Two-Stage Decomposition Algorithms for Single Product Maritime Inventory Routing

Dimitri J. Papageorgiou[1], Ahmet B. Keha[2], George L. Nemhauser[3], Joel Sokol[3]

[1]Corporate Strategic Research
ExxonMobil Research and Engineering Company
1545 Route 22 East, Annandale, NJ 08801
dimitri.j.papageorgiou@exxonmobil.com

[2]Automation, Optimization, and Global Support Department
ExxonMobil Research and Engineering Company
3225 Gallows Rd., Fairfax, VA 22037
ahmet.b.keha@exxonmobil.com

[3]H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology
765 Ferst Drive NW, Atlanta, Georgia, 30332
{gnemhaus,jsokol}@isye.gatech.edu

Abstract

We present two decomposition algorithms for single product deep-sea maritime inventory routing problems (MIRPs) that possess a core substructure common in many real-world applications. The problem involves routing vessels, each belonging to a particular vessel class, between loading and discharging ports, each belonging to a particular region. Our algorithms iteratively solve 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 load and discharge quantities are determined. Computational experience shows that an integrated approach that combines these two algorithms is vastly superior to solving the problem directly with a commercial mixed-integer programming solver.

Keywords: aggregation, decomposition, deterministic inventory routing, lot-sizing, maritime transportation, mixed-integer linear programming.

1 Introduction

Inventory routing problems (IRPs) encompass a broad class of logistics problems that arise naturally in supply chain and distribution systems. IRPs have gained increasing attention in the past two decades as they are a fundamental tool in vendor managed inventory (VMI), a policy in which a supplier manages both the inventory and its distribution for its customers [5, 7, 10]. They involve the integration and coordination of two components of the logistics value chain: inventory management and vehicle routing. Although the use of IRPs in a maritime setting has grown since its inception in the mid-1990's, the algorithmic study of maritime IRPs (MIRPs) remains largely untapped with only 20 to 30 papers dedicated to optimization-based methods for solving application-specific problems. As discussed in [8] and [21], the range of applications is rather diverse given the economic viability of seaborne transportation for shipping many products. Even within the petrochemical industry, MIRPs for several products, including crude oil [23], fuel oil [2], liquefied natural gas (LNG) [14, 16, 25], and vacuum gas oil (VGO) [13], have been studied often leading to specially-tailored approaches. Our goal in this research is to develop a solution method to a general class of MIRPs known as deep-sea MIRPs with inventory tracking at every port. This class was extensively reviewed in [21].

The single product MIRP considered in this paper is best described in terms of its main components: ports, regions, vessels, and vessel classes. 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 after being discharged from vessels. Product can be stored in inventory at both types of ports. Each port has: exactly one classification type, "loading" or "discharging"; an inventory capacity that can change over time (e.g., due to closures for maintenance); a fixed number of berths limiting the number of vessels that can simultaneously load or discharge in a given period; and a deterministic per-period rate of production or consumption that can change over time. Each discharging port has a deterministic per-period unit price for the quantity discharged. Each port belongs to a predefined region. All ports in a region are of the same type giving rise to loading and discharging regions. In general, the travel times between regions is an order of magnitude greater than the travel times between ports in a region (e.g., 10 days vs. 1-2 days).

Vessels make voyages between ports by picking up inventory at one or more ports in a loading region and delivering inventory to one or more ports in a discharging region. For this reason, our MIRP can be classified as a split-pickup and split-delivery problem. We assume that vessels always travel from a loading region to a discharging region, then back to a (possibly different) loading region and on to a (possibly different) discharging region and so forth. As regions are geographically dispersed, trips between two regions of the same type are economically impractical from the outset. We focus on an industrial shipping setting in which all vessels are term-chartered vessels, meaning that they are typically operated by the company (the supplier) and are available to be used for the entire planning horizon. If a vessel attempts to load or discharge in a given period, the load/discharge quantity must be within prespecified bounds. A vessel attempting to load or discharge at a port must pay a fixed port fee every time the vessel enters the berth. Finally, each vessel belongs to a particular vessel class, which has a fixed capacity, a fixed cruising speed, and a fixed traveling cost. Port fees are assumed to be independent of vessel class.

The typical assumptions motivating this work are a planning horizon of 45 to 60 days (periods), 3 to 5 vessel classes, and a total of 5 to 20 vessels traveling between 5 to 15 ports. These instances are often larger than instances discussed in the literature. For example, the VGO models studied in [12, 13, 29] have at most 60 time periods but only one loading and one discharging region and fewer than 6 vessels. The problem considered in this paper is deterministic and is used for making tactical decisions. In practice, it is solved

repeatedly over time in a rolling horizon framework as new information becomes available.

Mathematical programming formulations of MIRPs are usually classified along at least two axes: discrete-time vs. continuous-time formulations and arc-flow vs. path-flow formulations. Whereas continuous-time formulations assume that time is a continuum so that inventory is produced and consumed at continuous rates and that decisions can be made at any instant in time throughout the process, discrete-time formulations discretize the planning horizon and assume that events can only take place at fixed points in time. Discrete-time formulations are more prevalent than continuous-time formulations. Meanwhile, whereas arc-flow formulations include decision variables to model the movement of vessels between individual ports, path-flow models include decision variables representing the entire sequence of ports visited by each vessel. In some models, a more detailed definition of a path is used to capture additional information, e.g., the amount of product loaded or discharged at each port visit.

By far the most common decomposition approach used for MIRPs is column generation (or branch-and-price) applied to a path-flow formulation [12, 16, 23]. 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. It is interesting to note that while empirically path-flow formulations for the VRP yield small root node gaps, in the order of 5 to 15 percent, path-flow formulations for a MIRP can yield very large gaps, often upwards of 100 percent [12].

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. In most column generation approaches described above, each subproblem can be interpreted 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 satisfied. Our approach 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. The way this is accomplished differs in our two approaches.

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 economical 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 contributions are: First, we use aggregation and decomposition to develop two hierarchical algorithms for solving large-scale MIRPs. Second, our algorithms emphasize and exploit model reusability. That is, a single mathematical formulation can be encoded for our first- and second-stage models, but instantiated with different data. Within an industrial setting, maintaining one mathematical model as opposed to many is an attractive feature. Third, we develop several enhancements that improve our ability to find feasible solutions and tighten dual bounds. Finally, computational experience demonstrates that, on instances that challenge a commercial solver (i.e., a single feasible solution cannot be found in 24 hours), an integrated approach that combines our algorithms can find provably high-quality solutions in roughly two hours.

The outline of this paper is as follows. In Section 2, an arc-flow MILP model of our MIRP is presented. In Section 3, we describe our first decomposition algorithm as a multi-start construction heuristic for finding a good feasible solution faster than a commercial solver. In Section 4, we describe an exact two-stage decomposition algorithm. Several enhancements are outlined in Section 5 followed by a sketch of our complete solution procedure in Section 6. Finally, we present computational experiments in Section 7.

2 An Arc-Flow Mixed-Integer Linear Programming Formulation

In this section, we describe an arc-flow MILP model of the MIRP considered in this paper. This model is a variant of the core MIRP model for MIRPs with inventory tracking at every port identified in [21]. Other variants have also been considered [1, 13, 15, 29]. The model is a discrete-time model involving an underlying time-space network. Its purpose is to identify optimal routing decisions for a heterogeneous fleet of vessels and optimal loading and discharging amounts by each vessel in each time period to ensure that inventory remains within prespecified bounds.

Vessels make voyages between ports by picking up inventory in a single loading region and delivering inventory in a single discharging region. A vessel will never load (discharge) in two different regions before discharging (loading) in another region. After discharging, however, a vessel may travel to a different loading region from the one it previously visited. A vessel may occupy a berth for one or more periods without loading or discharging. A vessel can remain idle the entire planning horizon in which case it travels directly from the source node to the sink node.

Indices and sets

$t \in \mathcal{T}$	set of time periods with $T = \lvert\mathcal{T}\rvert$
$v \in \mathcal{V}$ $(vc \in \mathcal{VC})$	set of vessels (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: $\mathcal{N} = \{n = (j,t) : j \in \mathcal{J}, t \in \mathcal{T}\}$
$n \in \mathcal{N}_{0,T+1}$	set of all nodes (including a source node n_0 and a sink node n_{T+1})
$a \in \mathcal{A}$	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 (set of all outgoing arcs) associated with node $n = (j,t) \in \mathcal{N}_{0,T+1}$ and vessel $v \in \mathcal{V}$ (vessel class $vc \in \mathcal{VC}$)
$a \in \mathcal{RS}_n^v$ $(\mathcal{RS}_n^{vc})$	reverse star (set of all incoming arcs) associated with node $n = (j,t) \in \mathcal{N}_{0,T+1}$ and vessel $v \in \mathcal{V}$ (vessel class $vc \in \mathcal{VC}$)

Data

$\alpha_{j,t}^{\max}$	upper bound on the amount of product that can be bought from or sold to the spot market at port $j \in \mathcal{J}$ in time period $t \in \mathcal{T}$
$\alpha_{j}^{\max}$	upper bound on the cumulative amount of product that can be bought from or sold to the spot market at port $j \in \mathcal{J}$ over the entire planning horizon
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 $v \in \mathcal{V}$ (a vessel in vessel class $vc \in \mathcal{VC}$) 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 time 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 $j \in \mathcal{J}^C$ ($r \in \mathcal{R}^C$)
ϵ_z	nonnegative cost parameter associated with attempting to load or discharge at a port
$F_{j,t}^{\min}$ ($F_{j,t}^{\max}$)	minimum (maximum) amount of product that can be loaded/discharged at port $j \in \mathcal{J}$ from a single vessel in time period $t \in \mathcal{T}$
$P_{j,t}$ ($P_{r,t}$)	nonnegative penalty parameter associated with one unit of lost production or stockout at port $j \in \mathcal{J}$ (region $r \in \mathcal{R}$) in time period $t \in \mathcal{T}$
Q^v (Q^{vc})	capacity of vessel $v \in \mathcal{V}$ (capacity of a vessel in vessel class $vc \in \mathcal{VC}$)
$R_{j,t}$	the unit sales revenue for product discharged at port $j \in \mathcal{J}$ in time period $t \in \mathcal{T}$
$S_{j,t}^{\min}$ ($S_{j,t}^{\max}$)	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

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

Network

The model takes place on an underlying time-space network as shown in Figure 1. The network has a set $\mathcal{N}_{0,T+1}$ of nodes and a set $\mathcal{A}$ of directed arcs. The node set is shared by all vessels, while each vessel has its own arc set $\mathcal{A}^v$. The set $\mathcal{N}_{0,T+1}$ 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_0 and a sink node n_{T+1}. The arc set $\mathcal{A}^v$ associated with each vessel v can be subdivided into source arcs (arcs emanating from the source node), sink arcs (arcs entering the sink node), waiting arcs (arcs representing that a vessel stays at the same port in two consecutive time periods), and travel arcs (arcs representing that a vessel travels between two distinct ports).

The Base Model that follows is an extension of the Core Model presented in Papageorgiou et al. [21] with the features discussed in Sections 3.2.1, 3.2.3, and 3.2.3 of [21] also included.

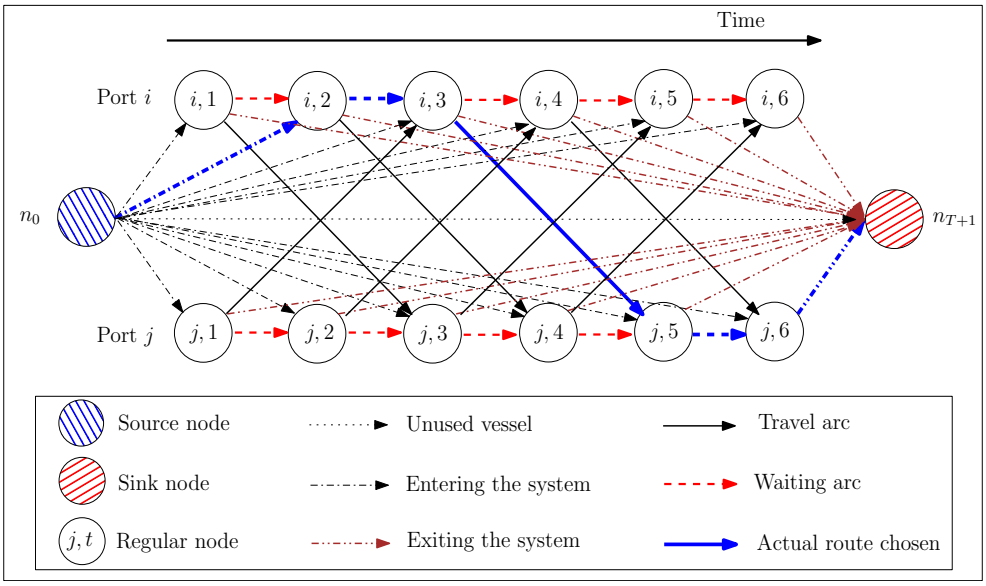

Figure 1: Example of time-space network structure for a single vessel

Base Model

$$\max \quad \sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}}\sum_{v\in\mathcal{V}} R_{j,t}f^v_{j,t} - \sum_{v\in\mathcal{V}}\sum_{a\in\mathcal{A}^v} C^v_a x^v_a - \sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}}\sum_{v\in\mathcal{V}}(t\epsilon_z)z^v_{j,t} - \sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}} P_{j,t}\alpha_{j,t} \tag{1a}$$

$$\text{s.t.} \quad \sum_{a\in\mathcal{FS}^v_n} x^v_a - \sum_{a\in\mathcal{RS}^v_n} x^v_a = \begin{cases} +1 & \text{if } n = n_0 \\ -1 & \text{if } n = n_{T+1} \\ 0 & \text{if } n \in \mathcal{N} \end{cases}, \qquad \forall\, n \in \mathcal{N}_{0,T+1}, \forall\, v \in \mathcal{V} \tag{1b}$$

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

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

$$\sum_{v\in\mathcal{V}} z^v_{j,t} \le B_j, \qquad \forall\, n = (j,t) \in \mathcal{N} \tag{1e}$$

$$z^v_{j,t} \le \sum_{a\in\mathcal{RS}^v_n} x^v_a, \qquad \forall\, n = (j,t) \in \mathcal{N}, \forall\, v \in \mathcal{V} \tag{1f}$$

$$s^v_t \ge Q^v x^v_a, \qquad \forall\, v \in \mathcal{V}, \forall\, a = ((j_1,t),(j_2,t')) \in \mathcal{A}^v : \quad j_1 \in \mathcal{J}^P, j_2 \in \mathcal{J}^C \cup \{n_{T+1}\} \tag{1g}$$

$$s^v_t \le Q^v(1 - x^v_a), \quad \forall\, v \in \mathcal{V}, \forall\, a = ((j_1,t),(j_2,t')) \in \mathcal{A}^v : \quad j_1 \in \mathcal{J}^C, j_2 \in \mathcal{J}^P \cup \{n_{T+1}\} \tag{1h}$$

$$\sum_{t\in\mathcal{T}} \alpha_{j,t} \le \alpha^{\max}_j \qquad \forall\, j \in \mathcal{J} \tag{1i}$$

$$0 \le \alpha_{j,t} \le \alpha^{\max}_{j,t} \qquad \forall\, j \in \mathcal{J}, \forall\, t \in \mathcal{T} \tag{1j}$$

$$F^{\min}_{j,t} z^v_{j,t} \le f^v_{j,t} \le F^{\max}_{j,t} z^v_{j,t}, \qquad \forall\, n = (j,t) \in \mathcal{N}, \forall\, v \in \mathcal{V} \tag{1k}$$

$$S^{\min}_{j,t} \le s_{j,t} \le S^{\max}_{j,t}, \qquad \forall\, n = (j,t) \in \mathcal{N} \tag{1l}$$

$$0 \le s^v_t \le Q^v, \qquad \forall\, v \in \mathcal{V}, \forall\, t \in \mathcal{T} \tag{1m}$$

$$x^v_a \in \{0,1\}, \qquad \forall\, v \in \mathcal{V}, \forall\, a \in \mathcal{A}^v \tag{1n}$$

$$z^v_{j,t} \in \{0,1\}, \qquad \forall\, n = (j,t) \in \mathcal{N}, \forall\, v \in \mathcal{V}. \tag{1o}$$

The objective is to maximize profit where revenue is earned at the time product is delivered to a port. Inventory costs are not included in the objective function because we assume that the shipper owns both the production and consumption sites. The term $\sum_{v\in\mathcal{V}}\sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}}-(t\epsilon_z)z^v_{j,t}$ is included to entice vessels to load or discharge as few times as possible as well as to load or discharge as soon as they arrive at a port (see Papageorgiou et al. [21]).

Constraints (1b) 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 (1c) are inventory balance constraints at the end of each time period at loading and discharging ports.. Constraints (1d) maintain inventory balance at the end of each time period on each vessel. Constraints (1e) limit the number of vessels that can attempt to load/discharge at a port in a given time period. Constraints (1f) ensure that a vessel does not attempt to load/discharge at a node unless the vessel is actually at that node. Constraints (1g) and (1h) 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; they also require vessels to leave the system empty or full. Constraints (1k) state that if a vessel attempts to load/discharge at node $n = (j, t)$, then the actual amount loaded/discharged is within predetermined port-specific bounds $[F^{\min}_{j,t}, F^{\max}_{j,t}]$. Constraints (1l) and (1m) require ending inventory in each time period at each port and on each vessel, respectively, to be within prespecified bounds.

In reality, there are often so-called spot markets available where product can be purchased or sold. Since accurately modeling spot market availability and price dynamics is difficult, we assume that a simplified spot market exists so that if a discharging port is on the brink of a stockout, it may purchase product from a nearby spot market to avoid stocking out. Similarly, if a loading port is at risk of reaching capacity, it may sell product to a spot market to avoid having to halt production. On the other hand, spot markets are not always available, so we also assume that there is a limit on the cumulative amount of product that can be purchased from or sold to a spot market from each port. In short, Constraints (1j) bound the amount of product that can be purchased from or sold to a spot market in a single period by a constant $\alpha^{\max}_{j,t}$. Constraints (1i) limit the cumulative amount of product that can be purchased from or sold to a spot market by each port over the entire planning horizon by a constant $\alpha^{\max}_{j}$.

Regarding the timing of operations, inventory levels on vessels and at ports are only monitored at the end of each period. Since this could lead to ambiguities, we assume that, in a given time period, production occurs *before* loading takes place and consumption occurs *after* discharging takes place. Consequently, 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.

There is no backlogging of inventory. We do not consider draft limits or port-specific operations.

3 A Two-Stage Construction Heuristic

In this section, we describe a two-stage multi-start heuristic for constructing an initial feasible solution to the Base Model (1). This heuristic is effective at generating solutions 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 feasible or 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 5.1.

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.

Before describing SystemModel and RegionalModel, it is important to note that both models are instantiations of the Base Model (1). This means that after writing the code (in an algebraic modeling language or in a programming language) for the Base Model (1) 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.

3.1 SystemModel: An Aggregate Model for Obtaining a Coarse Solution

To create SystemModel and to accommodate our aggregations based on region and vessel class, we use a variant of the network described in Section 2. 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 2. 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 Base Model (1) that are required:

1. Wherever the index j or v appears, an r or a vc should appear instead. Consequently, all decision

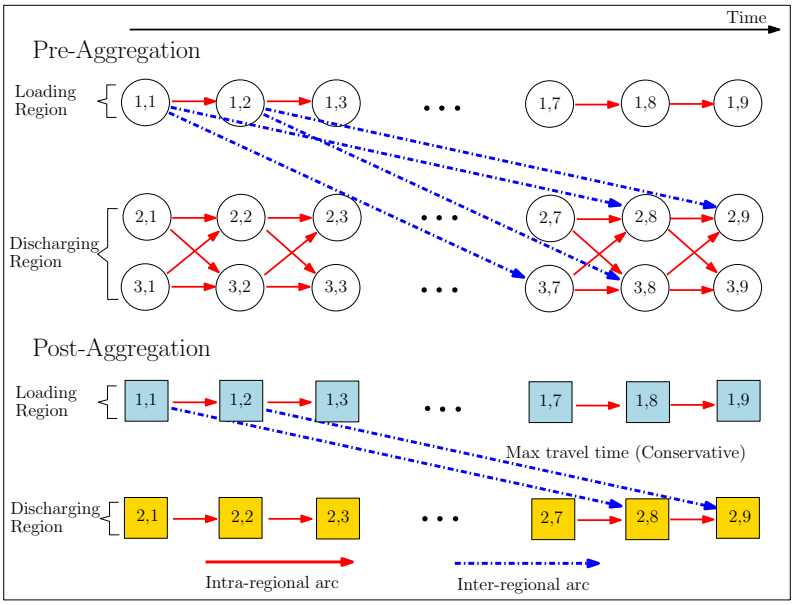

Figure 2: Example of a network before and after aggregation. Circles represent port-time pairs in the original network. Squares represent region-time pairs in the aggregate network. The two ports in the discharging region are aggregated together. No aggregation occurs in the loading region since there is only one loading port.

variables that are vessel-specific in the Base Model (1) become vessel class-specific in SystemModel, i.e., the variables become $f^{vc}_{j,t}, x^{vc}_a, z^{vc}_{j,t}$, and $s^{vc}_{r,t}$ (the latter is explained below).

2. Constraints (1n) become $x^{vc}_a \in \mathbb{Z}_+, \forall\ vc \in \mathcal{VC}, \forall\ a \in \mathcal{A}^{vc}$ to account for the fact that multiple vessels in the same class may travel along the same arc. Constraints (1o) become $z^{vc}_{r,t} \in \mathbb{Z}_+$ with $z^{vc}_{r,t} \leq B_r$, $\forall\ n = (r,t) \in \mathcal{N}, \forall\ vc \in \mathcal{VC}$, since multiple vessels in the same class may simultaneously attempt to load/discharge in the same region.

3. In the Base Model (1), each vessel has its own dedicated arc set $\mathcal{A}^v$. Since a vessel can only be in one location at a time, it is sufficient to use the decision variable s^v_t 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 $s^{vc}_{r,t}$ to keep track of the amount of inventory on each vessel class *in each region* in each time period. In addition, inventory balance constraints (1d) for each vessel are replaced by inventory balance constraints for each vessel class,

$$s^{vc}_{r,t} = s^{vc}_{r,t-1} + \Delta_r \left(f^{vc}_{r,t} - \sum_{a \in \mathcal{XS}^{vc,\text{inter}}_{r,t}} Q^{vc} x^{vc}_a \right) , \qquad \forall\ r \in \mathcal{R}, \forall\ t \in \mathcal{T}, \forall\ vc \in \mathcal{VC}\ , \tag{2}$$

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

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, inventory bound, and berth limit constraints at an aggregate (regional) level. Note the emphasis on "minimum duration." Since inter-regional arcs in this model are assumed to use the longest port-to-port arc, once individual vessel routes are determined in the RegionalModel, a vessel may 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, 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 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 where 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

$$\sum_{u=0}^{t+\tau_r^{vc}} \sum_{a \in \mathcal{FS}^{vc}_{(r,u)}} x_a^{vc} \leq \sum_{u=0}^{t} \sum_{a \in \mathcal{RS}^{vc}_{(r,u)}} x_a^{vc} \ , \qquad \forall\ r \in \mathcal{R}, \forall\ t \in \mathcal{T}, \forall\ vc \in \mathcal{VC} \ , \tag{3}$$

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 (3). 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 Base Model (1). We refer to this model as **Optimistic SystemModel**. This bound is easy to compute and almost always better than the lower bound obtained from solving the

Base Model (1) with a 24-hour time limit (as shown in the computational results section), but is typically not very tight as port specific information is ignored in favor of using regional information.

3.2 RegionalModel: A Model for Making Detailed Decisions within a Region

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 Base Model (1). 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 Base Model (1) 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.

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 (1b) for the cases when $n = n_0$ and $n = n_{T+1}$, which ensure that exactly one source and one sink arc for each vessel is chosen over the entire planning horizon, with the constraints

$$\sum_{a \in \mathcal{FS}^v_{n_0,k}} x_a^v = 1 \ , \quad \forall\ v \in \mathcal{V}, \forall\ k \in K_r^v \tag{4a}$$

$$\sum_{a \in \mathcal{RS}^v_{n_{T+1},k}} x_a^v = 1 \ , \quad \forall\ v \in \mathcal{V}, \forall\ k \in K_r^v \ , \tag{4b}$$

where $\mathcal{FS}^v_{n_0,k}$ and $\mathcal{RS}^v_{n_{T+1},k}$ 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 $\mathcal{FS}^v_{n_0,k}$ and $\mathcal{RS}^v_{n_{T+1},k}$ 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_k^v and u_k^v be the first and last time periods that vessel v may enter and exit region r during its kth visit. Thus, for each port $j \in r$, $\mathcal{RS}^v_{n_{T+1},k}$ includes sink arcs $((j, u_k^v), n_{T+1})$ (note that all tail nodes have the same departure time u_k^v) and $\mathcal{FS}^v_{n_0,k}$ includes source arcs $(n_0, (j, t^v_{j,k}))$, where $t^v_{j,k}$ is the latest possible time that vessel v can reach port j on visit k given that 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_r^v visits, we set $s^v_{t_k^v - 1} = 0$ if $r \in \mathcal{R}^P$ and $s^v_{t_k^v - 1} = Q^v$ if $r \in \mathcal{R}^C$. During all time periods $t \in \mathcal{T} \setminus \cup_{k=1}^{K_r^v} [t_k^v, \ldots, u_k^v]$ that do not involve a visit, the inventory balance constraints (1d) on vessels are omitted to reduce the size of the resulting model 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 or because a vessel does not have enough time to fully load or discharge during a visit

to a region. To avoid this situation, we may replace inventory balance constraints (1c) with the constraints

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

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 than a loading port has in inventory or discharge more than a discharging port's capacity. 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. This information can be exploited in our local search.

3.3 Summary of Construction Heuristic

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 Base Model (1) is now available. Finally, we apply local search (see Section 5.1) on this complete solution to repair it if it is infeasible or to improve it.

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 (3).
5: Set $\tau_r^{vc} = \min\{\tau_r^{vc} - 1, \tau_r^{vc,\min}\}$.
6: **until** no τ_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, solve RegionalModel with modified inventory balance constraints (5).
10: Merge solutions to each RegionalModel into a complete, but possibly infeasible, solution to Model (1).
11: Perform local search on complete solution to remove infeasibility (penalties) and improve solution.
12: **end for**
13: **return** The best solution found.

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 our framework, 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. Furthermore, if we were to insist that all vessels in vessel class vc_1 must remain an additional period in region r, 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.

4 A Two-Stage Algorithm with Feedback

In this section, we describe a 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 bound from the LP relaxation of the Base Model (1) is often 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 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 in deep-sea MIRPs 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 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. [11] 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 [23] 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 [4, 14, 22, 25]. In the LNG setting studied by [15], 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 [3, 6, 26, 27]. Hennig et al. [17, 18] 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.

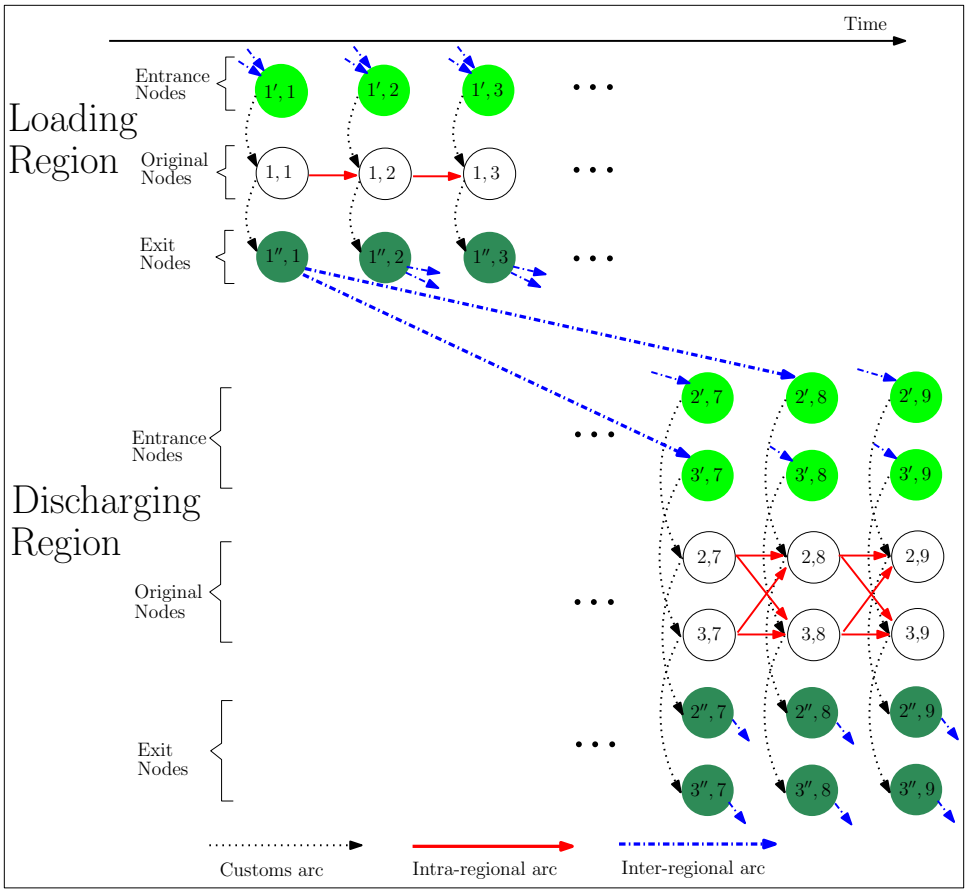

Figure 3: Augmented time-space network

4.1 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 original nodes are removed. An example of an augmented network having one loading region with a single port and one discharging region with two ports is shown in Figure 3.

4.2 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 Base Model (1). 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, (ii) the number of vessels in each region during each visit, and (iii) the duration of each visit by each vessel. 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.

4.2.1 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 only the routes of each vessesl class are modeled, 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 3, 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 4 depicts a portion of the augmented time-space network used in the first-stage problem. Only arcs for one particular vessel class are shown.

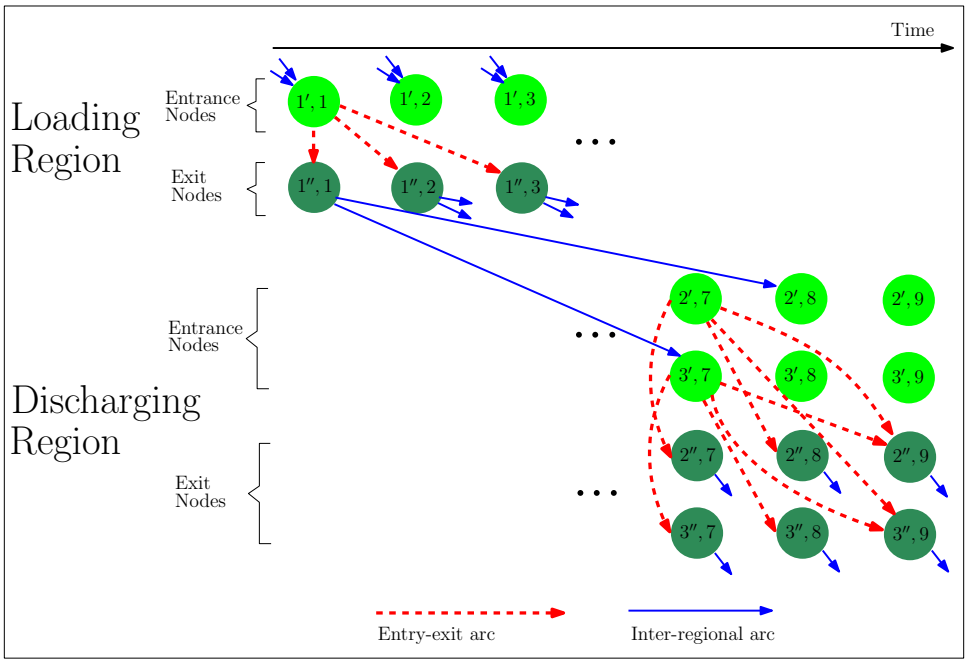

Figure 4: First-stage customs network for a single vessel class

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. 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 4 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^{vc}_a denoting integer flow on inter-regional arcs $a \in \mathcal{A}^{vc,\text{inter}}$ within vessel class vc, and w^{vc}_a 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 Base Model (1) under the restrictions imposed by the assumptions made at the outset:

SystemModel-2Port

$$\min_{\mathbf{x},\mathbf{w}} \quad \sum_{vc\in\mathcal{VC}} \sum_{a\in\mathcal{A}^{vc,\text{inter}}} C^{vc}_a x^{vc}_a + \sum_{vc\in\mathcal{VC}} \sum_{a\in\mathcal{A}^{vc,ee}} C^{vc}_a w^{vc}_a \tag{6a}$$

$$\text{s.t.} \quad \sum_{a\in\mathcal{FS}^{vc}_{n_0}} x^{vc}_a = |\mathcal{V}^{vc}|, \qquad \forall\ vc \in \mathcal{VC} \tag{6b}$$

$$\sum_{a\in\mathcal{RS}^{vc}_{n_{T+1}}} x^{vc}_a = |\mathcal{V}^{vc}|, \qquad \forall\ vc \in \mathcal{VC} \tag{6c}$$

$$\sum_{n''\in\mathcal{N}''} w^{vc}_{(n',n'')} - \sum_{a\in\mathcal{RS}^{vc}_{n'}} x^{vc}_a = 0, \qquad \forall\ vc \in \mathcal{VC}, \forall\ n' = (j', t) \in \mathcal{N}' \tag{6d}$$

$$\sum_{a\in\mathcal{FS}^{vc}_{n''}} x^{vc}_a - \sum_{n'\in\mathcal{N}'} w^{vc}_{(n',n'')} = 0, \qquad \forall\ vc \in \mathcal{VC}, \forall\ n'' = (j'', t) \in \mathcal{N}'' \tag{6e}$$

$$\text{First-stage berth limit constraints (see Section 4.2.2)} \tag{6f}$$

$$\text{Lot-sizing based covering and packing constraints (see Section 4.2.3)} \tag{6g}$$

$$w^{vc}_a \in \mathbb{Z}_+, \qquad \forall\ vc \in \mathcal{VC}, \forall\ a \in \mathcal{A}^{vc,ee} \tag{6h}$$

$$x^{vc}_a \in \mathbb{Z}_+, \qquad \forall\ vc \in \mathcal{VC}, \forall\ a \in \mathcal{A}^{vc,\text{inter}}\ . \tag{6i}$$

The first four sets of constraints ensure flow balance for each vessel classes. Constraints (6f) ensure that berth

limit constraints are not violated. Constraints (6g) 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.

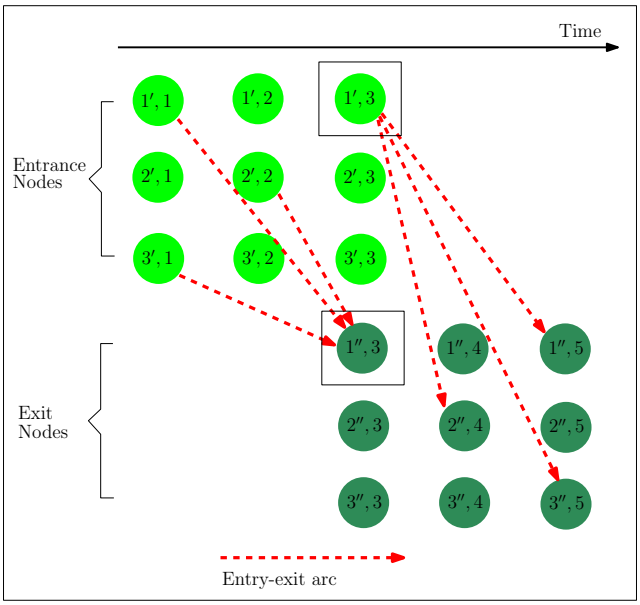

Figure 5: Example of first-stage berth limit constraints

4.2.2 First-Stage Berth Limit Constraints

In the Base Model (1), berth limit constraints at a port are explicitly handled through Constraints (1e). Although SystemModel-2Port (6) 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 SystemModel-2Port (6) 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*

$$\sum_{vc \in \mathcal{VC}} \left[\sum_{a \in \bar{\mathcal{FS}}^{vc,ee}_{(j',t)}} w^{vc}_a + \sum_{a \in \bar{\mathcal{RS}}^{vc,ee}_{(j'',t)}} w^{vc}_a \right] \leq B_j, \qquad \forall\, j \in \mathcal{J}, \forall\, t \in \mathcal{T}\ , \tag{7}$$

is valid for the first-stage master problem. Constraint (7) 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 5 and assume that there is only one vessel class. Suppose that all of the arcs

shown in Figure 5 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 5 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 are generated dynamically and appended to Model (6).

4.2.3 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. Similar ideas are discussed in [1, 2, 21, 26]. Consider the standard capacitated lot-sizing set (see, e.g., Pochet and Wolsey [24]) 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 $\mathcal{T}$:

$$s_{t-1} + x_t = d_t + s_t \ , \qquad \forall\ t \in \mathcal{T} \tag{8a}$$

$$0 \le x_t \le C_t y_t \ , \qquad \forall\ t \in \mathcal{T} \tag{8b}$$

$$y_t \in \{0,1\} \ , \qquad \forall\ t \in \mathcal{T} \tag{8c}$$

$$0 \le s_t \le s^{\max} \ , \qquad \forall\ t \in \mathcal{T} \ . \tag{8d}$$

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 (8a) and apply inequalities (8b) to obtain the relaxation

$$s_{t_1-1} + \sum_{u=t_1}^{t_2} C_u y_u \ge d_{[t_1,t_2]} + s_{t_2} \ , \qquad \forall\ 1 \le t_1 \le t_2 \le T \tag{9a}$$

$$y_t \in \{0,1\} \ , \qquad \forall\ t \in \mathcal{T} \tag{9b}$$

$$0 \le s_t \le s^{\max} \ , \qquad \forall\ t \in \mathcal{T} \ , \tag{9c}$$

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 $s^{\max}_{t_1-1}$ and s_{t_2} with its lower bound $s^{\min}_{t_2}$, we obtain a further relaxation

$$\sum_{u=t_1}^{t_2} C_u y_u \ge d_{[t_1,t_2]} + s^{\min}_{t_2} - s^{\max}_{t_1-1} \ , \qquad \forall\ 1 \le t_1 \le t_2 \le T \tag{10a}$$

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

Note that we prefer to include the term $(s^{\min}_{t_2} - s^{\max}_{t_1-1})$ in the right hand side of (10a) instead of $(0 - s^{\max})$ since instance data may reveal that $s^{\max}_{t_1-1} < s^{\max}$ and $s^{\min}_{t_2} > 0$.

Relaxation (10) 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 (10) has replaced the original lot-sizing set (8) 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 (9) onto the space of binary variables y.

Finally, note that if the capacitated lot-sizing set (8) 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

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

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

is valid. Constraints (10a) and (11a) 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 $\mathcal{A}^{vc,ee}_{I,[t_1,t_2]}$ 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 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 (10a) for the lot-sizing set, we can define subset covering constraints

$$\sum_{vc \in \mathcal{VC}} \sum_{a \in \mathcal{A}^{vc,ee}_{I,[t_1,t_2]}} C^{vc,\max}_{I,[t_1,t_2],a} w^{vc}_a \geq d_{I,[t_1,t_2]} + S^{\min}_{I,t_2} - S^{\max}_{I,t_1-1}\ , \qquad \forall I \in \mathcal{I}, \forall 1 \leq t_1 \leq t_2 \leq T\ . \tag{12}$$

Similarly, analogous to Constraints (11a), we can define subset packing constraints

$$\sum_{vc \in \mathcal{VC}} \sum_{a \in \mathcal{A}^{vc,ee}_{I,[t_1,t_2]}} C^{vc,\min}_{I,[t_1,t_2],a} w^{vc}_a \leq d_{I,[t_1,t_2]} + S^{\max}_{I,t_2} - S^{\min}_{I,t_1-1}\ , \qquad \forall I \in \mathcal{I}, \forall 1 \leq t_1 \leq t_2 \leq T\ . \tag{13}$$

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 (12) 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 6. Customs entrance and exit nodes are shown along with 12 entry-exit arcs. There are two ports in a discharging region, each with one berth,

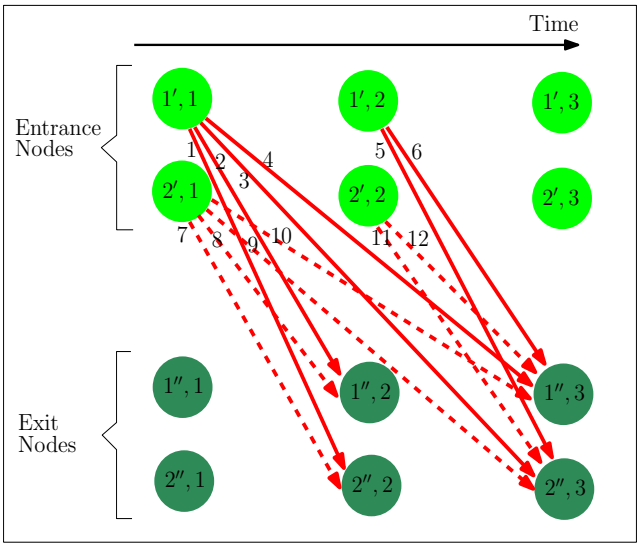

Figure 6: Example of customs network

Table 1: 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)

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 1 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.

4.3 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 $\mathcal{FS}^v_{n_0,k}$ and $\mathcal{RS}^v_{n_{T+1},k}$ associated with Constraints (4) 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.

4.4 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 $x^{vc}_a \in \{0, 1\}$ for all $vc \in \mathcal{VC}$ and $a \in \mathcal{A}^{vc}$, then an enumeration cut can be written as

$$\sum_{vc \in \mathcal{VC}} \left[\sum_{a \in \mathcal{A}^{vc}: \hat{x}^{vc}_a = 0} x^{vc}_a + \sum_{a \in \mathcal{A}^{vc}: \hat{x}^{vc}_a = 1} (1 - x^{vc}_a) \right] \geq 1 \tag{14}$$

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 $[t_1, t_2]$.

We close this section by comparing the construction heuristic of Section 3 and the algorithm of this section. Table 2 captures the salient differences. The first major difference between the two approaches is that the second incorporates a feedback mechanism so that information from the subproblems can be communicated back to the master problem. In keeping with the analogy of zooming in and out on a problem, this means that the construction heuristic can only zoom in whereas the two-stage approach with feedback can zoom in and out on a problem. Second, the details modeled in the respective master problems differ. In the construction heuristic, the master problem models inventory, loading/discharging, and inter-regional routing decisions, but not specific port information. In the approach with feedback, only routing decisions

Table 2: Algorithm Comparison. "Feedback" refers to communication between the subproblem and master problem.

Algorithm	**Purpose**	**Master Problem**	**Subproblem**	**Feedback**
(Zoom In Only) Multi-start construction heuristic	Find a good initial feasible solution	• Aggregate by region and by vessel class • Inventory, loading/discharging, and routing decisions are modeled • Explicit inventory balance constraints in each region	Arrival *time* is flexible, but departure *time* is fixed for each vessel	No
(Zoom In & Out) Two-stage algorithm with feedback	Improve an existing feasible solution and provide a useful dual bound	• Aggregate by vessel class • Only routing decisions are modeled • Lot-sizing based constraints and other cuts replace inventory balance constraints in regions and ports	Arrival *node* (port-time pair) and departure *node* are fixed for each vessel	Yes

are modeled, but lot-sizing constraints are used to ensure that regions and individual ports are visited with the right frequency. As a consequence, a regional manager has more flexibility (a larger solution space) in the construction heuristic than in the approach with feedback. This is the price we have chosen to pay in order to obtain useful bounds in the approach with feedback.

5 Enhancements: Improving Practical Performance

In this section, we discuss several techniques that we have found useful to improve the practical performance of both algorithms.

5.1 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 and to each solution stored in a solution pool of SystemModel-2Port (6). Several authors have shown how local search can be used to find high-quality solutions and improve existing solutions [14, 19, 22, 28, 29, 30]. Note that in [29], 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, time-chartered 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. [19]. 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.

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 Base Model (1).

repeat
  for `fixedRegionType` in {`Loading`,`Discharging`} **do**
    for each region of type `fixedRegionType` **do**
      Fix all vessel paths and all $z^v_{j,t}$ variables in this region to their current value.
    end for
    for each region not of type `fixedRegionType` **do**
      Solve RegionalModel with Constraints (4) where $\mathcal{FS}^v_{n_0,k}$ and $\mathcal{RS}^v_{n_{T+1},k}$ are defined in the text.
      (Optional) Force vessels to depart from a region in the time period of their last attempt to load/discharge.
    end for
  end for
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
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 $\mathcal{FS}^v_{n_0,k}$ and $\mathcal{RS}^v_{n_{T+1},k}$ in Constraints (4) are modified. We 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 $\mathcal{FS}^v_{n_0,k}$ as the set of source arcs $\{(n_0,(j,t^v_{j,k}))\}$ 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 $\mathcal{RS}^v_{n_{T+1},k}$ as the set of sink arcs $\{((j,u^v_{j,k}),n_{T+1})\}$ 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 Base Model (1) and then write an iterative procedure that solves reduced instances of the Base Model (1). 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. [19]. 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 $z_{j,t}^v$ variables are fixed at their current value for $v \in \mathcal{V} \setminus S$. 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.

5.2 Branching on Auxiliary Decision Variables

SystemModel-2Port (6) typically 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.

5.3 Integer Knapsack Polytope Constraints

It is sometimes possible to strengthen the formulation of SystemModel-2Port by including facets of the integer knapsack polytope derived from the lot-sizing based constraints of Section 4.2.3. If we isolate a packing constraint (13) in which slack variables are also included (i.e., if a spot market is present), we obtain the set

$$W = \left\{ x \in \{0,1\}^{n_1}, s \in \mathbb{R}_+^{n_s} : \sum_{j=1}^{n_1} \hat{a}_j x_j - \sum_{k=1}^{n_s} s_k \leq \hat{b} \right\} , \tag{15}$$

where the x_j variables correspond to arc variables, the s_k variables correspond to slack variables over an interval of n_s 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 (15). We could include auxiliary continuous variables to represent the sum $\sum_{k=1}^{n_s} s_k$, 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 $\sum_{k=1}^{n_s} s_k$ by its upper bound $s^{\max}$. Setting $b = \hat{b} + s^{\max}$, we obtain the relaxed pure binary knapsack set

$$X = \left\{ x \in \{0,1\}^{n_1} : \sum_{j=1}^{n_1} \hat{a}_j x_j \leq b \right\} . \tag{16}$$

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 (16), 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

$$Y = \left\{ y \in \mathbb{Z}^{n_2} : \sum_{j=1}^{n_2} a_j y_j \leq b \right\} . \tag{17}$$

When n_2 is small, e.g. $n_2 \leq 10$, it is possible to obtain the facets of the integer knapsack polytope (17) by calling PORTA [9].

There are at least three 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.

It is possible to try a similar idea using covering constraints (12) or by considering multiple constraints simultaneously as is done in [20] for the dynamic knapsack set. The problem, however, with both of these extensions is that the resulting sets have too high of a dimension for PORTA to handle. One could, of course, bypass PORTA and try to separate fractional solutions in a callback, but this experiment lies beyond the scope of this paper.

6 An Integrated Solution Procedure

Finally, we propose an integrated approach in which the algorithms of the previous sections are combined. Pseudocode of this integrated algorithmic approach is provided in Algorithm 3. Several observations are in order. In Step 4, warm-starting SystemModel-2Port for larger instances can save hundreds of seconds in presolve time and an additional hundreds of seconds in solving the root LP. We refer to Steps 5-15 as *Zoom*. This procedure is implemented using a LazyConstraintCallback as is typically required for a Benders-like strategy. In Steps 11-14, since we are searching for the best solution to the Base Model (1), 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 Base Model (1). We warm-start the solution process for each relaxed model in Step 12 with the solution obtained from the restricted model in Step 8, so that the additional CPU time for this search is often only a few seconds. In Step 16, we could perform local search after each new solution to the Base Model (1) is found, but we chose to use local search as a last step in the spirit of a solution polishing procedure.

Algorithm 3 Integrated Solution Procedure

Create an empty list of solutions `SolutionPool`.
Apply the two-stage multi-start construction heuristic of Section 3 to generate a list `LIST` of (possibly infeasible) solutions to the Base Model (1).
Perform local search on each solution in `LIST` to remove any infeasibilities and/or find an improving solution.
Warm-start SystemModel-2Port (6) with the best feasible solution found thus far.
for each integer feasible solution found **do**
  for each regional subproblem **do**
    Solve RegionalModel-2Port
    if RegionalModel-2Port is infeasible, **then** add a first-stage berth limit cut or enumeration cut.
  end for
  if all regional subproblems are feasible, **then** a new incumbent solution has been found.
  for each regional subproblem **do**
    Solve a relaxed version of RegionalModel-2Port with no two-port-with-no-revisits constraints.
  end for
  if all regional subproblems are feasible, **then** store this solution in `SolutionPool`.
end for
Perform local search on each solution in `SolutionPool` to find an improving solution.
return The best solution found and the bound provided by SystemModel-2Port (6).

7 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. Whenever local search was performed, emphasis on feasibility was also selected.

Computational experiments were conducted on the Group 1 instances of the Maritime Inventory Routing Problem Library (MIRPLib) [21], a library of MIRP instances available at `mirplib.scl.gatech.edu/`. Although inspired by real-world MIRPs, they do not represent any particular real-world data set. Each discharging region r has a constant unit revenue R_r throughout the planning horizon. Consequently, all ports within a discharging region have the same parameter $R_{j,t}$. The parameters $\alpha_j^{\max}$ are rather strict so that there is a tight limit on the cumulative amount of product that can be bought from or sold to the spot market over the entire horizon. Initial ports and times for all vessels are given as input. Vessels originating in loading regions initially have zero inventory, while those beginning in discharging regions start at capacity.

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_VC5_V16b`. `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. `VC5` means that there are five vessel classes. `V16` means that there are a total of 16 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.

Table 3: Comparison of lower bounds obtained from different models

Instance	Periods	Base Model	Optimistic SystemModel	RootNode LB SystemModel-2Port	Final LB SystemModel-2Port
LR1_1_DR1_3_VC1_V7a	45	-25261	-14410	-13368	**-13273**
LR1_1_DR1_4_VC3_V11a	45	-29340	-12994	-11736	**-11289**
LR1_1_DR1_4_VC3_V12a	45	-32642	-12329	-11226	**-10739**
LR1_1_DR1_4_VC3_V12b	45	-30299	-9578	-9096	**-9073**
LR1_1_DR1_4_VC3_V8a	45	-23507	-6153	-5234	**-5174**
LR1_1_DR1_4_VC3_V9a	45	-24035	-8242	-7369	**-6959**
LR1_2_DR1_3_VC2_V6a	45	-22749	-12763	-11218	**-11146**
LR1_2_DR1_3_VC3_V8a	45	-27897	-13625	-12123	**-12012**
LR2_11_DR2_22_VC3_V6a	45	-22843	*-10802*	-10896	**-9779**
LR2_11_DR2_33_VC4_V11a	45	-35743	-16445	-15254	**-15137**
LR2_11_DR2_33_VC5_V12a	45	-42508	-20668	-19258	**-19092**
LR2_22_DR2_22_VC3_V10a	45	-44707	-27803	-25876	**-25576**
LR2_22_DR3_333_VC4_V14a	45	-47114	-27216	-24427	**-23967**
LR2_22_DR3_333_VC4_V17a	45	-55285	-27628	-23739	**-23553**
LR1_1_DR1_3_VC1_V7a	60	-33091	-17847	-16792	**-16676**
LR1_1_DR1_4_VC3_V11a	60	-38536	-15020	-13666	**-13383**
LR1_1_DR1_4_VC3_V12a	60	-42972	-12832	-11599	**-11269**
LR1_1_DR1_4_VC3_V12b	60	-39843	-11287	-10541	**-10085**
LR1_1_DR1_4_VC3_V8a	60	-31306	-6691	-5735	**-5628**
LR1_1_DR1_4_VC3_V9a	60	-31800	-9383	-8128	**-7696**
LR1_2_DR1_3_VC2_V6a	60	-30163	-15841	-14108	**-13810**
LR1_2_DR1_3_VC3_V8a	60	-36458	-17379	-15343	**-14931**
LR2_11_DR2_22_VC3_V6a	60	-30207	*-14198*	-14283	**-13351**
LR2_11_DR2_33_VC4_V11a	60	-46759	-19565	-17491	**-17008**
LR2_11_DR2_33_VC5_V12a	60	-56078	-25988	-24513	**-24246**
LR2_22_DR2_22_VC3_V10a	60	-58951	-35873	-34305	**-34167**
LR2_22_DR3_333_VC4_V14a	60	-61737	-33503	-30000	**-29931**
LR2_22_DR3_333_VC4_V17a	60	-72108	-33909	-30247	**-30227**

7.1 Experiments with the Base Model Using a Commercial Solver

We conducted several experiments with the Base Model (1) to understand its strengths and limitations. The main finding was that, within a 24-hour time limit, Gurobi's MIP solver emphasizing feasibility could not find a single feasible solution to any of the instances. Moreover, as shown in Table 3, the lower bounds produced from the Base Model were very weak compared to those obtained from solving Optimistic SystemModel to provable optimality (see the last paragraph of Section 3.1), from solving just the root node of SystemModel-2Port (6), and from solving SystemModel-2Port for up to two hours. Using Gurobi's default settings did not help in obtaining feasible solutions and made negligible improvements to the lower bounds.

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 Base Model (1) 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 Base Model (1). Unfortunately, the results are worse. To reduce the solution space of the Base Model (1), we included

additional constraints, which were implemented in two ways. First, we included explicit constraints in the Base Model (1) 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 cut pool 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.

7.2 Main Computational Results

Table 4 shows the main results of our approaches for instances with planning horizons of 45 and 60 periods, respectively. Columns 2-5 show the **CPU Time** in seconds of each of the algorithms, where **CH+LS** = the multi-start construction heuristic followed by local search of Section 3 (Steps 2-3 of Algorithm 3); **Zoom** = the two-stage algorithm with feedback of Section 4 (Steps 5-15); **Polish** = a solution polishing procedure using the local search of Section 5.1 (Step 16). Time to the best known solution was not computed. Columns 6 and 7 show the objective function value of the best solution found once the associated procedure has completed.

Column 8 displays the lower bound z^{LB} (labeled **LB***) provided by SystemModel-2Port once the Zoom algorithm has terminated. Columns 9 and 10 show the relative and absolute gaps (labeled **relGap*** and **absGap***), which 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 solution found. An asterisk appears next to **LB***, **relGap***, and **absGap*** as a reminder that the lower bound and the gaps are 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.*

Before giving detailed comments about each of the algorithms, we begin with a summary of our main findings. First, the multi-start construction heuristic is a decent-to-good primal heuristic. Second, warmstarting SystemModel-2Port prior to entering Zoom is crucial; it reduces preprocessing times (up to several hundred seconds in some cases) and, without it, Zoom may not find a single feasible solution. Third, Zoom is able to improve, sometimes substantially, on an initial solution showing that it is not solely a "dual-side" algorithm. Most importantly, these observations suggest that our integrated approach is effective at obtaining provably high-quality solutions.

The construction heuristic followed by local search (**CH+LS**) performs reasonably well at finding good initial feasible solutions. For some instances, it does not produce solutions that are close to the best known solutions within the allotted time. However, additional experiments (not shown) indicate that its performance could improve if given additional time. 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 proving optimality. 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.

The Zoom algorithm (**Zoom**) is able to improve, sometimes significantly, upon the solution found by CH+LS. The average relative improvement in the objective function value for 45- and 60-period instances is 4.44% and 10.24%, respectively, where the relative improvement is computed as $(z^{CH} - z^{Zoom})/z^{CH}$. This

Table 4: Main algorithmic results

		CPU Time (s)				Objval				
Instance	**Periods**	**CH+LS**	**Zoom**	**Polish**	**Total**	**CH+LS**	**Zoom**	**LB***	**relGap* (%)**	**absGap***
LR1_1_DR1_3_VC1_V7a	45	27	132	18	177	-13271	-13272	-13273	0.01	1
LR1_1_DR1_4_VC3_V11a	45	1032	7200	300	8532	-10650	-11239	-11289	0.44	50
LR1_1_DR1_4_VC3_V12b	45	487	7200	300	7987	-10112	-10732	-10739	0.06	7
LR1_1_DR1_4_VC3_V12c	45	550	927	300	1777	-9013	-9069	-9073	0.05	5
LR1_1_DR1_4_VC3_V8a	45	481	3828	300	4609	-4945	-5106	-5176	1.37	68
LR1_1_DR1_4_VC3_V9a	45	191	267	54	512	-6730	-6891	-6959	0.98	68
LR1_2_DR1_3_VC2_V6a	45	473	5190	300	5963	-11062	-11134	-11146	0.10	11
LR1_2_DR1_3_VC3_V8a	45	134	7200	300	7634	-11747	-12010	-12012	0.02	2
LR2_11_DR2_22_VC3_V6a	45	649	7200	300	8149	-9571	-9718	-9779	0.63	61
LR2_11_DR2_33_VC4_V11a	45	1413	7200	300	8913	-13095	-14017	-15137	7.99	1120
LR2_11_DR2_33_VC5_V12a	45	1434	7200	300	8934	-16666	-18423	-19092	3.63	669
LR2_22_DR2_22_VC3_V10a	45	1226	7200	300	8726	-23929	-24789	-25576	3.18	787
LR2_22_DR3_333_VC4_V14a	45	1906	7200	300	9406	-20065	-21952	-23967	9.18	2015
LR2_22_DR3_333_VC4_V17a	45	2027	7200	300	9527	-19879	-21713	-23553	8.47	1840
LR1_1_DR1_3_VC1_V7a	60	386	28	30	444	-16588	-16675	-16676	0.01	1
LR1_1_DR1_4_VC3_V11a	60	779	6060	300	7139	-11407	-13257	-13383	0.94	125
LR1_1_DR1_4_VC3_V12b	60	487	7200	300	7987	-10120	-11040	-11269	2.08	229
LR1_1_DR1_4_VC3_V12c	60	1210	7200	300	8710	-8833	-10053	-10085	0.33	33
LR1_1_DR1_4_VC3_V8a	60	187	7200	300	7687	-4398	-5191	-5628	8.43	437
LR1_1_DR1_4_VC3_V9a	60	785	7200	300	8285	-6479	-7552	-7696	1.91	144
LR1_2_DR1_3_VC2_V6a	60	800	7200	300	8300	-12659	-13631	-13810	1.31	178
LR1_2_DR1_3_VC3_V8a	60	531	7200	300	8031	-12930	-14652	-14931	1.90	278
LR2_11_DR2_22_VC3_V6a	60	904	7200	300	8404	-12581	-12655	-13351	5.50	696
LR2_11_DR2_33_VC4_V11a	60	1443	7200	300	8943	-14852	-15387	-17008	10.53	2701
LR2_11_DR2_33_VC5_V12a	60	1503	7200	300	9003	-19646	-22730	-24246	6.67	1517
LR2_22_DR2_22_VC3_V10a	60	1303	7200	300	8803	-30876	-32627	-34167	4.72	1540
LR2_22_DR3_333_VC4_V14a	60	2042	7200	300	9542	-24239	-26873	-29931	11.38	3057
LR2_22_DR3_333_VC4_V17a	60	2115	7200	300	9615	-24166	-27000	-30227	11.95	3227

improvement comes at the cost of additional computational time. More importantly, the Zoom phase of the integrated algorithm provides a dual bound for gauging the quality of the best known solution. As stated earlier, the Zoom algorithm struggles to find a feasible solution if one is not provided in a warmstart.

The solution polishing procedure (**Polish**) calls our local search heuristics. It is given a total 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 is used to clean up the solution and is not responsible for any appreciable improvements after calling the Zoom algorithm. It also makes no improvements to the dual bound.

As one would expect, going from 45 to 60 periods leads to greater computational challenges as reflected in the gaps. For instances with one loading and one discharging region, our integrated 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 difficulty 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 half of a vessel to port 1 in a region

and the other half to port 2 in the same region, in its node LP relaxations to avoid intra-regional travel costs. Lot-sizing based cuts help to eliminate these fractional solutions with vessel splitting.

Table 4 shows that it is possible for the Zoom phase of the integrated algorithm to terminate before the two-hour time limit with a positive gap. This occurs because SystemModel-2Port uses lot-sizing based cuts that rely on *relaxations* of the original problem. Thus, the objective function value of an optimal solution to SystemModel-2Port may not account for all of the costs incurred when this solution is converted to a feasible solution to the Base Model (1). For example, an optimal solution to SystemModel-2Port may not require that any product is bought from the spot market, whereas the corresponding solution to the Base Model (1) may require for a small amount of product to be purchased from the spot market at a small cost.

Despite some relative gaps above 5%, we believe that the results are quite promising. For a relative comparison, Hewitt et al. [19] use a branch-and-price guided local search technique to find solutions to challenging 60-period instances presented in [12]. 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 integrated approach is successful at simultaneously finding good solutions and good bounds in just over two hours.

7.3 Comparison of Lower Bounds Obtained Using Different SystemModel-2Port Formulations

Table 5 compares the lower bounds generated by SystemModel-2Port (6) within a two-hour time limit under its default formulation, when auxiliary variables are included to allow for enhanced branching (see Section 5.2), and when constraints from the integer knapsack polytope are included (see Section 5.3). The upper bounds generated by these three variants were virtually the same. These tables show the number of first-stage berth limit cuts (**B**) and enumeration cuts (**E**) that are generated; the quality of the root node LP relaxation (**RLB***), i.e., the value of the LP solution obtained after all MIP 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 lower bound (**FLB***) is denoted with an asterisk as a reminder that the "two-port-with-no-revisits" assumption is in effect. For each instance, SystemModel-2Port (6) was warm-started with the same inital solution found by the construction heuristic with the objective function value reported in Table 4.

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. However, 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 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

Table 5: Comparison using SystemModel-2Port with enhancements. **B** = Berth limit cuts; **E** = Enumeration cuts; **RLB*** = Root node lower bound after all MIP processing is completed; **FLB*** = Final lower bound within a two-hour time limit.

		Default				**Auxiliary variable branching**				**Integer knapsack cuts**			
		Cuts		**Bounds**		**Cuts**		**Bounds**		**Cuts**		**Bounds**	
Instance	**Periods**	**B**	**E**	**RLB***	**FLB***	**B**	**E**	**RLB***	**FLB***	**B**	**E**	**RLB***	**FLB***
LR1_1_DR1_3_VC1_V7a	45	0	0	-13368	**-13273**	0	0	-13368	**-13273**	0	0	-13368	**-13273**
LR1_1_DR1_4_VC3_V11a	45	0	104	-11736	-11619	0	17	-11815	-11583	0	17	-11815	**-11289**
LR1_1_DR1_4_VC3_V12b	45	26	463	-11226	-11119	34	553	-10835	**-10739**	10	364	-11324	-11138
LR1_1_DR1_4_VC3_V12c	45	9	40	-9096	**-9073**	0	10	-9086	**-9073**	0	23	-9130	**-9073**
LR1_1_DR1_4_VC3_V8a	45	17	420	-5234	-5199	29	402	-5227	**-5174**	5	253	-5476	**-5174**
LR1_1_DR1_4_VC3_V9a	45	0	75	-7369	-7120	0	73	-7113	**-6959**	0	169	-7411	-7336
LR1_2_DR1_3_VC2_V6a	45	0	1127	-11218	-11147	0	1248	-11218	**-11146**	0	691	-11291	-11149
LR1_2_DR1_3_VC3_V8a	45	0	16	-12123	-12037	0	4	-12110	-12059	0	12	-12171	**-12012**
LR2_11_DR2_22_VC3_V6a	45	0	68	-10896	-10133	0	481	-10879	**-9779**	0	82	-11036	-10435
LR2_11_DR2_33_VC4_V11a	45	2	19	-15254	-15210	1	21	-15286	-15175	1	16	-15482	**-15137**
LR2_11_DR2_33_VC5_V12a	45	29	674	-19258	**-19092**	22	578	-19188	-19107	0	35	-19491	-19145
LR2_22_DR2_22_VC3_V10a	45	0	36	-25876	-25791	0	99	-25729	**-25576**	0	69	-26278	-25587
LR2_22_DR3_333_VC4_V14a	45	0	11	-24427	-24367	0	5	-24597	**-23967**	0	13	-24597	-24518
LR2_22_DR3_333_VC4_V17a	45	0	201	-23739	-23721	0	28	-23740	**-23553**	0	42	-23832	-23554
LR1_1_DR1_3_VC1_V7a	60	0	0	-16792	**-16676**	0	0	-16683	**-16676**	0	0	-16792	**-16676**
LR1_1_DR1_4_VC3_V11a	60	0	40	-13666	-13470	0	63	-13460	**-13383**	3	616	-13566	-13490
LR1_1_DR1_4_VC3_V12b	60	1	46	-11599	-11485	8	66	-11598	**-11269**	78	363	-11535	-11480
LR1_1_DR1_4_VC3_V12c	60	0	17	-10541	-10472	0	14	-10311	**-10085**	0	20	-10492	-10473
LR1_1_DR1_4_VC3_V8a	60	0	76	-5735	-5670	11	39	-5723	**-5628**	24	147	-5719	-5685
LR1_1_DR1_4_VC3_V9a	60	1	44	-8128	-8055	2	10	-7801	**-7696**	0	29	-8089	-8057
LR1_2_DR1_3_VC2_V6a	60	0	54	-14108	-14000	0	15	-14063	-13963	0	39	-14057	**-13810**
LR1_2_DR1_3_VC3_V8a	60	0	7	-15343	-15214	0	4	-15019	**-14931**	0	9	-15030	-15003
LR2_11_DR2_22_VC3_V6a	60	0	12	-14283	-13908	0	7	-14032	**-13351**	0	8	-14077	-13943
LR2_11_DR2_33_VC4_V11a	60	0	4	-17491	-17115	0	5	-17346	**-17008**	0	6	-17342	-17083
LR2_11_DR2_33_VC5_V12a	60	0	5	-24513	-24494	0	12	-24354	**-24246**	0	7	-24491	-24455
LR2_22_DR2_22_VC3_V10a	60	0	16	-34305	-34181	0	8	-34250	**-34167**	0	24	-34193	-34175
LR2_22_DR3_333_VC4_V14a	60	0	1	-30000	**-29931**	0	1	-30000	-29975	0	1	-30011	-29961
LR2_22_DR3_333_VC4_V17a	60	0	6	-30247	-30228	0	7	-30247	**-30227**	0	11	-30261	-30250

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 Base 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.

We also 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 (6) a five-hour time limit and the additional improvements to the bounds were minor. This suggests that more powerful cuts or model reformulation may be needed.

8 Conclusions

We introduced two-stage decomposition algorithms for solving single product deep-sea 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 feedback, the regional subproblems can be solved independently. It would be interesting to explore the computational gains from a parallel implementation of our algorithms.

Acknowledgments

We wish to thank Myun-Seok Cheon of ExxonMobil Research and Engineering Company and Jin-Hwa Song formerly of ExxonMobil for their collaboration and helpful suggestions. We also wish to thank ExxonMobil Research and Engineering Company for financial support of this research.

References

[1] A. Agra, H. Andersson, M. Christiansen, and L. A. Wolsey. A maritime inventory routing problem: Discrete-time formulations and valid inequalities. *To appear in Networks*, 2011.

[2] A. Agra, M. Christiansen, and A. Delgado. Mixed integer formulations for a short sea fuel oil distribution problem. *Transportation Science*, 47(1):108–124, 2013.

[3] H. Andersson. A maritime pulp distribution problem. *INFOR*, 49(2):125–138, 2011.

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

[5] H. Andersson, A. Hoff, M. Christiansen, G. Hasle, and A. Løkketangen. Industrial aspects and literature survey: Combined inventory management and routing. *Computers & Operations Research*, 37(9):1515–1536, 2010.

[6] B. Bilgen and I. Ozkarahan. A mixed-integer linear programming model for bulk grain blending and shipping. *International Journal of Production Economics*, 107(2):555–571, 2007.

[7] A. Campbell, L. Clarke, A. Kleywegt, and M. W. P. Savelsbergh. The inventory routing problem. In T. G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 95–113. Kluwer, 1998.

[8] M. Christiansen, K. Fagerholt, B. Nygreen, and D. Ronen. Maritime transportation. In C. Barnhart and G. Laporte, editors, *Transportation, Handbooks in Operations Research and Management Science*, volume 14, pages 189–284. Elsevier, 2007.

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

[10] L. C. Coelho, J.-F. Cordeau, and G. Laporte. Thirty years of inventory-routing. *To appear in Transportation Science*, 2013.

[11] S. Dauzère-Pérès, A. Nordli, A. Olstad, K. Haugen, U. Koester, P. O. Myrstad, G. Teistklub, and A. Reistad. Omya hustadmarmor optimizes its supply chain for delivering calcium carbonate slurry to european paper manufacturers. *Interfaces*, 37(1):39–51, 2007.

[12] F. G. Engineer, K. C. Furman, G. L. Nemhauser, M. W. P. Savelsbergh, and J.-H. Song. A Branch-Price-And-Cut algorithm for single product maritime inventory routing. *Operations Research*, 60(1):106–122, 2012.

[13] K. C. Furman, J.-H. Song, G. R. Kocis, M. K. McDonald, and P. H. Warrick. Feedstock routing in the ExxonMobil downstream sector. *Interfaces*, 41(2):149–163, 2011.

[14] V. Goel, K. C. Furman, J.-H. Song, and A. S. El-Bakry. Large neighborhood search for LNG inventory routing. *Journal of Heuristics*, 18(6):821–848, 2012.

[15] R. Grønhaug and M. Christiansen. Supply chain optimization for the liquefied natural gas business. *Innovations in Distribution Logistics, Lecture Notes in Economics and Mathematical Systems*, 619:195–218, 2009.

[16] R. Grønhaug, M. Christiansen, G. Desaulniers, and J. Desrosiers. A Branch-and-Price method for a liquefied natural gas inventory routing problem. *Transportation Science*, 44(3):400–415, 2010.

[17] F. Hennig, B. Nygreen, M. Christiansen, K. Fagerholt, K. Furman, J.-H. Song, G. Kocis, and P. Warrick. Maritime crude oil transportation - a split pickup and split delivery problem. *European Journal of Operational Research*, 218(3):764–774, 2012.

[18] F. Hennig, B. Nygreen, K. C. Furman, J.-H. Song, and G. R. Kocis. Crude oil tanker routing and scheduling. *INFOR*, 49(2):153–170, 2011.

[19] M. Hewitt, G. L. Nemhauser, M. W. P. Savelsbergh, and J.-H. Song. A Branch-and-Price guided search approach to maritime inventory routing. *Computers & Operations Research*, 40(5):1410–1419, 2013.

[20] M. Loparic, H. Marchand, and L. A. Wolsey. Dynamic knapsack sets and capacitated lot-sizing. *Mathematical Programming*, 95(1):53–69, 2003.

[21] D. J. Papageorgiou, M.-S. Cheon, A. B. Keha, G. L. Nemhauser, and J. Sokol. MIRPLib: A maritime inventory routing problem library. *Submitted*, 2012.

[22] D. J. Papageorgiou, M.-S. Cheon, G. L. Nemhauser, and J. Sokol. Approximate dynamic programming for a class of long-horizon maritime inventory routing problems. *Submitted*, 2013.

[23] J. A. Persson and M. Göthe-Lundgren. Shipment planning at oil refineries using column generation and valid inequalities. *European Journal of Operational Research*, 163(3):631–652, 2005.

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

[25] J. G. Rakke, M. Stålhane, C. R. Moe, M. Christiansen, H. Andersson, K. Fagerholt, and I. Norstad. A rolling horizon heuristic for creating a liquefied natural gas annual delivery program. *Transportation Research Part C: Emerging Technologies*, 19(5):896–911, 2011.

[26] R. Rocha, I. E. Grossmann, and M. V. P. de Aragão. Cascading knapsack inequalities: reformulation of a crude oil distribution problem. *Annals of Operations Research*, 203(1):1–18, 2013.

[27] D. Ronen. Marine inventory routing: shipments planning. *Journal of the Operational Research Society*, 53:108–114, 2002.

[28] Y. Shao, K. C. Furman, and V. Goel. Improved large neighborhood search for LNG inventory routing. *Submitted*, 2013.

[29] J.-H. Song and K. C. Furman. A maritime inventory routing problem: Practical approach. *Computers & Operations Research*, 40(3):657–665, 2013.

[30] M. Stålhane, J. G. Rakke, C. R. Moe, H. Andersson, M. Christiansen, and K. Fagerholt. A construction and improvement heuristic for a liquefied natural gas inventory routing problem. *Computers & Industrial Engineering*, 62(1):245–255, 2012.