Computers & Operations Research 40 (2013) 657–665

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

# A maritime inventory routing problem: Practical approach

Jin-Hwa Song *, Kevin C. Furman

*Corporate Strategic Research, ExxonMobil Research and Engineering Company, Annandale, NJ 08801, United States*

ARTICLE INFO

Available online 12 November 2010

*Keywords:*
IRP
Branch-and-Cut
Large neighborhood search
Maritime inventory routing

ABSTRACT

Despite of the practicality of the motivation of the inventory routing problem (IRP), there are few successful implementation stories of IRP based decision support systems which utilize optimization algorithms. Besides the fact that the IRP is an extremely challenging optimization problem, simplifications and assumptions made in the definition of typical IRP in the literature make it even more difficult to take advantage of the developed technologies for IRP in practice. This paper introduces a flexible modeling framework for IRP, which can accommodate various practical features. A simple algorithmic framework of an optimization based heuristic method is also proposed. A case study on a practical maritime inventory routing problem (MIRP) shows that the proposed modeling and algorithmic framework is flexible and effective enough to be a choice of model and solution method for practical inventory routing problems.



## 1. Introduction

When vendor managed inventory re-supply is employed, the opportunity of cost savings in product distribution comes with the freedom of choice of which customers to visit on a given day and how much product to deliver to the selected customers. Since the vendor is responsible for the inventory management for customers, the problem the vendor faces is how to efficiently integrate inventory management with the product distribution. This problem is known as the inventory routing problem (IRP) in the literature, and there exist a fair number of articles studying this problem. A recent review on the inventory routing problem can be found in a section of the review of vehicle routing by Cordeau et al. [8].

A typical IRP concerns the distribution of a single product from a single supply facility to a set of customers using a homogeneous fleet of vehicles. Unlimited product is available at the supply facility, and each customer has its own storage capacity and consumption rate. The IRP tries to minimize total transportation cost over a given planning horizon while guaranteeing no stockouts at any of the customers. It is usually assumed that the length of each route is short enough to be completed in a single time period (e.g. one day). Therefore, for each day, the optimization makes decisions regarding the selection of customers to visit, amount of product delivered to each selected customer, and the routes of vehicles to perform the deliveries. Even with the simplifications and assumptions made in the definition of the typical IRP, it is well known that the IRP is an extremely challenging optimization problem.

From a practical perspective, there exist many simplifications and assumptions made in this definition of the typical IRP: a single supply or production site, unlimited production capacity, unlimited storage capacity at the production site, constant consumption rate over the planning horizon at customers, etc. Since the length of each route is assumed to be short enough to be performed in a single day, the routing decision can be separated from the customer selection decision and the amount of product delivered to the customers. Once the decisions for which customers are visited on a given day and how much product to deliver are made, the remaining elements of the problem form the capacitated vehicle routing problem, which leads to many ideas for solution methods. These assumptions and simplifications in the definition of the IRP impose a substantial limitation in the use of the model for real world applications. Typically the solution approaches are developed such that they take advantage of these characteristics of the problem, which also substantially limits their use in practice.

There exist variations of this typical IRP in order to consider the practical features of real world problems. Savelsbergh and Song [11,12] study a variation of the IRP with the consideration of multiple production sites, storage capacities at the production sites and customer sites, a constant production rate at each production site, and routes longer than a single time unit. Christiansen and Nygreen [6,7], and Christiansen [3] study a maritime inventory routing problem, which involves multiple production ports with constant production rates and limited storage capacities, multiple loads and discharges, and the consideration of travel time in the inventory update constraints due to long voyages. Most of these variations are very application oriented. Therefore, the solution methods tend to take advantage of the problem specification such as constant production and consumption rates for the entire planning horizon. This is a reasonable approach since their goal is to develop a solution

* Corresponding author.
*E-mail address:* jinhwa.song@exxonmobil.com (J.-H. Song).

0305-0548/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2010.10.031

method for their particular problem of interest. As pointed out by Andersson et al. [2], unlike other routing problems with clear definitions and assumptions, almost every paper concerning combined inventory management and routing has a new version of the problem. Therefore, one of the major difficulties of applying IRP techniques from the literature to a new real world IRP problem is that it is not obvious how to extend these models and solution methods to a more general problem setup, such that they are flexible enough to accommodate various practical features which may appear in other real life inventory routing applications.

When seagoing vessels are responsible for the transportation in IRP, the IRP becomes a maritime inventory routing problem (MIRP). Christiansen and Fagerholt [4] describe a basic MIRP as the transportation of a single product that is produced at loading ports and consumed at discharging ports where each port has a given inventory storage capacity and a production or consumption rate. Christiansen et al. [5] thoroughly review maritime transportation from the operations research perspective. Al-Khayyal and Hwang [1] also describe various aspects of maritime transportation from a modeling point of view. Recently, Furman et al. [9] introduced a MIRP based on a real business application in the petroleum industry. This problem is similar to the problem studied by Christiansen et al. [3,6,7], but the complex cost structure for voyage charter vessels, daily changing production and consumption schedules, and various practical requirements such as draft limits make the problem studied in Furman et al. [9] unique. Demurrage costs are incurred when allowed laytime (i.e. time at port) is exceeded, for example, by waiting to load or to discharge more product. A fixed cost is charged to each leg regardless of the amount of product on board. However, an overage freight cost is applied to any amount of production loaded in excess of a pre-specified cargo amount known in the shipping industry as the part-cargo minimum. This overage cost is charged for the entire voyage. These additional cost components make the cost structure for voyage charter vessels complex. Daily changing production and consumption rates are some of the most important differentiating practical features in the problem because an assumption of constant production and consumption would significantly hamper the operational use of such a model. Draft limits are quite interesting features to marine transportation problems, since a vessel may or may not be allowed to visit a port depending on the weight of the vessel which is affected by the amount of product it carries. Various other practical features can also be introduced into the problem. The problem studied in this paper fits nicely as a test case for the practicality of the proposed modeling and algorithmic framework due to its inclusion of several differentiating characteristics not typically seen in the literature. The problem studied in this paper is the same MIRP discussed by Furman et al. [9]. Through the study of this practical problem, we introduce a flexible modeling framework and a simple optimization based heuristic approach, both of which capture the fundamental features of the MIRP. We also show how various practical features of the MIRP can be effectively handled in the proposed framework. Computational experiments on several realistic test cases illustrate the effectiveness of the proposed method.

The remainder of the paper is organized as follows. In Section 2, the formal problem description is introduced. Section 3 discusses various techniques for the solution methods as well as the development of a large scale neighborhood search procedure as a practical solution technique. Section 4 illustrates how various additional practical features can be incorporated into the proposed model. In Section 5, computational experiments are conducted with the model and algorithm described in this paper. Some comparison of results between the method in this paper and the one in Furman et al. [9] are also presented. Finally, in Section 6 the conclusions are discussed.

## 2. Problem description

The goal of the problem is to find an optimal schedule for routing a heterogeneous pool of seagoing ships in the loading, transporting and discharging of a single bulk product to and from multiple ports while maintaining all capacities, constraints and restrictions related to inventory or to specific ports. This problem is defined as follows.

Given:

- Set of ports at which product is either produced or consumed.
- Set of heterogeneous vessels (some of them may be already chartered).
- Time horizon for planning.
- Daily amount of production or consumption of product at each port.
- Initial inventories of product and inventory limits at each port.
- Load or discharge quantity ranges at each port.
- Travel time between ports.
- Draft limits and other port restrictions.
- Cargo capacity and part cargo minimum for vessels.
- Transportation cost calculation parameters.

The objective is to minimize total transportation cost minus added value by transported product while the routes of specific vessels, the timing of each particular leg of the voyages, and quantities of product loaded and discharged are determined by the solution of the problem such that all constraints are satisfied. This problem already has several complicating characteristics including: flexible cargo sizes, port draft limits, daily changing production and consumption rates, vessels may load and discharge at multiple ports, vessels may revisit ports, limited berth availability at ports, and route-, cargo size- and timing-based transportation costs.

A bulk product is distributed from a set $\mathcal{J}^P$ of production ports to a set $\mathcal{J}^C$ of consumption ports over a planning horizon $T$. The set $\mathcal{J}$ of all ports is the union of $\mathcal{J}^P$ and $\mathcal{J}^C$. A set $\mathcal{J}^D$, a subset of $\mathcal{J}$, represents a set of ports with draft limits. The draft limit restricts the amount of product onboard when a ship enters and leaves the port. The number of loads/discharges by a vessel at port $j$ may be limited. This means that each vessel cannot load/discharge at port $j$ more than $m_j$ times. In maritime transportation, a vessel rarely visits a port more than once within a voyage. However, $m_j > 1$ allows a vessel to load/discharge from/to a port more than the maximum amount of loading/discharging per time unit. When a vessel capacity is relatively larger than the storage capacity at a port, this feature can be quite useful.

The model developed in this paper is a discrete time mixed integer linear programming model in which each time index $t$ belongs to the set $\{1,2,\ldots,T\}$. For practical purposes, the time unit used in the computational experiments is a day. However, different time units can easily be applied depending on necessity. Each port $j \in \mathcal{J}$ has an initial inventory $I_{(j,0)}$ at the beginning of the planning period. Each port $j \in \mathcal{J}$ has also a minimum inventory level $I^{\min}_{(j,t)}$ at time $t$ and a maximum inventory level $I^{\max}_{(j,t)}$ at time $t$ for each $t \in \{1,2,\ldots,T\}$. When a load or a discharge happens at port $j \in \mathcal{J}$, its amount has to be greater than or equal to $f^{\min}_j$ and less than or equal to $f^{\max}_j$. Each port $j \in \mathcal{J}^P$ produces $p_{(j,t)}$ amount of product from time $t-1$ to time $t$, and each port $j \in \mathcal{J}^C$ consumes $d_{(j,t)}$ amount of product from time $t-1$ to time $t$. The travel time between ports $j$ and $j'$ is $t_{jj'}$, and we assume that $t_{jj'}$ is a multiple of the time unit employed. We also assume that travel time includes the time a vessel spends at a port. This port time may consist of berthing time, loading time, discharging time, de-berthing time, and so on.

A set $\mathcal{V}$ of voyage charter vessels is available for transporting the product. A voyage begins when an empty vessel visits a port and

ends when a vessel becomes empty again. A voyage charter vessel is hired to cover a single voyage. Each vessel $v \in \mathcal{V}$ has an initial inventory $I_{v0}$ at the beginning of the planning period. Unless Vessel $v$ is a carried over vessel from the previous planning period, $I_{v0}$ is in general zero. Each vessel $v$ has a maximum amount of product $C_v$ it can carry. A vessel $v \in \mathcal{V}$ belonging to the set $\mathcal{V}^C$ of chartered vessels becomes available at time $t_v^C$ and must be used in the solution. Each vessel $v \in \mathcal{V} \backslash \mathcal{V}^C$ may or may not be used. For each $v \in \mathcal{V}, j \in \mathcal{J}^D$, and $t \in \{1,2,\ldots,T\}$, the inlet draft limit $D_{vjt}^I$ and outlet draft limit $D_{vjt}^O$ need to be satisfied. These draft limits are time dependent parameters because maximum drafts for vessels can change with seasons. For each vessel $v \in \mathcal{V}$, $b_v$, $w_v$, $r_v^D$, and $r_v^O$ represent part cargo minimum tons (for short, PC tons), world scale multiplier, demurrage rate, and overage multiplier, respectively. The flat rate for traveling from port $j \in \mathcal{J}$ to port $j' \in \mathcal{J}$ is $c_{jj'}$. If vessel $v$ travels from port $j$ to port $j'$, the flat cost for this leg can be represented by $b_v w_v c_{jj'}$. The demurrage cost is charged when a vessel is waiting for its next activity or when it is moving slower than its expected speed. The demurrage cost for vessel $v$ is calculated by $r_v^D$ multiplied by the number of demurrage days of vessel $v$'s voyage. When vessel $v$ carries product more than its part cargo minimum (a.k.a. PC tons) $b_v$, overage cost is charged. If any leg of vessel $v$'s voyage incurs overage, the overage rate $r_v^O w_v c_{jj'}$ applies to the entire legs of vessel $v$'s voyage based on the largest amount of overage of the voyage.

For the objective function, we introduce a parameter $\eta$ which represents the value (or netback) of a ton of product transported. The objective is to minimize transportation costs minus $\eta$ multiplied by the amount of product transported over the planning horizon $T$ while satisfying all the requirements, constraints and restrictions.

### 2.1. Formulation

This section develops a new time–space network formulation for the MIRP described above. Although this particular formulation is specific to the MIRP, the basic concepts are generally applicable to many classes of routing problems. Further, due to its general nature, this modeling framework can easily accommodate side constraints and practical restrictions and requirements for routing problems. Finally, these concepts also lend themselves well in the development of efficient Branch-and-Cut algorithms and optimization-based heuristic techniques as will be described in later sections.

As presented by Savelsbergh and Song [12], the time–space network formulation can be viewed as an integer multi-commodity network flow formulation. A vessel is a commodity and a node represents a possible visit to a port at a particular time. The network has a set of nodes and a set of arcs. The node set is shared by all vessels, and each vessel has its own arc set. The set of nodes $N$ consists of one origin node (0,0), one sink node (0,$T$+1), and a set of regular nodes $N^R = \{(j,t) : j \in \mathcal{J}, t \in \{1,2,\ldots,T\}\}$. The arc set $A_v$ represents a set of arcs belonging to vessel $v$, and the set of arcs $A$ is defined by $\bigcup_{v \in \mathcal{V}} A_v$.

There are five types of arc sets in the set of arcs $A$. A travel arc $(v,(j,t),(j',t+t_{jj'}))$ such that $v \in \mathcal{V}, (j,t) \in N^R, (j',t+t_{jj'}) \in N^R$, and $j \neq j'$ represents a possibility of vessel $v$ traveling from port $j$ to port $j'$, leaving at time $t$ and arriving at time $t+t_{jj'}$. Let $A_v^T$ denote the set of all travel arcs for vessel $v$, then $A^T = \bigcup_{v \in \mathcal{V}} A_v^T$. A demurrage arc $(v,(j,t),(j,t+1))$ with $v \in \mathcal{V}, (j,t) \in N^R$ and $(j,t+1) \in N^R$ represents a possibility of vessel $v$ waiting at port $j$ from time $t$ to time $t$+1. Let $A_v^D$ denote the set of all demurrage arcs for vessel $v$, then $A^D = \bigcup_{v \in \mathcal{V}} A_v^D$. An arc $(v,(0,0),(j,t))$ with $v \in \mathcal{V}$ and $(j,t) \in N^R$ represents when and where vessel $v$ starts its voyage. An arc $(v,(j,t),(0,T+1))$ with $v \in \mathcal{V}$ and $(j,t) \in N^R$ represents when and where vessel $v$ finishes its voyage. An arc $(v,(0,0),(0,T+1))$ represents a possibility of vessel $v$ not being used. Let $c_a$ represent the cost of using arc $a$. The cost of using travel arc $a \in A_v^T$ which goes from node $(j,t)$ to node $(j',t+t_{jj'})$ is $b_v w_v c_{jj'}$. The cost of using demurrage arc $a \in A_v^D$ is $r_v^D$. We set the cost of the remaining arcs to zero. If there exists a fixed cost for using a vessel, it can be applied to either arc $(v,(0,0),(j,t))$ or arc $(v,(j,t),(0,T+1))$. Let $\delta^+(n)$ denote the set of arcs that have node $n$ as their tail node, and let $\delta^-(n)$ denote the set of arcs that have node $n$ as their head node.

Each vessel has its own set of arcs. Therefore, each vessel may have a different set of arcs if it is necessary. For example, if two

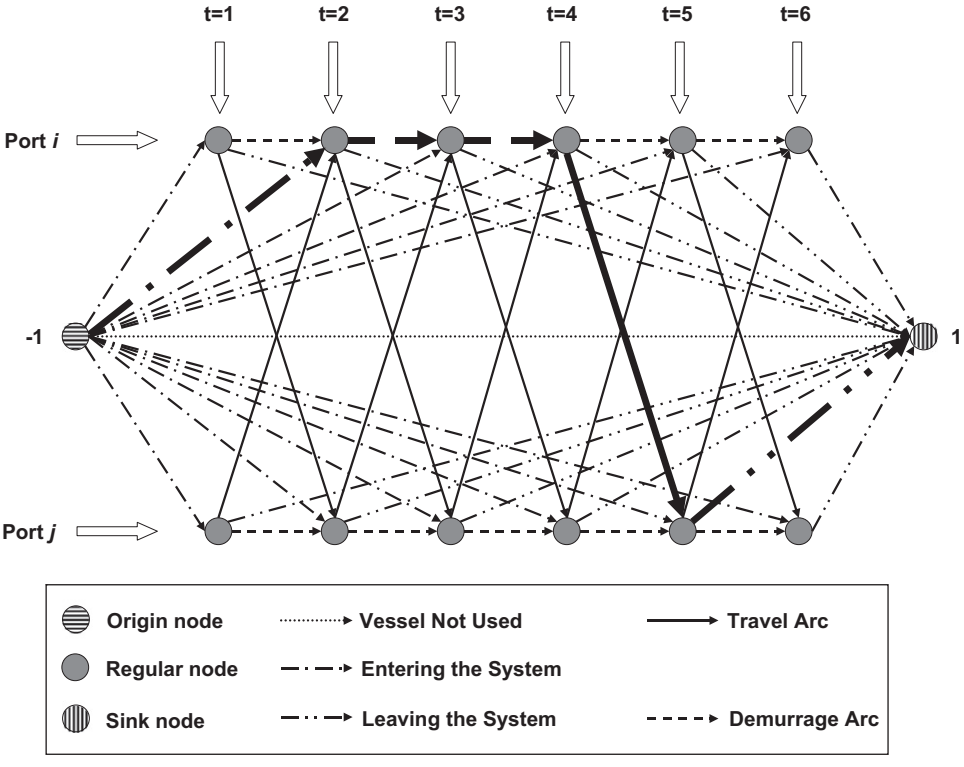


**Fig. 1.** Example of time–space network structure.

vessels have noticeably different nominal speeds, $t_{jj'}$ requires an additional vessel index to accommodate the differences in speed among vessels. In this case, the set of arcs for vessel $v$ can be constructed by using $t^v_{jj'}$, which is a vessel specific travel time.

Fig. 1 shows an example of the network structure described above. The bold line in the figure represents a possible voyage for a vessel. In this example, a vessel enters the system at time $t=2$ by arriving at port $i$. After spending 2 days of demurrage, it visits port $j$ at time $t=5$ and leaves the system.

A binary variable $x_a$ for each $a \in A_v$ takes one if vessel $v$ uses arc $a$ and takes zero otherwise. A binary variable $z_{vn}$ for each vessel $v \in \mathcal{V}$ and each node $n=(j,t) \in N^R$ indicates whether or not vessel $v$ loads product from port $j$ if $j \in \mathcal{J}^P$ and discharges product to port $j$ if $j \in \mathcal{J}^C$ at time $t$. A continuous variable $f_{vn}$ with $n=(j,t) \in N^R$ represents flow of product between port $j$ and vessel $v$. If $j \in \mathcal{J}^P$ where $n=(j,t)$, then $f_{vn}$ is a loading amount of product from port $j$ by vessel $v$ at time $t$, and if $j \in \mathcal{J}^C$ where $n=(j,t)$, then $f_{vn}$ is a discharging amount of product to port $j$ by vessel $v$ at time $t$. A continuous variable $I_n$ with $n=(j,t) \in N^R$ denotes the inventory level at port $j$ right after time $t$. Another continuous variable $I_{vt}$ represents the amount of product on board for vessel $v$ right after time $t$. A continuous variable $o_v$ for each vessel $v \in \mathcal{V}$ represents the largest amount of overage of vessel $v$'s voyage. Another continuous variable $o_{va}$ for each travel arc $a \in A^T_v$ and $v \in \mathcal{V}$ is equal to $o_v$ if arc $a$ is used, and otherwise is zero. The variable $o_{va}$ is used in the objective function for the calculation of overage costs.

The presentation of a time–space network based formulation now begins with flow conservation constraints:

$$\sum_{\{a \in A_v : a \in \delta^-(n)\}} x_a - \sum_{\{a \in A_v : a \in \delta^+(n)\}} x_a = 0, \quad \forall v \in \mathcal{V},\ \forall n \in N^R, \tag{1}$$

$$\sum_{\{a \in A_v : a \in \delta^+((0,0))\}} x_a = 1, \quad \forall v \in \mathcal{V}, \tag{2}$$

$$\sum_{\{a \in A_v : a \in \delta^-((0,T+1))\}} x_a = 1, \quad \forall v \in \mathcal{V}. \tag{3}$$

The next set of constraints ensures inventory balance at the ports:

$$I_{(j,t-1)} - \sum_{v \in \mathcal{V}} f_{vn} + p_{(j,t)} = I_{(j,t)}, \quad \forall j \in \mathcal{J}^P,\ \forall n=(j,t) \in N^R, \tag{4}$$

$$I_{(j,t-1)} + \sum_{v \in \mathcal{V}} f_{vn} - d_{(j,t)} = I_{(j,t)}, \quad \forall j \in \mathcal{J}^C,\ \forall n=(j,t) \in N^R. \tag{5}$$

Inventory balance for product on the vessels are represented by the following expression:

$$I_{vt-1} + \sum_{\{n=(j,t): j \in \mathcal{J}^P\}} f_{vn} - \sum_{\{n=(j,t): j \in \mathcal{J}^C\}} f_{vn} = I_{vt}, \quad \forall t \in \{1,2,\ldots,T\},\ \forall v \in \mathcal{V}. \tag{6}$$

The following set of constraints ensures that a load or a discharge by a vessel can occur only when the vessel is at that port. If this occurs, the loading amount or the discharging amount has to be in between the port specific minimum and maximum amounts:

$$z_{vn} \le \sum_{\{a \in A_v : a \in \delta^-(n)\}} x_a, \quad \forall n \in N^R,\ \forall v \in \mathcal{V}, \tag{7}$$

$$f^{\min}_j z_{vn} \le f_{vn} \le f^{\max}_j z_{vn}, \quad \forall n=(j,t) \in N^R,\ \forall v \in \mathcal{V}. \tag{8}$$

In many practical cases, berthing rules specify that only one vessel may stop for loading or discharging at a port at any given time. If more than one vessel can stop at a port at a time, the right hand side of the constraint below can be adjusted appropriately:

$$\sum_{v \in \mathcal{V}} z_{vn} \le 1, \quad \forall n \in N^R. \tag{9}$$

The next set of constraints enforces the rule that a vessel may not load or discharge more than $m_j$ times at port $j$. This rule arises from practical limits set for revisiting a port. A typical reason to revisit a port would be that some port has an inventory limit lower than the cargo size limit of a vessel:

$$\sum_{\{t: n=(j,t) \in N^R\}} z_{vn} \le m_j, \quad \forall v \in \mathcal{V},\ \forall j \in \mathcal{J}. \tag{10}$$

The following set of constraints enforce draft limits which restrict the cargo capacity of a vessel entering or exiting a port due to water depth. Without loss of generality, it is assumed that the draft limits for a vessel $v$ are less than or equal to vessel $v$'s capacity $C_v$:

$$I_{vt-1} \le D^I_{vjt} + (C_v - D^I_{vjt})(1 - z_{vn}), \quad \forall n=(j,t) \in N^R,\ \forall v \in \mathcal{V}, \tag{11}$$

$$I_{vt} \le D^O_{vjt} + (C_v - D^O_{vjt})(1 - z_{vn}), \quad \forall n=(j,t) \in N^R,\ \forall v \in \mathcal{V}. \tag{12}$$

The overage calculations are handled by the next set of constraints. Overage is the greatest amount of cargo carried by a vessel in excess of the PC tons among any leg of a voyage. Without loss of generality, it is assumed that the capacity of vessel $C_v$ is greater than or equal to its PC tons $b_v$:

$$o_v \ge I_{vt} - b_v, \quad \forall v \in \mathcal{V},\ \forall t \in \{1,2,\ldots,T\}, \tag{13}$$

$$o_{va} \ge o_v - (C_v - b_v)(1 - x_a), \quad \forall v \in \mathcal{V},\ \forall a \in A^T_v. \tag{14}$$

The final set of constraints ensures that all variables remain within specific bounds:

$$x_a \in \{0,1\}, \quad \forall a \in A, \tag{15}$$

$$z_{vn} \in \{0,1\}, \quad \forall v \in \mathcal{V},\ \forall n \in N^R, \tag{16}$$

$$I^{\min}_n \le I_n \le I^{\max}_n, \quad n \in N^R, \tag{17}$$

$$f_{vn} \ge 0, \quad \forall v \in \mathcal{V},\ \forall n \in N^R, \tag{18}$$

$$0 \le I_{vt} \le C_v, \quad \forall v \in \mathcal{V},\ \forall t \in \{1,2,\ldots,T\}, \tag{19}$$

$$0 \le o_{va} \le C_v - b_v, \quad \forall v \in \mathcal{V},\ \forall a \in A^T_v, \tag{20}$$

$$0 \le o_v \le C_v - b_v, \quad \forall v \in \mathcal{V}. \tag{21}$$

The objective function is to minimize total transportation costs minus the total product transported multiplied by the constant netback factor, $\eta$:

$$\min \sum_{a \in A} c_a x_a + \sum_{v \in \mathcal{V}} \sum_{\{a=(v,(j,t),(j',t')) \in A^T_v\}} r^Q_v w_v c_{jj'} o_{va} - \sum_{v \in \mathcal{V}} \sum_{\{n=(j,t): j \in \mathcal{J}^C\}} \eta f_{vn}. \tag{22}$$

## 3. Solution techniques

The general solution methods developed here can be adapted to any IRP problem that can be formulated with the time–space network described in the previous section. Preprocessing is used to reduce problem size by enforcing problem specific requirements. These requirements are directly handled by fixing some variables to certain values and/or by removing some arcs from the network rather than adding specific constraints in the model. For example, travel arcs from a consumption port to a production port are

removed and vessel inventory at the end of the planning horizon is set to zero since all the vessels are chartered for a single voyage in our problem.

Previously chartered vessels are those in which a charter contract is already in place. Therefore, if vessel $v$ is a previously chartered vessel, i.e., $v \in \mathcal{V}^C$, then vessel $v$ must be used in the solution. The contract also specifies the date $t_v^C$ on which the demurrage calculation begins. Thus $v$ enters the system at time $t_v^C$. In this case, all the arcs from the source for the vessel should be removed except the ones entering the system on $t_v^C$. Port blackout days are simply handled by fixing corresponding $z_{vn}$ variables to zero. These kinds of preprocessing steps should be used whenever possible.

Special branching rules and user defined cuts play important roles for the performance of a Branch-and-Cut algorithm. In Section 3.1, some effective branching rules and cuts for the problem are developed. Due to the complexity of the problem, heuristic algorithm development as a practical solution method is imperative. In Section 3.2, an optimization-based large neighborhood search procedure is developed in order to find good solutions within a reasonable amount of time. Finally, in the Section 3.3, the complete solution technique developed is presented.

### 3.1. Branch-and-Cut

The algorithm we develop in Section 3.2 is a practical optimization-based heuristic method that involves iteratively solving small subproblems to improve the incumbent solution. This heuristic algorithm solves subproblems using a Branch-and-Cut algorithm. The subproblem derived is a restricted version of the original problem, which means its feasible region is a subset of the feasible region of the original problem. Moreover, a subproblem is generated based on an integer feasible solution to the original problem. This heuristic method can thus be viewed as an optimization-based large neighborhood search.

In this section, we discuss some valid inequalities which can tighten the formulation of the original problem. A special branching rule is also discussed. These improvements to the solution method can be applied to both the original problem and the subproblems utilized in the heuristic.

For any $t \in \{1,2,\ldots,T\}$ and $j \in \mathcal{J}^P$, $I_{(j,0)}+\sum_{t' \le t} p_{(j,t')}$ represents the cumulative amount of product available at port $j$ up to time $t$. Any vessel on any time no later than $t$ cannot load more than $\min\{f_j^{max}, \max_{t' \le t}\{I_{(j,t')}^{max}+p_{(j,t')}-I_{(j,t')}^{min}\}\}$ per load from port $j$. This gives a lower bound

$$\left\lceil \frac{I_{(j,0)}+\sum_{t' \le t} p_{(j,t')}-I_{(j,t)}^{max}}{\min\{f_j^{max}, \max_{t' \le t}\{I_{(j,t')}^{max}+p_{(j,t')}-I_{(j,t')}^{min}\}\}} \right\rceil,$$

on the number of loads at port $j$ up to time $t$ from the beginning of the planning period. The following set of inequalities is then valid:

$$\sum_{v \in \mathcal{V}} \sum_{\{n=(j,t') \in N^R | t' \le t\}} z_{vn} \ge \left\lceil \frac{I_{(j,0)}+\sum_{t' \le t} p_{(j,t')}-I_{(j,t)}^{max}}{\min\{f_j^{max}, \max_{t' \le t}\{I_{(j,t')}^{max}+p_{(j,t')}-I_{(j,t')}^{min}\}\}} \right\rceil,$$
$$\forall j \in \mathcal{J}^P, \ \forall t \in \{1,2,\ldots,T\}. \quad (23)$$

Similarly, the following inequalities are also valid for consumption ports:

$$\sum_{v \in \mathcal{V}} \sum_{\{n=(j,t') \in N^R | t' \le t\}} z_{vn} \ge \left\lceil \frac{I_{(j,t)}^{min}-(I_{(j,0)}-\sum_{t' \le t} d_{(j,t')})}{\min\left\{f_j^{max}, max_{t' \le t}\{I_{(j,t')}^{max}+d_{(j,t')}-I_{(j,t')}^{min}\}\right\}} \right\rceil,$$
$$\forall j \in \mathcal{J}^C, \forall t \in \{1,2,\ldots,T\}. \quad (24)$$

Another set of valid inequalities is based on an input from the user involving a limit on the number of loads/discharges at port $j$ by a vessel. The number of loads/discharge for vessel $v$ at port $j$ cannot exceed $u_j$. If $u_j$ is large, the following cuts may not be effective. In many practical cases, however, $u_j$ is one, in which condition these cuts can be effective to cut off the optimal solution of the LP relaxation:

$$\sum_{\{n=(j,t) | t \in \{1,2,\ldots,T\}\}} z_{vn} \le u_j \left( \sum_{\{a=(v,(j,t),(j',t')) \in A_v^T\}} x_a + \sum_{\{a=(v,(j,t),(0,T+1)) \in A_v\}} x_a \right),$$
$$\forall j \in \mathcal{J}, \ \forall v \in \mathcal{V}. \quad (25)$$

Even though no illustrative example is provided here, it is easy to see that the above cuts are valid because a vessel should visit a port to perform loading/discharging and it should leave the port afterward. The above inequalities can cut off some fractional solutions which send a fraction of a ship to a port and performing a fraction of loading/discharging in order to save some transportation costs.

As mentioned above, a vessel $v$ may load/discharge multiple times at a port $j$. When this occurs, vessel $v$ may be prohibited from visiting any other port between loads/discharges at port $j$. This is called *direct revisit*. If *direct revisit* is a requirement, the following set of constraints are added to the model:

$$\left( \sum_{\{a=(v,(j',t'),(j,t)) \in A_v^T\}} x_a + \sum_{\{a=(v,(0,0),(j,t)) \in A_v\}} x_a \right) \le 1, \quad \forall j \in \mathcal{J}, \ \forall v \in \mathcal{V}. \quad (26)$$

When the requirement of *direct revisit* is in place, a special branching rule that takes advantage of *direct revisit* can be employed. The solution space is divided into $\sum_{\{a=(v,(j,t),(j',t')) \in A^T\}} x_a = 1$ and $\sum_{\{a=(v,(j,t),(j',t')) \in A^T\}} x_a = 0$ for each $v \in \mathcal{V}$, $j \in \mathcal{J}$, and $j' \in \mathcal{J}$ such that $j \ne j'$. A similar branching rule is applied to the arcs coming out of the origin node and arcs going into the sink node. This branching rule makes the branching tree more balanced than conventional branching on binary variables. In many practical applications, a vessel (or a vehicle in general) does not load or discharge from a site more than once. This is a special case of *direct revisit* because, with $m_j = 1$ and the costs of arcs satisfying the triangle inequality, *direct revisit* ensures that a vessel does not visit a site more than once.

### 3.2. A large neighborhood search

Even though solving practical instances in a reasonable amount of time with the proposed model and Branch-and-Cut algorithm is computationally prohibitive, the optimization algorithm developed still works efficiently with small instances. This makes it possible to use the optimization algorithm in the framework of a large neighborhood search. The basic idea is similar to the one introduced by Savelsbergh and Song [12].

The search starts with a feasible integer solution. By fixing some of the binary variables to the values of the current feasible solution, a small sub-problem can be constructed and solved. This procedure is continued until it is determined that a better solution cannot be found with this method. The details of the algorithm are described in Algorithm 1.

The concept of this large neighborhood search is quite general because it generates small Mixed Integer Programming subproblems by fixing some binary variables from the best solution found so far. However, the construction of sub-problems in Algorithm 1 is using problem specific structures. From the incumbent solution, a sub-problem is generated by fixing all of the binary variables except those associated with a selected pair of vessels. Each vessel's routing and scheduling determines its transportation cost, and the sum of the transportation cost of each ship is the total cost in the objective function. Within a restricted inventory profile

in a sub-problem, the transportation cost of two selected vessels is optimized in a sub-problem. Since two vessels are selected in a sub-problem, there is also the possibility of switching and/or consolidating the ports each vessel visits, which can yield a better solution. As later described in Section 5, this heuristic algorithm is quite effective. This is another example of showing the importance of using the problem structure in the solution methods.

**Algorithm 1.** Large neighborhood search.

```
count=0
while count < |V|*(|V|−1)/2 do
    Select two vessels v and v′ which have never been selected since the best solution found so far
    Fix binary variables which are not associated with vessel v and v′
    Solve MIP
    if Better solution found then
      Update best solution
      count=0
    else
      count=count+1
    end if
end while
```

Note that there are many factors which can affect the performance of the search procedure. The initial solution affects the performance of this search. The choice of two vessels for each sub-problem can also produce different solutions. A time limit can be enforced for the sub-problems in order to manage the run-time, or a different stopping rule can be employed. The best settings for these parameters can be different from instance to instance, and thus requires some iterative investigations. The setting of parameters for Algorithm 1 is tuned through computational experimentation with real instances.

### 3.3. Optimization-based heuristic method

By using the large neighborhood search method described above, an optimization-based heuristic method has been designed. The whole procedure is briefly described in Algorithm 2. ILOG CPLEX is used as the MIP solver. CPLEX runs until it finds an integer feasible solution. Finding an integer feasible solution for the instances in Section 5 was not an issue. However, if finding a first feasible solution is difficult, a feasibility pump can replace this step. Once a feasible solution is found, the CPLEX Solution Polishing algorithm is used for a small, predetermined amount of time in order to improve the solution. For the experiments in Section 5, the algorithm spent 30 seconds with Solution Polishing. After these two steps, the best solution found so far is used as an initial solution to the large neighborhood search Algorithm 1.

Since Algorithm 2 finds a good solution within a reasonable amount of time, it can be used as a standalone solution method for practical purposes. Algorithm 2 can also be used for finding a good initial solution for a global optimization algorithm. Optimization algorithms such as Branch-and-Cut can perform better when initiated with a good initial solution.

**Algorithm 2.** Optimization-based heuristic.

```
Run MIP solver until it finds an integer feasible solution
Run Solution Polishing for a pre-determined amount of time
Run Algorithm 1
```

Solution Polishing is a proprietary development of ILOG, now part of IBM. It is a local search heuristic [10]. Algorithm 2 works well even without using Solution Polishing, therefore, a MIP solver other than CPLEX may be used to implement this method and the step involving Solution Polishing could simply be omitted. However, computational experience indicates that in most cases using Solution Polishing finds an improved solution, which improves the starting point for Algorithm 1 and ultimately aids the Branch-and-Cut algorithm.

## 4. Incorporation of various practical requirements

In practice, many additional practical requirements can arise to the model introduced earlier. This section discusses how several of these additional features may be incorporated into the model.

Sometimes it may be possible to buy the product from or sell the product to a third party site. Third party ports need to be handled differently because tracking inventory levels is not necessary. Let $\mathcal{J}^3$ be a set of third party ports. For each third party port $j \in \mathcal{J}^3$, time windows and available amount of product for loading/discharging for each time window are given. If $j \in \mathcal{J}^P \cap \mathcal{J}^3$, it means product can be loaded from port $j$ up to the pre-specified amount within the given time window. If $j \in \mathcal{J}^C \cap \mathcal{J}^3$, it means product can be discharged at port $j$ up to the prespecified amount within the given time window. Let $T^1_{jk}$ and $T^2_{jk}$ represent the beginning and the ending of $k$th time window for a third party port $j \in \mathcal{J}^3$, respectively. Let $Q^k_j$ for $j \in \mathcal{J}^3$ also represent the available amount for loading/discharging during $k$th time window. It is assumed that time windows for a third party port are mutually exclusive. Then instead of Constraint (4) and (5), the following constraint for each time window $k$ of each port $j \in \mathcal{J}^3$ is necessary:

$$\sum_{v \in \mathcal{V}} \sum_{\{n=(j,t) | T^1_{jk} \le t \le T^2_{jk}\}} f_{vn} \le Q^k_j.$$

A lower limit $\underline{N}$ and an upper limit $\overline{N}$ on the number of vessels used in the solution can easily be considered in the model:

$$\underline{N} \le \sum_{v \in \mathcal{V}} \sum_{\{a=(v,(0,0),(0,T+1)) \in A_v\}} (1-x_a) \le \overline{N}.$$

It may also be necessary to enforce a minimum amount of product $M$ to be transported. The following constraint handles this situation.

$$\sum_{v \in \mathcal{V}} \sum_{\{n=(j,t) \in N^R | j \in \mathcal{J}^C\}} f_{vn} \ge M.$$

For each vessel $v \in \mathcal{V}$, there may exist a limit $\overline{D}_v$ on the demurrage days. The following set of constraints enforces this limit:

$$\sum_{\{a \in A^D_v\}} x_a \le \overline{D}_v, \quad \forall v \in \mathcal{V}.$$

Each vessel $v \in \mathcal{V}$ may need to load at least $l_v$ percent of its PC tons. The following set of constraints ensures that the solution satisfies this requirement:

$$\sum_{\{n=(j,t) \in N^R | j \in \mathcal{J}^P\}} f_{vn} \ge l_v b_v \sum_{\{a=(v,(0,0),(0,T+1)) \in A_v\}} (1-x_a), \quad \forall v \in \mathcal{V}.$$

Product loaded from port $j \in \mathcal{J}^P$ may need to be discharged at port $j' \in \mathcal{J}^C$. Based on this relation, the set $G$ is defined such that if $(j,j') \in G$, then product loaded from port $j$ has to be discharged at port $j'$. This can

be satisfied by the following set of constraints:

$$\sum_{\{n=(j,t)\in N^R | (j,j')\in G\}} f_{vn} \le \sum_{\{n=(j',t)\in N^R\}} f_{vn}, \quad \forall v \in \mathcal{V},\ \forall j' \in \mathcal{J}^C.$$

The above inequality needs to be an equality if port $j'$ cannot receive product from port $j$ when $(j,j')\notin G$.

A port may have a special requirement on the minimum amount of time between consecutive loads/discharges. Let port $j$ need at least $T_j^{Con}$ amount of time between any consecutive loads/discharges. For each $t \in \{1,2,\ldots,T-T_j^{Con}\}$, the following constraint can be added in order to satisfy the requirement:

$$\sum_{v\in\mathcal{V}} \sum_{\{n=(j,t')\in N^R | t\le t'\le t+T_j^{Con}\}} z_{vn} \le 1.$$

## 5. Computational experimentation

The model and algorithm described so far has been developed and implemented as a decision support system, which is an enhancement to the previous existing system introduced by Furman et al. [9]. All the practical features discussed in Section 4 are implemented per users' request. These practical features are implemented as options, such that the users can choose to activate any particular feature based on their needs. Computational experiments have been performed with 10 instances generated based on actual operations of a major oil and gas company through the decision support system. Each instance utilizes most of the practical features described in Section 4.

**Table 1**
Description of instances.

| Instance | Production ports (Third party) | Consumption ports (Third party) | Vessels (Chartered) |
|---|---|---|---|
| P1 | 3 (0) | 3 (0) | 6 (3) |
| P2 | 3 (0) | 3 (0) | 6 (3) |
| P3 | 4 (1) | 3 (0) | 6 (3) |
| P4 | 4 (1) | 3 (1) | 6 (3) |
| P5 | 4 (1) | 3 (1) | 6 (3) |
| P6 | 4 (1) | 4 (1) | 5 (3) |
| P7 | 4 (1) | 4 (1) | 5 (3) |
| P8 | 4 (1) | 4 (1) | 5 (3) |
| P9 | 4 (1) | 4 (1) | 5 (3) |
| P10 | 4 (1) | 4 (1) | 5 (3) |

The machine used for the experiments is a 3.4 GHz Intel Xeon CPU desktop with 3 GB of RAM. CPLEX 10 has been used as a MIP solver. Table 1 illustrates the scale and the size of the instances used in the experimentation. The planning horizons for all instances are about 2 months. The experimental results clearly show the effectiveness achieved using the model and the algorithm developed.

In order to study the value of the cuts and the special branching rule developed in Section 3.1, a set of experiments has been conducted. The columns with *User Cuts* heading in Table 2 show the number of Branch-and-Cut nodes and CPU time in seconds for proving optimality when the cuts are added but the special branching rule is not employed. The comparison of these two columns with the last two columns in the table clearly shows the value of the special branching rule. The columns with *Special Branching* heading in Table 2 also show the number of Branch-and-Cut nodes and CPU time in seconds for proving optimality when the cuts introduced in this paper are not used, but the special branching rule is employed. The results in Table 2 demonstrate the value of the cuts developed. More importantly, it is clear from Table 2 that using both the cuts and the special branching makes a big difference in performance of the algorithm.

Table 3 contains CPU times in seconds for various solution methods. The first two columns show the results when Branch-and-Cut is used, and the next three columns show the results when Branch-and-Cut is combined with the optimization based heuristic method, Algorithm 2, as proposed in Section 3.3. In the latter case, the Branch-and-Cut algorithm begins with the solution found by Algorithm 2. The column with *Optimal Solution* heading shows how many seconds the algorithm spent in finding what was later determined to be the optimal solution. The column with *Proven Optimality* heading shows how many seconds the algorithm spent to prove optimality. The column with Algorithm 2 shows how many seconds Algorithm 2 required. If the value in the *Optimal Solution* column is less than the value in the Algorithm 2, it means that Algorithm 2 found an optimal solution prior to its termination. In 9 out of 10 cases, Algorithm 2 found optimal solutions. The results here illustrate the benefits of using Algorithm 2.

Table 4 contains comparison results between the model and solution approach presented in this paper and the prior approach [9]. Beyond the first two instances, the previous method cannot even solve the instances and generate any meaningful data. Therefore, the comparison results shown in Table 4 are based only on the experimentation of instances P1 and P2. The method by Furman et al. failed to close out the optimality gap after 10 h for both instance P1 and P2. However, the new method proposed in this paper could prove the optimality in 6 min for instance P1 and in 35 min for instance P2. Fig. 2 shows how lower bounds and upper

**Table 2**
Comparison of branching rules and user cuts.

| | User cuts | | Special branching | | Special branching and user cuts | |
|---|---|---|---|---|---|---|
| | B&C nodes | Time (s) | B&C nodes | Time (s) | B&C nodes | Time (s) |
| P1 | 15,165 | 414 | 56,399 | 1177 | 42,009 | 931 |
| P2 | 388,667 | 42,200 | 96,280 | 10,411 | 33,311 | 7568 |
| P3 | 951,624 | 142,211 | 81,838 | 10,237 | 47,859 | 5874 |
| P4 | 676,951 | 116,798 | 149,108 | 25,570 | 26,826 | 6925 |
| P5 | 85,289 | 9677 | 43,710 | 7229 | 11,592 | 1720 |
| P6 | 134,976 | 25,028 | 627,830 | 131,405 | 35,477 | 8333 |
| P7 | 62,971 | 7750 | 297,485 | 58,936 | 12,399 | 2759 |
| P8 | 53,676 | 9654 | 271,205 | 54,782 | 39,780 | 10,203 |
| P9 | 325,709 | 40,552 | 554,784 | 128,698 | 72,066 | 15,854 |
| P10 | 62,765 | 13,390 | 265,409 | 42,396 | 19,106 | 4073 |
| Avg | 275,779 | 40,767 | 244,405 | 47,084 | 34,043 | 6424 |

**Table 3**
CPU times (seconds) of Branch-and-Cut with and without the heuristic.

| | Branch-and-Cut | | Algorithm 2+Branch-and-Cut | | |
|---|---|---|---|---|---|
| | Optimal solution | Proven optimality | Optimal solution | Algorithm 2 | Proven optimality |
| P1 | 922 | 931 | 28 | 86 | 343 |
| P2 | 7304 | 7568 | 78 | 145 | 2049 |
| P3 | 4028 | 5874 | 59 | 120 | 6234 |
| P4 | 6427 | 6925 | 110 | 165 | 4700 |
| P5 | 1235 | 1720 | 186 | 334 | 1805 |
| P6 | 3398 | 8333 | 203 | 258 | 13,840 |
| P7 | 2123 | 2759 | 171 | 216 | 1587 |
| P8 | 3156 | 10,203 | 224 | 273 | 6249 |
| P9 | 14,581 | 15,854 | 178 | 240 | 9104 |
| P10 | 2907 | 4073 | 1226 | 287 | 2514 |
| Avg | 4608 | 6424 | 246 | 212 | 4843 |

**Table 4**
Comparison with the old model.

| | Old method by Furman et al. [9] | | New method (Heuristic+Branch-and-Cut) | |
|---|---|---|---|---|
| | Optimal solution | Gap after 10 h (%) | Optimal solution | Proven optimality |
| P1 | 756 | 86.9 | 28 | 343 |
| P2 | –[a] | 112.7 | 78 | 2049 |

[a] Means failed to find optimal solution.

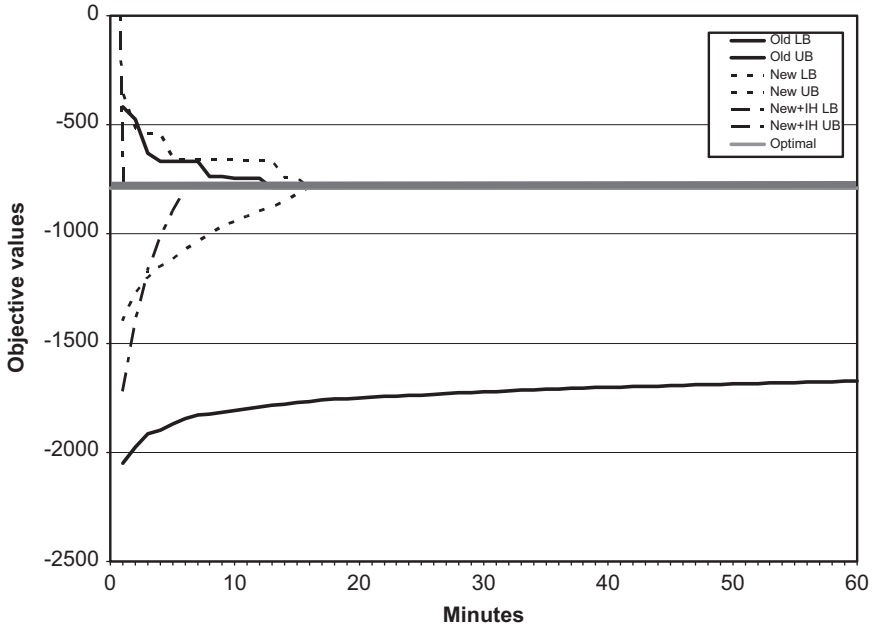


**Fig. 2.** Performance comparison of models and algorithms for instance P1.

bounds for instance P1 from various methods converge to its optimal objective function value. In Fig. 2, *Old* represents the method in Furman et al. [9], and *New* represents the model and the Branch-and-Cut algorithm presented in this paper. When Algorithm 2 is used as an initial heuristic for the Branch-and-Cut, it is represented by *New+IH*. Lower bounds, *LB*, are the line below the *Optimal* line, and upper bounds, *UB*, are the lines above the *Optimal* line. The gap between lower bound and upper bound is closed out much faster with the *New* method than with the *Old* method. This figure also shows that Algorithm 2 helped close out the optimality gap even faster. Table 4 and Fig. 2 clearly show that the method described in this paper outperforms the method used by Furman et al. [9]

## 6. Conclusion

This paper presents a model and a solution method for a practical maritime inventory routing problem. The model is based on the time–space expanded network, and the solution methods consist of an optimization-based large neighborhood search and a Branch-and-Cut algorithm. Computational results show that the method presented in this paper solves the real operations-based instances in a reasonable amount of time.

The main ideas of the proposed modeling and algorithmic framework are general enough to be a choice in implementations for other optimization-based IRP applications. From a modeling perspective, a heterogeneous pool of vehicles for the transportation, daily changeable

production and consumption rates, multiple pickup and delivery locations are noteworthy general features among IRP models. From an algorithmic perspective, the main concepts for the cuts, branching rules, and the optimization-based large neighborhood search are based on features that are common to most classes of IRP. The primary cut is based on a combination of the minimum number of pickups/deliveries considering storage capacity, production and consumption schedules, and vessel (or vehicle) capacities, which tend to be common elements of combined inventory and routing problems. The branching rule attempts to maintain a balanced branching tree via branching on the routes while ignoring the timing specific information of the route. The two ship (or two vehicle) subproblem-based large neighborhood search can be easily applied to any of class of routing problem with multiple vehicles treated individually. Using this modeling and algorithmic framework while accommodating many additional requirements that may be encountered in practice, a very practical and complex maritime inventory routing problem can be solved exceedingly efficiently. This illustrates the strength and wide applicability of the proposed modeling and algorithmic framework.

## Acknowledgments

We would like to thank Mike McDonald, Chad Reimann, Gary Kocis, and Phil Warrick for their contributions as part of the project team.

## References

[1] Al-Khayyal F, Hwang S-J. Inventory constrained maritime routing and scheduling for multi-commodity liquid bulk, part I: applications and model. European Journal of Operational Research 2007;176:106–30.
[2] Andersson H, Hoff A, Christiansen M, Hasle G, Løkketangen A. Industrial aspects and literature survey: combined inventory management and routing. Computers & Operations Research 2010;37:1515–36.
[3] Christiansen M. Decomposition of a combined inventory and time constrained ship routing problem. Transportation Science 1999;33(1):3–16.
[4] Christiansen M, Fagerholt K. Maritime inventory routing problems. In: Floudas CA, Pardalos PM, editors. Encyclopedia of Optimization. Springer; 2009. p. 1947–55.
[5] Christiansen M, Fagerholt K, Nygreen B, Ronen D. Maritime transportation. In: Barnhart C, Laporte G, editors. Transportation, Handbooks in Operations Research and Management Science. Amsterdam: Elsevier, North-Holland; 2007. p. 189–284.
[6] Christiansen M, Nygreen B. A method for solving ship routing problems with inventory constraints. Annals of Operations Research 1998;81:357–78.
[7] Christiansen M, Nygreen B. Modelling path flows for a combined ship routing and inventory management problem. Annals of Operations Research 1998;82:391–412.
[8] Cordeau J-F, Laporte G, Savelsbergh MWP, Vigo D. Vehicle routing. In: Barnhart C, Laporte G, editors. Transportation, Handbooks in Operations Research and Management Science. Amsterdam: Elsevier, North-Holland; 2007. p. 367–428.
[9] Furman KC, Song J-H, Kocis GR, McDonald MK, Warrick PH. Feedstock routing in the ExxonMobil Downstream Sector, Interfaces, in press, doi:10.1287/inte.1100.0508.
[10] Rothberg E. An evolutionary algorithm for polishing mixed integer programming solutions. INFORMS Journal on Computing 2007;19:534–41.
[11] Savelsbergh MWP, Song J-H. Inventory routing with continuous moves. Computers & Operations Research 2007;34(6):1744–63.
[12] Savelsbergh MWP, Song J-H. An optimization algorithm for the inventory routing problem with continuous moves. Computers & Operations Research 2008;35(7):2266–82.