problem is studied in which requests for vehicles to move items from one location to another occur randomly over time and expire after a certain number of periods. Once a vehicle arrives at its destination node (location-time pair), it is available for servicing another request or for traveling empty to a new location. A single vehicle type with single-period travel times is considered and an ADP algorithm in which a separable piecewise linear concave value function approximation is shown to yield strong performance. This work is extended in [46] to handle multi-period travel times between locations. Further extensions are made to allow for deterministic multi-period travel times with multiple vehicle types [102], random travel times with a single vehicle type [100], and random travel times with multiple types [101]. In all of these studies, separable piecewise linear concave value function approximations are used and shown to work well.

There are two important observations to make regarding the above papers. First, they all treat dynamic fleet management problems, not inventory routing problems. That is, the movement of vehicles is critical, while the amount of a product on the vehicles or at each location is not an issue and, therefore, is not modeled. Second, they all use value function approximations that are only a function of the *vehicle state*. That is, they value the number of each vehicle type that will be available at each location over future time periods. In contrast, Toriello et al. [103] use value function approximations that are a function of the *inventory state* at each location in order to address a deterministic inventory routing problem with a planning horizon of 60 periods. Their problem involves a homogeneous fleet of vehicles that transport a single product between a single loading region and a single discharging region. Each region may have multiple ports. They assume that (1) the inter-regional travel time is a constant regardless of which location is last visited in the loading region and which location is the first visited in the discharging region, and that (2) all locations visited in a region by the same vehicle are visited in the same time period. With these assumptions, the problem reduces to an inventory routing problem with single-period travel times. In addition, after traveling from the loading region to the discharging region, vehicles exit the system as they are assumed to behave like voyage chartered vessels as in [36, 41, 57, 92]. They employ separable piecewise linear concave value functions of the inventory to generate high-quality solutions much faster than solving a large MIP model with a commercial solver.

In this chapter, we extend the ideas above by considering a deterministic inventory routing problem with multiple discharging regions and multi-period travel times. One-way travel times range between 5 and 37 periods. Like Toriello et al. [103], we employ value function approximations that are only a function of the inventory state. However, the presence of multiple discharging regions, multi-period travel times, and longer time horizons makes our problem arguably more complex. This chapter is organized as follows. In Section 4.3, we present a mixed-integer linear programming formulation and a dynamic programming formulation of our problem. In Section 4.4, we provide our solution methodology using an ADP framework. Finally, computational results illustrate the effectiveness of our ADP approach in Section 4.5.

Assumptions: For ease of reference, we collect the assumptions made throughout this chapter: (1) there is exactly one port within each region; (2) port capacity always exceeds the capacity on vessels, e.g., $S_{j,t}^{\max} \geq \max\{Q^{vc} : vc \in \mathcal{VC}\}$; and (3) vessels can fully load or discharge in a single period, e.g. $F_j^{\max} \geq \max\{Q^{vc} : vc \in \mathcal{VC}\}$. (4) Production and consumption rates are known, e.g., $D_{j,t}^{\min} = D_{j,t}^{\max} = d_{j,t}$, (5) Revenues are not considered, e.g., $R_{j,t} = 0$, for all j and t, leaving us with a cost minimization problem. (6) There is a single loading port as is typically the case for LNG-IRPs problems [47, 53, 80, 94]

4.3 Formulations

In this section, we present a mixed-integer linear programming formulation as well as a dynamic programming formulation of the problem. At all times, we try to use notation consistent with that of Chapter 2. As before, we have a time-expanded network where nodes represent port-time pairs. Arcs represent the flow of vessel classes from one node to another, rather than the flow of each individual vessel. Travel times between ports are fixed. Let $\mathcal{FS}_n^{vc,\text{inter}}$ denote the set of all outgoing inter-regional travel arcs and sink arcs from node $n \in \mathcal{N}$ associated with vessel class $vc \in \mathcal{VC}$.

4.3.1 Arc-Based Mixed-Integer Linear Programming Model

We consider the following MIP model:

LNG MIP Model

$$\max \quad \sum_{vc\in\mathcal{VC}}\sum_{a\in\mathcal{A}^{vc}} -C_a^{vc}x_a^{vc} + \sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}} -P_{j,t}\alpha_{j,t} \tag{38a}$$

$$\text{s.t.} \quad \sum_{a\in\mathcal{FS}_n^{vc}} x_a^{vc} - \sum_{a\in\mathcal{RS}_n^{vc}} x_a^{vc} = \begin{cases} +1 & \text{if } n = n_s \\ -1 & \text{if } n = n_t \\ 0 & \text{if } n \in \mathcal{N} \end{cases} , \qquad \forall\ n \in \mathcal{N}_{s,t}, \forall\ vc \in \mathcal{VC} \tag{38b}$$

$$s_{j,t} = s_{j,t-1} + \Delta_j \left(d_{j,t} - \sum_{vc\in\mathcal{VC}} \sum_{a\in\mathcal{FS}_n^{vc,\text{inter}}} Q^{vc}x_a^{vc} - \alpha_{j,t} \right) , \quad \forall\ n = (j,t) \in \mathcal{N} \tag{38c}$$

$$\sum_{vc\in\mathcal{VC}} \sum_{a\in\mathcal{FS}_n^{vc,\text{inter}}} x_a^{vc} \le B_j , \qquad \forall\ n = (j,t) \in \mathcal{N} \tag{38d}$$

$$\alpha_{j,t} \ge 0 , \qquad \forall\ n = (j,t) \in \mathcal{N} \tag{38e}$$

$$s_{j,t} \in [0, S_{j,t}^{\max}] , \qquad \forall\ n = (j,t) \in \mathcal{N} \tag{38f}$$

$$x_a^{vc} \in \mathbb{Z}_+ , \qquad \forall\ vc \in \mathcal{VC} , \forall\ a \in \mathcal{A}^{vc} \tag{38g}$$

The objective is to minimize the sum of all transportation costs and penalties for lost production and stockout. Constraints (38b) require flow balance of vessels within each vessel class. Constraints (38c) are inventory balance constraints at loading and discharging ports, respectively. Berth limit constraints (38d) restrict the number of vessels that can attempt to load/discharge at a port at a given time. This formulation requires that a vessel must travel at capacity from a loading region to a discharging region and empty from a discharging region to a loading region. In contrast to the Core Model (2), the LNG MIP Model (38) does not require decision variables for tracking inventory on vessels (vessel classes), nor does it include decisions variables for the quantity loaded/discharged in a given period.

In order for the LNG MIP Model (38) to furnish the correct lost production and stockout values, the penalty parameters $P_{j,t}$ must be monotonically decreasing in time, i.e., $P_{j,t} > P_{j,t+1}$. This ensures that a solution will not involve lost production (stockout) until the inventory level reaches capacity (falls to zero).

This model is similar to the one studied in Goel et al. [47]. The major differences are that they do not include travel costs in the objective function; they model each vessel individually (in other words, there is only one vessel per vessel class); they model consumption rates as decision variables with upper and lower bounds; and they include an additional set of continuous decision variables to account for cumulative unmet demand at each consumption port.

4.3.2 Dynamic Programming Formulation

We now formulate our MIRP as a finite-horizon dynamic programming problem. It is convenient to interpret this DP formulation as a sequence of dispatching problems. At each point in time, a regional manager has a set of vessels available for dispatching in his region. If enough inventory is available for a vessel to fully load or enough excess capacity is available for a vessel to fully discharge, then the manager faces three options for each available vessel: send the vessel to another region, have the vessel remain in the region, or force the vessel to exit the system.

With this interpretation in mind, we now describe the DP formulation. The state of the system at time t is given by the vector tuple $(\mathbf{r}_t, \mathbf{s}_t)$ where

$\mathbf{r}_t$	$=$	$\{r^{vc}_{j,u,t} : j \in \mathcal{J}, u = t, \ldots, T, vc \in \mathcal{VC}\}$
$\mathbf{s}_t$	$=$	$\{s_{j,u,t} : j \in \mathcal{J}, u = t, \ldots, T\}$
$r^{vc}_{j,u,t}$	$=$	Just before making decisions in time period t (i.e., in the time t subproblem), the number of vessels in vessel class vc that are or will be available for service at location j in the beginning of time period u when decisions are made in time period u ($u \geq t$)
$s_{j,u,t}$	$=$	The number of units of inventory "available" at location j at the end of time period u, after making and executing all decisions in the time t subproblem.

Here, "available" inventory refers to inventory that is either in storage at the port (i.e., has already been discharged) or is on vessels that are at the port but have yet to discharge. The initial state of the system, i.e., inventories and vessel positions, is given. Let $s_{j,t-1,t}$ denote the initial inventory available at port j in the beginning of time period t prior to any events (e.g., decisions, deliveries, consumptions, etc.) taking place.

Given a time period t and the state of the system, we have restrictions on the number and weighted combination of vessels that may leave a port in a given time period:

$$\sum_{vc \in \mathcal{VC}} \sum_{a \in \mathcal{FS}_n^{vc,\text{inter}}} x_a^{vc} \leq B_j \ , \qquad \forall\ n = (j,t) \in \mathcal{N} \tag{39a}$$

$$\sum_{vc \in \mathcal{VC}} \sum_{a \in \mathcal{FS}_n^{vc,\text{inter}}} Q^{vc} x_a^{vc} \leq \begin{cases} s_{j,t-1,t} + d_{j,t} & \text{if } j \in \mathcal{J}^P \\ S_{j,t}^{\max} - s_{j,t-1,t} - d_{j,t} & \text{if } j \in \mathcal{J}^C \end{cases} , \ \forall\ n = (j,t) \in \mathcal{N} \ . \tag{39b}$$

Constraints (39a) are berth limit restrictions (identical to Constraints (38d)) and limit the number of vessels that may take an inter-regional or sink arc in time period t. Constraints (39b) ensure that the maximum amount of inventory that can be loaded (discharged) onto all vessels leaving a port does not exceed the amount of available inventory (remaining capacity) at that port.

Next, we have to model the dynamics of the system, i.e., the transition of vessels and inventory over time. To model the flow of vessels, we have the following requirements:

$$\sum_{a \in \mathcal{FS}_n^{vc}} x_a^{vc} = r_{j,t,t}^{vc} \ , \qquad \forall\ n = (j,t) \in \mathcal{N}, \forall\ vc \in \mathcal{VC} \tag{40a}$$

$$r_{j,u,t+1}^{vc} - \sum_{a=((i,t),n) \in \mathcal{RS}_n^{vc}} x_a^{vc} = r_{j,u,t}^{vc} \ , \qquad \forall\ n = (j,u) \in \mathcal{N} : u > t, \forall\ vc \in \mathcal{VC} \ . \tag{40b}$$

Equations (40a) state that all vessels available at time t must transition by remaining at the same port, moving to another port, or exiting the system. Equations (40b) keep track of the number of vessels in each vessel class that will become available in some future time period $u > t$. Inventory at ports is updated according to the equations

$$s_{j,u,t} = \begin{cases} s_{j,u-1,t} + d_{j,u} - \alpha_{j,u} - q_{j,u}^{\text{out}} & \text{if } j \in \mathcal{J}^P \\ s_{j,u-1,t} - d_{j,u} + \alpha_{j,u} + q_{j,u}^{\text{in}} & \text{if } j \in \mathcal{J}^C \end{cases} , \ \forall\ n = (j,t) \in \mathcal{N}, \ \forall\ u \geq t \tag{41a}$$

where $q_{j,u}^{\text{in}}$ and $q_{j,u}^{\text{out}}$ represent the quantity of inventory incoming to and outgoing from port j at time u after decisions in time t have been made. Specifically, define $q_{j,u}^{\text{out}} = \sum_{vc \in \mathcal{VC}} Q^{vc} \left(\sum_{a \in \mathcal{FS}_{(j,u)}^{vc,\text{inter}}} x_a^{vc} \right)$ if $u = t$ and 0 if $u > t$, and $q_{j,u}^{\text{in}} = \sum_{vc \in \mathcal{VC}} Q^{vc} \left(\sum_{a \in \mathcal{XS}} x_a^{vc} + r_{j,u,t}^{vc} \right)$ with $\mathcal{XS} = \mathcal{FS}_{(j,u)}^{vc,\text{inter}}$ if $u = t$ and $\mathcal{XS} = \mathcal{RS}_{(j,u)}^{vc}$ if $u > t$. Lastly, before transitioning from the time t subproblem to the time $t+1$ subproblem, we must initialize $s_{j,t,t+1} = s_{j,t,t}$ for all $j \in \mathcal{J}$.

Using the principle of optimality, we can write our time t optimization problem as

$$V_t(\mathbf{r}_t, \mathbf{s}_{t-1}) = \max \quad \sum_{vc \in \mathcal{VC}} \sum_{j \in \mathcal{J}} \sum_{a \in \mathcal{FS}^{vc}_{(j,t)}} - C^{vc}_a x^{vc}_a - \sum_{j \in \mathcal{J}} P_{j,t} \alpha_{j,t} + V_{t+1}(\mathbf{r}_{t+1}, \mathbf{s}_t) \tag{42a}$$

$$\text{s.t.} \quad (39), (40), (41) \tag{42b}$$

$$\alpha_{j,u} \geq 0 \ , \qquad \forall \ n = (j,u) \in \mathcal{N} : u \geq t \tag{42c}$$

$$s_{j,u,t} \geq 0 \ , \qquad \forall \ n = (j,u) \in \mathcal{N} : u \geq t \tag{42d}$$

$$x^{vc}_a \in \mathbb{Z}_+ \ , \qquad \forall \ vc \in \mathcal{VC} \ , \forall \ a \in \mathcal{A}^{vc} : a = ((\cdot, t), (\cdot, \cdot)) \tag{42e}$$

Note that V_t is a function of $\mathbf{r}_t$ and $\mathbf{s}_{t-1}$, not $\mathbf{s}_t$. This is because we have followed the standard notation in inventory models where a variable s_t denotes the *ending* inventory in time period t. Also note that we only require the inventory variables $s_{j,u,t}$ to be nonnegative and not below port capacity. This is because, according to our definition, $s_{j,u,t}$ represents the amount of inventory in storage or on a vessel at port j in some future time period u, and therefore could easily exceed capacity at a port.

4.4 Solution Methodology

Solving stochastic dynamic programming problems is notoriously challenging due to the curse of dimensionality: As the dimension of the state space grows, the time required to solve the problem exactly grows exponentially quickly. The MIRP studied here is no exception. Attempting to solve Bellman's equation (42) exactly is futile. Instead, we try to solve it approximately using ADP methods.

We accomplish this by replacing the future value function V_{t+1} with a suitable approximation $\hat{V}_{t+1}$ and solve the approximate problem

$$\tilde{V}_t(\mathbf{r}_t, \mathbf{s}_{t-1}) = \max \quad \sum_{vc \in \mathcal{VC}} \sum_{j \in \mathcal{J}} \sum_{a \in \mathcal{FS}^{vc}_{(j,t)}} -C^{vc}_a x^{vc}_a - \sum_{j \in \mathcal{J}} P_{j,t} \alpha_{j,t} + \hat{V}_{t+1}(\mathbf{r}_{t+1}, \mathbf{s}_t) \tag{43a}$$

$$\text{s.t.} \quad (42b) - (42e) \tag{43b}$$

With an approximate value function in place, we now have the main ingredients for an ADP algorithm. The only missing piece is to describe how the value function is updated in each iteration. This is discussed below.

Pseudocode of our approach is shown in Algorithm 4.4. The most common ADP methods step forward in time. The decisions made in the time t subproblem are guided by the current value function approximation, as shown in Step 5. After a solution to the time t subproblem is obtained, we typically collect some sort of marginal information to determine what the marginal benefit would be from having an additional vessel or an addition unit of inventory available at a given port and future time. Next, we update the state of the system. Once all subproblems have been solved, we update the value function approximations using information obtained from the current solution and from each of the subproblems.

Algorithm 4 `Basic Deterministic ADP Algorithm`

```
 1: Initialization: Choose an approximation $\hat{V}_t$ for all $t \in \mathcal{T}$.
 2: for $n = 1$ to $N$ do
 3:     Initialize the state of the system $(\mathbf{r}_1, \mathbf{s}_0)$.
 4:     for $t = 1$ to $T$ do
 5:         Solve the time $t$ subproblem
            $$\max \sum_{vc \in \mathcal{VC}} \sum_{j \in \mathcal{J}} \sum_{a \in \mathcal{FS}^{vc}_{(j,t)}} -C^{vc}_a x^{vc}_a - \sum_{j \in \mathcal{J}} P_{j,t}\alpha_{j,t} + \hat{V}_{t+1}(\mathbf{r}_{t+1}, \mathbf{s}_t) .$$
 6:         Obtain marginal value information.
 7:         Update the state of the system.
 8:     end for
 9:     Update the value function approximation: $\hat{V}_t \leftarrow \text{Update}(\hat{V}_t, \mathbf{r}_t, \mathbf{s}_t, \pi_t)$ for all $t \in \mathcal{T}$.
10: end for
11: return The best solution found and its corresponding value function approximations.
```

In the next subsection, we discuss our value function approximations. After which, we discuss our updating procedure.

4.4.1 Value Function Approximations

For dynamic resource allocation maximization problems, separable piecewise linear concave value function approximations have enjoyed much success. Toriello et al. [103] note that piecewise linear concave functions are appropriate for several reasons. From a modeling viewpoint, they can easily be embedded into a MIP (when solved as a maximization problem). From a practical perspective, concavity captures the diminishing returns one expects to gain from future inventories. Finally, from a theoretical perspective, they are the "closest" continuous functions to true MIP value functions, which are known to be piecewise linear, superadditive, and upper semi-continuous, but possibly discontinuous [20, 21]. Separability in space/location is also quite natural for problems in which vehicles always fully load and fully discharge at a single location [101, 102]. Meanwhile, separability in time is less understood, but has proven to be effective in a stream of research paper for dynamic fleet management applications [46, 88, 101, 102].

In this work, we also use a value function approximation that is a separable piecewise linear concave function. Specifically, we replace $V_t(\mathbf{r}_t, \mathbf{s}_{t-1})$ with

$$\hat{V}_t(\mathbf{r}_t, \mathbf{s}_{t-1}) = \sum_{j \in \mathcal{J}} \sum_{u \geq t} \hat{V}_{j,u,t}(s_{j,u,t-1}) \ ,$$

where $\hat{V}_{j,u,t}$ is a univariate piecewise linear concave function defined by a sequence of decreasing slopes $\hat{v}^k_{j,u,t}$ and integral breakpoints $\beta^k_{j,u,t}$. Note that this approximation ignores the number of vessels that will be available in the future. Although this might at first seem like a significant amount of information is not being used, in fact, it is not the case. Since vessels always fully discharge, knowing the future amount of available inventory $s_{j,u,t}$ at a discharging port is more useful than knowing the number of vessels in each vessel class that will make the delivery. On the other hand, some information is lost at the loading port.

With this approximation, the term $\hat{V}_t(\mathbf{r}_t, \mathbf{s}_{t-1})$ in Equation (43) becomes

$$\sum_{j \in \mathcal{J}} \sum_{u \geq t} \sum_{k \in \mathcal{K}_{j,u,t}} \hat{v}^k_{j,u,t} w^k_{j,u,t} \ ,$$

where $\mathcal{K}_{j,u,t} = \{1, \ldots, K_{j,u,t}\}$ is an index set of slopes and $w^k_{j,u,t}$ are continuous decision variables satisfying

$$\begin{aligned} s_{j,u,t} &= \sum_{k \in \mathcal{K}_{j,u,t}} w^k_{j,u,t} \\ 0 \leq w^k_{j,u,t} &\leq \beta^k_{j,u,t} \qquad \forall\, k \in \mathcal{K}_{j,u,t} \ . \end{aligned}$$

As a final approximation, rather than consider the value of all future inventories after time period t, we limit ourselves to a shorter time horizon based on travel times and so-called capacity-to-rate ratios. In particular, let τ_j be the travel time between the loading port and discharging port j. Let $\text{C2R}_{j,t}$ be the capacity-to-rate ratio at discharging port j beginning in time period t, i.e., the number of periods it will take for port j to run out of inventory when starting full in time period t. Then, in time period t, we only value inventory up to time period $t + u_{j,t}$ where $u_{j,t} = \tau_j + \text{C2R}_{j,t}$. The rationale for this truncation is to avoid giving ports with a high consumption rate an artificially high value. Thus, the term $\hat{V}_t(\mathbf{r}_t, \mathbf{s}_{t-1})$ in Equation (43) becomes

$$\sum_{j \in \mathcal{J}} \sum_{u=t}^{t+u_{j,t}} \sum_{k \in \mathcal{K}_{j,u,t}} \hat{v}^k_{j,u,t} w^k_{j,u,t} \ .$$

Figure 14 helps to illustrate how our approximation is used. Given an available vessel in the loading region (LR) in time period 1, the time 1 subproblem considers the tradeoff between the immediate cost of moving the vessel and the reward associated with satisfying future demands. In this example, discharging region 1 (DR1) has a capacity-to-rate ratio of four periods, whereas that of discharging region 2 (DR2) is three periods. With separable value functions, we sum the value function approximations over future time periods in which a piecewise concave linear function is shown.

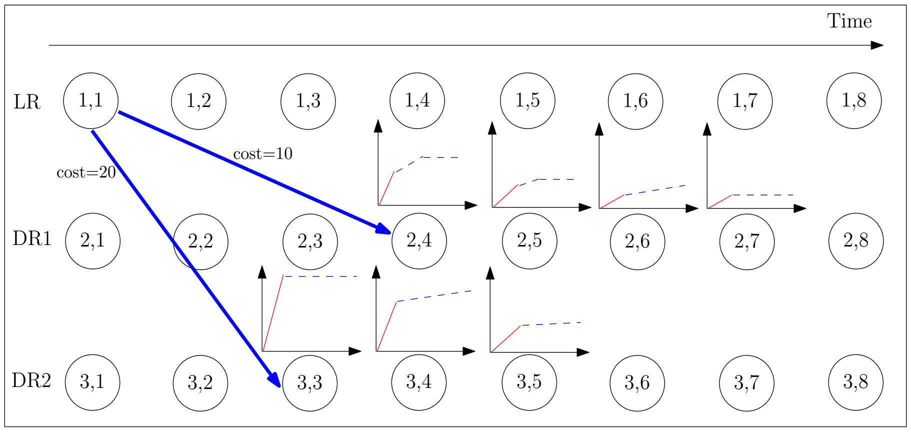


Figure 14: Example using separable piecewise linear concave value function approximations.

It is important to mention that our approach does not allow vessels to leave the system until the very last time period of the horizon and therefore there is no value for this option. Consequently, some needless trips at the end of the horizon may take place. The rationale for removing the option to take a vessel out of service is due to the fact that, in our instances, there is not an overabundance of vessels and so all vessels are continually in operation. Moreover, some might argue that our problem is an infinite horizon problem and should not be truncated. Thus, as a final step in our solution approach, we have a simple routine, which we call “end effect polishing,” to remove these needless trips that are an artifact of the finite horizon.

4.4.2 Updating the Value Function Approximation

Just as there are numerous choices for designing a value function approximation, there are also a number of techniques commonly found for updating the value function approximations (see, e.g., George and Powell [42]). Perhaps, the most important consideration is to determine what the goal of the update is. In early iterations of an ADP algorithm, it is often beneficial to explore the solution space. Thus, it is usually prefered to have a fast update rule that results in substantive changes to the value function. On the other hand, in later iterations, some sort of convergence is often desired, in which case small changes are sought. Regression and batch least-squares are sometimes used [78]. Toriello et al. [103] suggest other fitting procedures.

Our focus is on generating one or more good solutions quickly; convergence is less of a concern. Consequently, we prefer to make rapid changes to the approximation. To this end, we adapt the concave adaptive value estimation (CAVE) algorithm introduced in [45]. This method creates and maintains a univariate piecewise linear concave function for each of the approximations.

We associate a dual variable $\pi_{j,u,t}$ with each inventory balance equation (41). Since our time t subproblem is a MIP, dual variable information is not immediately available. We choose to obtain the value for $\pi_{j,u,t}$ by solving the so-called fixed model associated with the solution of that subproblem. That is, after obtaining a solution to the time t subproblem, we fix all integer decision variables to their optimal values and solve the resulting LP. We use the $\pi_{j,u,t}$ values associated with the LP of the fixed model. An alternative approach is to use the dual values of the root node LP relaxation. One could also spend more effort and write an auxiliary method to compute the true marginal benefits of having an additional unit of inventory.

Given a time period t, the current value function approximation $\hat{V}_t$, the inventory levels $\mathbf{s}_{t-1}$, and dual values π_t, we would like to update the value function locally, while also preserving global concavity. To do this, we take a convex combination of the current slope and an estimate of what the slope should be based on the information collected. Let $\alpha_n \in [0,1]$ be the stepsize parameter in the nth iteration of the ADP algorithm, i.e., the outermost loop in the ADP Algorithm 4.4. Let k be the index of the slope $(\hat{v}^k_{j,u,t})$ that we would like to update given the inventory level $s_{j,u,t-1}$. Then the slope is updated using the rule

$$\hat{v}^k_{j,u,t} = (1-\alpha_n)\hat{v}^k_{j,u,t} + \alpha_n \pi^{\max}_{j,u,t} \ , \qquad \forall \ j,u,t \ (u \geq t) \tag{44}$$

where

$$\pi^{\max}_{j,u,t} = \max_{s=t,\ldots,u} \{\pi_{j,u,s}\} \ , \qquad \forall \ j,u,t \ (u \geq t) \ .$$

Using the parameter $\pi^{\max}_{j,u,t}$ in the updating step, as opposed to $\pi_{j,u,t}$, was shown to be effective in [46] and [102]. Essentially, the parameter $\pi^{\max}_{j,u,t}$ attempts to estimate the marginal value of having an additional unit of inventory at port j in the future time period u (when solving the time t subproblem) by taking maximum dual value at (j,u) computed over the subproblems in time periods t, $t+1$, …, u.

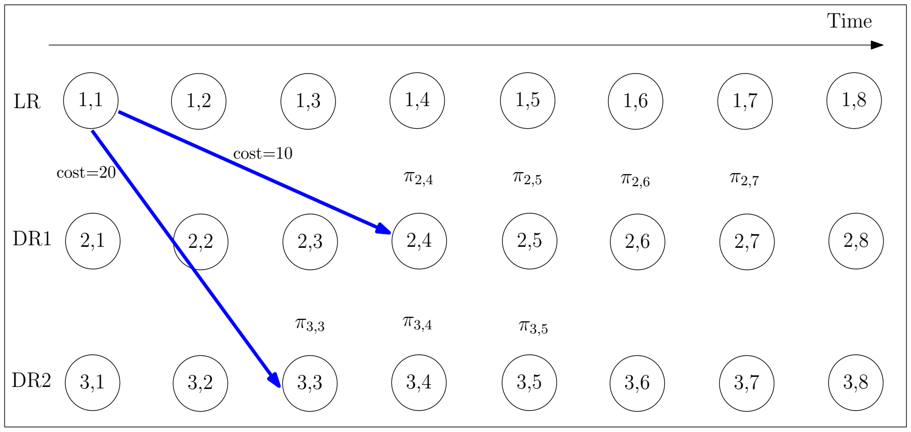


Figure 15: Example showing the collection of dual information in future time periods.

4.5 Computational Experiments

In this section, we compare the performance of our ADP method with that of the commericial MIP solver Gurobi 5.0 solving the LNG MIP Model (38) over shorter and shorter planning horizons. In all experiments, we set Gurobi's MIPFocus parameter to 1 to emphasize feasibility so that more time is spent trying to find good feasible solutions. All models and algorithms were coded in Python. All experiments were carried out on a Linux machine with kernel 2.6.18 running a 64-bit x86 processor equipped with two 2.27 GHz Intel Xeon E5520 chips and 32GB of RAM.

Four methods are compared. Method M0 refers to Gurobi 5.0 with emphasis on feasibility. Methods M1, M2, and M3 are ADP methods with different rules for updating the value function approximations. All ADP algorithms were run for 50 iterations. All value function approximations are initialized with zero slopes. The different stepsizes α_n that are used in Equation (44) for updating slopes are shown in Table 12.

Method M1 uses a simple harmonic stepsize rule to ensure that the algorithm will converge to a set of value function approximations if the total number of iterations were not limited. Methods M2 and M3 are meant to be more aggressive in searching for good solutions. Let $rgap_n$ denote the relative gap computed on the nth iteration of ADP algorithm. That is, $rgap_n = \min\{(z_n - z_{Best})/z_n, 1\}$ where z_{Best} is the best known objective function value and z_n is the objective function value found on the nth iteration of the algorithm. The basic idea behind the stepsize update for methods M2 and M3 is that, after a certain number of iterations (here, 25), we would like to re-invigorate the search for a better set of value function approximations. Thus, if the objective function value z_n of the solution found in iteration n is poor, we expect the relative gap $rgap_n$ to be closer to 1 and so the stepsize chosen will be close to $1/C$, leading to a more drastic change in the value function approximations. On the other hand, if the relative gap $rgap_n$ is small, we would like the stepsize to be closer to $1/(C+n)$ so that the value function updates are modest. Finally, it should be noted that we tried other constants, but these parameter settings convey the most important observations.

Table 12: Stepsizes used for ADP variants

Method	**Stepsize**	**Constant**
M1:	$\alpha_n = [C_1 + n]^{-1}$	$C_1 = 10$
M2:	$\alpha_n = \begin{cases} [C_2 + n]^{-1} & \text{if } n \leq 25 \\ [C_2 + (1 - rgap_n)n]^{-1} & \text{o.w.} \end{cases}$	$C_2 = 8$
M3:	$\alpha_n = \begin{cases} [C_3 + n]^{-1} & \text{if } n \leq 25 \\ [C_3 + (1 - rgap_n)n]^{-1} & \text{o.w.} \end{cases}$	$C_3 = 5$

4.5.1 Instances with a 180-period horizon

Our first experiment considers instances with a 180-period horizon. In practice, it is doubtful that one would solve the LNG MIP Model (38) directly when the planning horizon is so large. However, it is worth exploring the differences in the two approaches. Figures 16 through 18 compares the objective function value of the incumbent as a function of CPU time for three instances: `LR1_1_DR5_11111_VC5_V25b`, `LR1_1_DR8_11111111_VC5_V40b`, and `LR1_1_DR12_111111111111_VC5_V70b`. The last instance has 12 discharging regions (`DR12`), 5 vessel classes (`VC5`), and 70 vessels (`V70`).

The first observation from these figures is that our ADP method is capable of generating good solutions quickly. As the number of vessels and discharging regions grow, the time it takes Gurobi to find a solution of comparable quality increases. The second observation is that using a smaller constant C in the stepsize update appears to drive the objective function value down faster.

Table 13 shows the additional time required for Gurobi to find a solution whose objective function value equals or is superior to the that of the best ADP variant for a larger set of instances. A '>36000' means that Gurobi could not find a better solution within a 10-hour time limit. We see that an increase in the number of discharging ports and vessels typically results in more time for Gurobi to achieve equal or better performance. Finally, it is worth mentioning that lower bounds for these problems are usually poor and so it is difficult to say anything conclusive about optimality gaps.

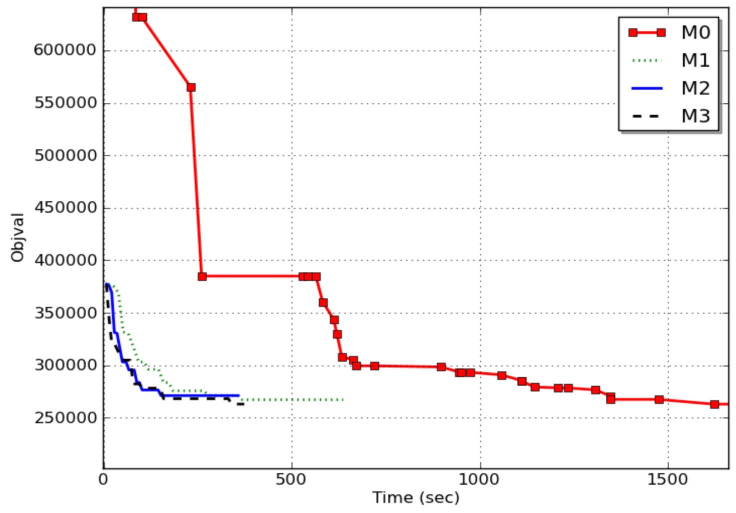


Figure 16: Comparison of solution times for an instance with 180 periods, 5 discharging regions, 5 vessel classes, and 25 vessels.

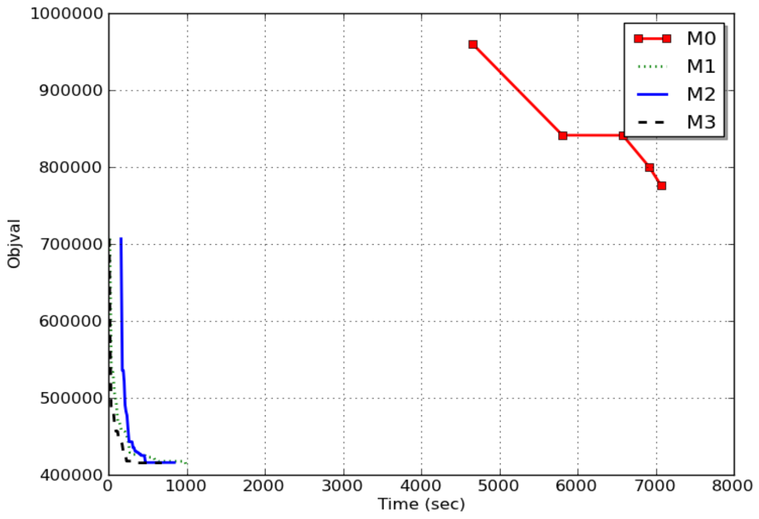


Figure 17: Comparison of solution times for an instance with 180 periods, 8 discharging regions, 5 vessel classes, and 40 vessels.

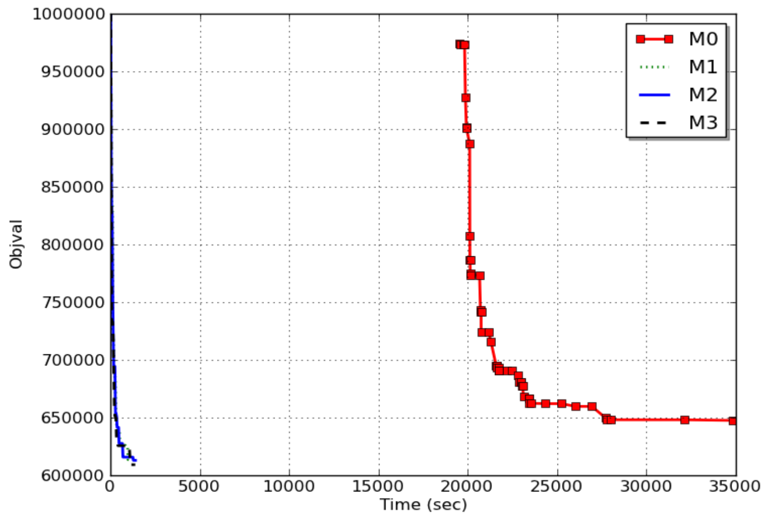


Figure 18: Comparison of solution times for an instance with 180 periods, 12 discharging regions, 5 vessel classes, and 70 vessels.

Table 13: 180-period instances: Additional time (sec) required by Gurobi to reach a solution of equal or better quality

		ADP	**GRB**
Instance	**Objval**	**Time to Best**	**Additional Time**
`LR1_1_DR3_111_VC3_V10b`	139815	40	>36000
`LR1_1_DR3_111_VC3_V13b`	176976	139	2771
`LR1_1_DR3_111_VC3_V16a`	245376	52	1251
`LR1_1_DR4_1111_VC3_V15a`	129821	145	5013
`LR1_1_DR4_1111_VC3_V15b`	200344	211	>36000
`LR1_1_DR4_1111_VC5_V17a`	121802	58	>36000
`LR1_1_DR4_1111_VC5_V17b`	177488	111	>36000
`LR1_1_DR5_11111_VC5_V25a`	174167	290	>36000
`LR1_1_DR5_11111_VC5_V25b`	263088	339	1283
`LR1_1_DR8_11111111_VC5_V38a`	482365	433	497
`LR1_1_DR8_11111111_VC5_V40a`	322494	202	>36000
`LR1_1_DR8_11111111_VC5_V40b`	415464	346	30836
`LR1_1_DR12_111111111111_VC5_V70a`	543483	975	>36000
`LR1_1_DR12_111111111111_VC5_V70b`	609194	1112	>36000

4.5.2 Instances with a 120-period horizon

In our second experiment, we test our ADP method on instances with a 120-period horizon in order understand if could be competitive with a rolling horizon framework. Recall that in a rolling horizon framework, a sequence of small MIPs with overlapping time intervals are solved to generate a solution over the entire planning horizon. For example, to generate solutions to planning problems with a 360-period horizon, Rakke et al. [81] solve subproblems involving 90 periods and piece together the solutions to these subproblems to create a solution for the full planning horizon. For several of our instances, one-way inter-regional travel times are over 30 periods in duration and we found that solving a reduced MIP with a 90-period time horizon could lead to solutions with odd end behavior. Extending these horizons over 120 periods seemed to yield more stable results.

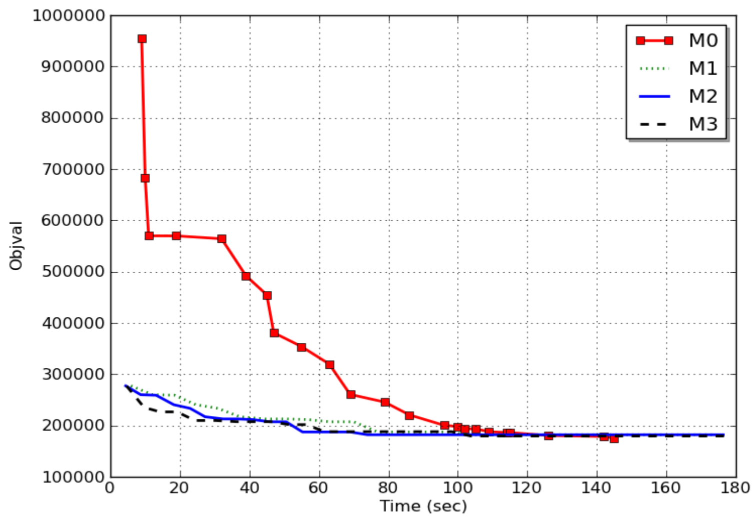


Figure 19: Comparison of solution times for an instance with 120 periods, 5 discharging regions, 5 vessel classes, and 25 vessels.

In this experiment, we compare our ADP methods with Gurobi 5.0 emphasizing feasibility on instances with 120-period planning horizons. The results indicate that our methods are still capable of generating good solutions quickly. It should be noted that in a rolling horizon framework, the solution process for each subsequent subproblem can be warm-started using the current solution. Indeed, we could warm-start the solution process using

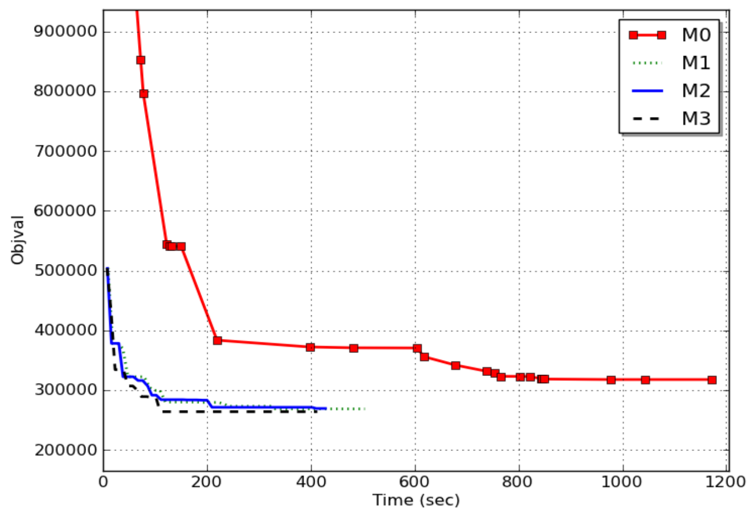


Figure 20: Comparison of solution times for an instance with 120 periods, 8 discharging regions, 5 vessel classes, and 40 vessels.

the best known value function approximations.

Table 14 shows the time required for our ADP algorithm to find its best solution and the additional time that Gurobi needed to find a solution of equal or better quality. As

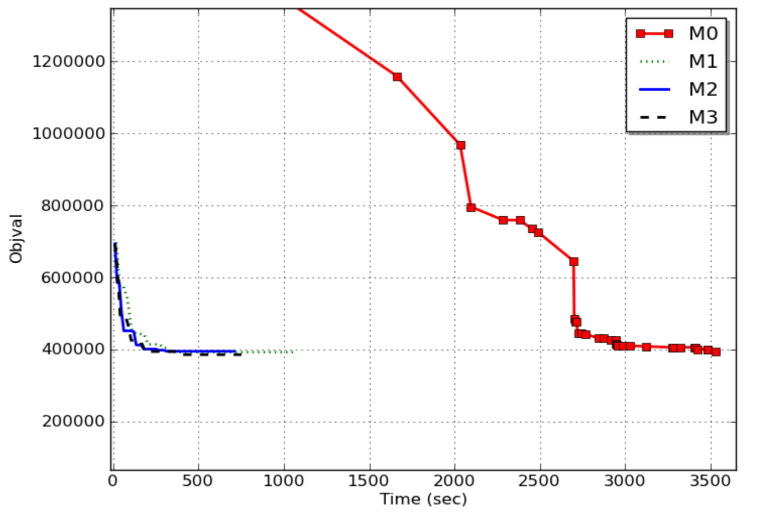


Figure 21: Comparison of solution times for an instance with 120 periods, 12 discharging regions, 5 vessel classes, and 70 vessels.

before, a '>36000' means that Gurobi could not find a better solution within a 10-hour time limit. Compared to the 180-period instances, we see that Gurobi is able to find solutions of equal or better quality to more instances with the 10-hour time limit. On one instance, it was able to find a better solution than any of our ADP methods and in less time.

Table 14: 120-period instances: Additional time (sec) required by Gurobi to reach a solution of equal or better quality

		ADP	GRB
Instance	**Objval**	**Time to Best**	**Additional Time**
LR1_1_DR3_111_VC3_V10b	112827	12	674
LR1_1_DR3_111_VC3_V13b	134247	106	1753
LR1_1_DR3_111_VC3_V16a	142096	72	-56
LR1_1_DR4_1111_VC3_V15a	80468	82	1110
LR1_1_DR4_1111_VC3_V15b	131780	49	>36000
LR1_1_DR4_1111_VC5_V17a	76554	16	938
LR1_1_DR4_1111_VC5_V17b	125626	21	56
LR1_1_DR5_11111_VC5_V25a	107750	79	>36000
LR1_1_DR5_11111_VC5_V25b	179538	103	3
LR1_1_DR8_11111111_VC5_V38a	314924	17	497
LR1_1_DR8_11111111_VC5_V40a	193207	141	>36000
LR1_1_DR8_11111111_VC5_V40b	263791	109	2198
LR1_1_DR12_111111111111_VC5_V70a	303241	138	>36000
LR1_1_DR12_111111111111_VC5_V70b	385661	383	>36000

4.5.3 Instances with a 360-period horizon

As a final experiment, we test our ADP method on instances with a 360-period horizon. We place this experiment last because we do not expect solving the 360-period LNG MIP Model (38) to be useful for these large instances. The solutions to each instance are used to generate the data in Table 5.

Table 15 shows the time required for our ADP algorithm to find its best solution and the additional time that Gurobi needed to find a solution of equal or better quality. A '>86400' means that Gurobi could not find a better solution within a one-day time limit.

As a final comment, we believe that as the number of vessel classes increases, the time it takes to solve the LNG MIP Model (38) should increase more rapidly than that of our ADP algorithm. Since some applications may not allow vessels to be aggregated by vessel

Table 15: 360-period instances: Additional time (sec) required by Gurobi to reach a solution of equal or better quality

Instance	Objval	ADP Time to Best	GRB Additional Time
LR1_1_DR3_111_VC3_V10b	315358	249	>86400
LR1_1_DR3_111_VC3_V13b	364484	93	>86400
LR1_1_DR3_111_VC3_V16a	574617	262	10322
LR1_1_DR4_1111_VC3_V15a	276954	133	39346
LR1_1_DR4_1111_VC3_V15b	380737	320	>86400
LR1_1_DR4_1111_VC5_V17a	259226	113	>86400
LR1_1_DR4_1111_VC5_V17b	365883	313	>86400
LR1_1_DR5_11111_VC5_V25a	404377	83	>86400
LR1_1_DR5_11111_VC5_V25b	532162	424	>86400
LR1_1_DR8_11111111_VC5_V38a	962344	525	>86400
LR1_1_DR8_11111111_VC5_V40a	750681	455	>86400
LR1_1_DR8_11111111_VC5_V40b	904454	1342	>86400
LR1_1_DR12_111111111111_VC5_V70a	1159183	423	>86400
LR1_1_DR12_111111111111_VC5_V70b	1399026	1197	>86400

class, our ADP approach may become more attractive.

4.5.4 Profiling the ADP Algorithm

Table 16 shows the percentage of time our ADP algorithm spends in each of its major functions, averaged over all instances and all time horizons considered. In software engineering, this type of "profiling" is useful as it informs a developer of which functions are consuming the bulk of the CPU effort. Almost 70% of the solution time is spent either initializing or solving all of the MIP subproblems. This large percentage is expected since MIP solving is costly compared to all of the other operations. However, a more intelligent implementation should be able to shrink the percentage of time spent on MIP initialization since this involves nothing more than severals loops. End effect polishing refers to a simple procedure that we apply after obtaining a solution for the full horizon. Since our algorithm does not allow vessels to leave the system, needless trips at the end of the horizon may take place. End effect polishing seeks to remove these needless trips that are an artifact of truncating what some might argue is an infinite horizon problem. In our approach, value

Table 16: Average percentage of time spent in each ADP-related function

Function	% of Time
MIP initialization	21.58
MIP solving	47.51
State updating	17.64
End effect polishing	6.02
Slope updating	0.84
Miscellaneous	6.41

function (or slope) updating requires virtually no time as convex combinations of information are used. This percentage of time would increase if one were to use more sophisticated schemes for updating the value function, e.g., regression.

4.6 Conclusions and Future Work

This chapter introduced an approximate dynamic programming framework for generating good solutions quickly to maritime inventory routing problems with a long planning horizon. The ADP approach appears to be one of the first of its kind in the maritime routing and scheduling domain and represents a significant departure from previous methods for this class of problems. Rather than putting the burden on a MIP solver to produce good solutions or improving solutions, our approach shifts this effort to identifying value function approximations that lead to good solutions. Computational experiments indicate that this framework is capable of obtaining better solutions than a commercial MIP solver tasked with considering many periods simultaneously.

As far as future research directions are concerned, Powell and his associates have laid the groundwork for an algorithmic approach that can incorporate stochastic elements, e.g., stochastic demands, travel times, etc. [78]. Although we have not explored these extensions, the attractive feature of the proposed ADP framework is that it requires minor changes. In particular, it considers sample realizations of the uncertain elements to solve each time t subproblem, and then proceeds as normal. When the value function approximations are updated with a convex combination procedure as we have done, the stepsize may become dependent on the noise in the estimates obtained over the course of the algorithm.

Another interesting experiment would be to assess the benefit from storing value function approximations when the model is re-optimized. For example, within the context of a general decision support tool that is called every month to obtain an updated long-term plan, it seems likely that warm-starting our ADP framework with the best known value function approximations would lead to better solutions faster.

Yet another experiment could explore using our ADP framework as a local search technique for re-optimization. As mentioned in the introduction, building an actual annual delivery plan usually involves several rounds of negotiations between the producer and the consumers. At various points in the negotiation process, one customer may agree to the timing of his deliveries, while another does not. In this case, we could fix the paths associated with a subset of vessels and then attempt to apply our ADP algorithm on a reduced problem.

CHAPTER V

FIXED-CHARGE TRANSPORTATION WITH PRODUCT BLENDING

5.1 Introduction and Problem Statement

In many operational and planning models within the chemical, petroleum, and process industries, a common issue involves blending raw materials with varying attributes and concentration levels into homogeneous intermediate or end products. Blending raw materials affords an organization the opportunity to realize sizable cost savings, while meeting demand for an array of final products and satisfying pre-determined specification requirements for each type of product [82]. The inherent flexibility of the blending process can be exploited to optimize the allocation and transportation of raw materials to production facilities. This motivates the study of what we call the *fixed-charge transportation problem with product blending* (FCTPwB). The feasible region of this problem arises as a substructure within many applications in the petrochemical industry, and potentially in other areas including supply chain management, agriculture, and the energy sector [67].

A general form of the standard fixed-charge transportation problem for a single product can be described as follows [71]. Consider a set of suppliers $S = \{1, \ldots, m\}$ and a set of consumers $C = \{1, \ldots, n\}$. Each supplier $i \in S$ has a minimum and maximum supply of a given product, denoted l_i and u_i, respectively. Similarly, each consumer $j \in C$ has a minimum and maximum demand for the product, denoted l_j and u_j, respectively. Product can be sent from suppliers to consumers on an underlying directed bipartite graph $G = (S \cup C, \mathcal{A})$, where $\mathcal{A}$ is the set of arcs. For each arc $(i, j) \in \mathcal{A}$, let c_{ij} denote the unit revenue for flow shipped from supplier i to consumer j and u_{ij} denote the capacity of flow on arc (i, j). What makes this problem more interesting than the classical transportation problem is the additional assumption that a fixed cost f_{ij} is incurred if arc (i, j) is opened. It is important to emphasize that fixed costs are incurred when arcs are opened as opposed to when suppliers are opened, as would happen in the facility location problem.

The FCTPwB incorporates an additional *proportionality requirement* on the quality of the product. Specifically, let $\tilde{p}_i$ denote the nominal quality (or purity) of product available from supplier $i \in S$ and $\tilde{p}_j^{\min}$ denote the minimum quality required at consumer $j \in C$. Then the additional constraint, which we refer to as a *linear blending constraint*, requires that the average quality of all product received by consumer j must be at least $\tilde{p}_j^{\min}$, where we assume that product received by a consumer can be *blended* together to meet this requirement. A similar constraint could be imposed based on a maximum quality requirement $\tilde{p}_j^{\max}$.

In this variant of the problem we assume that there is a single product as well as a single attribute associated with that product. The blending constraint applies to this single attribute. More generally, there could be multiple products/commodities each with multiple attributes, and consumers could demand different products with varying minimum and maximum quality requirements. In addition, the problem described above consists of a single period in which a product is distributed, but one could envision a multi-period problem in which the supply and demand inventories are affected by exogeneous factors, which is why we have chosen to describe the supply and demand as having to satisfy pre-determined inventory level requirements.

To cast this problem as a mixed-integer program, we introduce continuous decision variables x_{ij} to denote the amount of product sent from supplier i to consumer j and binary decision variables y_{ij} which take value 1 if arc (i, j) is opened and 0 otherwise. Let $p_{ij} := \tilde{p}_i - \tilde{p}_j^{\min}$ be the "purity difference" between supplier i and consumer j, $\forall (i, j) \in \mathcal{A}$.

This yields the arc-based formulation:

$$\text{(FCTPwB)} \qquad \max_{\mathbf{x},\mathbf{y}} \quad \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij} - \sum_{(i,j)\in\mathcal{A}} f_{ij}y_{ij} \tag{45a}$$

$$\text{s.t.} \quad \sum_{i\in S} p_{ij}x_{ij} \geq 0, \qquad \forall j \in C \tag{45b}$$

$$l_i \leq \sum_{j\in C} x_{ij} \leq u_i, \qquad \forall i \in S \tag{45c}$$

$$l_j \leq \sum_{i\in S} x_{ij} \leq u_j, \qquad \forall j \in C \tag{45d}$$

$$0 \leq x_{ij} \leq u_{ij}y_{ij}, \qquad \forall (i,j) \in \mathcal{A} \tag{45e}$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in \mathcal{A} \ . \tag{45f}$$

The objective of this formulation is to maximize profit, defined as the revenue from shipping product from suppliers to consumers minus the fixed cost incurred from opening the arcs on which goods are sent. Constraint (45b) models the linear blending constraint since it is a re-statement of the blending constraint

$$\frac{\sum_{i\in S} \tilde{p}_i x_{ij}}{\sum_{i\in S} x_{ij}} \geq \tilde{p}_j^{\min} \ , \qquad \forall \ j \in C$$

as it would appear in its natural form.

An interesting history of blending in the petroleum industry is given in [35] and [82]. These two works, along with [84], describe successful deployments of decision support systems in which blending is an integral component and underscore the importance of mathematical programming methodologies. In the chemical, petroleum, and wastewater treatment industries, several blending and pooling problems have undergone extensive study. The survey paper by Misener and Floudas [70] discusses five relevant classes of pooling problems.

When formulated as mathematical programs, most practical blending problems are modelled as mixed-integer nonlinear mathematical programming problems (MINLPs). However, because of the difficulty in solving these MINLPs, mixed-integer linear programming (MIP) formulations are commonly used to approximate MINLP formulations [61, 65]. In these MIP models, nonlinearities that arise from blending constraints are linearized (through reformulation) or approximated (sometimes iteratively) [65, 69].

The fixed-charge transportation problem (FCTP) *without* blending has been studied for years, with early work dating back to Balinski [12]. In the standard FCTP, each supplier $i \in S$ has a fixed supply $s_i = l_i = u_i$ and each consumer $j \in C$ has a fixed demand $d_j = l_j = u_j$. This problem is known to be $\mathcal{NP}$-hard. As a consequence, the FCTPwB is $\mathcal{NP}$-hard since if $p_{ij} > 0, \forall (i,j) \in \mathcal{A}$, then the blending constraints (45b) become redundant and the resulting problem is simply the FCTP. By and large, researchers have focused on developing heuristics and exact algorithms for solving the FCTP [4, 15, 44, 58, 62, 63, 73, 97]. More generally, the FCTP is a special case of the fixed-charge network flow problem for which substantial polyhedral theory and numerous algorithms have been developed. Notable inequalities derived from studying the single-node fixed-charge flow model include flow cover cuts [50, 72], flow path cuts [107], and flow pack cuts [9]. These cutting planes are now standard in many commercial MIP solvers. The relation between our facets and flow cover cuts is discussed in Section 5.2.4. We are not aware of any literature in which blending constraints are also considered.

Despite the abundance of research on blending and fixed-charge problems, there is a dearth of literature in which both themes are studied simultaneously from a polyhedral vantage point. In this paper, we strive to fill this void by investigating polyhedral aspects of the uncapacitated FCTPwB in which fixed charges and linear blending constraints are present. Our contributions are a polyhedral analysis of the FCTPwB, including two new families of facet-defining valid inequalities which fully exploit the presence of a linear blending requirement, and computational results that demonstrate the effectiveness of the inequalities. In Section 2, we introduce two exponentially-sized facet classes for the single-consumer uncapacitated FCTPwB polytope and provide intuition for their validity using arguments based on lifting facets of lower-dimensional sets. We also show that these facets can be separated with a low-order polynomial-time separation routine. In Section 3, we prove that in two special cases these facet classes, along with the continuous relaxation of the original formulation constraints, yield the convex hull of the feasible region. These results lend theoretical support to our claim that our two facet classes are strong. In Section 4, computational results are presented to illustrate the effectiveness of our facets at reducing the integrality gap and solution time on instances with multiple consumers and arc capacities. These results also provide empirical support that our separation procedure is extremely fast in practice. Some discussion of the relevance and applicability of these cuts to other models is provided in Section 5.

5.2 An Uncapacitated Single-Consumer Model

In this section, we study polyhedral aspects of an uncapacitated single-consumer model. We begin by collecting several assumptions that we will use throughout the remainder of the paper. We assume that each supplier can send product to a single consumer, that the consumer's (supplier's) lower bound on demand (supply) is 0, and that the consumer's (supplier's) upper bound on demand (supply) is 1, which is without loss of generality since we can scale parameters accordingly. Having unequal lower and upper bounds is not critical, but will permit us to work with a set that is full dimensional. We assume that arc capacities are arbitrarily large. Given that only one consumer is present, we drop the subscript for the consumer. We assume $p_1 > p_2 > \cdots > p_m$ and $p_i \neq 0, \forall i \in S$. This, again, is done for mathematical convenience. In fact, when we return to the multi-consumer case we will continue to assume that $p_{ij} \neq p_{kj}$ and $p_{ij} \neq 0$, $\forall i, k \in S, \forall j \in C$. Let $S^+ = \{1, \cdots, m_+\}$ be the set of *good* suppliers (i.e., suppliers whose purity difference p_i is positive) and analogously define $S^- = \{m_+ + 1, \cdots, m\}$ to be the set of *bad* suppliers. Let $S = S^+ \cup S^-$ be the set of all suppliers. We assume $m_+ = |S^+| \geq 1$ and $m_- = |S^-| \geq 1$.

The feasible region, denoted by X_{m_+,m_-}, of the single-consumer uncapacitated FCTPwB is the set of points $(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^m_+ \times \{0,1\}^m$ satisfying

$$\text{(blending constraint)} \qquad \textstyle\sum_{i\in S^+} q_i x_i - \sum_{k \in S^-} r_k x_k \geq 0 \tag{46a}$$

$$\text{(demand constraint)} \qquad \textstyle\sum_{i \in S} x_i \leq 1 \tag{46b}$$

$$x_i \leq y_i, \forall\, i \in S, \tag{46c}$$

where $q_i = p_i, \forall\, i \in S^+$, $r_k = -p_k, \forall\, k \in S^-$. Note that $q_1 > \cdots > q_{m_+} > 0$ and $0 < r_{m_+ + 1} < \cdots < r_m$. We have introduced the parameters q_i and r_i for convenience so that all coefficients are positive. Our primary goal is to obtain a polyhedral description of the convex hull of X_{m_+,m_-}, denoted by $\text{conv}(X_{m_+,m_-})$.

5.2.1 Extreme Points

We now characterize the extreme points of $\mathrm{conv}(X_{m_+,m_-})$. The intuition behind their structure is simple. The extreme points of the projection of $\mathrm{conv}(X_{m_+,m_-})$ onto the continuous variables correspond to one of the three following cases: (i) the origin, (ii) one good supplier sending one unit of flow to satisfy demand while all other suppliers send nothing, or (iii) one good supplier and one bad supplier each sending product in such a way that both the blending and demand constraints are tight. When we return to the original space $\mathrm{conv}(X_{m_+,m_-})$, we must also consider the y variables.

Proposition 1. *The extreme points of* $\mathrm{conv}(X_{m_+,m_-})$ *are*

$$\left(\mathbf{0}, \sum_{i \in T} \mathbf{e}_i\right), \qquad \forall\, T \subseteq S \tag{47a}$$

$$\left(\mathbf{e}_i, \mathbf{e}_i + \sum_{j \in T} \mathbf{e}_j\right), \qquad \forall\, i \in S^+, \forall\, T \subseteq S \setminus \{i\} \tag{47b}$$

$$\left(\frac{r_k}{q_i + r_k}\mathbf{e}_i + \frac{q_i}{q_i + r_k}\mathbf{e}_k, \mathbf{e}_i + \mathbf{e}_k + \sum_{j \in T} \mathbf{e}_j\right), \qquad \forall\, i \in S^+, k \in S^-, \forall\, T \subseteq S \setminus \{i, k\}, \tag{47c}$$

where $\mathbf{e}_i \in \mathbb{R}^m$ *is the* i*-th unit vector. All nontrivial extreme points of* $\mathrm{conv}(X_{m_+,m_-})$ *have exactly one positive value among the variables* $x_i, i \in S^+$*, and possibly one additional positive value among the variables* $x_k, k \in S^-$*.*

Proof It suffices to prove that the extreme points of $\{\mathbf{x} \in \mathbb{R}^m_+ : (46a);\ (46b)\}$, the continuous projection of $\mathrm{conv}(X_{m_+,m_-})$, have the desired structure. This follows because the set only has two nontrivial constraints (46a) and (46b), and therefore when choosing which constraints to fix at equality at an extreme point, at most two variables (satisfying the specified conditions) will be positive. $\square$

Corollary 1. *The set* $\mathrm{conv}(X_{m_+,m_-})$ *is full-dimensional.*

Corollary 2. $x_i \geq 0$ *and* $y_i \leq 1$ *for all* $i \in S$ *are trivial facets of* $\mathrm{conv}(X_{m_+,m_-})$*.*

Corollary 3. *The blending constraint* $\sum_{i \in S^+} q_i x_i - \sum_{k \in S^-} r_k x_k \geq 0$ *is a facet of* $\mathrm{conv}(X_{m_+,m_-})$*. The inequalities* $\sum_{i \in S} x_i \leq 1$ *and* $x_i \leq y_i$ *for* $i \in S^+$ *are facets of* $\mathrm{conv}(X_{m_+,m_-})$ *when* $m_+ \geq 2$*.*

Proof We can easily pick $2m+1$ affinely independent extreme points for Corollary 1 and $2m$ such points for Corollaries 2 and 3. $\square$

5.2.2 Facets of the Uncapacitated Single-Consumer FCTPwB Polytope

We now state and prove our main result.

Theorem 1. (Facet Class 1: Lifted Blending Facets) *The inequalities*

$$\sum_{i\in T} x_i + \sum_{k\in S^-} \min\left\{1, \frac{r_k}{r_l}\right\} x_k \le \sum_{i\in S^+\setminus T} \left(\frac{q_i}{r_l}\right) x_i + \sum_{i\in T} y_i \ , \qquad \forall\ T \subseteq S^+, \forall\ l \in S^- \ , \tag{48}$$

are valid for $\operatorname{conv}(X_{m_+,m_-})$*. They are facet-defining in all cases except when (a)* $T=\emptyset$ *and* $l<m$*, or (b)* $T=S^+$ *and* $l>m_+ + 1$.

Theorem 2. (Facet Class 2: Lifted Variable Upper Bound Facets) *Let* $S_j^+ = \{1,\ldots,j\}$ *for* $j \in S^+ \cup \{0\}$*, with* $S_0^+ = \emptyset$*. Let* $S_l^- = \{m_+ + 1, \ldots, l\}$ *for* $l \in S^- \cup \{m_+\}$*, with* $S_{m_+}^- = \emptyset$*. The inequalities*

$$\sum_{i\in T^+} \frac{r_l(q_i-q_j)}{q_i} x_i + \sum_{k\in T^-\cup\{l\}} (q_j + r_k) x_k \le$$

$$\sum_{k\in T^-\cup\{l\}} q_j y_k + \sum_{i\in S_{j-1}^+\setminus T^+} (q_i - q_j) x_i + \sum_{i\in T^+} \frac{r_l(q_i-q_j)}{q_i} y_i \ , \tag{49}$$

$$\forall\ T^+ \subseteq S_{j-1}^+, \forall\ T^- \subseteq S_{l-1}^-, \forall\ j \in S^+, \forall\ l \in S^-$$

are valid for $\operatorname{conv}(X_{m_+,m_-})$*. If the conditions* $T^+ = S_{j-1}^+$ *and* $T^- \neq \emptyset$ *do not hold simultaneously, then the inequalities* (49) *are also facet-defining for* $\operatorname{conv}(X_{m_+,m_-})$.

Before proving these theorems, we give a brief explanation about their derivation as well as an illustrative example. Note that in Facet Class 1 when $l=m$ and $T=\emptyset$, the constraint becomes the original blending constraint (46a). Similarly, note that in Facet Class 2 when $j=1$, $l\in S^-$, and $T^- = T^+ = \emptyset$, the constraint becomes a variable upper bound constraint $x_l \le \frac{q_1}{q_1+r_l} y_l$ on a bad supplier $l \in S^-$. Wherever possible, we will use subscripts i and j when indexing good suppliers and k and l when indexing bad suppliers.

We refer to these inequalities as *lifted* facets because they can be derived from lifting blending or variable upper bound inequalities from lower-dimensional sets. Specifically, for Facet Class 1, if we fix $T \subseteq S^+$ and $l \in S^-$, and set $x_i = y_i = 0, \forall i \in T$, and $x_k = y_k = 0, \forall k \in S^-, k \neq l$, we may lift the pairs of variables (x_t, y_t), which were fixed at 0, by considering the lifting function associated with the blending constraint $\sum_{j \in S^+ \setminus T} q_j x_j - r_l x_l \geq 0$, which is a facet on this restricted set. Similarly, for Facet Class 2, we fix a good supplier $j \in S^+$, a bad supplier $l \in S^-$, and set $x_i = y_i = x_k = y_k = 0, \forall i \in S^+_{j-1}, \forall k \in S^-_{l-1}$. We may then lift the pairs of variables (x_t, y_t), which were fixed at 0, by considering the lifting function associated with the variable upper bound constraint $x_l \leq \frac{q_j}{q_j + r_l} y_l$, which is a facet on this restricted set. Moreover, it can be shown that this lifting function is superadditive, hence, we obtain the computationally attractive property known as sequence independent lifting [51].

Example. There are two good suppliers, $S^+ = \{1, 2\}$, two bad suppliers, $S^- = \{3, 4\}$, and $\mathbf{p} = (11, 7, -3, -5)$. The lifted blending facets are

	T	l	
$3x_1 - 7x_2 + 3x_3 + 3x_4 \leq 3y_1$	$\{1\}$	3	(LB 3a)
$-11x_1 + 3x_2 + 3x_3 + 3x_4 \leq 3y_2$	$\{2\}$	3	(LB 3b)
$x_1 + x_2 + x_3 + x_4 \leq y_1 + y_2$	$\{1, 2\}$	3	(LB 3c)
$-11x_1 - 7x_2 + 3x_3 + 5x_4 \leq 0$	$\emptyset$	4	(LB 4a)
$5x_1 - 7x_2 + 3x_3 + 5x_4 \leq 5y_1$	$\{1\}$	4	(LB 4b)
$-11x_1 + 5x_2 + 3x_3 + 5x_4 \leq 5y_2$	$\{2\}$	4	(LB 4c)

As described above, these facets are obtained by "turning off" all good suppliers in T and all bad suppliers besides l, and then lifting back in the pairs (x_t, y_t) of variables that were "turned off" starting from the lower-dimensional blending constraint $\sum_{j \in S^+ \setminus T} q_j x_j - r_l x_l \geq 0$. Note that facet (LB 4a) is the original blending constraint. Facet (LB 3c) states that at least one good supplier must be "turned on" if any product is sent from a supplier.

The lifted variable upper bound facets are

	T^+	T^-	j	l	
$14x_3 \leq 11y_3$	$\emptyset$	$\emptyset$	1	3	(LVUB 13)
$16x_4 \leq 11y_4$	$\emptyset$	$\emptyset$	1	4	(LVUB 14)
$-4x_1 + 10x_3 \leq 7y_3$	$\emptyset$	$\emptyset$	2	3	(LVUB 23a)
$12x_1 + 110x_3 \leq 12y_1 + 77y_3$	$\{1\}$	$\emptyset$	2	3	(LVUB 23b)
$-4x_1 + 12x_4 \leq 7y_4$	$\emptyset$	$\emptyset$	2	4	(LVUB 24a)
$-4x_1 + 10x_3 + 12x_4 \leq 7y_3 + 7y_4$	$\emptyset$	$\{3\}$	2	4	(LVUB 24b)
$20x_1 + 132x_4 \leq 20y_1 + 77y_4$	$\{1\}$	$\emptyset$	2	4	(LVUB 24c)

Note that when $j = 1$, the variable upper bound inequality $(q_1 + r_l)x_l \leq q_1 r_l$, for $l \in S^-$, is already facet-defining. When $j = 2$ and $l = 3$, i.e., when supplier 1 alone is "turned off" at the outset, there are two ways to lift in the pair (x_1, y_1) to obtain a facet as shown in (LVUB 23a) and (LVUB 23b). When $j = 2$ and $l = 4$, i.e., when suppliers 1 and 3 are "turned off" at the outset, there are three ways to lift in the pairs (x_1, y_1) and (x_3, y_3) to obtain a facet as shown in (LVUB 24a) – (LVUB 24c).

In addition to the bound inequalities, inequalities (LB) and (LVUB), the following three facets are needed to describe the convex hull of $X_{m+,m-}$ for this example:

$$\begin{array}{rcrcrcrcrcrcrcr}
-110x_1 & + & 30x_2 & + & 30x_3 & + & 162x_4 & \leq & & + & 30y_2 & & & + & 77y_4 \\
30x_1 & - & 42x_2 & + & 128x_3 & + & 30x_4 & \leq & 30y_1 & & & + & 77y_3 & & \\
60x_1 & & & + & 550x_3 & + & 396x_4 & \leq & 60y_1 & & & + & 385y_3 & + & 231y_4 \ .
\end{array}$$

Proof of Theorem 1: Let $(\mathbf{x}^*, \mathbf{y}^*) \in X_{m+,m-}$, $T \subseteq S^+$, and $l \in S^-$. If $y_i^* = 0, \forall\, i \in T$, then inequality (48) reduces to a weakened version (because of the min operator) of the blending constraint (46a) under the restriction $x_i = y_i = 0, \forall\, i \in T$. Otherwise, we have

$$\sum_{i \in T} x_i^* + \sum_{k \in S^-} \min\left\{1, \frac{r_k}{r_l}\right\} x_k^* \leq \sum_{i \in T} x_i^* + \sum_{k \in S^-} x_k^* \leq 1 \leq \sum_{i \in T} y_i^* \leq \sum_{i \in S^+ \setminus T} \left(\frac{q_i}{r_l}\right) x_i^* + \sum_{i \in T} y_i^* \ .$$

In all but the two exceptional cases, to prove that inequality (48) is facet-defining for a given choice of $T \subseteq S^+$ and $l \in S^-$, let $u \in S^+ \setminus T$ and $v \in T$. One can verify that the following $2m-1$ points, along with the origin, are affinely independent:

$$\Big(\mathbf{0}, \mathbf{e}_i\Big), \qquad \forall\ i \in S^+ \setminus T \tag{50a}$$

$$\Big(\mathbf{e}_i, \mathbf{e}_i\Big), \qquad \forall\ i \in T \tag{50b}$$

$$\Big(\frac{r_l}{q_i + r_l}\mathbf{e}_i + \frac{q_i}{q_i + r_l}\mathbf{e}_l, \mathbf{e}_i + \mathbf{e}_l\Big), \qquad \forall\ i \in S^+ \tag{50c}$$

$$\Big(\mathbf{0}, \mathbf{e}_k\Big), \qquad \forall\ k \in S^- \tag{51a}$$

$$\Big(\frac{r_k}{q_u + r_k}\mathbf{e}_u + \frac{q_u}{q_u + r_k}\mathbf{e}_k, \mathbf{e}_u + \mathbf{e}_k\Big), \qquad \forall\ k \in S^-, k < l \tag{51b}$$

$$\Big(\frac{r_k}{q_v + r_k}\mathbf{e}_v + \frac{q_v}{q_v + r_k}\mathbf{e}_k, \mathbf{e}_v + \mathbf{e}_k\Big), \qquad \forall\ k \in S^-, k > l \tag{51c}$$

Note that (50a)–(50c) contribute $2m_+$ points and (51a)–(51c) contribute $2m_- - 1$ points. $\square$

Proof of Theorem 2: Let $(\mathbf{x}^*, \mathbf{y}^*)$ be an extreme point of $\text{conv}(X_{m_+,m_-})$. Let $j \in S^+, l \in S^-, T^+ \subseteq S^+_{j-1}$, and $T^- \subseteq S^-_{l-1}$. If $x^*_i = 1$ for some $i \in S^+$ or if $x^*_k > 0$ for some $k \in S^- \setminus (T^- \cup \{l\})$, then validity is immediate. So suppose $(\mathbf{x}^*, \mathbf{y}^*)$ takes the form (47c) for some $i \in S^+$ and some $k \in T^- \cup \{l\}$.

Case 1: If $i \geq j$ $(q_j \geq q_i)$, then $(q_j + r_k)x^*_k = (q_j + r_k)\left(\frac{q_i}{q_i+r_k}\right) \leq q_j = q_j y^*_k$.

Case 2: If $i \in S^+_{j-1} \setminus T^+$, then since $x^*_i + x^*_k = 1$ and $r_k x^*_k - q_i x^*_i = 0$, we obtain

$$(q_j - q_i)x^*_i + (q_j + r_k)x^*_k = q_j(x^*_i + x^*_k) + r_k x^*_k - q_i x^*_i = q_j = q_j y^*_k\ .$$

Case 3: If $i \in T^+$, then $\left(\frac{r_l(q_i-q_j)}{q_i}\right)x^*_i + (q_j + r_k)x^*_k = \left(\frac{r_l(q_i-q_j)}{q_i}\right)\left(\frac{r_k}{q_i+r_k}\right) + (q_j + r_k)\left(\frac{q_i}{q_i+r_k}\right) \leq q_j + \frac{r_l(q_i-q_j)}{q_i} = q_j y^*_k + \left(\frac{r_l(q_i-q_j)}{q_i}\right)y^*_i$, with equality holding only when $k = l$.

In all but the exceptional cases, to prove that inequality (49) is facet-defining for a given choice of $j \in S^+, l \in S^-, T^+ \subseteq S^-_{j-1}$, and $T^- \subseteq S^-_{l-1}$, let $u \in S^+ \setminus S^+_{j-1}$ and $v \in S^+_{j-1} \setminus T^+$. One can verify that the following $2m-1$ points, along with the origin, are affinely independent:

$$\Big(\mathbf{0}, \mathbf{e}_i\Big), \qquad \forall\ i \in (S^+_{j-1} \setminus T^+) \cup (S^+ \setminus S^+_{j-1}) \tag{52a}$$

$$\Big(\mathbf{e}_i, \mathbf{e}_i\Big), \qquad \forall\ i \in T^+ \cup (S^+ \setminus S^+_{j-1}) \tag{52b}$$

$$\Big(\frac{r_l}{q_i + r_l}\mathbf{e}_i + \frac{q_i}{q_i + r_l}\mathbf{e}_l, \mathbf{e}_i + \mathbf{e}_l\Big), \qquad \forall\ i \in S^+_{j-1} \tag{52c}$$

$$\left(\mathbf{0}, \mathbf{e}_k\right), \qquad \forall\, k \in (S^-_{l-1} \setminus T^-) \cup (S^- \setminus S^-_l) \tag{53a}$$

$$\left(\frac{r_k}{q_j + r_k}\mathbf{e}_j + \frac{q_j}{q_j + r_k}\mathbf{e}_k, \mathbf{e}_j + \mathbf{e}_k\right), \qquad \forall\, k \in T^- \cup \{l\} \cup (S^- \setminus S^-_l) \tag{53b}$$

$$\left(\frac{r_k}{q_u + r_k}\mathbf{e}_u + \frac{q_u}{q_u + r_k}\mathbf{e}_k, \mathbf{e}_u + \mathbf{e}_k\right), \qquad \forall\, k \in S^-_{l-1} \setminus T^- \tag{53c}$$

$$\left(\frac{r_k}{q_v + r_k}\mathbf{e}_v + \frac{q_v}{q_v + r_k}\mathbf{e}_k, \mathbf{e}_v + \mathbf{e}_k\right), \qquad \forall\, k \in T^- \tag{53d}$$

Note that (52a)–(52c) contribute $2m_+$ points and (53a)–(53d) contribute $2m_- - 1$ points. $\square$

5.2.3 Separation

The next proposition shows that separation of the lifted blending constraints (48) and the lifted variable upper bound constraints (49) can be done in polynomial time, i.e., the former can be done in $O(m^2)$ time while the latter can be done in $O(m^3)$ time.

Proposition 2. *Let $(\mathbf{x}^*, \mathbf{y}^*)$ be an optimal solution to the LP relaxation.*

1. *Fix $l \in S^-$. If*

$$\zeta(l) := \sum_{k \in S^-} \min\left\{1, \frac{r_k}{r_l}\right\} x_k^* + \sum_{i \in S^+} \left(\left(1 + \left(\frac{q_i}{r_l}\right)\right) x_i^* - y_i^*\right)^+ - \left(\frac{q_i}{r_l}\right) x_i^* \tag{54}$$

is positive, where $(x)^+ := \max\{0, x\}$, then the most violated lifted blending inequality (48) for this $l \in S^-$ is given by the subset $T := \{i \in S^+ : \left(\left(1 + \left(\frac{q_i}{r_l}\right)\right) x_i^* - y_i^*\right) > 0\}$. If $\zeta(l) \leq 0, \forall l \in S^-$, then there is no violated lifted blending inequality (48).

2. *Fix $j \in S^+$ and $l \in S^-$. If*

$$\begin{aligned}\psi(j,l) := -\sum_{i=1}^{j-1} \frac{(q_i - q_j)}{r_l} x_i^* + \sum_{i=1}^{j-1} \left(\frac{(q_i - q_j)}{q_i}(x_i^* - y_i^*) + \frac{(q_i - q_j)}{r_l} x_i^*\right)^+ \\ + \sum_{k=m_+ + 1}^{l} \left(\frac{(q_j + r_k)}{r_l} x_k^* - \frac{q_j}{r_l} y_k^*\right)^+\end{aligned} \tag{55}$$

is positive, then the most violated lifted variable upper bound inequality (49) for this $j \in S^+$ and $l \in S^-$ is given by the subsets

$$T^+ := \left\{i \in \{1, \ldots, j-1\} : \left(\frac{(q_i - q_j)}{q_i}(x_i^* - y_i^*) + \frac{(q_i - q_j)}{r_l} x_i^*\right) > 0\right\}$$

and

$$T^- := \left\{ k \in \{m_+ + 1, \ldots, l\} : \left(\frac{(q_j + r_k)}{r_l} x_k^* - \frac{q_j}{r_l} y_k^* \right) > 0 \right\} .$$

If $\psi(j,l) \leq 0, \forall j \in S^+, \forall l \in S^-$, *then there is no violated lifted variable upper bound inequality* (49).

Proof. 1. For each bad supplier $l \in S^-$, one can find the most violated blending inequality (48), or determine that no such violated inequality exists, by checking if

$$\zeta(l) = \kappa + \max_{T \subseteq S^+} \sum_{i \in T} (x_i^* - y_i^*) - \sum_{i \in S^+ \setminus T} \left(\frac{q_i}{r_l} \right) x_i^*$$

is positive, where $\kappa = \sum_{k \in S^-} \min \left\{ 1, \frac{r_k}{r_l} \right\} x_k^*$ is a constant independent of the subset T. Notice that the maximization is trivial: if $x_i^* - y_i^* > -\left(\frac{q_i}{r_l} \right) x_i^*$, set $i \in T$; otherwise, $i \in S^+ \setminus T$. Consequently, if $\zeta(l)$, as defined in (54), is positive, set $T = \{ i \in S^+ : \left(\left(1 + \left(\frac{q_i}{r_l} \right) \right) x_i^* - y_i^* \right) > 0 \}$. Since $\zeta(l)$ can be computed by summing over all good suppliers $j \in S^+$, of which there are at most m, and this operation must be done for each bad supplier $l \in S^-$, of which there are also at most m, we can determine the most violated lifted blending cuts (48) in $O(m^2)$ time.

2. For each good supplier $j \in S^+$ and each bad supplier $l \in S^-$, one can find the most violated variable upper bound inequality (49), or determine that no such violated inequality exists, by checking if

$$\begin{aligned} \psi(j,l) = \max_{\{T^+ \subseteq S^+_{j-1}, T^- \subseteq S^-_{l-1}\}} & \sum_{i \in T^+} \frac{(q_i - q_j)}{q_i} (x_i^* - y_i^*) + \sum_{k \in T^- \cup \{l\}} \left(\frac{(q_j + r_k)}{r_l} x_k^* - \frac{q_j}{r_l} y_k^* \right) \\ & - \sum_{i \in S^+_{j-1} \setminus T^+} \frac{(q_i - q_j)}{r_l} x_i^* \end{aligned}$$

is positive. As above, this maximization problem is trivial: if $\left(\frac{r_l (q_i - q_j)}{q_i} \right) (x_i^* - y_i^*) > -(q_i - q_j) x_i^*$ for $i \in S^+_{j-1}$, set $i \in T^+$; otherwise, set $i \in S^+_{j-1} \setminus T^+$. Similarly, if $(q_j + r_k) x_k^* - q_j y_k^* > 0$ for $k \in S^-_{l-1}$, set $k \in T^-$; otherwise, set $k \in S^-_{l-1} \setminus T^-$. Hence, if $\psi(j,l)$, as defined in (55), is positive, set T^+ and T^- accordingly. In the worst case, it requires $O(m^3)$ time to find the most violated lifted variable upper bound facets over all (j,l) pairs. This follows because looping over all (j,l) pairs, for $j \in S^+$ and $l \in S^-$, requires $O(m^2)$ time, and for a given (j,l) pair, the above summation requires $O(m)$ time. $\square$

5.2.4 Relation to Single-Node Flow Covers

We close this section by comparing the constraint set $X_{m+,m-}$ with that of the single-node flow model since the latter has been studied extensively in the literature [50, 72]. The constraint set for a single-node flow model is given by

$$F := \left\{ (\mathbf{x}, \mathbf{y}) \in \mathbb{R}^m_+ \times \{0,1\}^m : \sum_{j \in N^+} x_j - \sum_{j \in N^-} x_j \le b, x_j \le a_j y_j, \forall\ j \in N \right\},$$

where the set N of arcs has been partitioned into incoming arcs N^- and outgoing arcs N^+, each arc j has a fixed capacity $a_j \in \mathbb{R}_+$ if opened, and $b \in \mathbb{R}$ is the exogeneous supply/demand at this node. There are two ways to relate the set $X_{m+,m-}$ to F.

- Interpretation 1: After setting $a_j = 1, \forall j \in S$, $b = 1$, and $N^- = \emptyset$, one can treat the demand constraint $\sum_{i \in S} x_i \le 1$ as the constraint $\sum_{j \in N^+} x_j - \sum_{j \in N^-} x_j \le b$ in F and intersect F with a single homogeneous linear inequality $\sum_{i \in S^+} q_i x_i - \sum_{k \in S^-} r_k x_k \ge 0$ to obtain the set $X_{m+,m-}$ as it was originally defined in (46).

- Interpretation 2: After setting $a_j = |p_j|, \forall j \in S$, and $b = 0$, and introducing an auxiliary decision variable $z_j = |p_j| x_j, \forall j \in S$, one can rewrite $\sum_{j \in S} p_j x_j \ge 0$ as $\sum_{j \in S^-} z_j - \sum_{j \in S^+} z_j \le b$. Thus, S^- and S^+ play the role of N^+ and N^-, respectively, in F. In addition, one must intersect these constraints with the demand constraint, which becomes $\sum_{i \in S} \frac{z_j}{|p_j|} \le 1$, to obtain the set

$$Z := \left\{ (\mathbf{z}, \mathbf{y}) \in \mathbb{R}^m_+ \times \{0,1\}^m : \sum_{j \in S^-} z_j - \sum_{j \in S^+} z_j \le 0, \sum_{j \in S} \frac{z_j}{|p_j|} \le 1, z_j \le |p_j| y_j, \forall\ j \in S \right\}.$$

Since $X_{m+,m-}$ and Z are subsets of F, valid cuts generated by well known procedures for the single-node flow covers, e.g., lifted flow cover inequalities, are valid for $X_{m+,m-}$ and Z. However, it is easy to verify that our two facet classes *cannot* be obtained as flow cover inequalities from $X_{m+,m-}$ or Z when the additional side constraint is omitted.

5.3 Special Cases: One Good or One Bad Supplier

In this section, we consider two special cases of the FCTPwB in which S^+ or S^- is a singleton. In both cases, we show that the continuous relaxation of $X_{m+,m-}$ along with

Facet Classes 1 and 2 yield the convex hull of X_{m_+,m_-}. These results lend theoretical support to our claim that inclusion of our two facet classes lead to strong formulations of the FCTPwB. Note that, as shown in the example from Section 5.2.2, when $|S^+| > 1$ and $|S^-| > 1$, the continuous relaxation of the original formulation constraints and the two facet classes are not enough to describe $\text{conv}(X_{m_+,m_-})$.

5.3.1 One Good Supplier and Many Bad Suppliers

First consider the simplified single-consumer model in which there is a single good supplier and one or more bad suppliers, i.e., $S^+ = \{1\}$ and $S^- = \{2, \dots, m\}$. In this case, the lifted blending and variable upper bound facets for $X_{1,m-1}$ become:

$$\sum_{i \in S} x_i \le y_1 \tag{56a}$$

$$x_k \le \frac{q_1}{q_1 + r_k} y_k, \forall\, k \in S^-. \tag{56b}$$

Constraint (56a) states that if any product is sent, then the arc originating from the lone good supplier must be "on" (otherwise, the blending constraint cannot be met). Similarly, the maximum amount of product that can be sent from a bad supplier $k \in S^-$ is bounded above by the ratio $\frac{q_1}{q_1+r_k}$.

Theorem 3. [A Polyhedral Description of $\text{conv}(X_{1,m-1})$] *Let* $P := \{(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^m_+ \times [0,1]^m :$ (46a), (56a), (56b)$\}$. *Then* $P = \text{conv}(X_{1,m-1})$.

Proof Let $(\mathbf{x}^*, \mathbf{y}^*) \in P$ with some fractional $y_i^* \in (0, 1)$. We show that $(\mathbf{x}^*, \mathbf{y}^*)$ cannot be an extreme point of P (see, e.g., Approach 2 on p.145 of [108]). Without loss of generality, we assume that the p_i's have been normalized so that $q_1 = 1$. The proof is split into four cases:

Case 1: Suppose $i \in S^-$ and $x_i^* < \frac{y_i^*}{r_i+1}$. Then for some $\varepsilon > 0$ we have $(\mathbf{x}^*, \mathbf{y}^* \pm \varepsilon \mathbf{e}_i) \in P$. Therefore $(\mathbf{x}^*, \mathbf{y}^*)$ is not extreme.

Case 2: Suppose $\sum_{k \in S} x_k^* = \alpha < 1$. Then the points

$$x_k^1 = \frac{x_k^*}{\alpha}, \quad y_k^1 = \min\left\{1, \frac{y_k^*}{\alpha}\right\}, \forall\, k \in S, \qquad \text{and} \qquad x_k^2 = 0, \quad y_k^2 = \max\left\{0, \frac{y_k^* - \alpha}{1 - \alpha}\right\}, \forall\, k \in S,$$

satisfy $(\mathbf{x}^1,\mathbf{y}^1),(\mathbf{x}^2,\mathbf{y}^2) \in P$ and yield $(\mathbf{x}^*,\mathbf{y}^*) = \alpha(\mathbf{x}^1,\mathbf{y}^1) + (1-\alpha)(\mathbf{x}^2,\mathbf{y}^2)$. Thus, $i \neq 1$ and we must have $\sum_{k\in S} x_k^* = 1$ at any nontrivial extreme point of P.

Case 3: Suppose $\sum_{k\in S} x_k^* = 1$ (which implies $y_1^* = 1$), $x_i^* = \frac{y_i^*}{r_i+1}$ and $x_1^* - \sum_{k\in S^-} r_k x_k^* > 0$. The point $(\mathbf{x}^1,\mathbf{y}^1)$ with

$$x_1^1 = \frac{r_i + \sum_{k\neq 1,i}(r_k - r_i)x_k^*}{r_i+1}, \quad x_i^1 = \frac{1-\sum_{k\neq 1,i}(r_k+1)x_k^*}{r_i+1},$$
$$y_1^1 = 1, \quad y_i^1 = 1 - \sum_{k\neq 1,i}(r_k+1)x_k^*, \quad (x_k^1, y_k^1) = (x_k^*, y_k^*), \forall\ k \neq 1, i$$

and $(\mathbf{x}^2,\mathbf{y}^2)$ with

$$x_1^2 = 1 - \sum_{k\neq 1,i} x_k^*, \quad x_i^2 = 0, \quad y_1^2 = 1, \quad y_i^2 = 0, \quad (x_k^2, y_k^2) = (x_k^*, y_k^*), \forall\ k \neq 1, i$$

belong to P and there is some $\lambda \in (0,1)$ with $(\mathbf{x}^*,\mathbf{y}^*) = \lambda(\mathbf{x}^1,\mathbf{y}^1) + (1-\lambda)(\mathbf{x}^2,\mathbf{y}^2)$.

Case 4: Suppose $\sum_{k\in S} x_k^* = 1$, $x_i^* = \frac{y_i^*}{r_i+1}$ and $x_1^* - \sum_{k\in S^-} r_k x_k^* = 0$. Then $y_i^* + \sum_{k\in S^-\setminus\{i\}}(r_k+1)x_k^* = 1$, which implies that $0 \leq x_l^* < \frac{1}{r_l+1}, \forall l \in S^- \setminus \{i\}$, and that there exists some $k \in S^- \setminus \{i\}$ such that $x_k^* > 0$. Since $0 < x_k^* < \frac{1}{r_k+1}$, $y_k^* = 1$ (otherwise, we are in Case 1). Define the direction vector $\mathbf{d} \in \mathbb{R}^m$ as

$$d_1 = \left(\frac{r_i+1}{r_k+1} - 1\right), \quad d_i = 1, \quad d_k = -\frac{r_i+1}{r_k+1}, \quad d_j = 0, \forall j \notin \{1, i, k\},$$

and note that $\sum_{j\in S} d_j = 0$ and $d_1 - \sum_{l\in S^-} r_l d_l = 0$. For $\varepsilon > 0$, define $y_i^1 = (r_i+1)(x_i^* + \varepsilon)$, $y_i^2 = (r_i+1)(x_i^* - \varepsilon)$, and let $\mathbf{x}^1 = \mathbf{x}^* + \varepsilon\mathbf{d}$, $\mathbf{x}^2 = \mathbf{x}^* - \varepsilon\mathbf{d}$, $y_j^1 = y_j^2 = y_j^*, \forall\ j \neq i$. Then if ε is small enough, $(\mathbf{x}^1,\mathbf{y}^1),(\mathbf{x}^2,\mathbf{y}^2) \in P$, and $(\mathbf{x}^*,\mathbf{y}^*)$ is their midpoint, so it cannot be extreme. □

5.3.2 Many Good Suppliers and One Bad Supplier

A polyhedral description of $\text{conv}(X_{m-1,1})$ is more complex than $\text{conv}(X_{1,m-1})$, in which there were only a polynomial number of facets. When $S^+ = \{1,\dots,m-1\}$ and $S^- = \{m\}$, the lifted blending and variable upper bound facets for $X_{m-1,1}$ become:

$$\sum_{i\in T} x_i + x_m \le \sum_{i\in S^+\setminus T}\left(\frac{q_i}{r_m}\right)x_i + \sum_{i\in T} y_i\ , \qquad \forall\ T\subseteq S^+ \tag{57a}$$

$$\sum_{i\in T}\frac{r_m(q_i-q_j)}{q_i}x_i + (q_j+r_m)x_m \le q_j y_m + \sum_{i\in S^+_{j-1}\setminus T}(q_i-q_j)x_i + \sum_{i\in T}\frac{r_m(q_i-q_j)}{q_i}y_i\ , \qquad \forall\ T\subseteq S^+_{j-1}\ , \forall\ j\in S^+ \tag{57b}$$

Theorem 4. [A Polyhedral Description of $\operatorname{conv}(X_{m-1,1})$] *Let* $P := \{(\mathbf{x},\mathbf{y})\in\mathbb{R}^m_+\times[0,1]^m : x_i\le y_i, \forall i\in S^+, (46b), (57a), (57b)\}$*. Then* $P=\operatorname{conv}(X_{m-1,1})$.

The next two propositions are used in the proof of Theorem 4. Let $\alpha_i = \frac{r_m}{q_i+r_m}, \forall\ i\in S^+$.

Proposition 3. *The extreme points of* $\operatorname{conv}(X_{m-1,1})$ *that lie in a lifted blending facet* (57a) *defined by the subset $T\subseteq S^+$ are:*

$$\left(\mathbf{0}, \sum_{u\in U}\mathbf{e}_u\right), \qquad \forall\ U\subseteq S\setminus T \tag{58a}$$

$$\left(\mathbf{e}_i, \mathbf{e}_i + \sum_{u\in U}\mathbf{e}_u\right), \qquad \forall\ i\in T, \forall\ U\subseteq S^+\setminus T \tag{58b}$$

$$\left(\alpha_i\mathbf{e}_i + (1-\alpha_i)\mathbf{e}_m, \mathbf{e}_i+\mathbf{e}_m + \sum_{u\in U}\mathbf{e}_u\right), \qquad \forall\ i\in S^+, \forall\ U\subseteq S^+\setminus T, \tag{58c}$$

Proof By inspection. Substitute each extreme point of $\operatorname{conv}(X_{m-1,1})$ into the lifted blending facet defined by the subset $T\subseteq S^+$ and verify that the facet is only satisfied at equality by the above extreme points. □

Proposition 4. *The extreme points of* $\operatorname{conv}(X_{m-1,1})$ *that lie in a lifted variable upper bound facet* (57b) *defined by $j\in S^+$ and the subset $T\subseteq S_{j-1}$ are:*

$$\left(\mathbf{0}, \sum_{u\in U}\mathbf{e}_u\right), \qquad \forall\ U\subseteq S^+\setminus T \tag{59a}$$

$$\left(\mathbf{e}_i, \mathbf{e}_i + \sum_{u\in U}\mathbf{e}_u\right), \qquad \forall\ i\in (S^+\setminus S_{j-1})\cup T, \forall\ U\subseteq S^+\setminus (T\cup\{i\}) \tag{59b}$$

$$\left(\alpha_i\mathbf{e}_i + (1-\alpha_i)\mathbf{e}_m, \mathbf{e}_i+\mathbf{e}_m + \sum_{u\in U}\mathbf{e}_u\right), \qquad \forall\ i\in S_j, \forall\ U\subseteq S^+\setminus (T\cup\{i\}), \tag{59c}$$

Proof By inspection. Substitute each extreme point of $\text{conv}(X_{m-1,1})$ into the lifted variable upper bound facet defined by $j \in S^+$ and the subset $T \subseteq S_{j-1}$ and verify that the facet is only satisfied at equality by the above extreme points. $\square$

Proof of Theorem 4. We show that for any cost vector $(\mathbf{c}, \mathbf{f}) \in \mathbb{R}^{m \times m}, (\mathbf{c}, \mathbf{f}) \neq (\mathbf{0}, \mathbf{0})$, the set $M(\mathbf{c}, \mathbf{f})$ of optimal solutions to the problem $\max\{\mathbf{c}^T\mathbf{x} - \mathbf{f}^T\mathbf{y} : (\mathbf{x}, \mathbf{y}) \in X_{m-1,1}\}$ coincides with at least one of the hyperplanes associated with an inequality defining P (see, e.g., Approach 6 on p.146 of [108]). Since the inequalities defining P are all facets of $\text{conv}(X_{m-1,1})$, P is a minimal polyhedral representation of $\text{conv}(X_{m-1,1})$. The proof, which is outlined in Figure 22, proceeds by partitioning the space of cost vectors and by gradually eliminating cost vectors from consideration. Initially, cost vectors that lead to optimal solutions that lie on one of the trivial or formulation facets are considered. Finally, cost vectors that lead to the case in which we are indifferent between sending product exclusively from a single good supplier and from a good supplier and the bad supplier are considered. The following notation will be used:

- $\alpha_i = \frac{r_m}{q_i + r_m}, (1 - \alpha_i) = \frac{q_i}{q_i + r_m}, \forall\, i \in S^+$
- $g_i = \alpha_i c_i + (1 - \alpha_i)c_m - (f_i + f_m), \forall\, i \in S^+$
- $CF = \arg\max\{c_i - f_i : (\mathbf{x}, \mathbf{y}) \in X_{m-1,1}\}$
- $G = \arg\max\{g_i : (\mathbf{x}, \mathbf{y}) \in X_{m-1,1}\}$

Note that CF and G are sets, not indices. Here $c_i - f_i$ denotes the cost of sending all supply exclusively from good supplier $i \in S^+$, whereas g_i denotes the cost of sending a nontrivial convex combination of supply from supplier i and the lone bad supplier m so that $\sum_{i \in S} x_i = 1$ and $\sum_{i \in S} p_i x_i = 0$. We say that g_i is the cost associated with a "blended" solution. Each bullet below corresponds to a branch in the tree presented in Figure 22.

- If $f_i < 0$ for some $i \in S$, then $y_i = 1$ in every optimal solution, i.e., $M(\mathbf{c}, \mathbf{f}) = \{(\mathbf{x}, \mathbf{y}) \in X_{m-1,1} : y_i = 1\}$. Thus, we may assume that $f_i \geq 0, \forall i \in S$.
- If $c_m < 0$, then $x_m = 0$ in every optimal solution, i.e., $M(\mathbf{c}, \mathbf{f}) = \{(\mathbf{x}, \mathbf{y}) : x_m = 0\}$. Thus, we may assume that $c_m \geq 0$.

- If $c_m = 0$, then
  - if $c_i - f_i < 0$ for some $i \in S^+$, then $x_m = 0$ in every optimal solution. Thus, we may assume that $c_i - f_i \geq 0, \forall i \in S^+$.
  - if $c_i - f_i > 0$ for some $i \in S^+$, then $x_i = 0$ in every optimal solution. Thus, we may assume that $c_i - f_i = 0, \forall i \in S^+$.
  - if $c_i - f_i = 0, \forall i \in S^+$, then $x_i = y_i$ in every optimal solution.

  Thus, we may assume that $c_m > 0$. In the remainder of the proof, we omit the statement "Thus, we may assume ..." to refer to the complement case as the details are shown in the tree structure of Figure 22.

- If $g_j < 0, \forall j \in G$, then $x_m = 0$ in every optimal solution.
- If $c_i - f_i > g_j, \forall i \in CF, \forall j \in G$, then $\sum_{i \in S} x_i = 1$ and $x_m = 0$ in every optimal solution.
- If $c_i - f_i < 0, \forall i \in CF$, then a "blended" solution is always optimal in which case $\sum_{i \in S} p_i x_i = 0$ in every optimal solution.
- Similarly, if $c_i - f_i < g_j, \forall i \in CF, \forall j \in G$, then a "blended" solution is always optimal in which case $\sum_{i \in S} p_i x_i = 0$ in every optimal solution.
- If $c_i - f_i > 0, \forall i \in CF$, then a solution in which all product is sent exclusively from a good supplier is optimal in which case $\sum_{i \in S} x_i = 1$ in every optimal solution.
- If $i \notin CF \cup G$, then $x_i = 0$ in every optimal solution.

Finally, we arrive at the last black box in Figure 22 in which we only have to consider cost vectors that satisfy $\mathbf{c} \in \mathbb{R}^{m-1} \times \mathbb{R}_{++}, \mathbf{f} \in \mathbb{R}^m_+,\ 0 = c_i - f_i = g_j, \forall i \in CF, \forall j \in G; CF \cup G = S^+$. Let $F_0 = \{i \in S^+ : f_i = 0\}$ and $F_+ = \{i \in S^+ : f_i > 0\}$. We now consider two cases, $f_m = 0$ and $f_m > 0$, and show that the former leads to extreme points that lie on a lifted blending facet and the latter to extreme points on a lifted variable upper bound facet.

Suppose $f_m = 0$. Set $T = CF$ and note that $f_i > 0, \forall i \in T$, i.e., $T \subseteq F_+$. This follows since for all $k \in CF \cap G$, $0 = c_k - f_k = g_k$ implies $c_k = f_k = c_m (> 0)$. Similarly, for all

$i \in CF \setminus G$, we have $c_i = f_i \geq 0$ by assumption. Suppose, to the arrive at a contradiction, that $f_i = 0$. Since $0 > g_i = \alpha_i c_i + (1-\alpha_i)c_m$ and $(1-\alpha_i)c_m > 0$ by assumption, it must be the case that $c_i < 0$, which is a contradiction. Then, in accordance with Proposition 3, the following extreme points lie on the lifted blending facet defined by T:

$$\left(\mathbf{0}, \sum_{k\in U} \mathbf{e}_k\right), \qquad \forall\, U \subseteq F_0 \tag{60a}$$

$$\left(\mathbf{e}_i, \mathbf{e}_i + \sum_{k\in U} \mathbf{e}_k\right), \qquad \forall\, i \in CF, \forall\, U \subseteq S^+ \setminus F_+ \tag{60b}$$

$$\left(\alpha_i \mathbf{e}_i + (1-\alpha_i)\mathbf{e}_m, \mathbf{e}_i + \mathbf{e}_m + \sum_{j\in U} \mathbf{e}_j\right), \qquad \forall\, i \in G, \forall\, U \subseteq S^+ \setminus F_+. \tag{60c}$$

Suppose $f_m > 0$. Set $j = \max\{t \in G\}$ and $T = CF \cap S_{j-1}$ so that $CF \subseteq (S^+ \setminus S_{j-1}) \cup T$ and $G \subseteq S_j$. Then, in accordance with Proposition 4, the following extreme points lie on the lifted variable upper bound facet defined by j and $T \subseteq S_{j-1}$:

$$\left(\mathbf{0}, \sum_{k\in U} \mathbf{e}_k\right), \qquad \forall\, U \subseteq F_0 \tag{61a}$$

$$\left(\mathbf{e}_i, \mathbf{e}_i + \sum_{k\in U} \mathbf{e}_k\right), \qquad \forall\, i \in CF, \forall\, U \subseteq S^+ \setminus (T \cup \{i\} \cup F_+) \tag{61b}$$

$$\left(\alpha_i \mathbf{e}_i + (1-\alpha_i)\mathbf{e}_m, \mathbf{e}_i + \mathbf{e}_m + \sum_{k\in U} \mathbf{e}_k\right), \qquad \forall\, i \in G, \forall\, U \subseteq S^+ \setminus (T \cup \{i\} \cup F_+). \tag{61c}$$

The only fact that we need to justify is that $F_0 \subseteq S^+ \setminus T$, or, equivalently, $T \subseteq F_+$. Suppose, to arrive at a contradiction, that this is not the case, i.e., that $T \neq \emptyset$ and $\exists i \in T$ such that $f_i = 0$. Then, since $i \in CF$ and $f_i = 0$, we have $c_i - f_i = c_i = f_i = 0$ and $0 \geq g_i = \alpha_i c_i + (1-\alpha_i)c_m - f_i - f_m = (1-\alpha_i)c_m - f_m$, which implies that $f_m \geq (1-\alpha_i)c_m$. Since $j \notin T$ by construction and $1-\alpha_1 > \cdots > 1-\alpha_{m-1}$ by assumption, we see that $f_m \geq (1-\alpha_i)c_m > (1-\alpha_j)c_m$, or

$$(1-\alpha_j)c_m - f_m < 0\ . \tag{62}$$

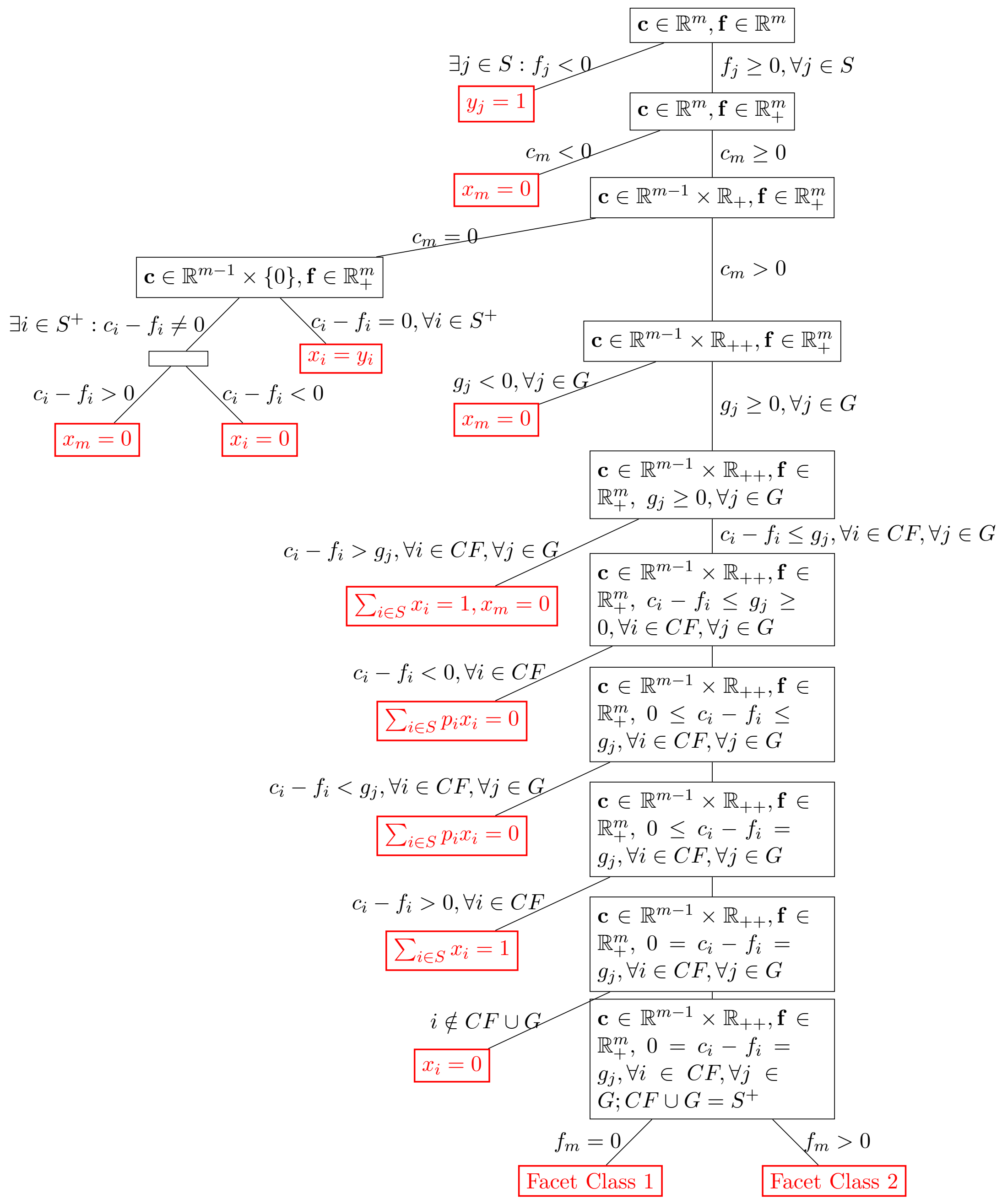
In addition, we have $c_j - f_j \leq 0$, which means that $f_j \geq c_j$ and

$$\alpha_j c_j - f_j \leq 0\ . \tag{63}$$

It follows from inequalities (62) and (63) that

$$0 = g_j = \underbrace{\alpha_j c_j - f_j}_{\leq 0} + \underbrace{(1-\alpha_j)c_m - f_m}_{<0} < 0\ , \tag{64}$$

which is a contradiction. $\square$



$\mathbf{c} \in \mathbb{R}^m, \mathbf{f} \in \mathbb{R}^m$
$\exists j \in S : f_j < 0$
$y_j = 1$
$f_j \geq 0, \forall j \in S$
$\mathbf{c} \in \mathbb{R}^m, \mathbf{f} \in \mathbb{R}^m_+$
$c_m < 0$
$x_m = 0$
$c_m \geq 0$
$\mathbf{c} \in \mathbb{R}^{m-1} \times \mathbb{R}_+, \mathbf{f} \in \mathbb{R}^m_+$
$c_m = 0$
$c_m > 0$
$\mathbf{c} \in \mathbb{R}^{m-1} \times \{0\}, \mathbf{f} \in \mathbb{R}^m_+$
$\exists i \in S^+ : c_i - f_i \neq 0$
$c_i - f_i = 0, \forall i \in S^+$
$x_i = y_i$
$c_i - f_i > 0$
$c_i - f_i < 0$
$x_m = 0$
$x_i = 0$
$\mathbf{c} \in \mathbb{R}^{m-1} \times \mathbb{R}_{++}, \mathbf{f} \in \mathbb{R}^m_+$
$g_j < 0, \forall j \in G$
$x_m = 0$
$g_j \geq 0, \forall j \in G$
$\mathbf{c} \in \mathbb{R}^{m-1} \times \mathbb{R}_{++}, \mathbf{f} \in \mathbb{R}^m_+,\ g_j \geq 0, \forall j \in G$
$c_i - f_i > g_j, \forall i \in CF, \forall j \in G$
$\sum_{i \in S} x_i = 1, x_m = 0$
$c_i - f_i \leq g_j, \forall i \in CF, \forall j \in G$
$\mathbf{c} \in \mathbb{R}^{m-1} \times \mathbb{R}_{++}, \mathbf{f} \in \mathbb{R}^m_+,\ c_i - f_i \leq g_j \geq 0, \forall i \in CF, \forall j \in G$
$c_i - f_i < 0, \forall i \in CF$
$\sum_{i \in S} p_i x_i = 0$
$\mathbf{c} \in \mathbb{R}^{m-1} \times \mathbb{R}_{++}, \mathbf{f} \in \mathbb{R}^m_+,\ 0 \leq c_i - f_i \leq g_j, \forall i \in CF, \forall j \in G$
$c_i - f_i < g_j, \forall i \in CF, \forall j \in G$
$\sum_{i \in S} p_i x_i = 0$
$\mathbf{c} \in \mathbb{R}^{m-1} \times \mathbb{R}_{++}, \mathbf{f} \in \mathbb{R}^m_+,\ 0 \leq c_i - f_i = g_j, \forall i \in CF, \forall j \in G$
$c_i - f_i > 0, \forall i \in CF$
$\sum_{i \in S} x_i = 1$
$\mathbf{c} \in \mathbb{R}^{m-1} \times \mathbb{R}_{++}, \mathbf{f} \in \mathbb{R}^m_+,\ 0 = c_i - f_i = g_j, \forall i \in CF, \forall j \in G$
$i \notin CF \cup G$
$x_i = 0$
$\mathbf{c} \in \mathbb{R}^{m-1} \times \mathbb{R}_{++}, \mathbf{f} \in \mathbb{R}^m_+,\ 0 = c_i - f_i = g_j, \forall i \in CF, \forall j \in G; CF \cup G = S^+$
$f_m = 0$
$f_m > 0$
Facet Class 1
Facet Class 2

Figure 22: Proof outline of Theorem 4

5.4 Computational Results

In this section, computational results are presented to illustrate the effectiveness of our two facet classes. In our first experiment, we investigate the reduction in the root node integrality gap due to our blending facets on uncapacitated single-consumer FCTPwB instances. Since our facets do not give the convex hull of X_{m_+,m_-} when $m_+ > 1$ and $m_- > 1$, this experiment provides empirical evidence concerning the strength of our facets with respect to the set X_{m_+,m_-}. In our second experiment, we solve capacitated multi-consumer FCTPwB instances to provable optimality and show that integrating our cuts in a branch-and-cut algorithm yields significant reductions in the overall solution time and the number of nodes explored in the search tree.

All experiments have the following characteristics: All computations were carried out on a Linux machine with kernel 2.6.18 running on a 64-bit x86 processor equipped with two Intel Xeon E5520 chips, which run at 2.27 GHz, and 32GB of RAM. The LP and MIP solvers of Gurobi 3.0 were used [52]. For every set of parameters, 100 instances were randomly generated. All cuts are generated via the separation routine described in Proposition 2. Specifically, for each good and each bad supplier, the most violated blending cuts are generated and are only added if the violation is at least $\epsilon := 0.0001$. Note that when multiple consumers are present, the number and set of good and bad suppliers differ for each consumer. Separation is performed for each consumer.

5.4.1 Uncapacitated Single-Consumer FCTPwB

In our first experiment, we present results for instances of the uncapacitated single-consumer FCTPwB. In light of Theorems 3 and 4, all instances have at least two good and bad suppliers so that the convex hull is not already known. Since our facets, along with the original formulation constraints, do not yield the convex hull of X_{m_+,m_-}, our main curiosity in this experiment is to obtain empirical evidence concerning how effective our cuts are at tightening the LP relaxation. Specifically, we aim to answer the following question: What is the reduction in the integrality gap due to our two facet classes and how many of these cuts are necessary to achieve this gap reduction? The integrality gap is defined as $(z^* - z^{LP})/z^*$, where z^* is the true optimal objective function value (computed in advance) and z^{LP} is the objective function value of the LP relaxation.

To answer this question, we could compare the integrality gap of the LP relaxation with that of a cutting plane algorithm in which only blending cuts are separated. However, in addition to this comparison, we may also want to know the value of our blending cuts when they are embedded in a MIP solver in which standard MIP cuts are used. To this end, we compare the integrality gap at the root node for four different options: the LP relaxation (denoted by 'LP' in the tables), Gurobi on its own, i.e., without blending cuts, ('GRB'), a user-implemented cutting plane algorithm ('User') in which only our blending cuts are added to the model until the LP relaxation ceases to improve by at least ϵ or no violated cuts are found, and Gurobi with both standard MIP cuts enabled and blending cuts added through a callback ('GRB+User'). We also experimented with turning off all default Gurobi cuts and having Gurobi use only our cuts through a callback. However, this option was almost always worse than default Gurobi and was always worse than our cutting plane implementation. Note that in this first experiment MIP preprocessing ('presolve') is turned *off* to understand how our blending cuts improve the quality of the original formulation.

A particular instance is generated as follows. First, we select the number of good and bad suppliers m_+ and m_-, respectively. Fixed costs are set such that $f_i = m - i + 1, \forall i \in S$. Unit cost are set such that $c_i = m + 1, \forall i \in S^+$, and $c_k = m + 1 + \Delta\text{bad}, \forall k \in S^-$, where $\Delta\text{bad} \in \mathbb{Z}_+$ is a parameter representing an increase in revenue (i.e., an incentive) for using bad suppliers. It is important to note that without an appreciable incentive for using bad suppliers, the optimal solution is trivial: send everything from a single good supplier. In this case, our blending cuts will not help. Nominal purity levels are generated as $\tilde{p}_i \sim \text{Normal}(0, 1), \forall i \in S$. To have exactly m_+ good and m_- bad suppliers, respectively, we sort the $\tilde{p}_i$'s in decreasing order, re-index so that $\tilde{p}_1 > \cdots > \tilde{p}_m$ and set $\tilde{p}^{\min} = (\tilde{p}_{m_+} + \tilde{p}_{m_+ + 1})/2$. Finally, we set $p_i = \tilde{p}_i - \tilde{p}^{\min}, \forall i \in S$.

The results are shown in Tables 17 and 18. The heading '# Good' refers to the number of good suppliers. The next four columns indicate the average integrality gap (%) at the root node of the branch-and-bound tree for the four different options discussed above.

To reiterate, this gap is exact since it is relative to the true optimal MIP solution. The remaining columns show cut-specific information. ‘Cuts (User)’ and ‘Cuts (GRB+User)’ refer to cut information associated with the ‘User’ and ‘GRB+User’ option, respectively. ‘LB’ and ‘LVUB’ denote the average number of lifted blending cuts (48) and lifted variable upper bound cuts (49) that were generated through separation, respectively. ‘Rounds’ refers to the average number of separation rounds, i.e., the average number of times an attempt to separate the current optimal solution to the LP relaxation with a blending cut.

The results in Tables 17 and 18 suggest that our blending cuts are effective at reducing the integrality gap of the model. In fact, the smallest gap is often achieved when only blending cuts are added. These results provide compelling empirical evidence that the subset of facets of X_{m_+,m_-} identified in Theorems 1 and 2 work well by themselves. We also see that when the number of suppliers is larger and when the incentive for using bad suppliers (Δbad) increases, our cuts are more valuable, i.e., the difference between the integrality gap of ‘GRB’ and ‘User’ and between ‘GRB’ and ‘GRB+User’ becomes more pronounced.

Given that blending cuts alone are so effective, one might assume that coupling blending cuts with standard MIP cuts added by Gurobi would further reduce the integrality gap. The results indicate that this is not the case when we simply add blending cuts as user cuts through a callback in Gurobi. It appears that with default settings Gurobi prefers not to generate cuts as aggressively as our implemented cutting plane method. Two possible explanations for this behavior are: (i) if the absolute value of the ratio (violation of cut)/(norm of cut) does not exceed Gurobi’s default tolerance, the cut may be rejected, and (ii) if two cuts are close to parallel, one of them may be rejected (Z. Gu, personal communication, August 13, 2010). At any rate, these results also serve as a useful reminder: Care has to be taken when setting up computational experiments and with interpreting computational results. If we had just used a callback implementation, we would have drawn completely different conclusions about the value of our blending cuts!

Table 17: Root information for the uncapacitated single-consumer FCTPwB with 20 suppliers

Data		Root Gap (%)				Cuts (User)			Cuts (GRB+User)		
Δbad	# Good	LP	GRB	User	GRB+User	LB	LVUB	Rounds	LB	LVUB	Rounds
5	5	1.19	**0.00**	**0.00**	**0.00**	0	6	1	0	4	1
5	10	1.45	**0.00**	0.01	**0.00**	3	49	3	1	17	1
5	15	3.63	**0.08**	0.32	0.19	38	208	15	7	226	7
15	5	16.72	0.42	**0.12**	1.27	33	86	8	51	56	5
15	10	29.67	13.93	**5.51**	14.44	308	652	43	102	801	11
15	15	17.47	2.66	**2.20**	7.21	315	300	69	58	712	12
25	5	24.52	2.28	**0.50**	4.97	110	196	14	119	438	9
25	10	22.59	9.92	**1.09**	8.57	301	439	39	108	874	11
25	15	16.71	2.70	**1.08**	7.13	281	155	59	56	716	12
50	5	9.29	0.66	**0.01**	0.87	73	51	6	79	301	5
50	10	13.51	4.19	**0.15**	2.66	221	154	26	96	742	11
50	15	13.04	4.82	**0.32**	4.63	227	79	47	58	715	13
100	5	4.20	0.22	**0.00**	0.23	49	24	3	56	215	3
100	10	7.49	0.68	**0.04**	0.35	142	68	15	69	536	8
100	15	8.97	2.84	**0.13**	2.08	175	63	36	55	642	12

Table 18: Root information for the uncapacitated single-consumer FCTPwB with 40 suppliers

Data		Root Gap (%)				Cuts (User)			Cuts (GRB+User)		
Δbad	# Good	LP	GRB	User	GRB+User	LB	LVUB	Rounds	LB	LVUB	Rounds
5	10	0.25	**0.00**	**0.00**	**0.00**	0	15	1	0	8	0
5	20	0.22	**0.00**	**0.00**	**0.00**	0	78	2	0	20	0
5	30	0.39	**0.00**	**0.00**	**0.00**	1	248	3	0	464	7
15	10	3.99	0.15	**0.05**	0.31	36	247	9	20	41	2
15	20	3.53	0.28	**0.18**	0.53	62	1214	16	29	246	3
15	30	12.74	7.83	**5.98**	8.68	997	1065	105	100	2471	10
25	10	19.04	4.23	**0.30**	5.76	111	744	17	135	138	5
25	20	24.65	12.73	**8.96**	12.46	1300	1939	74	283	4422	14
25	30	14.13	8.71	**3.91**	8.87	1295	1556	136	161	4346	16
50	10	23.89	14.94	**1.75**	12.10	1068	1610	51	304	2415	10
50	20	16.13	11.46	**2.77**	7.83	1741	2059	99	261	4583	13
50	30	12.69	7.53	**2.31**	6.86	1111	772	114	136	3738	14
100	10	9.60	4.91	**0.05**	5.10	597	487	24	219	1940	7
100	20	9.89	6.06	**0.77**	4.03	1439	999	78	255	4320	14
100	30	10.19	6.32	**0.69**	5.56	1050	357	106	134	3610	13

5.4.2 Capacitated Multi-Consumer FCTPwB

In our next experiment, we show the strength of our two cut classes for capacitated multi-consumer FCTPwB instances described by Formulation (45). In this capacitated setting, our inequalities remain valid, but may no longer be facet-defining. The set-up for this experiment resembles what was done above, except in addition to investigating the root

Table 19: FCTPwB Data Sets

Data Set	# Consumers	# Good Suppliers per Consumer
1	10	15,14,13,12,11,10,9,8,7,6
2	17	18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2

relaxation, we also observe that our cuts are effective at solving these instances to provable optimality. In some cases, embedding blending cuts within Gurobi reduces solution time by two orders of magnitude.

A particular instance is generated as follows. There are $m = 20$ suppliers and the number of consumers varies depending on data set used. Table 19 specifies the number of consumers as well as the number of good suppliers for each consumer. For example, in Data Set 1, the first consumer has 15 good suppliers; the last consumer has 6. As above, nominal purity levels are generated as Normal(0,1) random variables and purity differences are computed so that the appropriate number of good suppliers aligns with what is stated in Table 19. For each arc $(i, j) \in \mathcal{A}$, we set $f_{ij} = m-i-(j/m)$; $c_{ij} = m+1$ if $p_{ij} > 0$ and $c_{ij} = m+1+i+\Delta\text{bad}$ if $p_{ij} < 0$. We set $l_i = l_j = 0$, $u_i = n$, and $u_j = 1, \forall i \in S, j \in C$. Finally, we distinguish between *weakly* and *highly* capacitated instances in which arc capacities u_{ij} are randomly generated as Uniform(0.80,0.95) and Uniform(0.25,0.50), respectively. In the tables, weakly and highly capacitated instances are denoted with a 'W' and an 'H,' respectively.

The results are shown in Tables 20–23. Tables 20 and 21 present information related to the root node of the search tree while Tables 22 and 23 focus on information related to solving the instances to provable optimality. 'Cap' refers to the capacity of the instance. Note that MIP preprocessing ('presolve') is turned *on*, just as a user would do. Tables 20 and 21 report the same information reported in the first set of experiments. In Tables 22 and 23, under the '# Cuts' heading, 'LB' and 'LVUB' refer to the number of lifted blending and lifted variable upper bound cuts that were ever generated. '# Nodes' refers to the number of nodes that were explored in the search tree.

After solving the capacitated multi-consumer FCTPwB model to provable optimality and averaging the results, the following observations are apparent. No blending (user) cuts were ever generated after the root node. This does not necessarily mean that there are no violated blending cuts at nodes other than the root node. However, with default parameter settings Gurobi chooses never to execute our cut callback beyond the root node and therefore never attempts to generate blending cuts at nodes other than the root node. It should also be noted that default Gurobi cuts were almost never generated beyond the root node. In every case, fewer nodes in the branch-and-cut tree were explored when blending cuts were generated alongside default Gurobi cuts. This reduction in the number of nodes explored often led to an order of magnitude improvement in the overall solution time.

In contrast to what was observed in our first experiment, Gurobi often performed many more rounds of separation at the root node than our implemented cutting plane method in this second experiment. One possible explanation for this is that when arc capacities are introduced, our inequalities are no longer facet defining and are unable to reduce the integrality gap as much per iteration as in our first experiment. Meanwhile, with the introduction of arc capacities and multiple consumers, Gurobi is able to generate more of its own inequalities (30-40% of which are Gomory mixed-integer cuts and 25-35% of which are flow cover cuts). Note that arc capacities lead to multiple single-node flow cover sets and, therefore, greater potential for flow cover inequalities to be separated. This leads to more opportunities for us to generate more (weaker) inequalities, which in turn leads to more opportunities for Gurobi to generate more inequalities, and so forth. Thus we end up with many more separation rounds and slow convergence.

In preliminary experimentation, we also learned that when the parameter Δbad was large, it was important to place an upper bound on the number of each type of blending cut that can be generated or on the number of separation rounds. Without such a constraint, an excessive number of blending cuts could be generated at the root node, bogging down the computations at subsequent iterations, ultimately resulting in longer solution times than default Gurobi. To avoid this, we imposed an upper bound of 5000 rounds of separation for all of the instances solved in this second experiment. As a final comment, in general, weakly capacitated instances are much easier to solve. Since our cuts were developed for an uncapacitated model, it seems natural that they should perform better on weakly capacitated instances.

Table 20: Root information for FCTPwB Data Set 1

Data		Root Gap (%)				# Cuts (User)			# Cuts (GRB+User)		
Cap	Δbad	LP	GRB	User	GRB+User	LB	LVUB	Rounds	LB	LVUB	Rounds
W	5	40.55	31.65	15.24	**9.40**	1219	4380	50	31342	178102	2128
W	15	35.37	27.30	14.11	**9.62**	856	3062	27	2860	10049	143
W	25	29.69	23.80	12.76	**8.10**	339	1714	3	1522	6783	27
W	50	22.60	16.47	10.99	**5.93**	327	1773	3	1696	7787	46
W	100	16.93	12.01	9.79	**4.75**	316	1712	3	2897	9681	174
H	5	43.60	20.89	23.71	**13.65**	91	57	12	574	466	466
H	15	36.13	23.19	20.67	**12.31**	212	48	25	1169	302	417
H	25	30.15	21.08	18.35	**11.23**	311	64	34	1796	376	1159
H	50	24.35	20.37	14.25	**11.86**	431	82	47	742	159	106
H	100	15.59	11.65	9.11	**6.81**	415	74	38	675	128	65

Table 21: Root information for FCTPwB Data Set 2

Data		Root Gap (%)				# Cuts (User)			# Cuts (GRB+User)		
Cap	Δbad	LP	GRB	User	GRB+User	LB	LVUB	Rounds	LB	LVUB	Rounds
W	5	34.39	20.23	13.03	**7.97**	1473	4924	40	65557	281830	3665
W	15	30.49	18.59	12.38	**7.78**	629	2608	6	1717	7037	19
W	25	27.32	17.76	11.82	**7.76**	449	2035	3	1833	8021	20
W	50	21.36	13.50	10.37	**5.77**	447	2170	3	1760	8063	20
W	100	16.56	9.50	8.86	**3.71**	395	1879	3	2084	7346	51
H	5	24.54	7.37	13.99	**4.79**	240	69	30	1354	594	798
H	15	27.94	10.03	12.79	**5.30**	330	87	39	7061	1717	3646
H	25	26.22	11.71	12.80	**5.84**	450	98	53	6795	4927	2829
H	50	22.37	11.75	11.85	**6.29**	679	118	76	26381	3914	4568
H	100	15.06	7.96	9.29	**6.71**	684	118	68	12042	2878	3158

Table 22: Full solve information for FCTPwB Data Set 1

Data		Time (sec)		# Cuts		# Nodes	
Cap	Δbad	GRB	GRB+User	LB	LVUB	GRB	GRB+User
W	5	271.06	**7.42**	31342	178102	2018646	**3204**
W	15	217.10	**0.91**	2860	10049	1538488	**159**
W	25	59.97	**0.47**	1522	6783	443672	**17**
W	50	19.40	**0.55**	1696	7787	114445	**37**
W	100	59.42	**0.80**	2897	9681	300811	**167**
H	5	**0.40**	0.61	574	466	1433	**603**
H	15	2.41	**0.65**	1169	302	18523	**469**
H	25	28.48	**1.55**	1796	376	249101	**1485**
H	50	43.49	**0.26**	742	159	317443	**95**
H	100	29.95	**0.21**	675	128	218113	**53**

Table 23: Full solve information for FCTPwB Data Set 2

Data		Time (sec)		# Cuts		# Nodes	
Cap	Δbad	GRB	GRB+User	LB	LVUB	GRB	GRB+User
W	5	931.71	**12.45**	65557	281830	4176033	**5627**
W	15	221.64	**0.61**	1717	7037	962686	**8**
W	25	123.12	**0.56**	1833	8021	465171	**8**
W	50	63.83	**0.49**	1760	8063	230501	**8**
W	100	11.18	**0.60**	2084	7346	42606	**38**
H	5	2.11	**1.57**	1354	594	10382	**1089**
H	15	**5.90**	6.58	7061	1717	27501	**5500**
H	25	169.14	**46.41**	6795	4927	843040	**148599**
H	50	265.60	**159.21**	26381	3914	1028405	**444680**
H	100	273.77	**129.75**	12042	2878	1076130	**372344**

5.5 Future Research

We would like to extend our two facet classes in two ways. First, it would be interesting to determine similar cuts for the *capacitated* FCTPwB. We attempted to do this for the case of a single good supplier and many bad suppliers. However, even for this simple set, the form of the cuts became complicated. Second, it would be interesting to construct facet classes when the right-hand-side b, which in our model is set to 0, of the blending constraint $\sum_{i\in S} p_i x_i \geq b$ takes nonzero values. Obtaining facets for this set, i.e., $X := \{(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^m_+ \times \{0,1\}^m : \sum_{i\in S} p_i x_i \geq b, \sum_{i\in S} x_i \leq 1, x_i \leq y_i, \forall i \in S\}$, could have greater appeal to the MIP community as they could be used to solve general MIP instances in which this structure appears. Our initial efforts into the question suggest that when $b > 0$ Facet Class 2 inequalities remain valid and facet-defining. However, we also found that "new" facets surface. We believe that lifting arguments will help to resolve this issue.

Although not presented here, we have also tested our blending inequalities when there are multiple blending constraints present. Specifically, suppose that the single blending constraint $\sum_{i\in S} p_i x_i \geq 0$ is replaced by $\sum_{i\in S} p_i^a x_i \geq 0, \forall a \in A$, where A is a set of attributes and p_i^a is the purity difference for supplier i with respect to attribute $a \in A$. We have found that applying our cuts for each attribute independently can reduce the root integrality gap by 80% on instances similar to those considered in Section 5.4.1. It would be interesting to explore how our cuts perform on multi-period models as well as multi-period models with multiple attributes.

CHAPTER VI

CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

6.1 Conclusions

The thrust of this thesis is the development of effective solution methods for large-scale inventory routing problems arising in a maritime setting. With a monopoly on moving large volumes of goods between continents [28], the maritime transportation industry is poised for new technology and innovations to improve efficiency. The models and methods introduced here have the potential to improve the decision support systems responsible for the movement of many valuable goods that are shipped by sea.

In Chapter 2, we introduce a core mixed-integer programming model for a particular class of MIRPs known as MIRPs with inventory tracking at every port. By and large, the literature on maritime routing and scheduling has been concerned with new applications with specific constraints and modeling features closely tied to each application. While these papers are important contributions as they expose the growing interest in the domain, it is often difficult to determine if the associated algorithmic techniques apply more generally. Our first contribution is to isolate a core substructure found in numerous applications and to describe ways of approaching this substructure. A comprehensive survey of this class of MIRPs is included along with a unified discussion of common enhancements. Finally, we present the first publicly available library of benchmark instances for this class of MIRPs. We hope that this library helps spur interest and development in this research domain.

Chapter 3 develops a two-stage decomposition method for solving the class of MIRPs presented in Chapter 2. There is a fundamental difference in the philosophy of our decomposition with that of traditional approaches for maritime (inventory and cargo) routing problems. Traditional decomposition methods use column generation strategies to decompose these problems into a master problem and multiple subproblems. Each subproblem can be intrepreted as the problem a *vessel* manager solves to decide if there exists a more profitable route for his vessel. The master problem is the problem a system manager solves to assign vessels to routes and ensure that all other constraints at ports are satified. Our approach more closely resembles Benders decomposition and interprets each subproblem as the problem a *regional* manager solves to route vessels through his region and ensure that all port-specific constraints are satisfied. Meanwhile, our master problem represents the problem a system manager solves to determine how vessels are routed from region to region. Empirically, our algorithm yields high-quality solutions and good bounds to large-scale instances.

Our contributions in Chapter 4 take advantage of the recent successes of approximate dynamic programming (ADP) in dynamic fleet management problems to quickly generate good solutions to maritime inventory routing planning problems with a long horizon. ADP has emerged as a powerful framework for solving stochastic dynamic problems in the transportation community. Borrowing from several scientific disciplines, including artificial intelligence, mathematical programming, and simulation, it transforms a multi-stage problem into a sequence of smaller problems. That is, it uses a form of time (or stage) decomposition. Using suitably chosen value function approximations, we solve a subproblem in each time period to decide how available vessels should be routed, and, as a consequence, how inventory should be distributed. For large instances with many ports and dozens of vessels, our ADP framework produces good solutions much more quickly than a commercial solver solving a single problem that involves all time periods in the planning horizon.

Chapter 5 addresses a problem that arises in maritime inventory routing, but is applicable in an even broader context. Fixed-charge transportation problems involve the routing of goods through a distribution network and require that a fixed charge is incurred when a particular arc is used. In some practical applications, it is also desirable to mix or blend these goods to lower costs. In this chapter, we present the first polyhedral analysis of the fixed-charge transportation problem with product blending (FCTPwB). This analysis includes two new families of facet-defining valid inequalities which fully exploit the presence of a linear blending requirement. We prove that in two special cases these facet classes, along with the continuous relaxation of the original formulation constraints, yield the convex hull of the feasible region. These results lend theoretical support to our claim that our two facet classes are strong. Computational results illustrate the effectiveness of our cuts at reducing the integrality gap and solution time on instances when they are no longer facet-defining.

6.2 Future Research Directions

We end with a discussion of several avenues for further research.

6.2.1 Arc-based vs. path-based formulations

As mentioned in Chapters 2 and 3, a useful contribution would be to provide guidelines as to when path-based formulations of a MIRP are superior to arc-based models. For complicated MIRPs having multiple split pickups and split deliveries, we believe that arc-based formulations are preferred. This is based on several observations. First, MIP-based local search heuristics appear to be better suited for making small local changes, e.g., modifying part of a vessel's path/route, as opposed to making sweeping changes to the current solution. Second, path-based formulation that use column generation often have to generate many columns. On the other hand, MIRPs with time windows (i.e., inventory is not tracked at every port; instead, minimum quantities must be loaded or discharged in a set of given time windows) are more amenable to path-based methods. It would be instructive to know precisely why one approach is favorable in certain settings.

6.2.2 Integration of more supply chain components

Supply chain networks involve more than simply routing vehicles from supply points to consumption points. From initial procurement to final delivery of goods to customers, there are many decisions to be made and these decisions are often intertwined. Although we have focused exclusively on one component of the supply chain, moving products from production regions to regions of high demand, it is interesting to consider a larger integrated network in which inventory routing problems are solved to answer more strategic questions related to fleet size and new business opportunities. Motivating factors are also discussed in [8] and [47].

6.2.3 Uncertainty and robust solutions

Just as with other modes of transport, the maritime sector is rife with uncertain data and decisions that need to be made without perfect knowledge. With this reality, one could argue that the prevailing emphasis on deterministic problems within the maritime transportation community is misaligned. The case studies described in Chapter 2 indicate that there is value in solving deterministic problems. Likewise, it is typical to first study deterministic versions of a problem before addressing stochastic variants. Nevertheless, building decision support tools that generate robust solutions is always of great practical interest.

Within the maritime sector, the primary sources of uncertainty include: demand fluctuations due to seasonality and variability in climate; weather delays during travel; mechanical delays at ports; strikes at ports; spot charter rates; spot market demands and prices. However, the most prominent sources of uncertainty depend on the application. If delays due to strikes and weather are most important, then one may seek to find certain types of solutions, or policies, that give greater flexibility in the long run. For example, the airline industry typically assigns aircraft to fly simple cyclic routes, e.g., Boston-Los Angeles and Los Angeles-Boston, as opposed to complicated multi-leg routes, in order to allow for smoother recoveries after a disruption. It is not clear that dedicating certain vessels to cyclic routes leads to good solutions or is even necessary in a maritime setting.

Several attempts have already been made, e.g., [24,30,59,68], but researchers in this area have really only scratched the surface. Ultimately, there needs to be a way of producing solutions that balance risk and reward. One approach is to generate many near-optimal solutions so that a decision-maker has a menu of options from which to choose. Another approach, akin to what is done in portfolio optimization, is to explicitly model risk and reward so that an efficiency frontier can be created and a solution fitting the decision-maker's risk profile can be selected.

6.2.4 Approximate dynamic programming for maritime problems

Our application of ADP was in the spirit of a fast heuristic as it was used to generate good solutions quickly without any regard for a bound. However, another reason we chose to employ an ADP approach is due to the fact that the ADP framework can easily be extended to allow for uncertainty in the data. This ease of transitioning from a deterministic setting to a stochastic one is typically absent when working with MIP models. It would be interesting to explore some of the themes mentioned above on uncertainty using the ADP approach.

A second research stream is to pursue parallelization. As recognized in [99, 101], when regional decisions can be made independently, the time t subproblem can be decomposed by region and all regional problems can be solved in parallel. Alternatively, one could attempt to run many ADP solvers in parallel with different value function approximations and different updating procedures to converge to a good approximation as quickly as possible.

A third open question is: Why do piecewise linear *concave* value functions approximations lead to good solutions? Practically, concavity leads to models that are can be solved with relative ease. Empirically, numerous studies have found that these approximations work well. Theoretically, however, it is still not clear why piecewise linear concave functions perform so well. Studying the true dynamic programming value function of the dynamic resource allocation problem may help to address this question.

It would also be interesting to investigate the benefit of incorporating vessel states into the value function approximation. Our value function approximations essentially "throw away" information about the state of each vessel and only keep track of future inventories at ports. It may be possible to generate superior solutions by valuing vessel states in addition to inventory states.

Finally, a more general and lofty research question concerns the development of ADP as a modeling and solving framework. A drawback of ADP is the lack of commercial and non-commercial software associated with it. Today, the mixed integer programming community is fraught with software for modeling and solving MIP models. A layman with access to Microsoft Excel could potentially solve a small MIP simply by entering several lines of data into a spreadsheet. No such software is available for ADP methods. This means that special-purpose code must be written and debugged, leading to a potentially large investment of time. Moreover, each tweak to the model may lead to an algorithmic change. Worse yet, even when the algorithm is working properly, this does not mean that the algorithm is working well. Leaders in the field [78] admit that fine-tuning an ADP algorithm can be a frustrating process. Many of these difficulties need to be overcome before ADP gains widespread acceptance as a tool for practitioners.

REFERENCES

[1] Achterberg, T. and Berthold, T., "Improving the feasibility pump," *Discrete Optimization*, vol. 4, no. 1, pp. 77 – 86, 2007.

[2] Achterberg, T. and Lodi, A., "Hybrid solving techniques," in *Hybrid Optimization* (van Hentenryck, P. and Milano, M., eds.), vol. 45 of *Springer Optimization and Its Applications*, pp. 169–190, Springer New York, 2011.

[3] Agra, A., Andersson, H., Christiansen, M., and Wolsey, L. A., "A maritime inventory routing problem: Time discrete formulations and valid inequalities," *Submitted to Networks*, 2011.

[4] Aguado, J. S., "Fixed charge transportation problems: a new heuristic approach based on lagrangean relaxation and the solving of core problems," *Annals of Operations Research*, vol. 172, no. 1, pp. 45–69, 2009.

[5] Al-Khayyal, F. and Hwang, S.-J., "Inventory constrained maritime routing and scheduling for multi-commodity liquid bulk, part I: Applications and model," *European Journal of Operational Research*, vol. 176, no. 1, pp. 106–130, 2007.

[6] Andersson, H., "A maritime pulp distribution problem," *INFOR*, vol. 49, no. 2, pp. 125–138, 2011.

[7] Andersson, H., Christiansen, M., and Fagerholt, K., "Transportation planning and inventory management in the LNG supply chain," in *Energy, Natural Resources and Environmental Economics*, Springer, 2010.

[8] Andersson, H., Hoff, A., Christiansen, M., Hasle, G., and Lokketangen, A., "Industrial aspects and literature survey: Combined inventory management and routing," *Computers & Operations Research*, vol. 37, no. 9, pp. 1515–1536, 2010.

[9] Atamtürk, A., "Flow pack facets of the single node fixed-charge flow polytope," *Operations Research Letters*, vol. 29, no. 3, pp. 107–114, 2001.

[10] Balas, E. and Martin, C., "Pivot and complement: a heuristic for 0-1 programming," *Management Science*, vol. 26, no. 1, pp. 86–96, 1980.

[11] Balas, E., Schmieta, S., and Wallace, C., "Pivot and shift: a mixed integer programming heuristic," *Discrete Optimization*, vol. 1, no. 1, pp. 3–12, 2004.

[12] Balinski, M. L., "Fixed-cost transportation problems," *Naval Research Logistics Quarterly*, vol. 8, no. 1, pp. 41–54, 1961.

[13] Ballou, R., *Business Logistics Management.* Upper Saddle River NJ: Prentice Hall, 4th ed., 1998.

[14] Bastert, O., Hummel, B., and de Vries, S., "A generalized Wedelin heuristic for integer programming," *INFORMS Journal on Computing*, vol. 22, no. 1, pp. 93–107, 2010.

[15] Bell, G. J., Lamar, B. W., and Wallace, C. A., "Capacity improvement, penalties, and the fixed charge transportation problem," *Naval Research Logistics*, vol. 46, no. 4, pp. 341–355, 1999.

[16] Benders, J., "Partitioning procedures for solving mixed-variables programming problems," *Numerische Mathematik*, vol. 4, no. 1, pp. 238–252, 1962.

[17] Berthold, T., "Primal heuristics for mixed integer programs," *Master's thesis, Technische Universitat Berlin*, 2006.

[18] Bertsekas, D. P. and Tsitsiklis, J. N., *Neuro-Dynamic Programming.* Belmont, MA: Athena Scientific, 1996.

[19] Bilgen, B. and Ozkarahan, I., "A mixed-integer linear programming model for bulk grain blending and shipping," *International Journal of Production Economics*, vol. 107, no. 2, pp. 555–571, 2007.

[20] Blair, C. and Jeroslow, R., "The value function of a mixed integer program: I," *Discrete Mathematics*, vol. 19, no. 2, pp. 121–138, 1977.

[21] Blair, C. and Jeroslow, R., "The value function of a mixed integer program: II," *Discrete Mathematics*, vol. 25, no. 1, pp. 7–19, 1979.

[22] Campbell, A., Clarke, L., Kleywegt, A., and Savelsbergh, M. W. P., "The inventory routing problem," in *Fleet Management and Logistics* (Crainic, T. G. and Laporte, G., eds.), pp. 95–113, Kluwer, 1998.

[23] Christiansen, M., "Decomposition of a combined inventory and time constrained ship routing problem," *Transportation Science*, vol. 33, no. 1, pp. 3–16, 1999.

[24] Christiansen, M. and Fagerholt, K., "Robust ship scheduling with multiple time windows," *Naval Research Logistics*, vol. 49, no. 6, pp. 611–625, 2002.

[25] Christiansen, M. and Fagerholt, K., "Maritime inventory routing problems," in *Encyclopedia of Optimization* (Floudas, C. A. and Pardalos, P. M., eds.), pp. 1947–1955, Springer-Verlag, second ed., 2009.

[26] Christiansen, M. and Fagerholt, K., "Some thoughts on research directions for the future: Introduction to the special issue in maritime transportation," *INFOR*, vol. 49, no. 2, pp. 75–77, 2011.

[27] Christiansen, M., Fagerholt, K., Flatberg, T., Haugen, O., Kloster, O., and Lund, E. H., "Maritime inventory routing with multiple products: A case study from the cement industry," *European Journal of Operational Research*, vol. 208, no. 1, pp. 86–94, 2011.

[28] Christiansen, M., Fagerholt, K., Nygreen, B., and Ronen, D., "Maritime transportation," in *Transportation, Handbooks in Operations Research and Management Science* (Barnhart, C. and Laporte, G., eds.), vol. 14, pp. 189–284, Elsevier, 2007.

[29] Christiansen, M., Fagerholt, K., Nygreen, B., and Ronen, D., "Ship routing and scheduling in a new millennium," *Submitted*, 2012.

[30] Christiansen, M. and Nygreen, B., "Robust inventory ship routing by column generation," in *Column Generation* (Desaulniers, G., Desrosiers, J., and Solomon, M. M., eds.), pp. 197–224, Springer US, 2005.

[31] Christof, T. and Løbel, A., "PORTA: a polyhedron representation transformation algorithm," 1997.

[32] Christophel, P. M., "An improved heuristic for the MOPS mixed integer programming solver," *Master's thesis, Unversität Paderborn*, 2005.

[33] Danna, E., Rothberg, E., and Pape, C., "Exploring relaxation induced neighborhoods to improve MIP solutions," *Mathematical Programming*, vol. 102, no. 1, pp. 71–90, 2005.

[34] Dauzère-Pérès, S., Nordli, A., Olstad, A., Haugen, K., Koester, U., Myrstad, P. O., Teistklub, G., and Reistad, A., "Omya hustadmarmor optimizes its supply chain for delivering calcium carbonate slurry to european paper manufacturers," *Interfaces*, vol. 37, no. 1, pp. 39–51, 2007.

[35] DeWitt, C. W., Lasdon, L. S., Waren, A. D., Brenner, D. A., and Melhem, S. A., "OMEGA - an improved gasoline blending system for texaco," *Interfaces*, vol. 19, no. 1, pp. 85–101, 1989.

[36] Engineer, F. G., Furman, K. C., Nemhauser, G. L., Savelsbergh, M. W. P., and Song, J.-H., "A Branch-Price-And-Cut algorithm for single product maritime inventory routing," *Operations Research*, vol. 60, no. 1, pp. 106–122, 2012.

[37] Fischetti, M. and Lodi, A., "Local branching," *Mathematical Programming*, vol. 98, no. 1, pp. 23–47, 2003.

[38] Fischetti, M., Glover, F., and Lodi, A., "The feasibility pump," *Mathematical Programming*, vol. 104, no. 1, pp. 91–104, 2005.

[39] Fischetti, M., Lodi, A., and Salvagnin, D., "Just MIP it!," in *Matheuristics* (Maniezzo, V., Stutzle, T., Voss, S., and Sharda, R., eds.), vol. 10 of *Annals of Information Systems*, pp. 39–70, Springer US, 2010.

[40] Fodstad, M., Uggen, K. T., Rømo, F., Lium, A., and Stremersch, G., "LNGScheduler: a rich model for coordinating vessel routing, inventories and trade in the liquefied natural gas supply chain," *Journal of Energy Markets*, vol. 3, no. 4, pp. 31–64, 2010.

[41] Furman, K. C., Song, J.-H., Kocis, G. R., McDonald, M. K., and Warrick, P. H., "Feedstock routing in the ExxonMobil downstream sector," *Interfaces*, vol. 41, no. 2, pp. 149–163, 2011.

[42] George, A. and Powell, W., "Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming," *Machine learning*, vol. 65, no. 1, pp. 167–198, 2006.

[43] Ghosh, S., "DINS, a MIP improvement heuristic," *Integer Programming and Combinatorial Optimization*, pp. 310–323, 2007.

[44] Glover, F., Amini, M., and Kochenberger, G. A., "Parametric ghost image processes for fixed-charge problems: A study of transportation networks," *Journal of Heuristics*, vol. 11, no. 4, pp. 307–336, 2005.

[45] Godfrey, G. and Powell, W., "An adaptive dynamic programming algorithm for dynamic fleet management, I: Single period travel times," *Transportation Science*, vol. 36, no. 1, pp. 21–39, 2002.

[46] Godfrey, G. A. and Powell, W. B., "An adaptive dynamic programming algorithm for dynamic fleet management, II: multiperiod travel times," *Transportation Science*, vol. 36, no. 1, pp. 40–54, 2002.

[47] Goel, V., Furman, K. C., Song, J.-H., and El-Bakry, A. S., "Large neighborhood search for LNG inventory routing," *Submitted*, 2012.

[48] Grønhaug, R. and Christiansen, M., "Supply chain optimization for the liquefied natural gas business," *Innovations in Distribution Logistics, Lecture Notes in Economics and Mathematical Systems*, vol. 619, pp. 195—-218, 2009.

[49] Grønhaug, R., Christiansen, M., Desaulniers, G., and Desrosiers, J., "A Branch-and-Price method for a liquefied natural gas inventory routing problem," *Transportation Science*, vol. 44, no. 3, pp. 400–415, 2010.

[50] Gu, Z. H., Nemhauser, G. L., and Savelsbergh, M. W. P., "Lifted flow cover inequalities for mixed 0-1 integer programs," *Mathematical Programming*, vol. 85, no. 3, pp. 439–467, 1999.

[51] Gu, Z. H., Nemhauser, G. L., and Savelsbergh, M. W. P., "Sequence independent lifting in mixed integer programming," *Journal of Combinatorial Optimization*, vol. 4, no. 1, pp. 109–129, 2000.

[52] Gurobi Optimization Inc. Gurobi Optimizer Reference Manual, Version 3.0, 2010.

[53] Halvorsen-Weare, E. and Fagerholt, K., "Routing and scheduling in a liquefied natural gas shipping problem with inventory and berth constraints," *Annals of Operations Research*, pp. 1–20, 2011.

[54] Hennig, F., Nygreen, B., Christiansen, M., Fagerholt, K., Furman, K., Song, J.-H., Kocis, G., and Warrick, P., "Maritime crude oil transportation a split pickup and split delivery problem," *European Journal of Operational Research*, vol. 218, no. 3, pp. 764 – 774, 2012.

[55] Hennig, F., Nygreen, B., Furman, K. C., and Song, J.-H., "Alternative approaches to the crude oil tanker routing and scheduling problem with split pickup and split delivery," *Submitted to Computers & Operations Research*, 2009.

[56] Hewitt, M., Nemhauser, G. L., and Savelsbergh, M. W. P., "Branch-and-price guided search for integer programs with an application to the multicommodity fixed-charge network flow problem," *INFORMS Journal on Computing*, 2012.

[57] Hewitt, M., Nemhauser, G. L., Savelsbergh, M. W. P., and Song, J.-H., "A Branch-and-Price guided search approach to maritime inventory routing," *Submitted*, 2011.

[58] Hirsch, W. M. and Dantzig, G. B., "The fixed charge problem," *Naval Research Logistics Quarterly*, vol. 15, no. 3, pp. 413–424, 1968.

[59] Hwang, H., Visoldilokpun, S., and Rosenberger, J., "A branch-and-price-and-cut method for ship scheduling with limited risk," *Transportation science*, vol. 42, no. 3, pp. 336–351, 2008.

[60] Hwang, S.-J., "Inventory constrained maritime routing and scheduling for multi-commodity liquid bulk," *Ph.D. thesis, Georgia Institute of Technology*, 2005.

[61] Karuppiah, R., Furman, K. C., and Grossmann, I. E., "Global optimization for scheduling refinery crude oil operations," *Computers & Chemical Engineering*, vol. 32, no. 11, pp. 2745–2766, 2008.

[62] Klose, A., "Algorithms for solving the single-sink fixed-charge transportation problem," *Computers & Operations Research*, vol. 35, no. 6, pp. 2079–2092, 2008.

[63] Lamar, B. W. and Wallace, C. A., "Revised-modified penalties for fixed charge transportation problems," *Management Science*, vol. 43, no. 10, pp. 1431–1436, 1997.

[64] Lawrence, S. A., *International Sea Transport: The Years Ahead.* Lexington, MA: Lexington Books, 1972.

[65] Lee, H. M., Pinto, J. M., Grossmann, I. E., and Park, S., "Mixed-integer linear programming model for refinery short-term scheduling of crude oil unloading with inventory management," *Industrial & Engineering Chemistry Research*, vol. 35, no. 5, pp. 1630–1641, 1996.

[66] Li, J., Karimi, I., and Srinivasan, R., "Efficient bulk maritime logistics for the supply and delivery of multiple chemicals," *Computers & Chemical Engineering*, vol. 34, no. 12, pp. 2118–2128, 2010.

[67] Liu, C.-M. and Sherali, H. D., "A coal shipping and blending problem for an electric utility company," *Omega*, vol. 28, no. 4, pp. 433–444, 2000.

[68] McKinnon, K. and Yu, Y., "Solving stochastic ship fleet routing problems with inventory management using branch and price," *University of Edinburgh Working Paper*, 2011.

[69] Mendez, C. A., Grossmann, I. E., Harjunkoski, I., and Kabore, P., "A simultaneous optimization approach for off-line blending and scheduling of oil-refinery operations," *Computers & Chemical Engineering*, vol. 30, no. 4, pp. 614–634, 2006.

[70] Misener, R. and Floudas, C. A., "Advances for the pooling problem: modeling, global optimization, and computational studies survey," *Applied and Computational Mathematics*, vol. 8, no. 1, pp. 3–22, 2009.

[71] Nemhauser, G. L. and Wolsey, L. A., *Integer and Combinatorial Optimization.* New York: Wiley, 1988.

[72] Padberg, M. W., Van Roy, T. J., and Wolsey, L. A., "Valid linear inequalities for fixed charge problems," *Operations Research*, vol. 33, no. 4, pp. 842–861, 1985.

[73] Palekar, U. S., Karwan, M. H., and Zionts, S., "A branch-and-bound method for the fixed charge transportation problem," *Management Science*, vol. 36, no. 9, pp. 1092–1105, 1990.

[74] Papageorgiou, D. J., Keha, A. B., Nemhauser, G. L., and Sokol, J., "Quickly finding good solutions to long-horizon maritime inventory routing problems," *Georgia Tech ISyE Working paper*, 2012.

[75] Papageorgiou, D. J., Keha, A. B., Nemhauser, G. L., and Sokol, J., "A two-stage decomposition algorithm for single product maritime inventory routing," *Georgia Tech ISyE Working paper*, 2012.

[76] Persson, J. A. and Göthe-Lundgren, M., "Shipment planning at oil refineries using column generation and valid inequalities," *European Journal of Operational Research*, vol. 163, no. 3, pp. 631–652, 2005.

[77] Pochet, Y. and Wolsey, L. A., *Production Planning by Mixed Integer Programming.* Springer Series in Operations Research and Financial Engineering, Springer, 2006.

[78] Powell, W., *Approximate Dynamic Programming: Solving the curses of dimensionality.* Wiley-Interscience, 2007.

[79] Psaraftis, H., "Foreword to the focused issue on maritime transportation," *Transportation Science*, vol. 33, 1999.

[80] Rakke, J., Andersson, H., Christiansen, M., and Desaulniers, G., "Branch-price-and-cut for creating an annual delivery program of multi-product liquefied natural gas," *Working Paper*, 2012.

[81] Rakke, J. G., Stålhane, M., Moe, C. R., Christiansen, M., Andersson, H., Fagerholt, K., and Norstad, I., "A rolling horizon heuristic for creating a liquefied natural gas annual delivery program," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 5, pp. 896–911, 2011.

[82] Rigby, B., Lasdon, L. S., and Waren, A. D., "The evolution of texaco blending systems - from OMEGA to STARBLEND," *Interfaces*, vol. 25, no. 5, pp. 64–83, 1995.

[83] Rodrigue, J., Comtois, C., and Slack, B., *The geography of transport systems.* Routledge, 2nd ed., 2009.

[84] Romo, F., Tomasgard, A., Hellemo, L., Fodstad, M., Eidesen, B. H., and Pedersen, B., "Optimizing the Norwegian natural gas production and transport," *Interfaces*, vol. 39, no. 1, pp. 46–56, 2009.

[85] Ronen, D., "Ship scheduling: The last decade," *European Journal of Operational Research*, vol. 71, pp. 325–333, 1993.

[86] Ronen, D., "Marine inventory routing: shipments planning," *Journal of the Operational Research Society*, vol. 53, pp. 108–114, 2002.

[87] Rothberg, E., "An evolutionary algorithm for polishing mixed integer programming solutions," *INFORMS Journal on Computing*, vol. 19, no. 4, pp. 534–541, 2007.

[88] Ruszczynski, A., "Post-decision states and separable approximations are powerful tools of approximate dynamic programming," *INFORMS Journal on Computing*, vol. 22, no. 1, pp. 20–22, 2010.

[89] Savelsbergh, M. W. P. and Song, J.-H., "An optimization algorithm for the inventory routing problem with continuous moves," *Computers & Operations Research*, vol. 35, no. 7, pp. 2266–2282, 2008.

[90] Shen, Q., Chu, F., and Chen, H., "A Lagrangian relaxation approach for a multimode inventory routing problem with transshipment in crude oil transportation," *Computers & Chemical Engineering*, vol. 35, no. 10, pp. 2113–2123, 2011.

[91] Siswanto, N., Essam, D., and Sarker, R., "Solving the ship inventory routing and scheduling problem with undedicated compartments," *Computers & Industrial Engineering*, vol. 61, no. 2, pp. 289–299, 2011.

[92] Song, J.-H. and Furman, K. C., "A maritime inventory routing problem: Practical approach," *Computers & Operations Research*, vol. To appear, 2010.

[93] Stahlbock, R. and Voss, S., "Operations research at container terminals: a literature update," *OR Spectrum*, vol. 30, no. 1, pp. 1–52, 2008.

[94] Stålhane, M., Rakke, J. G., Moe, C. R., Andersson, H., Christiansen, M., and Fagerholt, K., "A construction and improvement heuristic for a liquefied natural gas inventory routing problem," *Comput. Ind. Eng.*, vol. 62, no. 1, pp. 245–255, 2012.

[95] Steenken, D., Voss, S., and Stahlbock, R., "Container terminal operation and operations research-a classification and literature review," *OR spectrum*, vol. 26, no. 1, pp. 3–49, 2004.

[96] Stopford, M., *Maritime Economics.* New York, NY: Taylor & Francis, 3rd ed., 2008.

[97] Sun, M., Aronson, J. E., McKeown, P. G., and Drinka, D., "A tabu search heuristic procedure for the fixed charge transportation problem," *European Journal of Operational Research*, vol. 106, no. 2-3, pp. 441–456, 1998.

[98] Sutton, R. and Barto, A., *Reinforcement learning: An introduction.* Cambridge University Press, 1998.

[99] Topaloglu, H. and Powell, W., "A distributed decision-making structure for dynamic resource allocation using nonlinear functional approximations," *Operations Research*, vol. 53, no. 2, pp. 281–297, 2005.

[100] Topaloglu, H., "A parallelizable dynamic fleet management model with random travel times," *European Journal of Operational Research*, vol. 175, no. 2, pp. 782 – 805, 2006.

[101] Topaloglu, H., "A parallelizable and approximate dynamic programming-based dynamic fleet management model with random travel times and multiple vehicle types," *Dynamic Fleet Management: Concepts, Systems, Algorithms and Case Studies*, pp. 65–93, 2007.

[102] Topaloglu, H. and Powell, W. B., "Dynamic-programming approximations for stochastic time-staged integer multicommodity-flow problems," *INFORMS Journal on Computing*, vol. 18, no. 1, pp. 31–42, 2006.

[103] Toriello, A., Nemhauser, G. L., and Savelsbergh, M. W. P., "Decomposing inventory routing problems with approximate value functions," *Naval Research Logistics*, vol. 57, pp. 718–727, 2010.

[104] Uggen, K., Fodstad, M., and Nørstebø, V., "Using and extending fix-and-relax to solve maritime inventory routing problems," *TOP*, pp. 1–23, 2011.

[105] United Nations Conference on Trade and Development (UNCTAD), *Review of Maritime Transport.* 2011.

[106] United States Energy Information Administration, *Annual Energy Review.* 2009.

[107] Van Roy, T. J. and Wolsey, L. A., "Valid inequalities and separation for uncapacitated fixed charge networks," *Operations Research Letters*, vol. 4, no. 4, pp. 105–112, 1985.

[108] Wolsey, L. A., *Integer Programming.* Wiley, 1998.

VITA

Dimitri Jason Papageorgiou was born in Ann Arbor, Michigan on 9 March 1980. At the age of 11, he moved with his family to North Carolina, where he attended middle school, high school, and college. In 2002, he earned a B.S. in Mathematics (with highest distinction) from the University of North Carolina at Chapel Hill, where he was first introduced to the field of Operations Research. In 2004, he earned an M.S. in Operations Research and Industrial Engineering from the University of Texas at Austin. His masters research on problems in stochastic network interdiction was supervised by Professor David Morton. He worked as a summer researcher at the NASA Langley Research Center in 2001, the RAND Corporation in 2002, and Lawrence Livermore National Laboratory in 2003.

After completing his masters degree, he joined Raytheon's world class Center for Discrimination as an Operations Research analyst (2004 - 2011) in Woburn, MA. There, he served as a technical expert responsible for the design, implementation, testing, and analysis of sophisticated sensor resource management models and algorithms for multi-million dollar R&D programs. His optimization software played a key role in the Missile Defense Agency's 2008 X-Lab demonstration by performing real-time data fusion between multiple sensor systems; this work is published in IEEE and the Journal of Advances in Information Fusion. During his time in New England, he was a visiting graduate student at the Massachusetts Institute of Technology.

At Georgia Tech, Dimitri collaborated with researchers from ExxonMobil Corporate Strategic Research to develop cutting-edge mathematical optimization techniques for modeling and solving large-scale supply chain and logistics problems. Dimitri is a recipient of a U.S. National Science Foundation Graduate Research Fellowship; he was the sole recipient of this prestigious honor for the award year in the field of Operations Research.