Recent Progress Using Matheuristics for Strategic Maritime Inventory Routing

Dimitri J. Papageorgiou, Myun-Seok Cheon, Stuart Harwood, and Francisco Trespalacios

Corporate Strategic Research

ExxonMobil Research and Engineering Company

1545 Route 22 East, Annandale, NJ 08801 USA

{dimitri.j.papageorgiou,myun-seok.cheon,stuart.m.harwood,francisco.trespalacios}@exxonmobil.com

George L. Nemhauser

H. Milton Stewart School of Industrial and Systems Engineering

Georgia Institute of Technology

765 Ferst Drive NW, Atlanta, Georgia, 30332

gnemhaus@isye.gatech.edu

Abstract

This paper presents an extensive computational study of simple, but prominent matheuristics (i.e., heuristics that rely on mathematical programming models) to find high quality ship schedules and inventory policies for a class of maritime inventory routing problems. Our computational experiments are performed on a set of the publicly available MIRPLib instances. This class of inventory routing problems has few constraints relative to some operational problems, but has long planning horizons, which make the time dimension challenging. We compare several variants of rolling horizon heuristics, K-opt heuristics, local branching, solution polishing, and hybrids thereof. Many of these matheuristics substantially outperform the commercial mixed-integer programming solvers CPLEX 12.6.2 and Gurobi 6.5 in their ability to quickly find high quality solutions. New best known incumbents are found for 26 out of 70 yet-to-be-proved-optimal instances and new best known bounds on 56 instances.

Keywords: deterministic inventory routing, matheuristics, maritime transportation, mixed-integer linear programming, time decomposition.

1 Introduction

In 2014, the volume of world seaborne shipments was estimated to be 9.84 billion tons, accounting for roughly 80% of total world merchandise trade [48]. The petrochemical sector was responsible for transporting approximately 32% of this tonnage with 17% going to crude oil, 9% to petroleum products, and 6% to gas and chemicals [48, Figure 1.3]. The transportation costs associated with these commodities directly affect the economic viability of certain projects and thus effective supply chain decision support tools are critical to

ensure profitability. This paper discusses recent progress using matheuristics (i.e., heuristics that rely on a mathematical programming model) to find high quality ship schedules and inventory policies for a particular maritime transportation problem known as the Maritime Inventory Routing Problem (MIRP), which plays an integral role in global bulk shipping. As there have been several papers over the last five years that have investigated matheuristics for particular (often proprietary) applications, we review and evaluate several prominent methods on a publicly available data set.

Inventory routing problems (IRPs) involve the integration and coordination of two components of the logistics value chain: inventory management and vehicle routing. After their emergence in the industrial gases industry, IRPs are now prominent in a number of business sectors with maritime IRPs arising in the petrochemical sector being a primary source of real-world IRP applications [15, p.2]. IRPs have come to prominence because they are an integral component in vendor managed inventory (VMI), a policy in which a central decision maker coordinates both the inventory and its distribution within a supply chain [12]. The survey paper on combined inventory management and vehicle routing problems by Andersson et al. [5] provides a summary of research on IRPs in road and maritime settings. The book chapter by Christiansen et al. [14] provides an overview of maritime transportation along with many references.

Relative to other transportation industries (e.g., air, rail, and truck), the maritime sector has been slow to adopt optimization-based decision support tools for business planning and operations. As a consequence, relatively few commercial software tools have been developed to fill this niche market. Meanwhile, the complexity of such problems is often too great to be handled using spreadsheets and experience.

Matheuristics are heuristic algorithms made by the interoperation of metaheuristics and mathematical programming techniques [10]. They have garnered increasing attention in the last decade due to advances in mathematical programming software that can now reliably solve certain classes of optimization problems to near optimality within an acceptable time limit [9]. Matheuristics for IRPs are discussed in [8] and, more generally, for vehicle routing problems in [6]. They have been applied in other shipping applications, such as in liner shipping network design Brouer et al. [11].

The guiding tenet behind the matheuristics presented here is the desire to maintain a single mixed-integer linear programming (MILP) formulation and to find high quality solutions to it by iteratively fixing subsets of variables and solving the resulting MILP with a general purpose MILP solver. Specifically, consider a generic MILP

$$\min \quad \mathbf{c}^\top \mathbf{x} \tag{1a}$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{x} \geq \mathbf{b} \tag{1b}$$

$$\mathbf{x} \in \mathbb{Z}^{n_1} \times \mathbb{R}^{n_2} \tag{1c}$$

where $n = n_1 + n_2$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, and $\mathbf{c} \in \mathbb{R}^n$. Let $\mathcal{J} \subseteq \{1, \ldots, n\}$ be a subset of the decision variables. The class of matheuristics investigated here work by fixing $x_j = \hat{x}_j$ for all $j \notin \mathcal{J}$ and solving the restricted MILP

$$\min \quad \sum_{j \in \mathcal{J}} c_j x_j + \underbrace{\sum_{j \notin \mathcal{J}} c_j \hat{x}_j}_{\text{constant}} \tag{2a}$$

$$\text{s.t.} \quad \sum_{j \in \mathcal{J}} a_{ij} x_j + \underbrace{\sum_{j \notin \mathcal{J}} a_{ij} \hat{x}_j}_{\text{constant}} \geq b_i \qquad \forall i = 1, \ldots, m \tag{2b}$$

$$\mathbf{x} \in \mathcal{X}(\mathcal{J}) \ , \tag{2c}$$

where $\mathcal{X}(\mathcal{J}) = \{\mathbf{x} \in \mathbb{Z}^{n_1} \times \mathbb{R}^{n_2} : x_j = \hat{x}_j \text{ for all } j \notin \mathcal{J}\}$ is the neighborhood induced by $\mathcal{J}$. By judiciously selecting the set $\mathcal{J}$ of variables to be locally optimized, the induced neighborhoods (in this case, smaller, more tractable MILPs) can be searched more easily by a generic MILP solver. In other words, these matheuristics rely on "hard fixing" of variables to induce search neighborhoods. This is in contrast to, for example, local branching, a popular heuristic now available in all major MILP solvers, which searches restricted MILPs by adding "soft" constraints to (1) that allow at most some positive integer K variables to change from their current value [19]. This reliance on a single mathematical program is also in stark contrast to a branch-and-price algorithm in which two mathematical programs are maintained: one for the restricted master problem and one for the pricing subproblem. In the latter, if the problem statement changes, then both the master problem and pricing problems must be updated, debugged, and validated.

There are several noteworthy benefits of restricting ourselves to this class of matheuristics: 1) **Sustainment.** The issue of sustaining decision support software is rarely discussed in the academic domain, but is often a key factor in the acceptance, deployment, and success of a tool for an industrial application. It is typically easier for an optimization team to hand off a single mathematical programming model that relies on a general purpose solver, as opposed to low-level C code that may require an extended maintenance effort. This is particularly important as those who are tasked to support and maintain the software may change roles over time. 2) **Adapting to new business settings.** When the business environment and/or problem changes and new model features (e.g., constraints) are needed, it is preferable to update a single mathematical programming model as opposed to several models and the corresponding algorithms used to solve them. 3) **Handling extensions.** Deterministic optimization models can be extended to scenario-based stochastic programs with relative ease.

1.1 Literature review

1.1.1 Brief review of matheuristics

Archetti and Bertazzi [6] classify matheuristics for vehicle routing problems (a superset of IRPs) into three classes, which we state verbatim:

1. *Decomposition approaches.* In general, in a decomposition approach the problem is divided into smaller and simpler subproblems and a specific solution method is applied to each subproblem. In matheuristics, some or all these subproblems are solved through mathematical programming models to optimality or suboptimality.
2. *Improvement heuristics.* Matheuristics belonging to this class use mathematical programming models to improve a solution found by a different heuristic approach. They are very

common as they can be applied whatever heuristic is used to obtain a solution that the mathematical programming model aims at improving.

3. *Branch-and-price/column generation-based approaches.* Branch-and-price algorithms have been widely and successfully used for the solution of routing problems. Such algorithms make use of a set partitioning formulation, where a binary or integer variable is associated with each possible route (column). Due to the exponential number of variables, the solution of the linear relaxation of the formulation is performed through column generation. In the branch-and-price/column generation-based matheuristics the exact method is modified to speed up the convergence, thus losing the guarantee of optimality. For example, the column generation phase is stopped prematurely.

In this paper, we focus exclusively on matheuristics in the first two categories.

1.1.2 Brief review of MIRPs

Papageorgiou et al. [36] provide a detailed survey of MIRPs with inventory tracking at all ports, i.e., MIRPs in which inventory levels at all loading and discharging ports must stay within prespecified bounds during every time period throughout the entire planning horizon. To avoid significant overlap with that review, we discuss some applications and recent work in this section. We discuss several algorithmic papers in Section 3 when we discuss the matheuristics compared in this study so that highly relevant methods are discussed in context.

Several notable case studies involving real-world maritime inventory routing applications have appeared in the literature. Dauzère-Pérès et al. [17] describe a case study in VMI involving a Norwegian supplier of calcium carbonate slurry, a product used in paper manufacturing. The supplier is responsible for routing a fleet of heterogeneous vessels and for maintaining sufficient inventory levels of up to sixteen products at ten tank farms in Northern Europe. Christiansen et al. [13] present a MIRP encountered by a major cement producer involving bulk ships with multiple compartments that transport multiple non-mixable cement products. Furman et al. [21] present an arc-flow MILP model embedded in a decision support tool used to aid decision-makers in the tactical routing and inventory management of vacuum gas oil (VGO) at ExxonMobil. This class of single product MIRPs is a tramp shipping application involving voyage chartered vessels or spot charters, i.e., vessels that are chartered for a single voyage from a loading region to a discharging region.

To solve models akin to those described in [21], Song and Furman [42] apply a large neighborhood search to an arc-flow model that extends the ideas introduced in Savelsbergh and Song [40]. In particular, after an initial solution is generated, a local search procedure, akin to a 2-opt procedure, is applied in which the decision variables associated with all but two vessels are fixed and an exact optimization algorithm is called to locally optimize the decisions for these two vessels. This procedure is applied for up to $\binom{|\mathcal{V}|}{2}$ iterations, where $|\mathcal{V}|$ is the number of vessels and vessel pairs are chosen randomly in each iteration. Working off of a simpler problem than the one considered in [21] and [42], Engineer et al. [18] formulate a path-flow model and apply a branch-cut-and-price approach for solving the problem. Three types of valid inequalities are suggested that generalize valid inequalities presented in previous work. Hewitt et al. [30] also attempt to generate good solutions quickly for the instances considered in [18] with branch-and-price guided search (BPGS) [29], a technique that systematically searches restricted neighborhoods of a MILP using information from an extended formulation in the master problem. They consider a much richer set of local search neighborhoods than previously studied and show that, after parallelizing their code on four processors, BPGS is quite

effective at finding high-quality solution in 30 minutes for the MIRP instances considered.

With few exceptions, MIRPs appearing in the literature review can be cast as mixed-integer linear programs (MILPs). In general, as the number of components (e.g., vessels, ports, and time periods) increase so too does the computation time required to solve the corresponding problem instances. Worse, there is often a point at which attempting to solve a realistic instance using a commercially available solver yields unacceptable performance, e.g., a low quality solution in a given time limit. To combat these limitations, numerous heuristics have been developed that exploit mathematical programming solvers.

In the liquefied natural gas (LNG) domain, several heuristics for LNG-IRPs have been studied. Rakke et al. [38] propose a rolling horizon heuristic in which a sequence of overlapping MILP subproblems are solved. Each subproblem involves at most 3 months of data and consists of a one-month "central period" and a "forecasting period" of at most two months. Once a best solution is found (either by optimality or within a time limit), all decision variables in the central period are fixed at their respective values and the process "rolls forward" to the next subproblem. Stålhane et al. [43] propose a construction and improvement heuristic that creates scheduled voyages based on the availability of vessels and product while keeping inventory feasible. Goel et al. [24] also use construction and improvement heuristics within a large neighborhood search. To address huge instances, Asokan et al. [7] demonstrated how these neighborhood searches can be parallelized to significantly reduce computation time. Goel et al. [25] introduce a constraint programming approach for the same LNG-IRP that can achieve an order of magnitude speedup relative to CPLEX 11.1. Mutlu et al. [33] employ a vessel routing heuristic for an LNG-IRP that iteratively builds ship routes that ensure delivery times and volume requirements of each contract are met.

Solution techniques for MIRPs faced by a vertically integrated company are presented in [36], where MIRPs with inventory tracking at every port are surveyed. Fodstad et al. [20] solve a MILP directly while Uggen et al. [47] present a fix-and-relax heuristic. Goel et al. [24] present a simple construction heuristic and adapt the local search procedure of Song and Furman [42] to generate solutions to instances with 365 time periods. Their model seeks to minimize penalties and does not consider travel costs.

More recently, several papers have appeared on short-sea shipping applications. Agra et al. [2] consider and combine three heuristic procedures for a fuel oil distribution problem. A rolling horizon heuristic is used to decompose the original problem into smaller and more tractable problems, feasibility pump is used to find initial solutions for MILP problems, and local branching is used to improve feasible solutions. Hemmati et al. [28] evaluated the potential of having a VMI service in tramp shipping to provide flexibility in delivery time and cargo quantities. Hemmati et al. [27] propose a two-phase hybrid matheuristic to solve a multi-product short sea MIRP. They first convert the MIRP into a ship routing and scheduling problem and then apply an adaptive large neighborhood search to solve the resulting optimization problem. Jiang and Grossmann [31] evaluate a number of MILP formulations for an operational MIRP; no heuristics are applied.

1.2 Contributions of this paper

The contributions of this paper are:

1. Several variants of popular matheuristics used in the MIRP literature are presented and adapted to a particular class of long-horizon MIRPs.

2. An extensive computational study comparing a suite of simple matheuristics that rely on the same underlying mathematical programming formulation is performed.

3. New best known incumbents are found for 26 out of 70 yet-to-be-proved-optimal instances and new best known bounds on 56 instances.

The remainder of the paper is organized as follows: In Section 2, we present a discrete-time arc-flow formulation of the problem. In Section 3, we describe the matheuristics used in our computational study, while citing other works in which they have been used. Section 4 includes a detailed computational comparison of these matheuristics against one another as well as against commercial MILP solvers CPLEX 12.6.2 and Gurobi 6.5. Concluding remarks and suggestions for future research directions are presented in Section 5.

2 Problem Description and Formulations

We consider a single product MIRP that involves a planning horizon of T time periods, where $\mathcal{T} = \{1, \ldots, T\}$ is the set of all time periods in the horizon, as well as three main components: ports, vessels, and vessel classes. Each port is classified as a loading port (where product is produced) or as a discharging port (where product is consumed). Let $\mathcal{J}^P$ denote the set of loading (production) ports, $\mathcal{J}^C$ the set of consumption (or discharging) ports, and $\mathcal{J} = \mathcal{J}^P \cup \mathcal{J}^C$ the set of all ports. Let the parameter Δ_j be 1 if $j \in \mathcal{J}^P$ and -1 if $j \in \mathcal{J}^C$. Product can be stored in inventory at both types of ports. Each port has: an inventory capacity of $S^{\max}_{j,t}$; a fixed number of berths B_j limiting the number of vessels that can simultaneously load or discharge in a given period; and a deterministic, but possibly non-constant, per-period rate $D_{j,t}$ of production or consumption, i.e., $D_{j,t}$ denotes the amount of product produced or consumed at port j in time period t. The amount of inventory at the end of time period t must be between 0 and $S^{\max}_{j,t}$.

Since it may not be possible to satisfy all demand or avoid hitting tank top, we include a simplified spot market so that a consumption port may buy product and a production port can sell excess inventory whenever necessary. The penalty parameter $P_{j,t}$ denotes the unit cost associated with the spot market at port j in time period t. We assume that $P_{j,t} > P_{j,t+1}$ for all $t \in \mathcal{T}$ so that the spot market is only used as late as possible, i.e., to ensure that a solution will not involve lost production (stockout) until the inventory level reaches capacity (falls to zero). When the penalty parameters are large (like a traditional "Big M" value), inventory bounds can be considered "hard" constraints. When they are small, however, inventory bounds can be treated as "soft" constraints. This "soft" interpretation may be beneficial in strategic planning problems for several reasons. First, a user may attempt to solve an instance with a demand forecast that cannot be met by the existing fleet in order to understand the limitations of the current infrastructure (see also [24]). Second, the inventory bounds given as input may be overly conservative in order to make the solution more robust when, in fact, slight bound violations may be acceptable. Third, incurring a small penalty for a particular solution (as opposed to declaring it strictly infeasible) can mitigate minor unwanted effects of using a discrete-time model [36].

Vessels travel from port to port, loading and discharging product. We assume vessels fully load and fully discharge at a port and that direct deliveries are made. Each vessel belongs to a vessel class $vc \in \mathcal{VC}$. Vessel class vc has capacity Q^{vc}. Vessels are owned by the supplier or time-chartered for the entire planning horizon. We assume that port capacity always exceeds vessel capacity, i.e., $S^{\max}_{j,t} \geq \max\{Q^{vc} : vc \in \mathcal{VC}\}$ and that vessels can fully load or discharge in a single period. These assumptions allow vessels to load or discharge in the same period in which they leave a port so that loading and discharging decisions do not need to be explicitly modeled.

As is done in several other works, the problem is modelled on a time-expanded network. 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 vessel classes, while each vessel class has its own arc set $\mathcal{A}^{vc}$. The set $\mathcal{N}_{0,T+1}$ of nodes consists of a set $\mathcal{N} = \{(j,t) : j \in \mathcal{J}, t \in \mathcal{T}\}$ of "regular" nodes, or port-time pairs, as well as a source node n_0 and a sink node n_{T+1}.

Associated with each vessel class vc is a set $\mathcal{A}^{vc}$ of arcs, which can be partitioned into source, sink, waiting, and travel arcs. A source arc $a = (n_0, (j,t))$ from the source node to a regular node represents the arrival of a vessel to its initial destination. A sink arc $a = ((j,t), n_{T+1})$ from a regular node to the sink node conveys that a vessel is no longer being used and has exited the system. A waiting arc $a = ((j,t),(j,t+1))$ from a port j in time period t to the same port in time period $t+1$ represents that a vessel stays at the same port in two consecutive time periods. Finally, a travel arc $a = ((j_1,t_1),(j_2,t_2))$ with $j_1 \neq j_2$ represents travel between two distinct ports, where the travel time $(t_2 - t_1)$ between ports is given. If a travel or sink arc is taken, we assume that a vessel fully loads or discharges immediately before traveling. The cost of traveling on arc $a \in \mathcal{A}^{vc}$ is C_a^{vc}.

The set of all travel and sink arcs for each vessel class are denoted by $\mathcal{A}^{vc,\text{inter}}$ (where "inter" stands for "inter-regional"). The set of incoming and outgoing arcs associated with vessel $vc \in \mathcal{VC}$ at node $n \in \mathcal{N}_{0,T+1}$ are denoted by $\mathcal{RS}_n^{vc}$ (for reverse star) and $\mathcal{FS}_n^{vc}$ (for forward star), respectively. Similarly, $\mathcal{FS}_n^{vc,\text{inter}}$ denotes the set of all outgoing travel and sink arcs at node n for vessel class vc. For our strategic planning problem, modeling the flow of vessel classes avoids the additional level of detail associated with modeling each individual vessel. Moreover, we found that modeling vessel classes could remove symmetry and improve solution times by more than an order of magnitude on large instances.

Assumptions: For ease of reference, we collect the assumptions made throughout this paper: (1) There is exactly one port within each region; (2) Port capacity always exceeds the capacity of the vessels, e.g., $S_{j,t}^{\max} \geq \max\{Q^{vc} : vc \in \mathcal{VC}\}$; (3) Travel times are deterministic; (4) Vessels can fully load or discharge in a single period (in other words, the time to load/discharge is deterministic and built into the travel time); (5) Production and consumption rates are known; (6) There is a single loading port ($|\mathcal{J}^P| = 1$) as is typically the case for LNG-IRPs [37, 26, 43, 24]; (7) In a single time period, at most one vessel per vessel class may begin an outgoing voyage to a given discharging port or a return voyage to the loading port. The reason for this assumption is that, in practice, a strategic planner is interested in avoiding scheduling conflicts at a port when a plan is actually enacted. Said differently, a schedule in which two large vessels of the same class must load or discharge simultaneously at the exact same port is perceived to be a non-robust solution.

2.1 A Discrete-Time Arc-Flow Mixed-Integer Linear Programming Model

We next define the decision variables. Let x_a^{vc} be the number of vessels in vessel class vc that travel on arc $a \in \mathcal{A}^{vc}$. Let $s_{j,t}$ be the ending inventory at port j in time period t. Initial inventory $s_{j,0}$ is given as data. Finally, let $\alpha_{j,t}$ be the amount of inventory bought from or sold to the spot market near port j in time period t.

We consider the following discrete-time arc-flow MILP model, which was introduced in Papageorgiou et al. [34]:

$$\min \quad \sum_{vc\in\mathcal{VC}} \sum_{a\in\mathcal{A}^{vc}} C_a^{vc} x_a^{vc} + \sum_{j\in\mathcal{J}} \sum_{t\in\mathcal{T}} P_{j,t}\alpha_{j,t} \tag{3a}$$

$$\text{s.t.} \quad \sum_{a\in\mathcal{FS}_n^{vc}} x_a^{vc} - \sum_{a\in\mathcal{RS}_n^{vc}} x_a^{vc} = \begin{cases} +1 & \text{if } n = n_0 \\ -1 & \text{if } n = n_{T+1} \\ 0 & \text{if } n \in \mathcal{N} \end{cases}, \qquad \forall\, n \in \mathcal{N}_{0,T+1}, \forall\, vc \in \mathcal{VC} \tag{3b}$$

$$s_{j,t} = s_{j,t-1} + \Delta_j \left(D_{j,t} - \sum_{vc\in\mathcal{VC}} \sum_{a\in\mathcal{FS}_n^{vc,\text{inter}}} Q^{vc} x_a^{vc} - \alpha_{j,t} \right), \quad \forall\, n = (j,t) \in \mathcal{N} \tag{3c}$$

$$\sum_{vc\in\mathcal{VC}} \sum_{a\in\mathcal{FS}_n^{vc,\text{inter}}} x_a^{vc} \le B_j\,, \qquad \forall\, n = (j,t) \in \mathcal{N} \tag{3d}$$

$$\alpha_{j,t} \ge 0\,, \qquad \forall\, n = (j,t) \in \mathcal{N} \tag{3e}$$

$$s_{j,t} \in [0, S_{j,t}^{\max}]\,, \qquad \forall\, n = (j,t) \in \mathcal{N} \tag{3f}$$

$$x_a^{vc} \in \{0,1\}\,, \qquad \forall\, vc \in \mathcal{VC}\,, \forall\, a \in \mathcal{A}^{vc,\text{inter}} \tag{3g}$$

$$x_a^{vc} \in \mathbb{Z}_+\,, \qquad \forall\, vc \in \mathcal{VC}\,, \forall\, a \in \mathcal{A}^{vc} \setminus \mathcal{A}^{vc,\text{inter}}\,. \tag{3h}$$

The objective is to minimize the sum of all transportation costs and penalties for lost production and stockout. Constraints (3b) require flow balance of vessels within each vessel class. Constraints (3c) are inventory balance constraints at loading and discharging ports, respectively. Berth limit constraints (3d) restrict the number of vessels that can attempt to load/discharge at a port at a given time. This formulation requires that a vessel must travel at capacity from a loading region to a discharging region and empty from a discharging region to a loading region. This model does not require decision variables for tracking inventory on vessels (vessel classes), nor does it include decision variables for the quantity loaded/discharged in a given period.

This model is similar to the one studied in Goel et al. [24]. The major differences are that they do not include travel costs in the objective function; they model each vessel individually (in other words, there is only one vessel per vessel class); they model consumption rates as decision variables with upper and lower bounds; and they include an additional set of continuous decision variables to account for cumulative unmet demand at each consumption port.

2.2 Additional valid inequalities: Mixed integer rounding cuts

It is unclear if additional constraints are beneficial for finding better primal solutions. On the one hand, more constraints typically imply that a solver must do more work at each search tree node, e.g., during each dual simplex update since the constraint matrix $\mathbf{A}$ is larger. On the other hand, a tighter relaxation often reduces the number of nodes that must be explored. Below we describe a set of mixed integer rounding cuts that can be added to the formulation. These cuts were originally proposed in Engineer et al. [18] and later in Agra et al. [1] and Papageorgiou et al. [36]. For completeness, a formal derivation is provided in the appendix.

Let $R_{j,t}$ be the cumulative amount of product required for pick up or delivery at port j by time period t, i.e.,

$$R_{j,t} = \begin{cases} \sum_{t'\le t} D_{j,t'} - s_{j,0} & \text{if} \quad j \in \mathcal{J}^C, \\ \sum_{t'\le t} D_{j,t'} + s_{j,0} - S_j^{\max} & \text{if} \quad j \in \mathcal{J}^P. \end{cases} \tag{4}$$

Then, for any scalar $Q > 0$, the following mixed integer rounding cuts are valid:

$$\sum_{vc \in \mathcal{VC}} \sum_{t' \leq t} \sum_{a \in \mathcal{FS}^{vc,inter}_{j,t'}} \left(\left\lceil \frac{Q^{vc}}{Q} \right\rceil - \frac{(F_{vc} - F^0_{j,t})^+}{1 - F^0_{j,t}} \right) x^{vc}_a + \sum_{t' \leq t} \frac{\alpha_{j,t'}}{Q(1 - F^0_{j,t})} \geq \left\lceil \frac{R_{j,t}}{Q} \right\rceil \qquad \forall (j,t) \in \mathcal{N}, \qquad (5)$$

where $F^0_{j,t} = \frac{R_{j,t}}{Q} - \left\lceil \frac{R_{j,t}}{Q} \right\rceil$, $F_{vc} = \frac{Q^{vc}}{Q} - \left\lceil \frac{Q^{vc}}{Q} \right\rceil$. These constraints are valid as they result from a direct application of the mixed-integer rounding cut (11) (see the appendix) applied to the constraint

$$\sum_{vc \in \mathcal{VC}} \sum_{t' \leq t} \sum_{a \in \mathcal{FS}^{vc,inter}_{j,t'}} \frac{Q^{vc}}{Q} x^{vc}_a + \frac{1}{Q} \sum_{t' \leq t} \alpha_{j,t'} \geq \frac{R_{j,t}}{Q} \qquad \forall (j,t) \in \mathcal{N} \ ,$$

for any $Q > 0$. In our experiments, we set $Q = Q^{\max} = \max\{Q^{vc} : vc \in \mathcal{VC}\}$. Of course, one could choose several values of Q to derive an assortment of cuts. It is not yet known if including a vast array of problem-specific cuts will outperform the myriad of general-purpose cuts already available in a MILP solver's cut library.

Although constraints (5) only consider cumulative supply and demand in a time interval $[0, t]$, it is possible to include such constraints for any time interval $[t_1, t_2]$ for any $t_1, t_2 \in \mathcal{T}$ such that $t_1 < t_2$. To do so, one must define

$$R_{j,t_1,t_2} = \begin{cases} D_{j,[t_1,t_2]} - S^{\max}_{j,t_1-1} + S^{\min}_{j,t_2} & \text{if} \quad j \in \mathcal{J}^C, \\ D_{j,[t_1,t_2]} + S^{\min}_{j,t_1-1} - S^{\max}_{j,t_2} & \text{if} \quad j \in \mathcal{J}^P, \end{cases}$$

where $D_{j,[t_1,t_2]} = \sum_{t'=t_1}^{t_2} D_{j,t'}$, and $S^{\min}_{j,t}$ and $S^{\max}_{j,t}$ are known or improved bounds on $s_{j,t}$. This is particularly relevant when implemented within a rolling horizon approach since s_{j,t_1-1} will be known when t_1 is the starting time period of the interval being optimized. In this case, we have $S^{\min}_{j,t_1-1} = S^{\max}_{j,t_1-1} = s_{j,t_1-1}$.

3 Matheuristics explored

3.1 Construction heuristics

3.1.1 Reducing the number of time periods

Aggregation in space and/or time are commonplace when attempting to reduce the solution space of a large optimization problem. In this approach, we attempt to reduce the number of time periods in which vessels can travel between distinct ports. In our implementation, rather than manipulate the set $\mathcal{T}$ of time periods, we limited the set of travel arcs (i.e., arcs that involve two distinct ports) so that vessels may only travel in even-numbered periods. The berth limit constraints were not modified. We attempted several other variants, but did not see any benefits for this class of problems.

3.1.2 Rolling horizon heuristics

Arguably the most common form of time decomposition in engineering applications is some form of rolling horizon heuristic (RHH). In the IRP literature, Al-Ameri et al. [3] present a rolling horizon framework for vendor-managed inventory systems. In the MIRP literature, this approach was applied in Al-Khayyal and Hwang [4], Rakke et al. [38], Agra et al. [2], and others. The basic idea of an RHH tailored to time-based IRPs is to solve a sequence of overlapping MILP subproblems. Each subproblem consists of a "central

period" and a "forecasting period." Once an optimal solution to this subproblem is found (or sub-optimal solution within a time limit), all decision variables in the central period are fixed at their respective values and the process "rolls forward" to the next subproblem.

Our implementation is shown in Algorithm 1. Here τ^{central} and τ^{forecast} are the number of periods in the central and forecast periods, respectively. The set $\mathcal{T}^{\text{start}} = \{t \in \mathcal{T} : t = 1 + \nu\tau^{\text{central}} \text{ for } \nu \in \mathbb{N}\}$ denotes the set of all starting periods to consider in the procedure. The algorithm is initialized by forcing all product to be bought and sold from the spot market $\alpha_{j,t} = D_{j,t}$ for all (j,t), which implies that inventory remains constant $s_{j,t} = s_{j,t-1}$ at all ports over the entire planning horizon. Meanwhile, all vessels movements are initially forbidden, i.e., $x_a^{vc} = 0$ for all $vc \in \mathcal{VC}, a \in \mathcal{A}^{vc}$. Note that this means that vessel balance constraints (3b) are not feasible at the source node and the initial solution is infeasible. This infeasibility is immediately resolved within the first solve. The algorithm is straightforward except for one subtle feature not common in all rolling horizon heuristics: In Step 6, in all but the final solve, it is wise for this application to forbid vessels to exit the system in the central period. Without this fixing, it may be optimal with a myopic forecast to have some vessels exit the system (take a sink arc) in the central period, which will in turn make these vessels unavailable in all future periods even after the algorithm rolls forward.

Algorithm 1 Rolling Horizon Heuristic

1: Initialization: Set $\alpha_{j,t} = D_{j,t}\ \forall (j,t); s_{j,t} = s_{j,t-1}\ \forall (j,t); x_a^{vc} = 0\ \forall vc \in \mathcal{VC}, a \in \mathcal{A}^{vc}$
2: **for** $t^{\text{start}} \in \mathcal{T}^{\text{start}}$ **do**
3: $t^{\text{end}} = \min\{t^{\text{start}} + \tau^{\text{central}} + \tau^{\text{forecast}}, T\}$
4: Fix all decision variables with index $t < t^{\text{start}}$ or $t > t^{\text{end}}$ to their current value
5: Deactivate all constraints with index $t < t^{\text{start}}$ or $t > t^{\text{end}}$
6: If $\texttt{Ord}(t^{\text{start}}) < \texttt{Card}(\mathcal{T}^{\text{start}})$, fix all sink arc variables $x^{vc}_{((j,t),n_{\mathcal{T}+1})} = 0$ for all $t \leq t^{\text{start}} + \tau^{\text{central}}$
7: Solve Model (3)
8: **end for**

Several variants are worth mentioning. Uggen et al. [47] present a fix-and-relax heuristic in which the forecast period includes all remaining time periods and all integer decision variables in the forecast period are relaxed to be continuous. This did not work well for this class of problems, so we did not adopt it. Goel et al. [24] and Shao et al. [41] apply a rolling horizon framework within a construction heuristic, but without solving a mathematical program. From a high level, the approximate dynamic programming method proposed in Papageorgiou et al. [34], the first of its kind for maritime inventory routing problems, can be viewed as a type of rolling horizon heuristic that uses value functions to approximate the value of sending vessels to certain ports in the forecast period. It is more complex than the matheuristics evaluated in this work. Similarly, Toriello et al. [46] apply approximate dynamic programming to address a deterministic IRP. It is one of the first papers to apply value function approximations to an IRP, extending a body of work on dynamic fleet management problems [22, 23, 45, 44].

3.1.3 Restricting port-vessel class compatibilities

For instances with many ports and vessel classes, Model (3) requires many binary variables, namely, one for each tuple $(vc, j_1, t, j_2, t + \tau_{j_1,j_2})$ where τ_{j_1,j_2} is the travel time between ports j_1 and j_2. If one were to restrict certain vessel class from traveling to certain discharging ports, then fewer binary decisions would be required. The approach that follows attempts to do just that in a systematic manner. We formulate a MILP

whose purpose is to identify a set of port-vessel class incompatibilities that lead to a restricted version of Model (3) that is easier to solve, but still generates high-quality solutions. Interestingly, this approach could be viewed as a type of a "Cluster first - route second" approach discussed in [6].

The basic idea behind the MILP below is to solve a much simplified variant of Model (3) with the requirement that at least $K \in \mathbb{N}$ port-vessel class pairs are incompatible so that a subset of less promising travel arcs are eliminated. Unlike Model (3), this MILP models each individual vessel, but does not involve any sequencing. It exploits the fact that there is only one supply port by assuming all vessels originate at the supply port. For those vessels that originate at a demand port, we assume that they can discharge in the first period in which they are available so that they then travel immediately to the supply port. This travel time is then deducted from the vessel's remaining time available to service other ports. Waiting time (i.e., successive time periods spent at a port) is not modeled. The model is described below.

Indices and sets

$t \in \mathcal{T}'$	subset of time periods considered; we assume $T \in \mathcal{T}'$
$v \in \mathcal{V}$	set of individual vessels

Parameters

$T^{\text{Rem}}_{t,v}$	remaining time available up to time t for vessel v from its first possible arrival to the supply port
$T^{\text{OW}}_{j,v}$	one-way travel time from the supply port to demand port j for vessel v
$T^{\text{RT}}_{j,v}$	roundtrip travel time from the supply port to demand port j (and back) for vessel v
$C^{\text{OW}}_{j,v}$	travel cost of a one-way voyage from the supply port to demand port j for vessel v
$C^{\text{RT}}_{j,v}$	travel cost of a roundtrip voyage from the supply port to demand port j (and back) for vessel v
$D^{\text{LB}}_{j,t}$	cumulative excess demand at demand port j up to time period t
$D^{\text{UB}}_{j,t}$	maximum demand at demand port j up to time period t
P_j	penalty for unmet demand: $P_j = P_{j,1}$ for all j

The parameters $T^{\text{Rem}}_{t,v}$, $D^{\text{LB}}_{j,t}$, and $D^{\text{UB}}_{j,t}$ are set as follows: For a vessel originating at the supply port, $T^{\text{Rem}}_{t,v} = [t - T^{\text{start}}_v]^+$. For a vessel originating at a discharging port, we set $T^{\text{Rem}}_{t,v} = [t - T^{\text{start}}_v - T^{\text{OW}}_{j(v),v}]^+$, where $j(v)$ denotes the discharging port where vessel v originates. This is an optimistic value since the vessel may not be able to fully discharge in the same period in which it enters the system/network. We set $D^{\text{LB}}_{j,t} = R_{j,t} - \bar{S}_{j,t}$, where $R_{j,t}$, defined in (4), is the cumulative amount of product required to be delivered by time period t defined in (4) and $\bar{S}_{j,t} = \sum_{v \in \mathcal{V}: j(v)=j, t \leq T^{\text{start}}_v} Q_v$ is the expected amount discharged by vessels originating at discharging port j by time t before making their first voyage to the supply port. Meanwhile, $D^{\text{UB}}_{j,t} = D^{\text{LB}}_{j,t} + S^{\max}_j$. All other parameters listed above are given as data.

Decision variables

$\sigma_{j,t}$	(continuous) cumulative slack at demand port j up to time t
$x^{\text{OW}}_{j,t,v}$	(binary) cumulative number of one-way voyages to demand port j up to time period t made by vessel v
$x^{\text{RT}}_{j,t,v}$	(integer) cumulative number of roundtrip voyages to demand port j up to time period t made by vessel v
$z_{j,vc}$	(binary) takes value 1 if demand port j and vessel class vc are deemed incompatible; 0 otherwise

Port-Vessel Class Incompatibility MIP Model

$$\min \quad \sum_{j\in\mathcal{J}^C}\sum_{v\in\mathcal{V}}\left(C^{\mathrm{RT}}_{j,v}x^{\mathrm{RT}}_{j,T,v}+C^{\mathrm{OW}}_{j,v}x^{\mathrm{OW}}_{j,T,v}\right)+\sum_{j\in\mathcal{J}^C}P_j\sigma_{j,T} \tag{6a}$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{J}^C}T^{\mathrm{RT}}_{j,v}x^{\mathrm{RT}}_{j,t,v}+T^{\mathrm{OW}}_{j,v}x^{\mathrm{OW}}_{j,t,v}\le T^{\mathrm{Rem}}_{t,v}\ , \qquad \forall\ t\in\mathcal{T}', v\in\mathcal{V} \tag{6b}$$

$$\sum_{v\in\mathcal{V}}Q^v(x^{\mathrm{RT}}_{j,t,v}+x^{\mathrm{OW}}_{j,t,v})+\sigma_{j,t}\ge D^{\mathrm{LB}}_{j,t}\ , \qquad \forall\ j\in\mathcal{J}^C, t\in\mathcal{T}' \tag{6c}$$

$$\sum_{v\in\mathcal{V}}Q^v(x^{\mathrm{RT}}_{j,t,v}+x^{\mathrm{OW}}_{j,t,v})+\sigma_{j,t}\le D^{\mathrm{UB}}_{j,t}\ , \qquad \forall\ j\in\mathcal{J}^C, t\in\mathcal{T}' \tag{6d}$$

$$x^{\mathrm{RT}}_{j,t,v}+x^{\mathrm{OW}}_{j,t,v}\ge x^{\mathrm{RT}}_{j,t-1,v}+x^{\mathrm{OW}}_{j,t-1,v}\ , \qquad \forall\ j\in\mathcal{J}^C, t\in\mathcal{T}' \tag{6e}$$

$$\sigma_{j,t}\ge\sigma_{j,t-1}\ , \qquad \forall\ j\in\mathcal{J}^C, t\in\mathcal{T}' \tag{6f}$$

$$T^{\mathrm{RT}}_{j,v}x^{\mathrm{RT}}_{j,T,v}+T^{\mathrm{OW}}_{j,v}x^{\mathrm{OW}}_{j,T,v}\le T^{\mathrm{Rem}}_{T,v}(1-z_{j,vc})\ , \qquad \forall\ j\in\mathcal{J}^C, vc\in\mathcal{VC}, v\in\mathcal{V}: VC(v)=vc \tag{6g}$$

$$\sum_{j,vc}z_{j,vc}\ge K\ , \tag{6h}$$

$$\sigma_{j,t}\ge 0\ , \qquad \forall\ j\in\mathcal{J}^C, t\in\mathcal{T}' \tag{6i}$$

$$x^{\mathrm{RT}}_{j,t,v}\in\mathbb{Z}_+\ , \qquad \forall\ j\in\mathcal{J}^C, t\in\mathcal{T}', v\in\mathcal{V} \tag{6j}$$

$$x^{\mathrm{OW}}_{j,t,v}\in\{0,1\}\ , \qquad \forall\ j\in\mathcal{J}^C, t\in\mathcal{T}', v\in\mathcal{V} \tag{6k}$$

$$z_{j,vc}\in\{0,1\}\ , \qquad \forall\ j\in\mathcal{J}^C, vc\in\mathcal{VC}\ . \tag{6l}$$

The objective is to minimize the total cost of one-way and roundtrip voyages and cumulative unmet demand penalties up to time T. Constraints (6b) limit the number of one-way and roundtrip voyages each vessel can make. Constraint (6h) requires at least K discharging port-vessel class pairs to be incompatible. Constraints (6c) and (6d) bound the amount of product that can be discharged at each discharging port up to time $t\in\mathcal{T}'$. Note that the more periods considered (i.e., the larger $\mathcal{T}'$ is), the more difficult it becomes to solve this MILP to provable optimality. For example, if $\mathcal{T}'=\mathcal{T}$, then this MILP would be almost as difficult to solve as Model (3). Constraints (6e) ensure that the cumulative number of voyages is correct. Note that it allows for a single one-way voyage to be made up to time period $t-1$ (e.g., $x^{\mathrm{OW}}_{j,t-1,v}=1$), but then the return trip to be completed after time period $t-1$ such that no one-way trip is made up to time period t (e.g., $x^{\mathrm{OW}}_{j,t,v}=0$).

Since the interplay of one-way and roundtrip voyages is important, the following example is meant to clarify how the two are used in Model (6). Consider an instance with a single supply port, a single demand port, and a single vessel originating at the supply port and available in the first time period. Assume a 181-day planning horizon with $\mathcal{T}=\{0,1,\ldots,180\}$. Assume that a one-way (roundtrip) voyage requires 10 (20) days. Assume that, in an optimal solution, the demand port is visited in time periods t^* for $t^*\in\{10,30,50,70,90,110,130,150,170\}$. Now, let $\mathcal{T}'=\{90,180\}$ be the subset of time periods considered in Model (6). Omitting the subscripts j and v in $x^{\mathrm{RT}}_{j,t,v}$ and $x^{\mathrm{OW}}_{j,t,v}$ (since there is only one demand port and one vessel), this optimal solution corresponds to a solution $x^{\mathrm{RT}}_{90}=4$, $x^{\mathrm{OW}}_{90}=1$, $x^{\mathrm{RT}}_{180}=8$, and $x^{\mathrm{OW}}_{180}=1$. That is, up to time period 90, there are four roundtrip voyages in which the vessel delivers in time periods $t^*\in\{10,30,50,70\}$ and a single one-way voyage in which the vessel delivers in time period 90. Up to time period 180, there are eight roundtrip voyages and a single one-way voyage. Note that if one-way voyages were only permitted in the last time period (i.e., $t=180$ in this example), only four roundtrips would have been feasible up to time period 90 possibly causing slack σ_{90} to be positive. Since $\sigma_{j,t}$ denotes *cumulative*

slack, an incorrect amount of slack would have been incurred.

Parameter settings: In our computational experiments, we set $K = 0.2|\mathcal{J}^C||\mathcal{VC}|$ and $\mathcal{T}' = \{t \in \mathcal{T} : t \bmod T/T' = 0\}$ with $T' = 12$ so that $T' = |\mathcal{T}'| = 12$.

3.2 Improvement heuristics

Below we discuss several matheuristics that are commonly used as improvement heuristics, i.e., as tools to improve existing feasible, or perhaps infeasible solutions.

3.2.1 K-opt local search

Borrowing from the K-opt local search procedures commonly used in combinatorial optimization problems, this heuristic fixes the routes for all but K vessel classes, and solves MILP (3) for a better solution over the neighborhood defined by the K unfixed vessel classes. This matheuristic is remarkably simple to implement and was used in [24, 30, 35, 34, 42], although often with routes for vessels, rather than vessel classes, being fixed and optimized. Pseudocode is provided in Algorithm 2.

Algorithm 2 K-opt Heuristic

```
while elapsedTime ≤ timeLimit and improvementOccurred do
  for 𝒱𝒞′ ∈ 𝒮 do
    Fix all decision variables associated with vessels in vessel class vc ∉ 𝒱𝒞′ to their current value
    Solve Model (3)
  end for
end while
```

The most interesting step in the algorithm is deciding which vessel classes should be fixed and in what order. In this paper, we simply use a predefined ordering so that the set $\mathcal{S}$ of vessel class tuples is given as input. Goel et al. [24] suggested a tailored scheme for dynamically choosing pairs of vessels to include in a 2-opt approach for an LNG-IRP. It is important to note that the order in which vessel classes are selected can have a significant impact on the performance of the algorithm.

3.2.2 1-opt with D-day time window flexibility search

This algorithm is a variant of the time-window improvement heuristic introduced in Goel et al. [24] and used subsequently in Shao et al. [41]. As proposed in [24], this heuristic searches in a neighborhood consisting of all solutions in which any voyage departure is delayed or advanced by at most D days, compared to the departure date for that voyage in the incumbent solution. We extend this heuristic by increasing the size of the neighborhood to include a `1opt` search. Specifically, in addition to allowing voyages $2D$ days of additional flexibility (D days before and D days after), we also unfix all variables associated with a single vessel class. Thus, this heuristic searches over larger neighborhoods than `1opt`.

3.2.3 Solution pool-based polishing

This algorithm seeks to improve the best known solution given a pool of solutions by attempting to exploit the best properties of these solutions. It is similar in spirit to that of [39]. The method solves an MILP in which travel and demurrage arcs are active (unfixed) only if they were selected by one of the solutions

in the solution pool. All other travel and demurrage arcs are removed from (fixed to zero in) the problem formulation. The MILP is initialized with the best solution from the pool, and can be terminated by achieving proven optimality or by reaching the time limit. Unlike other heuristic methods, this heuristic can take multiple solutions as input.

The solution pool can be generated by MILP solvers, or by saving the solutions found through different approaches. This heuristic can be used as a last step to improve the solutions found with other methods.

3.3 Hybrid Approaches

Combining the construction and improvement heuristics presented above has the potential to further improve solution quality. Several hybridizations are discussed below. Before proceeding, we pause to mention that it is not clear a priori if solving each portion of a rolling horizon heuristic to provable optimality is, indeed, better than solving a portion to near optimality before rolling forward. For example, an optimal solution to a myopic instance may place vessels in a suboptimal future position from which it is impossible to recover.

3.3.1 Rolling horizon heuristic with a coarse time granularity forecast

In this approach, we keep the central period unperturbed, but restrict the times at which vessels may make port-to-port voyages in the forecast period. In all of our computational testing, we use a time grain of 2, which means that vessels may only leave in even numbered time periods. Increasing the time grain above 2 in these instances always resulted in a performance degradation. This reduces the number of travel arcs, but not the number of demurrage arcs.

3.3.2 Rolling horizon heuristic with interspersed calls to K-opt

Since solving a 90-period instance to provable optimality can be challenging in its own right, this hybrid is meant to clean up flaws in the solution that have accrued in the process of running a rolling horizon heuristic. Consequently, in this hybrid, a K-opt algorithm is called immediately after $T/2$ periods have been fixed and again after all T periods have been fixed.

3.3.3 Rolling horizon heuristic with restricted port-vessel class pairs

This hybrid applies a rolling horizon heuristic over the entire planning horizon after restricting port-vessel class pairs (i.e., after solving MILP (6)). Thus, if a port-vessel class pair is deemed incompatible by MILP (6), this pair is excluded for the entire solution process thereafter. The hope is that solving each portion of the rolling horizon heuristic is simplified due to these port-vessel class restrictions and that this simplification will result in superior solutions given a time limit.

Algorithm 3 Rolling horizon heuristic with restricted port-vessel class pairs

1: Solve MILP (6) to obtain port-vessel class incompatibility pairs $\hat{z}_{j,vc}$
2: Set $x_a^{vc} = 0\ \forall vc \in \mathcal{VC}, a = ((j_1,t_1),(j_2,t_2)) \in \mathcal{A}^{vc}$ s.t. $\hat{z}_{j_1,vc} = 1$ or $\hat{z}_{j_2,vc} = 1$
3: Call Algorithm 1

3.3.4 K-opt with restricted port-vessel class pairs

This hybrid applies K-opt over the entire planning horizon after restricting port-vessel class pairs (by solving MILP (6)).

Algorithm 4 K-opt with restricted port-vessel class pairs

1: Solve MILP (6) to obtain port-vessel class incompatibility pairs $\hat{z}_{j,vc}$
2: Set $x_a^{vc} = 0 \ \forall vc \in \mathcal{VC}, a = ((j_1, t_1), (j_2, t_2)) \in \mathcal{A}^{vc}$ s.t. $\hat{z}_{j_1,vc} = 1$ or $\hat{z}_{j_2,vc} = 1$
3: Call Algorithm 2

4 Computational experiments

Our computational experiments are performed on the Group 2 instances of the publicly available maritime inventory routing problem library (MIRPLib), available at `mirplib.scl.gatech.edu` and presented in Papageorgiou et al. [36]. Since its creation in 2012, the MIRPLib website has received over 75,000 visits and has gained the attention of all of the major MILP solvers, including CPLEX, Express, Gurobi, Mosek, and SCIP. This is important as two of the major goals for the library were: (1) to present benchmark instances for a particular class of MIRPs; (2) to provide the MILP community with a set of optimization problem instances from the maritime transportation domain. Indeed, we show that the MILP solvers have gained some ground as they are now able to find feasible solutions to several instances faster than before. Nevertheless, for large instances, matheuristics are far superior. Moreover, the progress in MILP solvers makes it possible to solve larger neighborhoods within the local search framework.

Our convention for naming instances is based on the number of loading and discharging ports/regions, the number of vessel classes, and the number of vessels. This convention is easily understood with an example. Consider an instance named `LR1_DR04_VC03_V15a`. `LR1` means that there is one loading region/port. `DR04` means that there are four discharging regions/ports. `VC03` means that there are three vessel classes. `V15` means that there are a total of 15 vessels (with at least one vessel belonging to each vessel class). Finally, the letter 'a' indicates that the production and consumption rates at all ports are constant over all time periods; the letter 'b' indicates that these rates may vary over time.

Table 1 provides information and statistics for each of the 72 instances. There are 24 Group 2 base instances that have been formulated with three different planning horizons (120, 180, and 360 time periods) yielding a total of 72 instances. The columns are:

- Column 1 (Horizon): Number of periods in the planning horizon
- Column 2 (Instance name)
- Column 3 (Difficulty): Easy (E), Medium (M), or Hard (H). An instance is declared Hard (H) if no commercial MILP solver (using default settings) can close more than 10% of the gap (defined in (7)) in 1800 seconds; Easy (E) if at least one commercial MILP solver can close more than 90% of the gap in 1800 seconds using default CPLEX; and Medium (M) otherwise.
- Columns 3-5: Number of constraints (Rows), total number of decision variables (Cols), and number of integer decision variables (Int Cols) in the original model before presolve is called

Table 1: Statistics and best known results for Group 2 MIRPLib instances

			Original Model			Presolved Model			Statistics			
Horizon	Instance name	Difficulty	Rows	Cols	Int Cols	Rows	Cols	Int Cols	Objval	Bound	Rgap	Agap
120	LR1_DR02_VC01_V6a	E	1081	1855	1134	1023	1839	1118	33809	33809	0.00	0
120	LR1_DR02_VC02_V6a	E	1441	2979	2258	1389	2909	2188	74982	74982	0.00	0
120	LR1_DR02_VC03_V7a	E	1801	4082	3361	1697	3867	3146	40446	**39318**	2.79	1128
120	LR1_DR02_VC03_V8a	E	1801	4004	3283	1713	3854	3133	43721	43717	0.01	4
120	LR1_DR02_VC04_V8a	E	2161	5216	4495	2047	4959	4238	41657	41277	0.91	380
120	LR1_DR02_VC05_V8a	E	2521	6316	5595	2308	5791	5070	36659	**36088**	1.56	571
120	LR1_DR03_VC03_V10b	M	2401	5643	4682	2297	5493	4532	92941	**83645**	10.00	9296
120	LR1_DR03_VC03_V13b	E	2401	5665	4704	2299	5515	4554	124921	118706	4.98	6215
120	LR1_DR03_VC03_V16a	E	2401	5372	4411	2273	5169	4208	82837	**71635**	13.52	11202
120	LR1_DR04_VC03_V15a	E	3001	7152	5951	2860	6940	5739	**73312**	**72108**	1.64	1204
120	LR1_DR04_VC03_V15b	E	3001	7188	5987	2849	6966	5765	117812	**102148**	13.30	15664
120	LR1_DR04_VC05_V17a	E	4201	11110	9909	3908	10538	9337	**72876**	**71572**	1.79	1304
120	LR1_DR04_VC05_V17b	E	4201	11123	9922	3919	10604	9403	**105766**	**84970**	19.66	20796
120	LR1_DR05_VC05_V25a	M	5041	13412	11971	4665	12673	11232	105328	102755	2.44	2573
120	LR1_DR05_VC05_V25b	E	5041	13368	11927	4707	12772	11331	137107	**125966**	8.13	11141
120	LR1_DR08_VC05_V38a	M	7561	20621	18460	7015	19616	17455	166615	**158499**	4.87	8116
120	LR1_DR08_VC05_V40a	E	7561	20759	18598	7074	19919	17758	**178593**	**171418**	4.02	7175
120	LR1_DR08_VC05_V40b	M	7561	20724	18563	7055	19848	17687	200746	**188843**	5.93	11903
120	LR1_DR08_VC10_V40a	M	12961	39298	37137	11729	36678	34517	185538	**180353**	2.79	5185
120	LR1_DR08_VC10_V40b	M	12961	39229	37068	11868	37173	35012	206315	**195690**	5.15	10626
120	LR1_DR12_VC05_V70a	H	10921	30166	27045	10169	28847	25726	**278647**	**267766**	3.90	10881
120	LR1_DR12_VC05_V70b	M	10921	30138	27017	10226	28907	25786	308555	**294441**	4.57	14114
120	LR1_DR12_VC10_V70a	H	18721	57811	54690	17162	54871	51750	**283154**	**274121**	3.19	9033
120	LR1_DR12_VC10_V70b	H	18721	57773	54652	17196	54880	51759	295126	**285440**	3.28	9686
180	LR1_DR02_VC01_V6a	E	1621	2815	1734	1563	2799	1718	52167	**52166**	0.00	1
180	LR1_DR02_VC02_V6a	E	2161	4539	3458	2109	4469	3388	129372	**128106**	0.98	1267
180	LR1_DR02_VC03_V7a	M	2701	6242	5161	2597	6027	4946	**60547**	**58790**	2.90	1757
180	LR1_DR02_VC03_V8a	E	2701	6164	5083	2613	6014	4933	**68153**	**66989**	1.71	1164
180	LR1_DR02_VC04_V8a	M	3241	7976	6895	3127	7719	6638	**66017**	**65274**	1.12	743
180	LR1_DR02_VC05_V8a	M	3781	9676	8595	3568	9151	8070	**58619**	**57260**	2.32	1359
180	LR1_DR03_VC03_V10b	M	3601	8643	7202	3497	8493	7052	**125638**	**106712**	15.06	18926
180	LR1_DR03_VC03_V13b	E	3601	8665	7224	3499	8515	7074	165764	**143695**	13.31	22069
180	LR1_DR03_VC03_V16a	M	3601	8372	6931	3473	8169	6728	**143178**	**120556**	15.80	22622
180	LR1_DR04_VC03_V15a	M	4501	10992	9191	4360	10780	8979	**118621**	**116211**	2.03	2410
180	LR1_DR04_VC03_V15b	E	4501	11028	9227	4349	10806	9005	189989	**170432**	10.29	19557
180	LR1_DR04_VC05_V17a	M	6301	17110	15309	6008	16538	14737	**117710**	**115530**	1.85	2180
180	LR1_DR04_VC05_V17b	M	6301	17123	15322	6019	16604	14803	**159168**	**130290**	18.14	28879
180	LR1_DR05_VC05_V25a	M	7561	20732	18571	7185	19993	17832	171620	**167766**	2.25	3854
180	LR1_DR05_VC05_V25b	M	7561	20688	18527	7227	20092	17931	205368	**186749**	9.07	18619
180	LR1_DR08_VC05_V38a	H	11341	31901	28660	10795	30896	27655	274244	261036	4.82	13208
180	LR1_DR08_VC05_V40a	H	11341	32039	28798	10854	31199	27958	296760	**284481**	4.14	12280
180	LR1_DR08_VC05_V40b	M	11341	32004	28763	10835	31128	27887	337559	**294265**	12.83	43294
180	LR1_DR08_VC10_V40a	H	19441	60778	57537	18209	58158	54917	304261	**297805**	2.12	6457
180	LR1_DR08_VC10_V40b	H	19441	60709	57468	18348	58653	55412	331775	**304668**	8.17	27107
180	LR1_DR12_VC05_V70a	H	16381	46726	42045	15629	45407	40726	460566	**444823**	3.42	15743
180	LR1_DR12_VC05_V70b	H	16381	46698	42017	15686	45467	40786	491160	**461302**	6.08	29858
180	LR1_DR12_VC10_V70a	H	28081	89371	84690	26522	86431	81750	466975	**454280**	2.72	12695
180	LR1_DR12_VC10_V70b	H	28081	89333	84652	26556	86440	81759	470172	**454552**	3.32	15620
360	LR1_DR02_VC01_V6a	E	3241	5695	3534	3183	5679	3518	108141	**107994**	0.14	147
360	LR1_DR02_VC02_V6a	E	4321	9219	7058	4269	9149	6988	283031	**275347**	2.71	7684
360	LR1_DR02_VC03_V7a	H	5401	12722	10561	5297	12507	10346	**124282**	**120346**	3.17	3936
360	LR1_DR02_VC03_V8a	M	5401	12644	10483	5313	12494	10333	**141166**	**139821**	0.95	1345
360	LR1_DR02_VC04_V8a	M	6481	16256	14095	6367	15999	13838	**138693**	**137449**	0.90	1244
360	LR1_DR02_VC05_V8a	H	7561	19756	17595	7348	19231	17070	**122598**	**120986**	1.31	1612
360	LR1_DR03_VC03_V10b	H	7201	17643	14762	7097	17493	14612	**259888**	**212621**	18.19	47267
360	LR1_DR03_VC03_V13b	M	7201	17665	14784	7099	17515	14634	**316441**	**262458**	17.06	53983
360	LR1_DR03_VC03_V16a	M	7201	17372	14491	7073	17169	14288	**327793**	**274753**	16.18	53040
360	LR1_DR04_VC03_V15a	M	9001	22512	18911	8860	22300	18699	**252710**	**248598**	1.63	4112
360	LR1_DR04_VC03_V15b	M	9001	22548	18947	8849	22326	18725	**339308**	**294483**	13.21	44825
360	LR1_DR04_VC05_V17a	H	12601	35110	31509	12308	34538	30937	**251623**	**247816**	1.51	3807
360	LR1_DR04_VC05_V17b	M	12601	35123	31522	12319	34604	31003	**304507**	**251864**	17.29	52643
360	LR1_DR05_VC05_V25a	H	15121	42692	38371	14745	41953	37632	368628	362894	1.56	5734
360	LR1_DR05_VC05_V25b	H	15121	42648	38327	14787	42052	37731	**410053**	**377388**	7.97	32665
360	LR1_DR08_VC05_V38a	H	22681	65741	59260	22135	64736	58255	596969	575369	3.62	21600
360	LR1_DR08_VC05_V40a	H	22681	65879	59398	22194	65039	58558	652380	**624179**	4.32	28201
360	LR1_DR08_VC05_V40b	H	22681	65844	59363	22175	64968	58487	709713	**630407**	11.17	79306
360	LR1_DR08_VC10_V40a	H	38881	125218	118737	37649	122598	116117	663245	649885	2.01	13360
360	LR1_DR08_VC10_V40b	H	38881	125149	118668	37788	123093	116612	724513	651195	10.12	73318
360	LR1_DR12_VC05_V70a	H	32761	96406	87045	32009	95087	85726	1021389	982619	3.80	38770
360	LR1_DR12_VC05_V70b	H	32761	96378	87017	32066	95147	85786	1093013	980410	10.30	112603
360	LR1_DR12_VC10_V70a	H	56161	184051	174690	54602	181111	171750	1024399	1002624	2.13	21775
360	LR1_DR12_VC10_V70b	H	56161	184013	174652	54636	181120	171759	1001541	985670	1.58	15871

- Columns 6-8: Same as columns 3-5 except for the presolved model, i.e., the model that is generated after invoking a MILP solver's "presolve" method, a preprocessing routine that reduces the number of variables and constraints, tightens coefficients and variable bounds, and more. The size of the presolve model is almost always more indicative of an instance's difficulty than that of the original model.
- Column 9 (Objval): Best known objective function value ever found (bold if improvement over best known)
- Column 10 (Bound): Best lower bound ever found (bold if improvement over best known)
- Column 11 (Rgap): Best known relative optimality gap $\frac{(z^{\text{best}} - z^{\text{bound}})}{z^{\text{best}}}$
- Column 12 (Agap): Best known absolute optimality gap $(z^{\text{best}} - z^{\text{bound}})$

The main metric that we use to compare solution methods is the average fraction of the relative gap closed over time, where the average is taken over all Easy, Medium, or Hard instances. Here the relative gap is defined as

$$\min\left\{\frac{(z^{\text{method}} - z^{\text{best}})}{z^{\text{best}}}, 1\right\} \tag{7}$$

where z^{method} is the objective function value of the method under evaluation and z^{best} is the objective function value of the best known solution ever found. It should be noted that the previous incumbent solutions as reported in [36] were found by warm-starting a local search procedure with the best solution found using the ADP methods described in [34], applying local search for five hours, and then warm-starting Gurobi with this solution and solving for 10 days.

All matheuristics were coded in AIMMS version 4.13.1.204. Because we set a time limit to search over each neighborhood in our local search implementation, our local search is no longer deterministic due to idiosyncrasies of MILP solvers.

The time limit for solving Model (1) in Step 7 of Algorithm 1 for the rolling horizon heuristic is 120 seconds, regardless of the number τ^c of central periods.

4.1 Benchmarking experiments with commercial solvers

For benchmarking purposes, we first compare the performance of the general purpose MILP solvers CPLEX 12.6.2 and Gurobi 6.5. This comparison is relevant because both solvers have been in possession of these instances since 2012 and have made improvements over the previous versions which were used in Papageorgiou et al. [36]. Two variants were run for each solver. The first directly solves Model (3), while the second solves Model (3) with MIR cuts (5) appended to the model for all $t \in \mathcal{T}$ for which the right hand side (5) is positive. It must be noted that CPLEX has an advantage when MIR cuts are included as these cuts are added as "User cuts" to a user cut pool, whereas they had to be included in the initial formulation for Gurobi. This distinction is important because when cuts are included in a cut pool, a solver only includes the cuts on an as-needed basis (i.e., when they are violated). Because we could not pass these cuts to Gurobi in a cut pool through the AIMMS-Gurobi interface, Gurobi had to include all of the cuts as structural constraints from the outset resulting in a larger constraint matrix that must be preprocessed and factorized throughout the branch-and-cut algorithm. These benchmarking results with CPLEX and Gurobi were run on a Sandy Bridge dual-socket Linux machine with kernel 2.6.32-279.el6.x86_64 equipped with eight 2.6 GHz cores and

Table 2: Algorithms Compared

Algorithm	Description	Section
`cpx`	CPLEX 12.6.2 default	–
`cpx_mir`	CPLEX 12.6.2 with MIR cuts (5)	–
`cpx_lb`	CPLEX 12.6.2 with local branching	–
`grb`	Gurobi 6.5 default	–
`grb_mir`	Gurobi 6.5 with MIR cuts (5)	–
`rhh_`τ^c`_`τ^f	Rolling horizon with τ^c central periods and τ^f forecast periods	3.1.2
`1opt`	Fix all but one vessel class per iteration: $\mathcal{S} = \mathcal{VC}$ in Algorithm 2	3.2.1
`2opt`	Fix all but two vessel classes per iteration: $\mathcal{S} =$ all $\binom{\lvert\mathcal{VC}\rvert}{2}$ pairs in Algorithm 2	3.2.1
`2opt_cyclic`	`2opt` with many fewer pairs: $\mathcal{S} = \{(vc_1, vc_2), (vc_2, vc_3), \ldots, (vc_{\lvert\mathcal{VC}\rvert}, vc_1)\}$	3.2.1
`1opt_1daytw`	`1opt` with 1-day time window flexibility search	3.2.2
`pool_polish`	Solution pool polishing	3.2.3
`rhh_cg_`τ^c`_`τ^f	`rhh_`τ^c`_`τ^f with coarse grain forecast	3.3.1
`rhh_`τ^c`_`τ^f`_kopt`	hybrid of `rhh_`τ^c`_`τ^f and K-opt	3.3.2
`rhh_`τ^c`_`τ^f`_restrict`	hybrid of `rhh_`τ^c`_`τ^f and port-vessel class restrictions	3.3.3
`kopt_restrict`	hybrid of `kopt` and port-vessel class restrictions	3.3.4

4GB of RAM per core (32GB of RAM). This machine is more powerful than the machine used to test our matheuristics.

Of the 72 total instances (24 base instances each formulated with a time horizon of 120, 180, and 360 periods), there are 26 Hard instances, 25 Medium instances, and 21 Easy instances. Figure 1 further breaks down the number of instances in each difficulty category based on the number of discharging ports and number of time periods in the planning horizon.

Figure 2 shows profile curves of the average fraction of the gap closed over time broken down by Easy, Medium, and Hard instances. We see that default CPLEX on average outperforms default Gurobi in each difficulty category. As mentioned above, the comparison between both solvers with MIR cuts is not a fair one since it is not an "apples-to-apples" comparison; nevertheless, we report it for completeness. CPLEX with MIR cuts appears to perform better than default CPLEX for Medium and Hard instances. It is important to note that our gap metric (7) is different from the traditional gap metric used by solvers. If the latter were used, the benefits of including MIR cuts would be more pronounced as the dual bound improvements are greater.

In all subsequent experiments evaluating matheuristics, CPLEX 12.6.2 was used.

4.2 Experiments with construction heuristics

Figure 3 presents performance profiles for several matheuristics and categorizes their performance based on their ability to solve Easy, Medium, and Hard instances. To improve readability, the RHH variants are shown in the top row, K-opt variant in the middle row, and the "best" RHH, K-opt, and commercial solver algorithms are compared in the bottom row. The word "best" is in quotations to emphasize that it is based on the performance up to 1800 seconds. If a shorter time limit were used, the "best" algorithm may change.

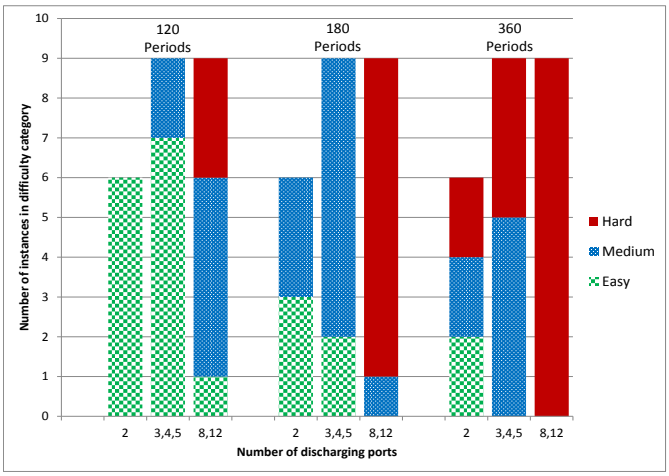

Figure 1: Difficulty categorization of instances evaluated

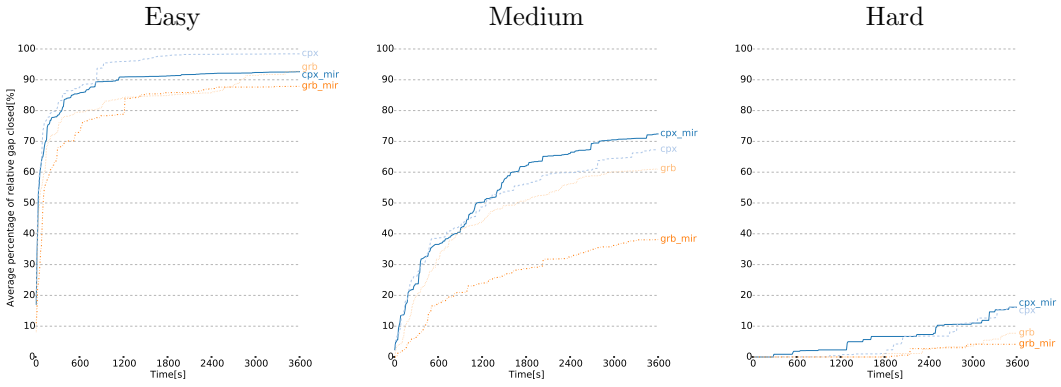

Figure 2: Performance profiles for commercial MILP solvers

Rolling horizon comparison. The top row of Figure 3 compares the performance of the various RHH variants. For all three difficulty levels, RHH variants outperform the best commercial solver within an 1800 second time limit. On Hard instances, `rhh_60_30` performs the best up to 1200 seconds, after which the two `rhh_30_60` variants begin to take over. Surprisingly, using a coarse grain forecast with `rhh_60_30` on hard instances is more of a bane than a boon. Using a 90-period forecast was never beneficial on Medium and Hard instances.

K-opt comparison. The middle row of Figure 3 compares the performance of the various K-opt approaches. Although this class of matheuristics was introduced as improvement heuristics in Section 3.2.1, they can just as well be applied in an iterative manner to generate high quality feasible solutions. In all of our experiments using a K-opt variant as a construction heuristic, we first solved an entire instance for 30 seconds before starting Algorithm 2. The results are somewhat mixed. It is not clear if using port-vessel class restrictions provides improvement. For Easy instances, restrictions seem to help, but they are not necessary since CPLEX's performance is satisfactory. For Hard instances, they seem to help for 1-opt as `1opt_restrict` outperforms `1opt`, but the reverse is true when comparing `2opt_restrict` and `2opt`

Best of the best comparison. The bottom row of Figure 3 reveals, not surprisingly, that the best RHH is superior to the best K-opt variant when used as a construction heuristic. Not shown in the bottom row Figure 3 is that on Hard instances, in 900 seconds, `rhh_60_30` performs better than `2opt`, which in turn does better than `rhh_30_60`. This is because `rhh_60_30` is taking larger steps when rolling forward compared to `rhh_30_60`. It should also be reiterated that "order matters" and that perhaps with a more intelligent ordering of vessel class pairs, `2opt` could outperform `rhh_60_30`.

Finally, it is worth calling attention to the fact that on Hard instances, the worst performing RHH and K-opt variant are far superior to all commercial solvers. We will revisit Hard instances in Section 4.4.

4.3 Experiments with improvement heuristics

In this section, we attempt to evaluate the performance of the improvement heuristics proposed in Section 3.2. To do so, during the course of our extensive computations, many feasible solutions were generated for each instance. We collected 917 feasible solutions and partitioned them into two groups, which we refer to as buckets: those whose objective function value is less than two times the best known objective function value for that instance (Bucket 1); and those whose objective function is between two and three times the best known objective function (Bucket 2). In other words, Bucket 1 contains "good" solutions and Bucket 2 contains "bad" solutions. For each of the 917 feasible solutions, several improvement heuristics were tested. The percentage improvement presented in this section is given by the ratio $(B - A)/B$, where B is the objective function value of the solution before the algorithm is applied, and A is the objective function value of the solution after the algorithm is applied. All improvement heuristics were tested with a time limit of 5 minutes.

Table 3 presents the performance of the different improvement heuristics for "good" solutions (Bucket 1) and "bad" solutions (Bucket 2). The table shows the average improvement of the different heuristics over all instances. Note that an instance can have multiple solutions. Therefore, the percentage improvement of each instance in each bucket is calculated as the average improvement of all the solutions for that particular instance and bucket. The table presents the comparison of 5 methods: `1opt`, `1opt_1daytw`, `2opt_cyclic`, CPLEX with local branching, and default CPLEX. The first three methods are described in Section 3.2. The last two methods involve solving the problems with CPLEX using local branching (`cpx_lb`) and default

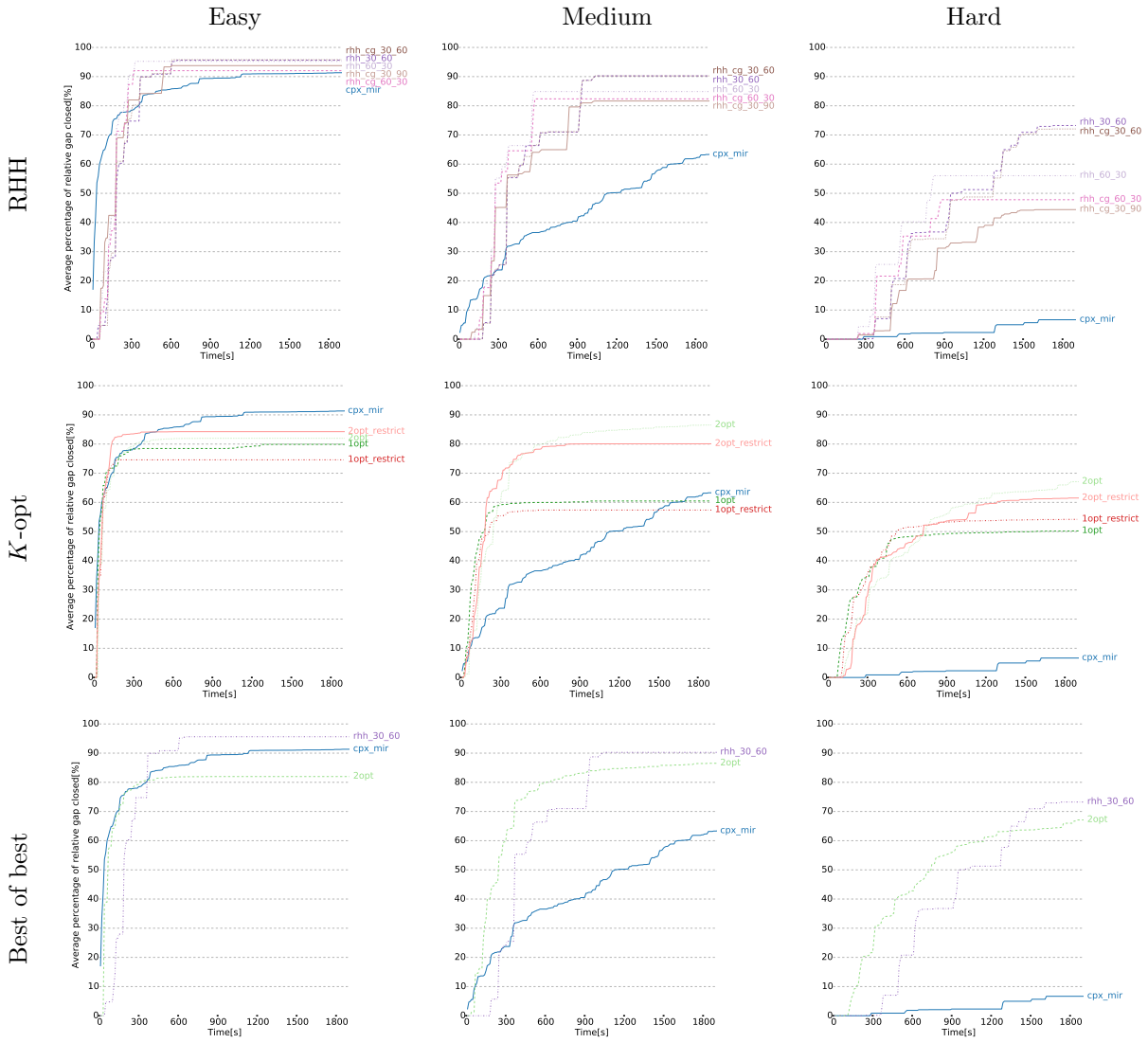

Figure 3: Performance profiles for construction heuristics

CPLEX (`cpx`), both using the "incumbent solution" as an initial integer solution.

The reason for evaluating these three K-opt heuristics is because they all require the same number of iterations (the cardinality of the set $\mathcal{S}$ in Algorithm 2 is the same, $|\mathcal{S}| = |\mathcal{VC}|$) with `1opt` searching the smallest neighborhood in each iteration, `1opt_1daytw` the next largest neighborhood, and `2opt_cyclic` the largest neighborhood of the three. The two CPLEX variants search the largest neighborhood as they solve the original problem using an initial solution to warmstart the solution process. Hence, this experiment attempts to answer the question: Empirically, is it better to search larger neighborhoods, which may not solve to provable optimality, or many smaller neighborhoods?

Table 3: Average percentage improvement of improvement heuristics

	Bucket 1			Bucket 2		
Method	Easy	Medium	Hard	Easy	Medium	Hard
`2opt_cyclic`	**10.9**	**7.9**	7.2	**50.5**	**47.7**	**34.4**
`1opt`	8.1	6.5	**8.6**	35.6	32.5	33.5
`1opt_1daytw`	9.8	5.3	2.8	42.9	35.5	15.5
`cpx_lb`	10.6	5.2	2.3	45.8	33.5	12.8
`cpx`	9.8	1.8	0	42.8	26.4	1.8

The first thing to note is that, as expected, all improvement heuristics are more successful at improving solutions in Bucket 2 than in Bucket 1. Moreover, all algorithms are better at improving easy instances, regardless of the bucket. Of all of the tested improvement heuristics, `2opt_cyclic` on average performs the best. The only exception is in the Hard instances from Bucket 1, in which `1opt` performs the best on average. It is interesting to note that CPLEX with the local branching option performs better than CPLEX with default settings. However, these two do not perform as well as other improvement heuristics, particularly in Medium and Hard instances.

Table 4 presents the number of instances for which a better solution was found after applying the different improvement heuristics. For example, if an instance has 9 solutions in the pool then the improvement is calculated using the best found solution (after applying the heuristic method to all 9 instances), compared to the best solution from the pool. The only exception is `pool_polish`, in which the best found solution is obtained by solving the solution pool polishing method presented in 3.2 (i.e. this problem is solved only once using all of the solutions, while the other methods are solved one time for each solution).

Table 4 shows that `2opt_cyclic` and `1opt` can improve the best solution in 43 out of the 72 instances.Out of these two, `2opt_cyclic` is better at improving Easy and Medium instances, while `1opt` is better at improving Hard instances. `1opt_1daytw` and `pool_polish` also perform well by improving the best solution in 35 and 26 of the instances, respectively. However, CPLEX has difficulties improving the best solution found. In particular, default CPLEX can only find a better solution in 9 of the instances, while CPLEX with local branching in none.

Table 5 presents the average percentage improvement over the best solution from the pool. The quality improvement of the instance in each basket is similar (in terms of behaviour) to the number of instances for which a better solution was found. In particular, `2opt_cyclic` performs the best in Easy and Medium instances and `1opt` performs the best in Hard instances. Default CPLEX makes little improvement in Easy and Medium instances, and no improvement in Hard instances.

Table 4: Number of instances for which the method found better solutions

	Easy	Medium	Hard	Total
2opt_cyclic	9	15	19	43
1opt	7	14	22	43
1opt_1daytw	7	13	15	35
cpx_lb	0	0	0	0
cpx	5	4	0	9
pool_polish	5	10	11	26

Table 5: Improvement of best solution using improvement heuristics

	Easy	Medium	Hard
2opt_cyclic	1.3	1.8	3
1opt	0.5	1.5	5.4
1opt_1daytw	1.2	0.8	1.2
cpx_lb	0	0	0
cpx	0.8	0.1	0
pool_polish	0.5	0.4	0.3

4.4 Experiments with hybrid heuristics for Hard instances only

In this section, we investigate hybrid matheuristics for finding good feasible solutions to the 26 Hard instances in this data set, almost all of which have eight or more discharging regions and five or more vessel classes. These instances are particularly challenging and require extra attention. Likewise, they are more representative of real-world instances.

Three of the hybrid heuristics outlined in Section 3.3 combine an RHH with a K-opt procedure or port-vessel class restrictions. Figure 4 shows the performance of these hybrids. An asterisk means that the CPLEX parameters "Emphasize feasibility over optimality" and "local branching" were engaged. The first observation to make is that all hybrids are able to close between 86% and 91% of the optimality gap in 3600 seconds. All three RHH methods with $\tau^c = 60$, i.e., 60 periods in the central period, close at least 82% of the optimality gap by 1800 seconds, whereas the best RHH method tested in Section 4.2 was able to close roughly 73% of the optimality gap in the same time limit. This results gives an affirmative answer to the question: Is it true that RHH-Kopt hybrids are successful at cleaning up flaws in the solution that have accrued in the process of running a rolling horizon heuristic?

Although the heuristic `rhh_60_30_1opt_restrict*`, which includes port-vessel class restrictions ends up at the bottom of the list after 3600 seconds, it is actually the best performing algorithm up to 1800 seconds. This suggests that restrictions can be useful for accelerating time to good feasible solutions.

4.5 Recommendations for obtaining good solutions quickly

We close this section by offering a set of recommendations for identifying good solutions reasonably quickly as a function of instance difficulty, the number of time periods, and user-imposed time limits. In general, solver performance is problem dependent, so our recommendations may not hold for other classes of vehicle routing problems.

First and foremost, when relying on a generic MILP solver within a matheuristic, it is important to determine the limits of the underlying solver. The Difficulty column of Table 1 reveals that instances with up to three vessel classes and up to 180 time periods are classified as Easy or Medium difficulty. With this

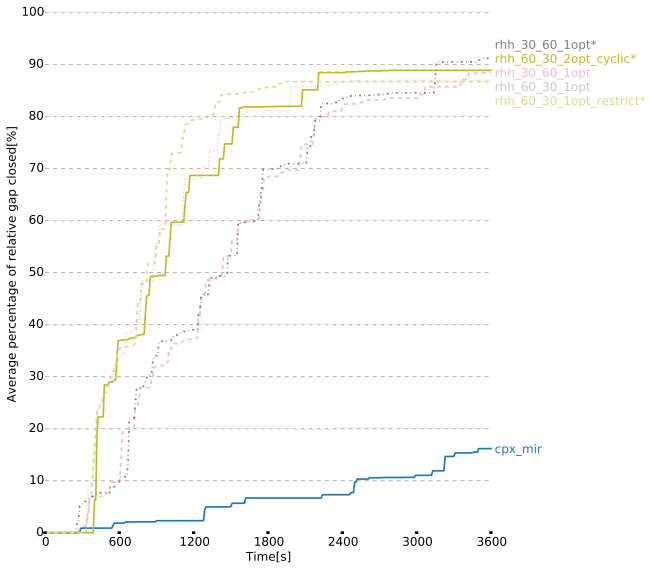

Figure 4: Performance profiles for hybrid heuristics

empirical data in hand, we felt confident that a MILP solver could make meaningful progress using a 1- or 2-opt local search (i.e., allowing no more than two vessel classes to be free at a time) or solving an RHH of up to 90 periods.

As far as improvement heuristics are concerned, Section 4.3 reveals that K-opt heuristics are superior to CPLEX with or without local branching for instances with 180 or more time periods. This suggests that these problem-specific heuristics should be used when trying to improve larger instances in one fell swoop. Alternatively, one can break down the problem into smaller chunks and apply a generic MILP solver directly. For example, one could modify the `rhh`$\_\tau^c\_\tau^f\_$`kopt` algorithm so that K-opt is applied only to the last $R\tau^c$ periods when it is invoked, where R is a positive scalar, instead of to the entire "past."

For Hard instances, and thus most industrial-scale problem instances, arguably the simplest, yet most effective algorithm is coupling a rolling horizon heuristic with a K-opt procedure (`rhh`$\_\tau^c\_\tau^f\_$`kopt`) as outlined in Section 3.3.2. If a relatively long time limit is set, then an RHH with a short central period τ^c (30 in our experiments) allows one to gradually build a solution and make numerous corrections over time. Interspersing calls to a K-opt procedure will further refine suboptimal decisions made in the "past." If a short time limit is imposed, then an RHH with a longer central period τ^c (60 in our experiments) is a wise choice because it rolls forward with larger steps and thus will lead to a complete solution (i.e., all time periods in the planning horizon will be considered) more quickly. If a very short time limit is imposed, a more specialized algorithm may be necessary (e.g., the ADP approach described in [34]).

5 Conclusions and future research directions

The matheuristics evaluated in this paper require a single mathematical program, are simple to implement, and outperform commercial MILP solvers on instances of practical size. Commercial MILP solvers have improved and continue to improve, but heuristics are still needed to solve large MIRP instances for real-

world applications for the foreseeable future. The matheuristics presented in this work rely heavily on the performance of MILP solvers to solve smaller, restricted MILPs. Consequently, an improvement in solver performance typically translates into a direct benefit to the matheuristics proposed here. It should also be noted that solvers continue to make progress in parallel implementations and they allow users to search for optimal parameter settings to improve performance on a class of problems. Eventually, these settings may suffice for large industrial applications. Meanwhile, there is on-going research aimed at developing general-purpose parallel large neighborhood search frameworks for finding high quality primal solutions for MILPs (see, e.g., Munguía et al. [32]).

It would be interesting to test matheurisics for continuous-time formulations, which are quite common in ship routing and scheduling problems. These problems typically involve time windows in lieu of inventory balance constraints. Our initial attempts at solving a continuous-time formulation for this problem failed miserably and thus we abandoned the effort. The most likely reason for this poor performance is due to the fact that this class of problems has few operational constraints and no time windows, two ingredients that typically help to reduce the size of a continuous-time formulation. It would also be interesting to have a detailed comparison of matheuristics for operational MIRPs, which typically have more scheduling related constraints.

Acknowledgments

We wish to thank two anonymous referees for their feedback, in particular Reviewer 1 whose perceptive comments helped improve the quality of the paper.

6 Nomenclature

Sets are represented using capital letters in a calligraphic font, such as $\mathcal{T}$ and $\mathcal{V}$. Where possible, parameters are denoted with capital letters in italic font or with Greek characters; however, some deviations are made to express constraints more easily, e.g., inventory balance constraints. Decision variables are always lower case.

Indices and sets

$t \in \mathcal{T}$	set of time periods with $T = \lvert\mathcal{T}\rvert$
$v \in \mathcal{V}$	set of vessels
$vc \in \mathcal{VC}$	set of vessel classes
$j \in \mathcal{J}^P$	set of production, a.k.a. loading, ports
$j \in \mathcal{J}^C$	set of consumption, a.k.a. discharging, ports
$j \in \mathcal{J}$	set of all ports: $\mathcal{J} = \mathcal{J}^P \cup \mathcal{J}^D$
$n \in \mathcal{N}$	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
$a \in \mathcal{A}^v$	set of arcs associated with vessel $v \in \mathcal{V}$
$a \in \mathcal{A}^{vc}$	set of arcs associated with vessel class $vc \in \mathcal{VC}$
$a \in \mathcal{FS}_n^{vc}$	forward star associated with node $n = (j,t) \in \mathcal{N}_{s,t}$ and vessel class $vc \in \mathcal{VC}$
$a \in \mathcal{RS}_n^{vc}$	reverse star associated with node $n = (j,t) \in \mathcal{N}_{s,t}$ and vessel class $vc \in \mathcal{VC}$

Data

B_j	number of berths (berth limit) at port $j \in \mathcal{J}$
C_a^{vc}	cost for vessel class vc to traverse arc $a = ((j_1,t_1),(j_2,t_2)) \in \mathcal{A}^{vc}$
$D_{j,t}$	number of units produced/consumed at port $j \in \mathcal{J}$ in period $t \in \mathcal{T}$
Δ_j	an indicator parameter taking value +1 if $j \in \mathcal{J}^P$ and -1 if $j \in \mathcal{J}^C$ at port j from a single vessel in a period
P_j	penalty for unmet demand/excess inventory at port $j \in \mathcal{J}$ 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)
$S_{j,t}^{\min}$ ($S_{j,t}^{\max}$)	lower bound (capacity) at port $j \in \mathcal{J}$ in time period $t \in \mathcal{T}$
$s_{j,0}$	initial inventory at port $j \in \mathcal{J}$

Decision Variables

$\alpha_{j,t}$	(continuous) number of units of unmet demand/excess inventory at port $j \in \mathcal{J}$ in time period $t \in \mathcal{T}$
$s_{j,t}$	(continuous) number of units of inventory at port $j \in \mathcal{J}$ available at the *end* of period t
x_a^{vc}	(integer) number of vessels in vessel class $vc \in \mathcal{VC}$ using arc $a \in \mathcal{A}^{vc}$

7 Appendix

Lemma 1 *Consider the 2-variable mixed-integer linear sets*

$$\mathcal{S}_{\leq} = \{(x,y) \in \mathbb{Z} \times \mathbb{R}_+ : x - y \leq b\} \ .$$

and

$$\mathcal{S}_{\geq} = \{(x,y) \in \mathbb{Z} \times \mathbb{R}_+ : x + y \geq b\} \ .$$

Then, the inequality

$$x - \frac{1}{1-f_0} y \leq \lfloor b \rfloor \tag{8}$$

is valid for $\text{conv}(\mathcal{S}_{\leq})$ *where* $f_0 := b - \lfloor b \rfloor$. *The inequality*

$$x + \frac{1}{1-f_0} y \geq \lceil b \rceil \tag{9}$$

is valid for $\text{conv}(\mathcal{S}_{\geq})$ *where* $f_0 := \lceil b \rceil - b$.

Proof See [16]. □

Consider a mixed-integer linear set defined by a single constraint:

$$\mathcal{S} = \left\{ (\mathbf{x}, \mathbf{y}) \in \mathbb{Z}^n_+ \times \mathbb{R}^p_+ : \mathbf{a}^T \mathbf{x} + \mathbf{g}^T \mathbf{y} \leq \mathbf{b} \right\} .$$

Proposition 1 *The* Mixed-Integer Rounding (MIR) *inequality*

$$\sum_{j=1}^{n} \left(\lfloor a_j \rfloor + \frac{(f_j - f_0)^+}{1-f_0} \right) x_j + \frac{1}{1-f_0} \sum_{j: g_j < 0} g_j y_j \leq \lfloor b \rfloor \tag{10}$$

is valid for $\text{conv}(\mathcal{S})$ *where* $f_0 = b - \lfloor b \rfloor$ *and* $f_j = a_j - \lfloor a_j \rfloor$.[1] *When the single constraint in* $\mathcal{S}$ *is written with* $a \geq$ *sign, the MIR inequality becomes*

$$\sum_{j=1}^{n} \left(\lceil a_j \rceil - \frac{(f_j - f_0)^+}{1-f_0} \right) x_j + \frac{1}{1-f_0} \sum_{j: g_j > 0} g_j y_j \geq \lceil b \rceil \tag{11}$$

where $f_0 = \lceil b \rceil - b$ *and* $f_j = \lceil a_j \rceil - a_j$.

Proof Since the proof of the first case with $\mathcal{S}$ expressed with a $\leq$ sign is given in [16], we prove the second case. Note that $a_j = \lceil a_j \rceil - f_j = \lfloor a_j \rfloor + (1 - f_j)$. Relax the constraint $\sum_{j=1}^{n} a_j x_j + \sum_{j=1}^{p} g_j y_j \geq b$ to

$$\sum_{j: f_j \leq f_0} \lceil a_j \rceil x_j + \sum_{j: f_j > f_0} a_j x_j + \sum_{j: g_j > 0} g_j y_j \geq b \ . \tag{12}$$

Here, we have thrown out the continuous variables with non-positive coefficients and we have partitioned the integer variables based on the value of their fractional part f_j being greater than or less than the fractional part f_0 of the right-hand side b. Next, rewrite the left hand side of (12) as $w + z \geq b$ where w and z are defined as

$$\begin{aligned} w &:= \sum_{j: f_j \leq f_0} \lceil a_j \rceil x_j + \sum_{j: f_j > f_0} \lfloor a_j \rfloor x_j \ , \text{ and} \\ z &:= \sum_{j: g_j > 0} g_j y_j + \sum_{j: f_j > f_0} (1 - f_j) x_j \ . \end{aligned}$$

Since $w \in \mathbb{Z}$, $z \in \mathbb{R}_+$, and $w + z \geq b$, we can apply inequality (9) from Lemma 1. This gives $w + \frac{1}{1-f_0} z \geq \lceil b \rceil$ or

$$\sum_{j: f_j \leq f_0} \lceil a_j \rceil x_j + \sum_{j: f_j > f_0} \left(\lfloor a_j \rfloor + \frac{1-f_j}{1-f_0} \right) x_j + \frac{1}{1-f_0} \sum_{j: g_j > 0} g_j y_j \geq \lceil b \rceil \ .$$

Note that $\lfloor a_j \rfloor + \frac{1-f_j}{1-f_0} = \lceil a_j \rceil - \frac{f_j - f_0}{1-f_0}$ for all $j \in \{1, \ldots, n\} : f_j > 0$. Note also that for those integer variables j whose fractional part f_j is greater than the fractional part f_0 of the right-hand side, the relaxed coefficient $\lceil a_j \rceil - \frac{(f_j - f_0)^+}{1-f_0}$ in the MIR cut (11) is stronger than for those j with $f_j \leq f_0$. □

[1] Note that the term $\frac{(f_j - f_0)^+}{1-f_0}$ can only increase the coefficient of x_j, which makes the constraint stronger.

References

[1] Agra, A., Andersson, H., Christiansen, M., Wolsey, L.: A maritime inventory routing problem: Discrete time formulations and valid inequalities. Networks 62(4), 297–314 (2013)

[2] Agra, A., Christiansen, M., Delgado, A., Simonetti, L.: Hybrid heuristics for a short sea inventory routing problem. European Journal of Operational Research 236(3), 924 – 935 (2014)

[3] Al-Ameri, T.A., Shah, N., Papageorgiou, L.G.: Optimization of vendor-managed inventory systems in a rolling horizon framework. Computers & Industrial Engineering 54(4), 1019–1047 (May 2008), http://www.sciencedirect.com/science/article/pii/S0360835207002744

[4] Al-Khayyal, F., Hwang, S.: Inventory constrained maritime routing and scheduling for multi-commodity liquid bulk, part i: Applications and model. European Journal of Operational Research 176(1), 106–130 (2007), http://www.sciencedirect.com/science/article/pii/S0377221705005771

[5] Andersson, H., Hoff, A., Christiansen, M., Hasle, G., Løkketangen, A.: Industrial aspects and literature survey: Combined inventory management and routing. Computers & Operations Research 37(9), 1515–1536 (2010), http://www.sciencedirect.com/science/article/pii/S0305054809002962

[6] Archetti, C., Speranza, M.G.: A survey on matheuristics for routing problems. EURO Journal on Computational Optimization 2(4), 223–246 (2014), http://dx.doi.org/10.1007/s13675-014-0030-7

[7] Asokan, B.V., Furman, K.C., Goel, V., Shao, Y., Li, G.: Parallel large-neighborhood search techniques for LNG inventory routing. Submitted for publication (2014)

[8] Bertazzi, L., Speranza, M.G.: Matheuristics for inventory routing problems. Hybrid algorithms for service, computing and manufacturing systems: routing and scheduling solutions. IGI Global, Hershey 488 (2011)

[9] Bixby, R., Rothberg, E.: Progress in computational mixed integer programminga look back from the other side of the tipping point. Annals of Operations Research 149(1), 37–41 (2007)

[10] Boschetti, M.A., Maniezzo, V., Roffilli, M., Röhler, A.B.: Matheuristics: optimization, simulation and control. In: Hybrid metaheuristics, pp. 171–177. Springer (2009)

[11] Brouer, B.D., Desaulniers, G., Pisinger, D.: A matheuristic for the liner shipping network design problem. Transportation Research Part E: Logistics and Transportation Review 72, 42–59 (2014)

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

[13] Christiansen, M., Fagerholt, K., Flatberg, T., Haugen, O., Kloster, O., Lund, E.H.: Maritime inventory routing with multiple products: A case study from the cement industry. European Journal of Operational Research 208(1), 86–94 (2011), http://www.sciencedirect.com/science/article/pii/S0377221710005606

[14] Christiansen, M., Fagerholt, K., Nygreen, B., Ronen, D.: Maritime transportation. In: Barnhart, C., Laporte, G. (eds.) Transportation, Handbooks in Operations Research and Management Science, vol. 14, pp. 189–284. Elsevier (2007), http://www.sciencedirect.com/science/article/pii/S0927050706140049

[15] Coelho, L.C., Cordeau, J.F., Laporte, G.: Thirty years of inventory-routing. Transportation Science 48(1), 1–19 (2014), http://pubsonline.informs.org/doi/abs/10.1287/trsc.2013.0472

[16] Cornuéjols, G.: Valid inequalities for mixed integer linear programs. Mathematical Programming 112(1), 3–44 (2007), http://dx.doi.org/10.1007/s10107-006-0086-0

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

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

[19] Fischetti, M., Lodi, A.: Local branching. Mathematical programming 98(1-3), 23–47 (2003)

[20] Fodstad, M., Uggen, K.T., Rømo, F., Lium, A., Stremersch, G.: LNGScheduler: a rich model for coordinating vessel routing, inventories and trade in the liquefied natural gas supply chain. Journal of Energy Markets 3(4), 31–64 (2010)

[21] Furman, K.C., Song, J.H., Kocis, G.R., McDonald, M.K., Warrick, P.H.: Feedstock routing in the ExxonMobil downstream sector. Interfaces 41(2), 149–163 (2011), http://interfaces.journal.informs.org.www.library.gatech.edu:2048/cgi/content/short/41/2/149

[22] Godfrey, G.A., Powell, W.B.: An adaptive dynamic programming algorithm for dynamic fleet management, I: Single period travel times. Transportation Science 36(1), 21–39 (2002)

[23] Godfrey, G.A., Powell, W.B.: An adaptive dynamic programming algorithm for dynamic fleet management, II: Multiperiod travel times. Transportation Science 36(1), 40–54 (2002)

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

[25] Goel, V., Slusky, M., van Hoeve, W.J., Furman, K.C., Shao, Y.: Constraint programming for LNG ship scheduling and inventory management. European Journal of Operational Research 241(3), 662–673 (2015)

[26] Halvorsen-Weare, E., Fagerholt, K.: Routing and scheduling in a liquefied natural gas shipping problem with inventory and berth constraints. Annals of Operations Research 203(1), 167–186 (2013), http://dx.doi.org/10.1007/s10479-010-0794-y

[27] Hemmati, A., Hvattum, L.M., Christiansen, M., Laporte, G.: An iterative two-phase hybrid matheuristic for a multi-product short sea inventory-routing problem. European Journal of Operational Research 252(3), 775 – 788 (2016), http://www.sciencedirect.com/science/article/pii/S0377221716300248

[28] Hemmati, A., Stålhane, M., Hvattum, L.M., Andersson, H.: An effective heuristic for solving a combined cargo and inventory routing problem in tramp shipping. Computers & Operations Research 64, 274 – 282 (2015), http://www.sciencedirect.com/science/article/pii/S0305054815001628

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

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

[31] Jiang, Y., Grossmann, I.E.: Alternative mixed-integer linear programming models of a maritime inventory routing problem. Computers & Chemical Engineering 77, 147–161 (2015)

[32] Munguía, L.M., Ahmed, S., Bader, D.A., Nemhauser, G.L., Shao, Y.: Alternating criteria search: A parallel large neighborhood search algorithm for mixed integer programs. Submitted for publication (2016)

[33] Mutlu, F., Msakni, M.K., Yildiz, H., Snmez, E., Pokharel, S.: A comprehensive annual delivery program for upstream liquefied natural gas supply chain. European Journal of Operational Research (2015), http://www.sciencedirect.com/science/article/pii/S0377221715009571

[34] Papageorgiou, D.J., Cheon, M.S., Nemhauser, G., Sokol, J.: Approximate dynamic programming for a class of long-horizon maritime inventory routing problems. Transportation Science 49(4), 870–885 (2014)

[35] Papageorgiou, D.J., Keha, A.B., Nemhauser, G.L., Sokol, J.: Two-stage decomposition algorithms for single product maritime inventory routing. INFORMS Journal on Computing 26(4), 825–847 (2014)

[36] Papageorgiou, D.J., Nemhauser, G.L., Sokol, J., Cheon, M.S., Keha, A.B.: MIRPLib – A library of maritime inventory routing problem instances: Survey, core model, and benchmark results. European Journal of Operational Research 235(2), 350–366 (2014), http://www.sciencedirect.com/science/article/pii/S0377221713009880

[37] Rakke, J.G., Andersson, H., Christiansen, M., Desaulniers, G.: A new formulation based on customer delivery patterns for a maritime inventory routing problem. Transportation Science 49(2), 384–401 (2014)

[38] Rakke, J.G., Stålhane, M., Moe, C.R., Christiansen, M., Andersson, H., Fagerholt, K., Norstad, I.: A rolling horizon heuristic for creating a liquefied natural gas annual delivery program. Transportation Research Part C: Emerging Technologies 19(5), 896–911 (2011), http://www.sciencedirect.com/science/article/pii/S0968090X10001427

[39] Rothberg, E.: An evolutionary algorithm for polishing mixed integer programming solutions. INFORMS Journal on Computing 19(4), 534–541 (2007)

[40] Savelsbergh, M.W.P., Song, J.: An optimization algorithm for the inventory routing problem with continuous moves. Computers & Operations Research 35(7), 2266–2282 (2008)

[41] Shao, Y., Furman, K.C., Goel, V., Hoda, S.: A hybrid heuristic strategy for liquefied natural gas inventory routing. Transportation Research Part C: Emerging Technologies 53, 151 – 171 (2015), http://www.sciencedirect.com/science/article/pii/S0968090X15000406

[42] Song, J.H., Furman, K.C.: A maritime inventory routing problem: Practical approach. Computers & Operations Research 40(3), 657–665 (2013), http://www.sciencedirect.com/science/article/pii/S0305054810002625

[43] Stålhane, M., Rakke, J.G., Moe, C.R., Andersson, H., Christiansen, M., Fagerholt, K.: A construction and improvement heuristic for a liquefied natural gas inventory routing problem. Computers & Industrial Engineering 62(1), 245–255 (2012), http://dx.doi.org/10.1016/j.cie.2011.09.011

[44] Topaloglu, H.: A parallelizable dynamic fleet management model with random travel times. European Journal of Operational Research 175(2), 782–805 (2006), http://www.sciencedirect.com/science/article/pii/S0377221705005278

[45] Topaloglu, H., Powell, W.B.: Dynamic-programming approximations for stochastic time-staged integer multicommodity-flow problems. INFORMS Journal on Computing 18(1), 31–42 (2006)

[46] Toriello, A., Nemhauser, G.L., Savelsbergh, M.W.P.: Decomposing inventory routing problems with approximate value functions. Naval Research Logistics 57(8), 718–727 (2010), http://onlinelibrary.wiley.com.www.library.gatech.edu:2048/doi/10.1002/nav.20433/abstract;jsessionid=4FEF633D9EC7A1D9A7E363

[47] Uggen, K., Fodstad, M., Nørstebø, V.: Using and extending fix-and-relax to solve maritime inventory routing problems. TOP 21(2), 355–377 (2013)

[48] UNCTAD: Review of maritime transport United Nations, New York and Geneva (2015)