**Submitted to *Transportation Science***
**manuscript (Please, provide the mansucript number!)**

# Bound Improvement for LNG Inventory Routing

Yufen Shao
ExxonMobil Upstream Research Company, yufen.shao@exxonmobil.com,

Kevin C. Furman
ExxonMobil Research and Engineering Company, kevin.c.furman@exxonmobil.com,

Vikas Goel
ExxonMobil Upstream Research Company, vikas.goel@exxonmobil.com,

Samid Hoda
previously at ExxonMobil Upstream Research Company,

Liquefied Natural Gas (LNG) is steadily becoming a common mode for commercializing natural gas. In this paper, we develop methods for improving both lower and upper bounds for a previously stated form of an LNG inventory routing problem. A Dantzig-Wolfe-based decomposition approach is developed for LNG inventory routing problem (LNG-IRP) attempting to overcome poor lower bounds. However, it fails to find feasible integer solutions that would provide good upper bounds. Advanced construction heuristics based on greedy randomized adaptive search procedure (GRASP) and rolling-time windows and several novel, MIP-based neighborhood search techniques are developed to achieve improved solutions in shorter computational time. The proposed algorithms are evaluated based on a set of realistic test instances that are very large relative to most of the problem instances seen in recent literature. Extensive computational results indicate that the proposed methods find improved lower bounds significantly faster than commercial solvers, and further, optimal or near optimal feasible solutions are achievable substantially faster than commercial optimization software as well as previously proposed heuristic methods.

*Key words*: maritime inventory routing; neighborhood search; GRASP; column generation; branch-and-price; liquefied natural gas

## 1. Introduction

Liquefied Natural Gas (LNG) is steadily becoming a common mode for commercializing natural gas. The LNG supply chain includes one or multiple production terminals where natural gas is produced, liquefied into LNG, and stored temporarily; a specialized ship fleet that loads and delivers LNG; and a set of regasification terminals where LNG is unloaded, stored temporarily, regasified and shipped to customers through pipelines.

Due to the historical illiquid nature of the LNG market, the LNG delivery contracts are structured such that an annual delivery schedule is agreed upon in advance by the LNG buyers and LNG producer every year. This annual delivery schedule is negotiated to best accommodate the expected requirements of each party for the planned year. The finalized schedule dictates when each cargo is to be delivered and by what ship. It is obvious that developing an optimized annual delivery schedule can be key to optimize the economics of an LNG project; moreover, it can also serve a key purpose from a supply chain design standpoint. Specifically, an optimized schedule for a given design can be used to evaluate the throughput and efficacy of that specific design. Due to the capital intensive nature of LNG projects, the optimal design of LNG supply chains is extremely important from a profitability perspective. In this paper, we address an LNG Inventory Routing Problem (LNG-IRP) where optimized ship schedules are developed for analyzing LNG supply chain design decisions.

In the research field of maritime transportation, most of the literature focuses on ship routing and scheduling problems (see Ronen [1983], Ronen [1993], Christiansen et al. [2004] and Christiansen et al. [2007]). The problem being considered here is closer to the more complex class of problems known as Maritime Inventory Routing Problems (MIRP). MIRP combines inventory management and ship routing, which are typically treated separately in industrial practice. A basic maritime inventory routing problem (see Christiansen and Fagerholt [2009]) involves the transportation of a single product from loading ports to unloading ports, with each port having a given inventory storage capacity, a pre-specified production or consumption rate, a restriction of number of visits to the port, and a limitation of quantity of product to be loaded or unloaded. A recent survey on the state-of-the-art in ship routing and scheduling research over the last decade is given by Christiansen et al. [2013].

LNG inventory routing can be considered as a special case of the MIRP. An excellent overview of the business cases and common characteristics for LNG inventory routing can be seen in Andersson et al. [2010]. The LNG inventory routing problem shares the fundamental properties of a single product MIRP with special features such as variable production and consumption rates, LNG specific contractual obligations, and berth constraints. The LNG-IRP seeks to generate schedules where each ship may make several voyages over a much longer than typical time horizon. Rakke et al. [2011] seems to be the first MIRP to address problems of developing annual delivery schedules for large LNG projects. Halvorsen-Weare and Fagerholt [2013] develops a decomposition-based heuristic scheme for an LNG-IRP problem similar to the one considered in this paper. Halvorsen-Weare et al. [2013] have recently studied the introduction of robustness strategies into the routing and scheduling methods for LNG-IRP.

In this paper, we develop improved lower bounds and heuristic solution methods for the LNG-IRP model proposed by Goel et al. [2012] which is based on the arc-flow formulation for maritime inventory routing model introduced by Song and Furman [2013]. The heuristic methods for LNG-IRP presented by Goel et al. [2012] were primarily based on very large neighborhood search heuristics (see Ahuja et al. [2002] for a general review). The 2-ship neighborhood operator used in that work was first introduced for MIRP by Song and Furman [2013], however the remainder of methods were novel. In this paper, several new heuristics for both constructing initial solutions and improving incumbent solutions are developed to solve this model more efficiently. New construction heuristics are based on rolling time methods and the greedy randomized adaptive search procedure (GRASP).

The rolling time heuristic is designed to solve problems with long time horizons via a sequence of subproblems, each of which is simplified to only consider a smaller time block with limited information of future time periods. It was first proposed by Baker [1977] for manufacturing scheduling problems, and has shown great success when applied by Stauffer and Liebling [1997], Mercé and Fontan [2003], Dimitriadis et al. [1997] and Araujo et al. [2007]. These algorithms, however, are specific for manufacturing problems and not generally applicable to MIRP. To the best of our knowledge, Bredström and Rönnqvist [2006] and Rakke et al. [2011] are the only works that apply rolling time heuristics to ship scheduling problems.

Bredström and Rönnqvist [2006] studies a combined supply chain and ship scheduling problem for pulp over a 40 day planning horizon. In the rolling time heuristic they developed, a subproblem includes an initial set of periods where schedules are frozen, a regular set of periods where schedules need to be solved in this subproblem, and a forecast extension where binary variables are relaxed. Computational results show that this algorithm works well for instances with realistic sizes. The differences between their problem and LNG-IRP studied in this paper is that, (a) our model is a arc flow model while theirs is a path flow model; (b) the cost structures are different, for example, we include penalty for unmet annual delivery requirements at each demand port; (c) we consider a large number of time periods (e.g. 365 days) while they consider a smaller number of time periods (e.g. 40 days); (d) our subproblem only includes the frozen periods and the regular periods, and we only lock in partial schedules after the subproblem is solved; and (e) we continue to improve our solution with other neighborhood searching procedures.

Rakke et al. [2011] developed a rolling time heuristic for LNG-IRP. In their problem, they consider a single production terminal with multiple types of LNG. Having a single production terminals allows for their MIP model formulation to use assignment variables to fully specify the decision for a ship to depart on a specific departure day and its return trip back to the production terminal. Their rolling time heuristic has the same structure as that used in Bredström and Rönnqvist [2006] where each subproblem includes three partitions of the time periods: a frozen section, a regular section, and a forecast section. However, they developed a scheme to improve the solution by searching in a limited area and adding additional constraints. In comparison, the LNG-IRP we consider allows multiple production and regasification terminals, however we do not consider maintenance events.

Unlike Bredström and Rönnqvist [2006] and Rakke et al. [2011] who use their original model with limited time horizon in their subproblem, we develop a specific model for the subproblem, in which all ships are traveling from production terminals and return back to production terminals. By enforcing this property, we know the exact available day for each ship at production terminals in each subproblem. From the producer's point of view, we can avoid creating a myopic schedule when considering a short time horizon which usually happens when there is a lack of information that some more suitable ships will be available deliver LNG soon.

GRASP was first introduced to solve combinatorial problems by Feo and Resende [1989]. It typically consists of iterations which include a construction of a greedy randomized solution and a local search to improve the solution. This algorithm has been applied successfully to many fields including routing (Kontoravdis and Bard [1995]) and transportation (Feo and Bard [1989] and Bard [1997]). A survey of GRASP can be found in Festa and Resende [2002].

To the best of our knowledge, we are the first to apply GRASP to the maritime inventory routing problems. In our method, the benefit function, which is used to construct the greedy randomized solution, includes the real cost at the leaving port and the estimated cost at the arriving port that cannot be avoided if a candidate is not selected. This benefit function structure has shown to be very efficient for our problem. Unlike most of other literature, adaptive rules are not used in our algorithm since our current setting is good enough for the instances that we tested.

Due to the difficulty of this class of problem, as well as those of similar classes, practical solution methodologies for addressing LNG-IRP involve constructing an initial solution heuristically and improving it with local search methods. The solutions obtained are obviously not guaranteed to be globally optimal. Both exploration on the lower bound, and improvements to the upper bounds are studied in this work to make progress toward better solutions with greater confidence in quality, and found in shorter time spans. Although applied to the examples of Goel et al. [2012], these methods should be generally applicable to many similar MIRP, especially those related to LNG.

The remainder of the paper is organized as follows: Section 2 gives a brief problem description; Section 3 addresses the model and formulation which are the same as in Goel et al. [2012]; Section 4 describes the decomposition procedure aiming to obtain good lower bounds; Section 5 presents the local searches for getting near-optimal solutions; Section 6 analyzes computational results; and Section 7 summarizes the accomplished work.

## 2. Problem Description

In this paper, we consider the problem addressed by Goel et al. [2012]. The problem under consideration is a combined inventory management and cargo routing problem for the delivery of LNG. Specifically, a problem instance includes a set of LNG production terminals with specified storage capacities. Each production terminal has a given production profile over the planning horizon, as well as a limited number of berths for ships to load cargoes. At the other end of the supply chain, there is a set of regasification (regas) terminals that have specified storage capacities and unloading berths. Each regas terminal has a specified contractual demand over the planning horizon and a specified profile based on which the LNG is regasified.

A problem instance includes a fleet of heterogeneous ships that may differ in terms of their LNG loading capacities and compatibility with the various terminals. We restrict our attention to full-load and full-discharge problems. In other words, we consider the scenario where LNG ships are filled to their maximum capacity prior to departing an LNG terminal and are completely discharged prior to departing regas terminals. This is common practice in the LNG industry for economic and safety reasons.

Similar to Goel et al. [2012], we focus on the need for developing optimized annual delivery schedules that will enable supply chain design analysis by quantifying the maximum throughput of a given design. Consequently, we focus on a year-long planning horizon with one day time discretization. Furthermore, the problem has the following goals: minimizing the under-delivery of LNG to each contract, minimizing lost production at stemming from running out of storage at production terminals, and minimizing stock-outs due to the lack of inventory for regasification at regas terminals. A more detailed version of the problem description can be found in Goel et al. [2012].

## 3. Model Formulation

In this section, we provide a brief description of the Mixed Integer Programming (MIP) formulation presented by Goel et al. [2012] for the above problem. The model in Goel et al. [2012] is based on a time-space network formulation proposed in Savelsbergh and Song [2008].

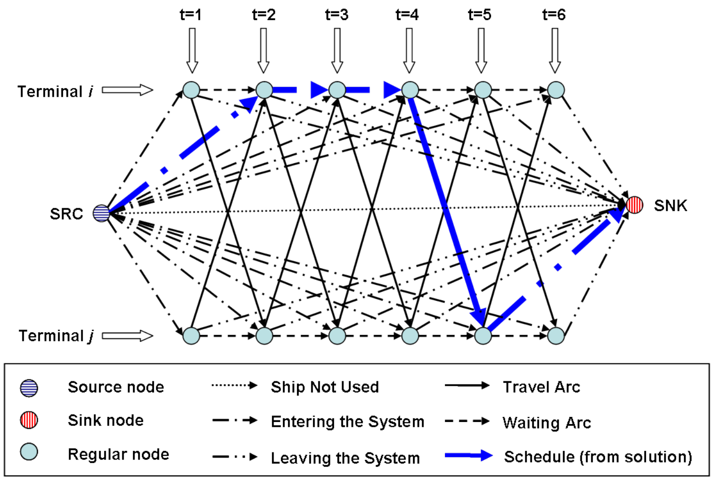

**Figure 1 Example of Time-Space Network Structure**

The time-space network in Figure 3 includes source ($SRC$) and sink ($SNK$) nodes to represent initial and final locations for the ships. The network also includes "regular" nodes to represent each terminal at a given time period. For each ship $v$, there are five types of arcs: an arc from the $SRC$ to $SNK$ node represents that a ship is not utilized; an arc from $SRC$ to a regular node represents the arrival of a ship to its initial destination; an arc from a regular node to $SNK$ represents the final departure of the ship; waiting arcs allow a ship to wait at a terminal without occupying a berth. These arcs connect regular nodes corresponding to the terminal at successive time periods. Finally, a travel arc from a regular node $n_1$ to a regular node $n_2$ represents the loading (or unloading) activity at the terminal corresponding to node $n_1$ immediately followed by travel to the node corresponding to node $n_2$. A solution for this LNG-IRP specifies a path for each ship within the network, together with the regas rates at each regas terminal during each time period.

The notations used in the MIP model developed by Goel et al. [2012] are presented at below.

| *Sets and Parameters* | |
|---|---|
| $L$ | Production terminals, $l \in L = \{1, 2, \dots, \lvert L \rvert\}$ |
| $R$ | Regas terminals, $r \in R = \{1, 2, \dots, \lvert R \rvert\}$ |
| $J$ | All terminals, $j \in J = L \cup R$ |
| $V$ | Vessels or Ships, $v \in V = \{1, 2, \dots, \lvert V \rvert\}$ |
| $T$ | Planning horizon, $t \in T = \{1, 2, \dots, \lvert T \rvert\}$ |
| $c_j$ | Storage capacity at terminal $j$ |
| $c_{v,j}$ | Volume loaded (discharged) by ship $v$ at production (regas) terminal $j$ |
| $D_r$ | Demand for LNG over planning horizon at regas terminal $r$ |
| $b_j$ | Number of berths at terminal $j$ |
| $p_{l,t}$ | Production rate at production terminal $l$ during time $t$ |
| $\underline{d}_{r,t}$ | Minimum regas rate of regas terminal $r$ during time $t$ |
| $\bar{d}_{r,t}$ | Maximum regas rate of regas terminal $r$ during time $t$ |
| $w_j$ | Penalty for lost production at production or stockout at regas terminal $j$ |
| $w_r^D$ | Penalty for unmet demand at regas terminal $r$ |
| *Network Elements* | |
| $N$ | Set of all nodes, $n \in N = (j, t)$ |
| $\bar{N}$ | Set of regular nodes. $\bar{N} = N \backslash \{SRC, SNK\}$ |
| $A_v^T$ | Set of travel arcs for ship $v$ |
| $A_v^S$ | Set of arcs from regular nodes to $SNK$ node for ship $v$ |
| $A_v$ | Set of all arcs for ship $v$ |
| $A$ | Set of arcs, $a \in A = \cup_v A_v$ |
| $\delta_n^+$ | Set of outgoing arcs from node $n$ |
| $\delta_n^-$ | Set of incoming arcs to node $n$ |
| *Decision Variables* | |
| $I_{j,t}$ | Inventory level at terminal $j$ at the end of time period $t$ |
| $d_{r,t}$ | Regas rate during time period $t$ at regas terminal $r$ |
| $o_{l,t}$ | Lost production at production terminal $l$ during time period $t$ |
| $s_{r,t}$ | Stockout at regas terminal $r$ during time period $t$ |
| $\delta_r^D$ | Unmet demand at regas terminal $r$ during planning horizon |
| $x_a^v$ | Binary variable for arc $a$ |

Now we present the MIP formulation which we will refer to as model (1) later.

$$min \sum w_l o_{l,t} + \sum w_r s_{r,t} + \sum w_r^D \delta_r^D \tag{1a}$$

$$s.t. \sum_{a \in A_v \cap \delta^+(n)} x_a^v - \sum_{a \in A_v \cap \delta^-(n)} x_a^v = 0, \qquad \forall v \in V, \forall n \in \bar{N}, \tag{1b}$$

$$\sum_{a \in A_v \cap \delta_{SRC}^+} x_a^v = 1, \qquad \forall v \in V, \tag{1c}$$

$$\sum_{a \in A_v \cap \delta_{SNK}^-} x_a^v = 1, \qquad \forall v \in V, \tag{1d}$$

$$I_{l,t} = I_{l,t-1} + p_{l,t} - \sum_v \sum_{a \in (A_v^T \cup A_v^S) \cap \delta^+(l,t)} c_{v,l} x_a^v - o_{l,t}, \qquad \forall l \in L, \forall t \in T, \tag{1e}$$

$$I_{r,t} = I_{r,t-1} - d_{r,t} + \sum_v \sum_{a \in (A_v^T \cup A_v^S) \cap \delta^+(r,t)} c_{v,r} x_a^v + s_{r,t}, \qquad \forall r \in R, \forall t \in T, \tag{1f}$$

$$\sum_v \sum_{a \in (A_v^T \cup A_v^S) \cap \delta^+(n)} x_a^v \le b_j, \qquad \forall n = (j, t) \in \bar{N}, \tag{1g}$$

$$\delta_r^D \geq D_r - \sum_t \sum_v \sum_{a \in (A_v^T \cup A_v^S) \cap \delta^+(r,t)} c_{v,r} x_a^v, \qquad \forall r \in R, \tag{1h}$$

$$\underline{d}_{r,t} \leq d_{r,t} \leq \overline{d}_{r,t}, \qquad \forall r \in R, \forall t \in T, \tag{1i}$$

$$0 \leq I_{j,t} \leq c_j, \qquad \forall j \in J, \forall t \in T, \tag{1j}$$

$$o_{l,t} \geq 0, \forall l \in L, \qquad t \in T, \tag{1k}$$

$$d_{r,t} \geq 0, \forall r \in R, \qquad t \in T, \tag{1l}$$

$$\delta_r^D \geq 0, \qquad r \in R, \tag{1m}$$

$$x_a^v \in \{0,1\}, \qquad \forall v \in V, a \in V_a \tag{1n}$$

The objective (1a) is to minimize the sum of weighted lost production, stockout, and unmet demand. Constraints (1b)-(1d) are the flow conservation constraints for the network. Constraints (1e) and (1f) ensure the inventory balance at the production terminals and regas terminals, respectively. Lost production and stockout variables provide slack for these constraints. The berth limits are enforced by constraints (1g). Constraint (1h), together with the objective function, takes care of the unmet demand calculation. Since the objective function seeks to minimize the total unmet demand, this constraint will be tight in every optimal solution that has a positive unmet demand. The final set of constraints (1i)-(1n) ensures that all the variables satisfy their specific bounds.

## 4. Decomposition Procedure–Improving Lower Bounds

Due to the complex nature of the above problem, commercially available MIP solvers cannot solve real world problems within a reasonable amount of time. The solution methodology developed by Goel et al. [2012] is practical, however little effort is focused on proving optimality or improving the lower bounds. In this section, we discuss the Dantzig-Wolfe-based decomposition approaches that we developed for LNG-IRP attempting to overcome this obstacle.

Our decomposition scheme is a branch and price (B&P) algorithm that applies branch and bound to find integral solutions while using Dantzig-Wolfe (DW) decomposition to solve the linear programming (LP) relaxation of each node. This branch and price framework has been studied and applied for MIRP by many researchers (see Christiansen [1999], Grønhaug et al. [2010], Engineer et al. [2012] and Hewitt et al. [2013]). In all of these papers, inventory capacities are treated as hard constraints which, together with other side hard constraints, are the basis of their valid cut generation or algorithm development. However, these cuts are not appropriate for our LNG-IRP since the inventory capacities are soft constraints with penalties in the objective function in this model. Furthermore, the instances that were tested in these paper primarily involve only 5-10 vessels and/or consider 60 or fewer time periods, thus successful approaches to solving these relatively small-sized instances have little value in approaching large, practical test cases with up to 100 vessels and 365 time periods. Nevertheless, these studies show that their optimality gaps can be closed quickly by using B&P. Thus, we attempt to develop our own B&P scheme with the aim of obtaining good lower bounds quickly, even if we cannot expect to prove optimality for large sized instances.

Our DW master problem is based on the management constraints (1e) - (1h). Each column represents the schedule for one ship over the entire planning horizon. When constraints (1e) - (1h) are removed from model (1), the resultant problem decomposes by ships into $|V|$ subproblems, each of which is a pure minimum cost flow network problem and can be solved in polynomial time in turn to produce additional master problem columns.

### 4.1. Master Problem (MP)

First we build the master problem by taking the management constraints (1e) - (1h) as linking constraints and converting the arc flow model a path flow model. The following notation is defined for developing the master problem.

| *Additional Notations for MP* | |
|---|---|
| $K$ | Set of all columns |
| $X^v_{k,a}$ | Binary parameter to indicate if arc $a$ is selected in $k^{th}$ column for ship $v$ |
| $\lambda^v_k$ | Binary variable to select the $k^{th}$ column for ship $v$ |

The Master Problem (MP) is defined as follows.

$$\begin{aligned}
min\ & \sum w_l o_{l,t} + \sum w_r s_{r,t} + \sum w^D_r \delta^D_r && && \text{(2a)}\\
s.t.\ & I_{l,t} + o_{l,t} + \sum_v \{c_{v,l} \sum_{a\in(A^T_v\cup A^S_v)\cap\delta^+(l,t)} \sum_k X^v_{k,a}\lambda^v_k\} - I_{l,t-1} = p_{l,t}, && \forall l\in L, \forall t\in T, && \text{(2b)}\\
& I_{r,t} - s_{r,t} - \sum_v \{c_{v,r} \sum_{a\in(A^T_v\cup A^S_v)\cap\delta^+(r,t)} \sum_k X^v_{k,a}\lambda^v_k\} - I_{r,t-1} = -d_{r,t}, && \forall r\in R, \forall t\in T, && \text{(2c)}\\
& \sum_v \sum_{a\in(A^T_v\cup A^S_v)\cap\delta^+(j,t)} \sum_k X^v_{k,a}\lambda^v_k \le b_j, && \forall j\in J, \forall t\in T, && \text{(2d)}\\
& \delta^D_r + \sum_t \sum_v \{c_{v,r} \sum_{a\in(A^T_v\cup A^S_v)\cap\delta^+(r,t)} \sum_k X^v_{k,a}\lambda^v_k\} \ge D_r, && \forall r\in R, && \text{(2e)}\\
& \sum_k \lambda^v_k = 1, && \forall v\in V, && \text{(2f)}\\
& \lambda^v_k \in \{0,1\}, \forall k\in K, \forall v\in V, && && \text{(2g)}\\
& \text{and constraints } (1i)-(1m) && && \text{(2h)}
\end{aligned}$$

This model is the path flow network version of model (1). It keeps the inventory constraints (2b)-(2c), berth limit constraints (2d) and unmet demand constraints (2e). In addition to these original constraints, constraint (2f) selects one path for each ship.

When we implement DW decomposition algorithm, we only include a restricted number of columns (paths) in MP and solve its corresponding linear relaxation. This problem is referred to as the Restricted Linear Master Problem (RLMP). The dual solution of RLMP is used by the pricing subproblems to generate new columns with negative reduced costs.

## 4.2. Pricing Subproblem (SP)

The purpose of the pricing subproblem in DW decomposition is to generate better columns with negative reduced costs for the ships with respect to the current MP solution. When MP is resolved after adding these new columns, the new solution should be improved compared to the current solution. Such columns can be found by solving the pricing subproblems if they exist. To describe the subproblem, let $\alpha_{j,t}$ be the free dual variable for inventory constraints (2b) and (2c), $\beta_{j,t}$ be the non-positive dual variable for berth constraints (2d), $\gamma_r$ be the non-negative dual variable for unmet demand constraints (2e), and $\mu_v$ be the free dual variable for path selection constraints (2f). Now we can formulate the pricing subproblem as follows for ship $v$ as follows.

$$\begin{aligned}
min\ & -\sum_{l,t}\{\alpha_{l,t} c_{v,l} \sum_{a\in(A^T_v\cup A^S_v)\cap\delta^+(l,t)} x^v_a\}\\
& + \sum_{r,t}\{\alpha_{r,t} c_{v,r} \sum_{a\in(A^T_v\cup A^S_v)\cap\delta^+(r,t)} x^v_a\}\\
& - \sum_{j,t}\{\beta_{j,t} \sum_{a\in(A^T_v\cup A^S_v)\cap\delta^+(j,t)} x^v_a\} && \text{(3a)}\\
& - \sum_r\{\gamma_r c_{v,r} \sum_t \sum_{a\in(A^T_v\cup A^S_v)\cap\delta^+(r,t)} x^v_a\}
\end{aligned}$$

$$
\begin{aligned}
&-\mu_v \\
s.t.\quad &(1b)-(1d) && \text{(3b)}
\end{aligned}
$$

Notice that problem (3) is a pure minimum cost flow problem; thus solving it to optimality can be done very efficiently in polynomial time. However, the associated lower bound that is achieved in the root node is not promising since it is equivalent to LP. In order to improve the lower bound quality, one method involves aggregating the ships into several groups by designing specific rules, and decomposing the problem by ship groups while appropriately adapting some of the management constraints (1e)-(1h) in the pricing subproblem. Nevertheless, this type of subproblem is usually very difficult to solve and can be as difficult as the original problem (1) with respect to complexity. When designing DW algorithms, a tradeoff between solution quality and computational difficulty must be balanced.

## 4.3. Branch and Price Algorithm

In this section, components of our B&P implementation are discussed. These include the initialization of the master problem, stabilization of column generation, the study of valid cuts, and branching.

**4.3.1. Initialization of MP** As we mention above, the formulation of model (1) is very loose which makes the master problem initialization trivial: the dummy paths from $SRC$ to $SNK$ for all ships are feasible to the master problem (2) initially. When branching cuts are added, however, the phase 1 problem is solved first to verify the feasibility of the branch and bound node and initialize the master problem as bellow.

$$
\begin{aligned}
\min\ & |V| - \sum_{v\in V}\sum_{k\in K} \lambda_k^v && \text{(4a)}\\
\text{s.t.}\ & \sum_{k} \lambda_k^v \le 1, \forall v \in V, && \text{(4b)}\\
& \text{constraints (2b)-(2e) and (2g)-(2h),} && \text{(4c)}\\
& \text{constraints associated to all cuts} && \text{(4d)}
\end{aligned}
$$

In this problem, the column selection constraints (2f) have been relaxed into constraints (4b). If constraints (2f) are violated, the objective in (4a) will return to a positive value. Thus, for any infeasible MP, its phase 1 problem is either infeasible or returns a positive value. Meanwhile, for any feasible MP, its phase 1 problem must return zero. Therefore, the feasibility of MP can be verified quickly and thus MP can be initialized successfully.

**4.3.2. Stabilization technique** Due to the high level of degeneracy of the problem (1), our decomposition procedure encounters the typical tailing-off effect of DW decomposition. In this phenomenon, MP objective value decreases rapidly in early iterations, but it takes a long time to converge. In order to overcome this deficiency, du Merle et al. [1999] developed an effective scheme, known as the stabilization technique, which uses two surplus and slack variables ($\omega^-$ and $\omega^+$) in an effort to reduce degeneracy. The stabilized version of model (2) is as follows.

$$
\begin{aligned}
\min\ & \text{objective (2a)} + \sum_{v} \{\sigma_+^v w_+^v - \sigma_-^v w_-^v\} && \text{(5a)}\\
\text{s.t.}\ & \text{constraints}(2b)-(2e), && \text{(5b)}\\
& \text{bounds}(2g)-(2h), && \text{(5c)}\\
& \sum_{k\in K} \lambda_k^v + w_+^v - w_-^v = 1, \forall v \in V, && \text{(5d)}\\
& 0 \le w_+^v \le \epsilon_+^v, 0 \le w_-^v \le \epsilon_-^v, \forall v \in V && \text{(5e)}
\end{aligned}
$$

Here, the parameters $\epsilon_+^v$ and $\epsilon_-^v$ are very small positive values so that the value of (5a) is not too far off from that of (2a). The parameters $\sigma_+^v$ and $\sigma_-^v$ are estimators of the dual variables $\mu_v$ with $\sigma_-^v < \sigma_+^v$ to allow variants. The solution of problem (5) is feasible to the original master problem (2) if $w_+^v = w_-^v = 0$.

This stabilization technique is known to be parameter dependent and the updating strategies for the parameters must ensure finite convergence. In the computations, we assume that the components of $\omega^-$ and $\omega^+$ vanish and hence (5d) is satisfied when their values are less that a ″small″ number $\epsilon$. In the implementation, we initialize $\epsilon_+^v$ to be less than 1 and set $\epsilon_-^v$ to be a very small random number in the interval $(0,\ \epsilon]$ at each iteration. If the subproblems do not return any columns with negative reduced costs, then we decrease $\epsilon_+^v$ at a rate $\eta \in (0,1)$, i.e., $\epsilon_+^v \leftarrow \eta\epsilon_+^v$; otherwise, no changes are made to $\epsilon_+^v$. The constantly changing random value of $\epsilon_-^v$ provides a small perturbation to the problem even when all other parameters remain fixed. The parameters $\sigma_+^v$ and $\sigma_-^v$ are estimated at the root node, and then set as the optimal dual solution of their parent nodes at other branch and bound nodes.

**4.3.3. Study of valid cuts** In our procedure, efficient valid cuts are not discovered due to the nature of model (1). Among all management constraints in model (1), only the berth constraints (1g) are hard constraints while other constraints (1e), (1f) and (1h), have slack variables which are penalized in the objective function. Compared to the inventory constraints (1e)-(1f), however, berth constraints (1g) are usually not dominant which makes the valid inequalities associated to berth constraints weak. Thus, other than branching cuts, cuts are not developed in our approach.

**4.3.4. Branching strategies** During the branch and bound search, both the Depth-First-Search (DFS) and Best-Bound-First (BBF) strategies are implemented and tested for node selections. Computational results show that DFS performs substantially worse than BBF. Given a good solution obtained from heuristic methods in Goel et al. [2012], DFS fails to find a better solution within an acceptable time, and the gap remains flat since the lower bound does not improve either. This result is intuitive since it is usually very hard to get even one integral solution by using branch and bound scheme for LNG-IRP due to the high level of degeneracy of the problem. However, BBF can reduce the gap significantly for some instances within an acceptable time as we observe the global lower bound is increased significantly. This observation increases our confidence in the heuristic solutions.

Regarding branching strategies, three categories have been developed and tested, including 14 rules in total. The first two categories aim to fix several variables simultaneously instead of one variable at a time as in the last category. Research study shows that fixing several variables simultaneously is usually more effective than fixing one variable at a time when 0-1 MIP is solved (see Wolsey [1998]).

The first category adopts Ryan-Foster branching (Ryan and Foster [1981]) for two scenarios. In one of the scenarios, a ship $v$ selects two or more schedules; among them, one schedule has traveling arcs from both day $t_1$ and day $t_2$; another schedule has only one traveling arc from day $t_1$ or day $t_2$. In one branch, ship $v$ has a traveling arc from day $d_1$ if and only if it has a traveling arc from day $d_2$; in the other branch, ship $v$ has one and only one traveling arc from day $t_1$ or $t_2$. In the second scenario, a ship $v$ also selects two or more schedules; among them, one schedule includes both arc $a_1$ and arc $a_2$; another schedule has only one arc between arc $a_1$ and arc $a_2$. In one branch, ship $v$ selects arc $a_1$ if and only if arc $a_2$ is selected; in the other branch, ship $v$ selects one and only one arc between arc $a_1$ and $a_2$.

The second category is associated to the fractional number of voyages. For example, Letting $y_t$ be the number of voyages that start on time $t$, its value $Y_t$ is fractional in the current LP solution, then the node can be branched into two partitions: one has $y_t \leq \lfloor Y_t \rfloor$, and the other has $y_t \geq \lceil Y_t \rceil$. The following are the rules included in this category: the total number of voyages, number of voyages starting on time $t$, number of voyages leaving from terminal $j$, number of voyages departing from a node $(j,t)$, number of voyages executed by ship $v$ or a ship group $g$, number of voyages for ship $v$ starting from terminal $j$, number of voyages entering the system through regular nodes, and number of voyages exiting the system through regular nodes etc.

The third category is the default strategy that branches on the most fractional arc in the current solution.

Performance of these branching rules differ significantly depending on individual cases. Among them, after some computational experimentation, we determined that branching on number of voyages starting from a terminal, number of voyages executed by a ship, together with the default rule, give the best bounds. The results in Section 6 are limited to these branching strategies.

## 5. Local Search–Improving Upper Bounds

As discussed in detail later in Section 6, although the Branch and Price algorithm developed in Section 4 significantly improves lower bounds for some of the test cases defined by Goel et al. [2012], however it fails to find feasible integer solutions that would provide upper bounds. This is not an unexpected result due to the known difficulty of this LNG-IRP. In order to further close the optimality gap, we propose several new heuristic approaches in order to improve upper bounds.

Like most heuristic methods, our heuristic scheme includes two basic phases – construction and improvement. In Phase I, two approaches are developed to construct good feasible solutions. In Phase II, different local searches are designed and implemented in order to improve the solutions. Among all of these construction approaches and local searches, the best strategy, which balances the solution quality and running time, is discovered. Now each phase is described.

### 5.1. Construction Heuristics

In previous efforts (Goel et al. [2012]) for solving LNG-IRP, a simple rolling time greedy algorithm was used to generate an initial solution. Construction of initial solutions has been extended by developing two more advanced techniques: round-trip rolling time algorithm and GRASP.

**5.1.1. Round-Trip Rolling Time Algorithm** In this algorithm, a feasible solution is constructed by solving a sequence of optimization subproblems, each of which corresponds to one day. In the subproblem, there is a focus on the production terminals and the algorithm attempts to schedule a round-trip for each considered ship. The definition of ″round-trip″ is described at below.

**Round-trip** A round-trip for a ship is defined as a sequence of nodes that the ship visits, among which only the first and last are production terminals. For example, if a ship visits $(l,t)$, $(r,t+3)$, $(r,t+4)$ and $(l,t+7)$ in a sequence, it has a round-trip. A round-trip is denoted as $g=[(l_0,t_0),(r,t_0^r),(r,t_1^r),(l_1,t_1)]$; it defines the sequence such that a ship loads LNG and leaves production terminal $l_0$ on day $t_0$, arrives at regas terminal $r$ on day $t_0^r$, discharges gas and leaves the regas terminal $r$ on day $t_1^r$, and arrives at production terminal $l_1$ on day $t_1$. If the ship waits at the production terminal, it has a waiting round-trip $g=[(l_0,t_0),(l_1,t_1)]$ with $l_0=l_1$.

We now start to describe our rolling time algorithm. Given a day $t^*$ and a length of time periods $\Delta$, the corresponding day $t^*$ subproblem is created as follows. All round-trips that are selected in the previous iteration and begin earlier than day $t^*$ are fixed; all ships whose next available time is later than day $t^*+\Delta$ are ignored; one round-trip will be assigned for each ship that is available during $[t^*,\ t^*+\Delta]$. After the subproblem is solved, each considered ship will either wait at a production terminal for one day, or visit a regas terminal, wait there for several days, and then return to a production terminal. The maximum number of days (denoted as $W_{max}$) that a ship is allowed to wait at regas terminals is one of the key parameters in this algorithm since it will affect the size of the subproblem and the quality of the solution. Nevertheless, the ship is forced to return as early as possible by penalizing the waiting days. After updating the production/regas, inventory and berth usage at each terminal, we can then create and solve the subproblem for the next subproblem $t^*+1$. A feasible initial solution is constructed when the last day $|T|$ subproblem is solved. The pseudocode for this algorithm appears as Algorithm 1.

**Algorithm 1** Round-Trip Rolling Time Algorithm

```
while $t^* \leq |T|$ do
    create subproblem for day $t^*$;
    solve subproblem for day $t^*$;
    accept the round-trips that begins on day $t^*$;
    update production/regas, inventory and berth usage at each terminal;
    set $t^* \leftarrow t^* + 1$;
end while
```

In addition to $t^*$, $\Delta$ and $W_{max}$, the following terms are used in defining the rolling time algorithm.

| *Sets for Rolling Time Algorithm* | |
|---|---|
| $V_{t^*}$ | set of available ships during $[t^*, t^*+\Delta]$ |
| $G$ | set of all round-trips, $g \in G$ |
| $G_v$ | set of round-trips for ship $v$, $\forall v \in V_{t^*}$ |
| $G_{t^*}$ | set of round-trips associated to subproblem $t^*$ |
| $G_{j,t}$ | set of round-trips leaving terminal $j$ on day $t$ |
| *Parameters for Rolling Time Algorithm* | |
| $b^*_{j,t}$ | Remained available number of berths at terminal $j$ on day $t$ for subproblem $t^*$ |
| $c^*_{j,t}$ | Remained available inventory capacity at terminal $j$ on day $t$ for subproblem $t^*$ |
| $p^*_{l,t}$ | Remained production rate at terminal $l$ during day $t$ for subproblem $t^*$ |
| $\underline{d}^*_{r,t}$ | Remained minimum regas rate of regas terminal $r$ during day $t$ for subproblem $t^*$ |
| $\overline{d}^*_{r,t}$ | Remained maximum regas rate of regas terminal $r$ during day $t$ for subproblem $t^*$ |
| $w_g$ | number of waiting days at regas terminal for a round-trip $g$ |
| *Variables for Rolling Time Algorithm* | |
| $x_{v,g}$ | Binary variable to select the round-trip $g$ for ship $v$ |

The subproblem corresponding day $t^*$ is defined as follows.

$$\min \sum_{t^* \leq t \leq t^*+\Delta} w_l o_{l,t} + \sum_{t \geq t^*} w_r s_{r,t} + \sum_{v \in V_{t^*}} \sum_{g \in G_{t^*} \cap G_v} \epsilon w_g x_{v,g} \tag{6a}$$

$$\sum_{g \in G_{t^*} \cap G_v} x_{v,g} = 1, \forall v \in V_{t^*}, \tag{6b}$$

$$I_{l,t} + o_{l,t} + \sum_{v \in V_{t^*}} \{c_{v,l} \sum_{g \in G_{t^*} \cap G_v \cap G_{l,t}} x_{v,g}\} - I_{l,t-1} = p^*_{l,t}, \forall l \in L, t^* \leq t \leq t^*+\Delta \tag{6c}$$

$$I_{r,t} - s_{r,t} - \sum_{v \in V_{t^*}} \{c_{v,r} \sum_{g \in G_{t^*} \cap G_v \cap G_{r,t}} x_{v,g}\} - I_{r,t-1} = -d_{r,t}, \forall r \in R, \forall t \geq t^* \tag{6d}$$

$$\sum_{v \in V_{t^*}} \sum_{g \in G_{t^*} \cap G_v \cap G_{j,t}} x_{v,g} \leq b^*_{j,t}, \forall j \in J, \forall t \geq t^* \tag{6e}$$

$$0 \leq I_{l,t} \leq c^*_{l,t}, o_{l,t} \geq 0, \forall l \in L, \forall t \geq t^* \tag{6f}$$

$$0 \leq I_{r,t} \leq c^*_{r,t}, s_{r,t} \geq 0, \underline{d}^*_{r,t} \leq d_{r,t} \leq \overline{d}^*_{r,t}, \forall r \in R, \forall t \geq t^* \tag{6g}$$

$$x_{v,g} \in \{0,1\}, \forall v \in V_{t^*}, \forall g \in G_{t^*} \cap G_v \tag{6h}$$

This model (6) is analogous to the full arc flow model (1), however, since it is greatly simplified with many fewer arcs and objective function terms, it can be solved quite efficiently by a commercial

MIP solver (e.g. CPLEX). In this model, the objective function (6a) is to minimize the production lost during $[t^*, t^* + \Delta]$, stockouts during $[t^*, |T|]$ and a penalty for waiting days at regas terminals. For each available ship, a round-trip is guaranteed through constraints (6b); the inventory and berth constraints are kept as in the original model through (6c)-(6e) with updated information for production/regas, inventory and berth usage associated with day $t^*$. However, the unmet demand constraints (1h) are not considered here since the unmet demand volume can only be calculated at the end of the time horizon.

Depending on the value of $\Delta$, two variants of this algorithm are posed: 1) a ***one-day*** variant in which we only consider the ships that are available on the current day $t^*$, i.e., $\Delta = 0$; and 2) a ***multi-day*** variant in which we consider ships that will be available over multiple days, i.e., $\Delta > 0$. When the ″multi-day″ variant is considered, it is apparent that the subproblem will be more difficult due enlarged problem size; however, a better solution quality is also expected. We will report the results for both of these two variants in Section 6.

**5.1.2. Greedy Randomized Adaptive Search Procedure (GRASP)** In this algorithm, the solution is also constructed day by day. Given a day $t^*$, we make a sequence of decisions, each of which corresponds to a selection of a new voyage(arc) for some ship that is available on day $t^*$. To select a voyage, we need to create the candidate list first, then calculate a benefit value for each candidate, and finally select one candidate at random based on a probability distribution which is associated to the benefit values. If a waiting arc is selected, the associated ship is forced to wait at its current location for another day and becomes available on day $t^* + 1$; otherwise, the ship will be sent to another terminal and will become available after a certain number of days depending on the traveling time. Along with this procedure, we ensure that a ship can only load or discharge a cargo when there exists an available berth and ample production inventory or discharge tank capacity. The pseudocode for the GRASP is exhibited as Algorithm 2.

**Algorithm 2** GRASP

```
while $t \leq |T|$ do
    while available ships exist on day $t^*$ do
        create candidate list of voyages starting on day $t^*$ for all available ships;
        calculate benefit values for these candidates;
        assign probability distribution based on the benefit values;
        select one candidate at random;
        remove the selected ship from the set of available ships on day $t^*$;
        update the next available location and time for the selected ship;
        update inventory and berth usage;
    end while
    set $t^* \leftarrow t^* + 1$;
end while
```

The terms which are used in describing GRASP are listed as follows.

| *Notations for GRASP* | |
|---|---|
| $V_{t^*}$ | set of available ships on day $t^*$ |
| $A_{v,t^*}$ | set of compatible arcs starting on day $t^*$ for ship $v$ |
| $CL_{t^*}$ | candidate list of voyages on day $t^*$, $CL_{t^*} = \{(v,a) \mid v \in V_{t^*}, a \in A_{v,t^*}\}$ |
| $j_v^{curr}$ | the current terminal of ship $v$ |
| $j_v^{arr}$ | the potential arrival terminal of ship $v$ |
| $t_v^{arr}$ | the potential arrival time of ship $v$ |
| $ben_{v,a}$ | the benefit value to select $(v,a)$ |
| $ben_{v,a}^{dep}$ | the departure benefit value to select $(v,a)$ |
| $ben_{v,a}^{arr}$ | the arrival benefit value to select $(v,a)$ |
| $\Delta^*$ | the number of days that we consider for loss estimation |

Now we start to describe the rules for picking a voyage $(v,a)$ from the candidate list $CL_{t^*}$. Given the current location $j_v^{curr}$ for ship $v \in V_{t^*}$, we can also denote the set of compatible arcs $A_{v,t^*} = \{(n_v^{curr}, n_v^{arr}) \mid n_v^{curr} = (j_v^{curr}, t^*), n_v^{arr} = (j_v^{arr}, t_v^{arr})\}$ with $n_v^{curr}$ as the current node and $n_v^{arr}$ as the potential arrival node for ship $v$. The benefit value $ben_{v,a}$ for each candidate pair $(v,a)$ is then calculated as follows.

If ship $v$ waits at terminal $j_v^{curr}$, i.e., $n_v^{arr} = (j_v^{curr}, t^*+1)$, we set $ben_{v,a} = 1$ for the associated candidate voyage $(v,a)$.

If ship $v$ departs from terminal $j_v^{curr}$, i.e., $j_v^{arr} \neq j_v^{curr}$, it essentially needs to load (discharge) LNG at the terminal $j_v^{curr}$ on day $t^*$ if $j_v^{curr}$ is a production (regas) terminal, and then it will potentially discharge (load) LNG at the terminal $j_v^{arr}$ on or after day $t_v^{arr}$. Whether or not the associated voyage $(v,a)$ can be selected depends on the resource availability at the terminal $j_v^{curr}$ on day $t^*$. If the inventory level is not high enough to load the ship $v$ or there is no available berth, then we set $ben_{v,a} = 0$ directly; otherwise, the benefit value $ben_{v,a}$ consists of two parts: departure benefit $ben_{v,a}^{dep}$ and arrival benefit $ben_{v,a}^{arr}$, i.e., $ben_{v,a} = ben_{v,a}^{dep} + ben_{v,a}^{arr}$.

**Departure benefit** If ship $v$ departs terminal $j_v^{curr}$ on day $t^*$, the departure benefit $ben_{v,a}^{dep}$ is defined as the amount of loss at terminal $j_v^{curr}$ during $[t^*, t^*+\Delta^*]$ that cannot be avoided if the associated voyage $(v,a)$ is not selected.

In the case that another ship $u$ becomes available at terminal $j_v^{curr}$ on day $t$ with $t \geq t^*$, we assume that the ship $u$ will take over the responsibility of ship $v$ for the loss after day $t$, i.e., we only estimate the loss during period $[t^*, \min\{t^*+\Delta^*, t-1\}]$ as the departure benefit for ship $v$.

Let $\delta^*$ be the updated number of days for loss estimation which is adjusted according to the rule above. It is not hard to conclude that, ship $v$ is the only available ship at terminal $j_v^{curr}$ during $[t^*, t^*+\delta^*]$; therefore there may exist some loss which can be avoided only if ship $v$ departs terminal $j_v^{curr}$ on day $t^*$. Depending on whether terminal $j_v^{curr}$ is a production terminal or regas terminal, the loss can be lost production or stockout. If terminal $j_v^{curr}$ is a production terminal, it is the lost production during $[t^*, t^*+\delta^*]$ which can be calculated based on the inventory level $I_{l,t^*-1}$ and the fixed production rate $p_{l,t} : t = t^*, \ldots, t^*+\delta^*$; here $l = j_v^{curr}$. If terminal $j_v^{curr}$ is a regas terminal, it is the stockout during $[t^*, t^*+\delta^*]$ which can be calculated based on the inventory level $I_{r,t^*-1}$ and the maximum regasification rate $\overline{d}_{r,t} : t = t^*, \ldots, t^*+\delta^*$; here $r = j_v^{curr}$. This loss is then taken as the departure benefit of the associated voyage $(v,a)$.

The pseudocode for departure benefit calculation is shown at below.

```
GRASP – Calculating Departure Benefit
if $j_v^{arr} = j_v^{curr}$ then
    $ben_{v,a}^{dep} = 1$
else if $j_v^{curr} \in L$ and $j_v^{arr} \in R$ then
    — departing from a production terminal —
```

```
        $totalStorage = c_{j_v^{curr}} - I_{tr_v^{curr}, t^*-1}$
        $totalProduction = 0$
        for $t = t^* \to t^* + \Delta^*$ do
            if there are other ships available on day $t$ then
                break
            else
                $totalProduction += p_{j_v^{curr}, t}$
            end if
        end for
        $totalLoss = \max\{0, totalProduction - totalStorage\}$
        $ben_{v,a}^{dep} = \min\{c_{v, j_v^{curr}}, totalLoss\}$
    else if $j_v^{curr} \in R$ and $j_v^{arr} \in L$ then
        — departing from a regas terminal —
        $totalInventory = I_{j_v^{curr}, t^*-1}$
        $totalDemand = 0$
        for $t = t^* \to t^* + \Delta^*$ do
            if there are other ships available on day $d$ then
                break
            else
                $totalDemand += \overline{d}_{j_v^{curr}, t}$
            end if
        end for
        $totalLoss = \max\{0, totalDemand - totalInventory\}$
        $ben_{v,a}^{dep} = \min\{c_{v, j_v^{curr}}, totalLoss\}$
    end if
```

**Arrival benefit** If ship $v$ arrives at terminal $j_v^{arr}$ on day $t_v^{arr}$, the arrival benefit $ben_{v,a}^{arr}$ is defined as the amount of loss at terminal $j_v^{arr}$ during $[t_v^{arr}, t_v^{arr} + \Delta^*]$ that cannot be avoided if the voyage $(v, a)$ is not selected.

This definition is similar as that of departure benefit except for the time period $[t_v^{arr}, t_v^{arr} + \Delta^*]$. For the ships which are available at terminal $j_v^{arr}$ during $[t^*, t_v^{arr} - 1]$, we assume that they will leave the terminal as soon as possible. Those ships that cannot leave before day $t_v^{arr}$ become available on day $t_v^{arr}$.

Similar as the definition of departure benefit, whenever a ship $u$ becomes available at terminal $j_v^{arr}$ on day $t \in [t_v^{arr}, t_v^{arr} + \Delta^*]$, we assume that the ship $u$ will take over ship $v$'s responsibility of the subsequent loss. Thus we only need to calculate the loss during $[t_v^{arr}, \min\{t_v^{arr} + \Delta^*, t - 1\}]$ for ship $v$; this amount of loss is taken as the arrival benefit since it is unavoidable if the voyage $(v, a)$ is not selected.

The pseudocode for arrival benefit calculation at produciton terminals is shown as follows. The calculation for regas terminals is similar.

GRASP – Calculating Arrival Benefit at Production Terminals

```
    if $j_v^{arr} = j_v^{curr}$ then
        $ben_{v,a}^{arr} = 0$
    else
        $totalInventory = I_{j_v^{curr}, t^*-1}$
        for $t = t^* \to t_v^{arr} - 1$ do
            $totalInventory += p_{j_v^{curr}, t}$
            for all available ship $u$ at terminal $j_v^{curr}$ on day $t$ do
                if ship $u$ can be loaded then
```

```
                    load ship $u$
                    update inventory level and berth usage at $j_v^{curr}$
                    remove ship $u$ from the set of available ships at terminal $j_v^{curr}$
                    $totalInventory -= c_{v,j_v^{curr}}$
                else
                    ship $u$ becomes available at terminal $j_v^{curr}$ on day $t+1$
                end if
            end for
            $totalInventory = \max\{c_{j_v^{curr}}, totalInventory\}$
        end for
        $totalStorage = c_{j_v^{curr}} - totalInventory$
        $totalProduction = 0$
        for $t = t_v^{arr} \to t_v^{arr} + \Delta^*$ do
            if other ship available on day $t$ then
                break
            else
                $totalProduction += p_{j_v^{curr},t}$
            end if
        end for
        $totalLoss = \max\{0, totalProduction - totalStorage\}$
        $ben_{v,a}^{arr} = \min\{c_{v,j_v^{curr}}, totalLoss\}$
    end if
```

**Probabilities** After the benefit value is calculated for each candidate voyage, we simply set $ben_{v,a}/sum_{(v',a')\in CL_{t^*}} ben_{v',a'}$ as the probability for voyage $(v,a) \in CL_{t^*}$, and then select a voyage randomly according to this distribution.

Depending on the number of steps to select a voyage, there are also two variants of GRASP: 1) a ***one-step*** variant which selects a voyage directly without distinguishing ships and destinations separately; 2) a ***two-step*** variant which selects a ship first and then decide where the ship should go. The one-step variant is the algorithm as described above. In the two-step variant, a ship is selected randomly based on its flexibility, and then its destination is picked randomly based on the benefit values calculated by using the same rules as described for the one-step variant. Results for both of these two variants are reported in Section 6.

## 5.2. Local Search

After constructing an initial solution, there are a number of ways to improve upon that solution. Neighborhood search improves solutions iteratively by searching the best neighbor in the current solution's neighborhood. In this article, MIP-based neighborhood search methods are developed to improve solutions. Since subproblems are MIP, the neighborhood size and structure can greatly affect the computational performance of a search. Four major categories of MIP-based neighborhood search are explored in this section.

**5.2.1. Singleton Search** The neighborhood structure in this category is defined by varying one specific aspect of the problem at a time. Four different structures are considered: one-ship search, one-terminal search, one-direction search and one-day-flexibility search. The algorithm pseudocode exhibited as Algorithms 3 through 6 only represents a basic version of these searches. In implementation, both the order of considered subproblems and the stopping criteria could be manipulated and tuned for better performance based on the problem characteristics.

***One-Ship Search*** This search (Algorithm 3) optimizes the schedule of one ship given that the schedules for all other ships have been fixed.

***One-Terminal Search*** This search (Algorithm 4) optimizes the schedule of one terminal by keeping all travel arcs associated with some terminal free while fixing the travel arcs for all other terminals. It should be emphasized that, if there is only one production terminal or one regasification terminal, then that terminal will be skipped in the search procedure; otherwise, the neighborhood would include the entire network. For example, if we try to optimize all voyages associated with a production terminal which is the only production terminal in the instance, it is equivalent to solving the entire original problem (1).

***One-Direction Search*** This search (Algorithm 5) focuses on production terminals. It either optimizes the inbound travel arcs while fixing the outbound travel arcs, or optimizes the outbound travel arcs while fixing the inbound travel arcs.

***One-Day-Flexibility Search*** This search (Algorithm 6) optimizes the schedule within a *one-day-flexibility neighborhood* which is created based on the current solution as follows. Given the current schedule, we have a sequence of *selected* traveling arcs. For each selected traveling arc, the ship is allowed to leave either one day earlier or one day later; these corresponding arcs are called as *relaxed* traveling arcs. In the one-day-flexibility neighhood, all traveling arcs, other than these selected and relaxed ones, are frozen to zero. For example, if a ship $v$ only has one traveling arc $[(l,t),(r,t+4)]$ in the current solution, then the ship can also take the relaxed arcs $[(l,t-1),(r,t+3)]$ and $[(l,t+1),(r,t+5)]$. Other than these three arcs, however, all other traveling arcs are not allowed for this ship to take in the neighborhood.

It should be noted that the term ″one-day″ is used only for the purpose of a simple description, it can also be extended to multiple days (say *k-day-flexibility neighborhood*). For example, if two-day-flexibility is given, the ship is allowed to take relaxed arcs $[(l,t-2),(r,t+2)]$, $[(l,t-1),(r,t+3)]$, $[(l,t+1),(r,t+5)]$ and $[(l,t+2),(r,t+6)]$ in its neighborhood.

The parameter $k$ is calculated based on the ship fleet size. Given a solution $X$, each ship has an annual schedule, i.e. each ship has one path in the entire network. The neighborhood defined by this algorithm is basically expanding along the path with radius $k$ days. Thus, the neighborhood might be very large if the ship fleet size is large. In that case, parameter $k$ should be set to a small value in order to balance the subproblem size.

**Algorithm 3** One-Ship Search

```
INPUT: a feasible solution $X$
while solution improved do
    for $v \in V$ do
        Fix schedules for all ships other than $v$ in model (1)
        Solve model (1) and get new solution $X_{new}$
        If $obj(X_{new}) < obj(X)$ then set $X \leftarrow X_{new}$
    end for
end while
OUTPUT: improved solution $X$
```

**5.2.2. Time Windows Search** This search is another improvement heuristic which involves the use of time windows in a rolling fashion. A time window here is a sequence of consecutive days. Starting from the beginning of the planning horizon, we define a time window $tw$ with $\Delta_{tw}$ days. In the associated subproblem, all arcs are freed up within the window, and all arcs are fixed outside the window. After the subproblem is solved, we will move to the next window which has $\Delta_{ovlap}$ ($\Delta_{ovlap} \leq \Delta_{tw} - 1$) overlapping days with the current one. These two parameters $\Delta_{tw}$ and $\Delta_{ovlap}$ are the key parameters for this search. The larger these parameters, the better the solution quality;

**Algorithm 4** One-Terminal Search

```
INPUT: a feasible solution $X$
while solution improved do
    for $j \in J | j$ is not the unique terminal in its type do
        Fix travel arcs for all terminals other than $j$ in model (1)
        Solve model (1) and get new solution $X_{new}$
        If $obj(X_{new}) < obj(X)$ then set $X \leftarrow X_{new}$
    end for
end while
OUTPUT: improved solution $X$
```

**Algorithm 5** One-Direction Search

```
INPUT: a feasible solution $X$
while solution improved do
    Fix outbound travel arcs from production terminals in model (1)
    Solve model (1) and get new solution $X_{new}$
    If $obj(X_{new}) < obj(X)$ then set $X \leftarrow X_{new}$
    Fix inbound travel arcs to production terminals in model (1)
    Solve model (1) and get new solution $X_{new}$
    If $obj(X_{new}) < obj(X)$ then set $X \leftarrow X_{new}$
end while
OUTPUT: improved solution $X$
```

**Algorithm 6** One-Day-Flexibility Search

```
INPUT: a feasible solution $X$
INPUT: a parameter $k$
while solution improved do
    for all selected arc $a$ in the current solution $X$ do
        Identify k-day-flexibility relaxed arcs for $a$
    end for
    Freeze all traveling arcs, other than the selected and relaxed arcs, to zero
    Solve model (1) and get new solution $X_{new}$
    If $obj(X_{new}) < obj(X)$ then set $X \leftarrow X_{new}$
end while
OUTPUT: improved solution $X$
```

however, the longer the running time. Therefore these parameters need to be set appropriately in order to get a balance between the solution quality and running time.

Instead of fixing all arcs, different rules can be designed to fix the arcs outside the time windows in the neighborhood construction. For example, we can fix the arcs in one direction at a time by fixing all arcs inbound to production terminals first and then all arcs outbound from production terminals. As is typical to all other neighborhood search methods, the fewer the number of fixed arcs, the larger the neighborhood yielding better the solution improvement as the cost of greater computational complexity. Although multiple construction rules have been designed and tested in our implementation, only the basic version in which all arcs outside the time windows are fixed is shown here in Algorithm 7.

**Algorithm 7** Time Windows Search

```
INPUT: a feasible solution $X$
INPUT: width of time window $\Delta_{tw}$
INPUT: overlapping days $\Delta_{ovlap}$
while solution improved do
    Set start time of the time window $t_{start} = 1$
    while $t_{start} \leq |T| - \Delta_{tw}$ do
        Set end time of the time window $t_{end} = t_{start} + \Delta_{tw}$
        Create time window $tw = [t_{start}, t_{end}]$
        Fix all arcs outside the time window $tw$ in model (1)
        Solve model (1) and get new solution $X_{new}$
        If $obj(X_{new}) < obj(X)$ then set $X \leftarrow X_{new}$
        Set $t_{start} \leftarrow t_{end} - \Delta_{ovlap}$
    end while
end while
OUTPUT: improved solution $X$
```

**5.2.3. Two-Ship Search** This neighborhood is essentially the multi-ship extension of the ″one-ship″ neighborhood. In this search method, we optimize schedules for two ships simultaneously while fixing schedules for all other ships. This method was originally proposed by Song and Furman [2013] for a different maritime inventory routing problem, and was extensively studied for LNG-IRP by Goel et al. [2012]. Neighborhoods of more than two ships could potentially provide greater improvements to solution quality, however, their computational expense is usually prohibitive. In order to speed up the search and improve upon previous methods, we propose three ship pair selection techniques for selecting the ship pairs intelligently.

The basic version of this search simply loops though all ship pairs $(u,v), \forall u \in V, v \in V, v \neq u$ in the original ordering of ships in the set until no further improvement is obtained. Goel et al. [2012] refer to this method as lexicographic selection. This can be considered as the metric by which to judge improvement in a ship pair selection technique.

***Correlation Selection*** In this technique, we first rank each ship pair $(u,v), \forall u \in V, v \in V, v \neq u$ with a metric called ″correlation″ which is defined as the number of days $t$, that ship $u$ is located at a terminal on day $t$ and ship $v$ is also located at the same terminal some time during $[t-k, t+k]$ with parameter $k \geq 0$. This correlation can be viewed as the extended number of days that both of the ships are available at the same terminal. When it is large, the chance to swap some of their voyages to improve the objective function value is also large. Therefore we sort and solve the ship pairs according to their correlations in non-increasing order. Whenever an improved solution is discovered, correlations are updated and ship pairs are resorted. Letting $corr_{u,v}$ be the correlation for ship pair $(u,v)$, its calculation is described in Algorithm 8.

***Benefit Selection*** In this technique, we rank each pair $(u,v), \forall u \in V, v \in V, v \neq u$ with a value called ″benefit″ which is defined as a weighted sum of the evaluated loss associated with ship $u$ and ship $v$ and their correlation. If a ship is compatible with a terminal that has a large loss in the current solution, we may have room to improve the solution if the ship's schedule is modified. Thus, we simply sum up the loss at all terminals that are compatible with ship $u$ or $v$ as the evaluated loss. Letting $w^{corr}$ be the weight for correlation and $ben_{u,v}$ be the benefit for ship pair $(u,v)$, its calculation is described in Algorithm 9.

***Random Ship Selection*** Another metric for evaluating variants of the two-ship neighborhood search is to randomly select ship pairs. For this implementation, probability $p_{u,v} = ben_{u,v} / \sum_{(u',v')} ben_{u',v'}$ is assigned to each ship pair $(u,v), \forall u \in V, v \in V, v \neq u$, and one pair is randomly selected to be rescheduled at each iteration. When improved solutions are detected, both

**Algorithm 8** Calculation for Two-Ship Correlation

```
INPUT: a feasible solution $X$
INPUT: a parameter $k$
Set $corr_{u,v} = 0, \forall u \in V, v \in V, u \neq v$
for $u = 1 \to |V| - 1$, $t = 1 \to |T|$ do
    if ship $u$ is at any terminal $j$ on day $t$ then
        for $v = u + 1 \to |V|$ do
            if ship $v$ is at the terminal $j$ some day during $[t - k, t + k]$ then
                $corr_{u,v} += 1$
            end if
        end for
    end if
end for
OUTPUT: $corr_{u,v}, \forall u \in V, v \in V, u \neq v$
```

**Algorithm 9** Calculation for Two-Ship Benefit

```
INPUT: a feasible solution $X$
INPUT: weight of correlation $w^{corr}$
Set $AssociatedLoss_{u,v} = 0, \forall u \in V, \forall v \in V, u \neq v$
for $u \in V, v \in V, u \neq v$ do
    for $j \in J$ do
        if ship $u$ or $v$ is compatible with terminal $j$ then
            $AssociatedLoss_{u,v} += Loss_j$
        end if
    end for
end for
Set $ben_{u,v} = AssociatedLoss_{u,v} + w^{corr} \cdot corr(u, v)$
OUTPUT: $ben_{u,v}, \forall u \in V, v \in V, u \neq v$
```

the benefits and probabilities are updated. The search is terminated after a pre-specified number of iterations.

## 6. Computational Results

In this section, we present results for all of the proposed methods for improving the lower and upper bounds of the LNG-IRP. All algorithms were coded in C++ with CPLEX 12.2 as the LP and MIP solvers and run on a workstation with Intel(R) Xeon(R) CPU X5687@3.60Ghz and 24 GB RAM. Test cases are adopted from Goel et al. [2012] and results are compared with the methods of Goel et al. [2012].

### 6.1. Testing Cases

Goel et al. [2012] designed and evaluated 14 problems which were categorized into three groups: 1) *easy* problems (P1-P5) for which CPLEX can generate the best solutions that were known by Goel et al. [2012] within 1 CPU hour; 2) *medium* problems (P6-P10) for which CPLEX can generate ″good″ solutions (within 15% of the best knowns) within 1-5 hours; and 3) *hard* problems (P11-P14) for which CPLEX cannot generate a good solution within 10 CPU hours. The characteristics of these problems are highlighted in Table 1 including the best known solutions reported in Goel et al. [2012].

| Case | Dimensions $(\|L\|,\|R\|,\|V\|)$ | Cont. Variables | 0-1 Variables | Constraints | Best knowns in Goel et al. [2012] |
|---|---|---|---|---|---|
| P1 | (1,2,6) | 2557 | 12812 | 6504 | 1172 |
| P2 | (1,3,8) | 4019 | 16988 | 8635 | 294 |
| P3 | (2,1,10) | 2557 | 21029 | 9303 | 5650 |
| P4 | (3,1,13) | 2922 | 27547 | 12196 | 0 |
| P5 | (1,4,15) | 5115 | 31449 | 14269 | 1701 |
| P6 | (4,1,4) | 3652 | 8377 | 6406 | 151 |
| P7 | (1,6,6) | 5117 | 18335 | 10812 | 1565 |
| P8 | (1,1,14) | 1827 | 28830 | 11205 | 116 |
| P9 | (1,2,17) | 2558 | 35142 | 14061 | 126 |
| P10 | (1,4,27) | 5115 | 95788 | 32744 | 29388 |
| P11 | (1,5,14) | 6228 | 34438 | 15625 | 0 |
| P12 | (1,4,18) | 3655 | 37752 | 16491 | 631 |
| P13 | (1,8,40) | 6579 | 132995 | 47364 | 1453 |
| P14 | (1,10,69) | 11691 | 232856 | 79425 | 53240 |

**Table 1 Problem Instances**

## 6.2. Results for Decomposition Procedure

The computational expense of the branch and price algorithm for finding lower bounds is excessive when compared to the local search methods for finding good upper bounds. None of the test cases can be solved to optimality, nor does the B&P algorithm produce any integer solutions for any of the test cases. However, the B&P algorithm does obtain better lower bounds with substantially better computational efficiency compared to CPLEX.

**DW Performance.** First we compare the performance of our DW decomposition with CPLEX at the root node. In our implementation, the entire problem is decomposed by individual ships. The termination criteria for convergence is defined as achieving a gap between the objective function value of MP and DW lower bound within 0.1%. Although we do not report the detailed performance of stabilization technique separately, our computational experimentation indicates that is cuts 15.12% of CPU time on average. Due to the structure of our subproblems, we would not expect better bounds at root node as discussed earlier. However, DW decomposition speeds up the computation time significantly when the problem size is extremely large when compared to CPLEX using its default dual simplex method on the arc-flow formulation. Table 2 illustrates these results.

In Table 2, columns 2-3 are the objective value and CPU time respectively solving the root node using CPLEX as the solver. Columns 4-9 address the DW lower bound, objective value of MP, total CPU time, CPU time for solving MP, CPU time for solving SP and the percent of MP CPU time over the entire DW procedure, respectively, for DW decomposition algorithm. Columns 10-11 present the speed up in which all subproblems are run in series, as well as the hypothetical speed up if we assume subproblems are all run in parallel with same CPU time (i.e. the CPU time for the MP plus the slowest of the subproblems).

Table 2 illustrates that both CPLEX (dual simplex) and DW procedure solved the root nodes successfully within 20 minutes, with the exception of case P14, which took CPLEX around 28 hours while DW only required 1.2 hours. When subproblems in DW are run in series, DW can solve easy, medium and hard test cases 1.09, 0.77 and 2.50 (geometric mean) times faster than CPLEX, respectively. This indicates that serial DW does not gain much in computational efficiency if an instance is not hard to solve or large. According to column "MP_CPU/CPU", however, DW spends 98.29% of time on average in solving subproblems. Parallelization of solving subproblem would yield

| | CPLEX | | DW decomposition | | | | | | Speed Up | |
|---|---|---|---|---|---|---|---|---|---|---|
| Case | Obj. | CPU (s) | DW LB | MP Obj. | CPU (s) | MP_CPU (s) | SP_CPU (s) | MP_CPU /CPU(%) | Serial | Synthetic Parallel |
| P1 | 1026 | 5 | 1026 | 1026 | 21.2 | 0.3 | 20 | 1.42 | 0.25 | 1.19 |
| P2 | 95 | 21 | 95 | 95 | 15.4 | 0.5 | 14 | 3.25 | 1.36 | 6.82 |
| P3 | 5457 | 21 | 5454 | 5457 | 29.9 | 0.3 | 28 | 1.00 | 0.71 | 4.21 |
| P4 | 0 | 32 | 0 | 0 | 12.5 | 0.3 | 11 | 2.40 | 2.54 | 15.32 |
| P5 | 1647 | 112 | 1647 | 1647 | 44.4 | 1.0 | 39 | 2.25 | 2.51 | 13.33 |
| P6 | 0 | 1 | 0 | 0 | 10.1 | 0.1 | 9 | 0.99 | 0.12 | 0.39 |
| P7 | 0 | 14 | 0 | 0 | 89.1 | 3.2 | 82 | 3.59 | 0.16 | 0.67 |
| P8 | 116 | 29 | 116 | 116 | 9.3 | 0.1 | 9 | 1.08 | 3.06 | 25.25 |
| P9 | 126 | 47 | 126 | 126 | 15.9 | 0.1 | 15 | 0.63 | 2.96 | 20.86 |
| P10 | 28766 | 648 | 28744 | 28768 | 403.4 | 1.4 | 367 | 0.35 | 1.61 | 13.01 |
| P11 | 0 | 100 | 0 | 0 | 47.5 | 1.4 | 43 | 2.95 | 2.11 | 13.1 |
| P12 | 0 | 65 | 0 | 0 | 105.9 | 1.2 | 92 | 1.13 | 0.61 | 3.39 |
| P13 | 1337 | 978 | 1337 | 1337 | 756.8 | 11.1 | 578 | 1.47 | 1.29 | 5.07 |
| P14 | 48049 | 101582 | 48028 | 48050 | 4342.9 | 64.6 | 2299 | 1.49 | 23.39 | 48.91 |

**Table 2 Comparison at Root Nodes: DW Decomposition v.s CPLEX**

significant improvements on CPU time, however since design of parallelization algorithm is not in the scope of this work, we simply report a hypothetical speed up metric to evaluate the potential for improvement. If subproblems were solved in parallel with such a hypothetical algorithm, in a best case scenario, DW can solve the root nodes 5.87, 4.48 and 10.24 times faster than CPLEX for easy, medium and hard problems respectively.

**B&P Performance.** Test case P14 is a large scale instance of a practical industrial scale with respect to size and difficulty. Since CPLEX requires over one day to solve the root node of instance P14, CPLEX is not a good choice for evaluating the quality of heuristic solutions for large scale LNG-IRP instances. Acknowledging that proving optimality is unlikely, the B&P algorithm proposed can be used to obtain lower bounds. As discussed previously, the results reported here are associated to the node selection rule BBF and the branching strategy which combines branching the number of voyages starting from terminals, number of voyages executed by ships and the default rules. Each instance was run with two hours, and results are shown in Table 3.

Table 3 lists the default LP bounds and the B&P bounds obtained within two hours, along with the corresponding gaps calculated using the following formula: (Best Known Solution Goel et al. [2012] - LB) / LB *100%. Except for the four instances whose value of LP relaxation is equivalent to that of MIP, B&P provides improved bounds for four instances, and these improvements are substantial for case P2, P7 and P12. These results show that B&P is an efficient method for obtaining lower bounds and evaluating the quality of heuristic solutions. Even though B&P does not improve lower bounds for other five cases, this deficiency could be overcome by designing a more sophisticated decomposition scheme as described previously, for example, by aggregating homogeneous ships into groups.

## 6.3. Results for Local Search

Next we present the results for local search including both the construction heuristics and solution improvement via neighborhood search.

**Construction Heuristics.** Table 4 compares the proposed construction heuristics with the previously developed methods by Goel et al. [2012]. The improved construction solutions derived

| | LP | | B&P (2 hours) | | |
|---|---|---|---|---|---|
| Case | LB | GAP(%) | LB | GAP(%) | Performance |
| P1 | 1026 | 14.23 | 1026 | 14.23 | - |
| P2 | 95 | 209.47 | 191 | 53.93 | improved |
| P3 | 5457 | 3.54 | 5470 | 3.29 | improved |
| P4 | 0 | 0 | 0 | 0 | optimal |
| P5 | 1646 | 3.34 | 1646 | 3.34 | - |
| P6 | 0 | inf | 0 | inf | - |
| P7 | 0 | inf | 1221 | 28.17 | improved |
| P8 | 116 | 0 | 116 | 0 | optimal |
| P9 | 126 | 0 | 126 | 0 | optimal |
| P10 | 28766 | 2.16 | 28670 | 2.16 | - |
| P11 | 0 | 0 | 0 | 0 | optimal |
| P12 | 0 | inf | 292 | 116.10 | improved |
| P13 | 1337 | 8.68 | 1337 | 8.68 | - |
| P14 | 48049 | 10.80 | 47868 | 10.80 | - |

**Table 3 Performance of B&P on Lower Bounds Improvement**

by Goel et al. [2012] are listed in column 2 with the corresponding CPU time in column 3; the previously determined final solutions are listed in column 4 at the CPU time in column 5. Columns 6-7 present the results of the one-day round-trip rolling time algorithm; 3 iterations are completed and the accumulated CPU time is reported. Columns 8-9 list the results of the multi-day round-trip rolling time algorithm. Columns 10-11 list the results for one-step GRASP; it is run for 10 iterations and accumulated CPU time is reported. Columns 12-13 list results for two-step GRASP; in this algorithm, higher priority is given to the more flexible ships; 10 iterations are completed; accumulated CPU time is reported. All of the CPU times are reported in the unit of ′seconds′.

In Table 4, those values in **bold** are no worse than the construction heuristic solutions of Goel et al. [2012] and those with an **underline** are no worse than the final solutions of Goel et al. [2012]. Taking case P1 as an example, the one-day rolling time algorithm achieves **1179** in 2 seconds which is better than the final solution 1187 found by Goel et al. [2012]; all other construction heuristics return a value of **1199** which is same as the construction solution of Goel et al. [2012]. From Table 4, we can conclude that, the initial solutions based on the new construction techniques are significantly better than those previously reported. Among the four techniques, the multi-day round trip rolling time algorithm performs best with respect to the objective values. By using this algorithm, the initial solutions for cases P4 and P14 are strictly better than the best known solutions reported in Goel et al. [2012]. However, this algorithm requires more CPU time than other construction methods due to its greater sophistication.

Further, several of the new construction solutions compete with the final solutions found via previous methods with significantly less CPU time, especially for these difficult cases. For example, for the three most challenging cases P10, P13 and P14, the solutions produced by the new construction methods are no worse than those found by Goel et al. [2012] while faster by at least a factor of 10. The comparison of the three most challenging cases are illustrated in Figure 2, in which the rolling time algorithm is donated as ″RTA″.

The results of Table 4 and Figure 2 illustrate that rolling time algorithms and GRASP have good performance across different cases. If we apply both heuristics in sequence, for example running multiple-day rolling time algorithm and then one-step GRASP, strong results are achieved for the majority of the test cases while maintaining the fast run times. Thus combining two different construction techniques can be a good strategy to generate a good initial solution for LNG-IRP.

| | Goel et al. [2012] | | | | Rolling Time Algorithm | | | | GRASP | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Constr. Sol. | | Final Sol. | | One-Day | | Multiple-Day | | One-Step | | Two-Step | |
| Case | Obj. | CPU (s) | Obj. | CPU (s) | Obj. | CPU (s) | Obj. | CPU (s) | Obj. | CPU (s) | Obj. | CPU (s) |
| P1 | 1199 | 6 | 1187 | 200 | **1179** | 2 | **1199** | 6 | **1199** | 5 | **1199** | 4 |
| P2 | 475 | 17 | 320 | 73 | 1226 | 7 | **348** | 15 | 749 | 4 | 681 | 6 |
| P3 | 5923 | 18 | 5696 | 74 | **5830** | 3 | 6281 | 11 | 5998 | 4 | 6220 | 6 |
| P4 | 288 | 12 | 0 | 197 | **0** | 13 | **49** | 44 | **202** | 6 | **174** | 7 |
| P5 | 2880 | 27 | 1961 | 332 | 2891 | 9 | **1818** | 42 | **2415** | 11 | **2640** | 11 |
| P6 | 151 | 3 | 151 | 235 | **151** | 3 | **151** | 12 | **151** | 3 | **151** | 3 |
| P7 | 4194 | 8 | 1565 | 2080 | **1602** | 5 | **1509** | 19 | **1709** | 5 | **1692** | 6 |
| P8 | 116 | 5 | 116 | 111 | **116** | 19 | **116** | 40 | **116** | 6 | **116** | 6 |
| P9 | 126 | 6 | 126 | 184 | 134 | 47 | 888 | 74 | **126** | 9 | **126** | 9 |
| P10 | 30331 | 96 | 29559 | 2645 | **29725** | 22 | **29403** | 149 | **29587** | 61 | **29413** | 57 |
| P11 | 504 | 7 | 0 | 446 | **108** | 36 | **0** | 75 | **436** | 14 | 650 | 13 |
| P12 | 656 | 127 | 631 | 735 | 668 | 9 | **656** | 50 | **631** | 14 | **637** | 13 |
| P13 | 6499 | 146 | 1722 | 13557 | **2067** | 181 | **1570** | 598 | **2830** | 89 | **3139** | 87 |
| P14 | 66387 | 349 | 53519 | 27595 | **53781** | 196 | **52899** | 1496 | **55321** | 206 | **55970** | 202 |

**Table 4** **Comparison: Construction Heuristics v.s. Goel et al. [2012]**

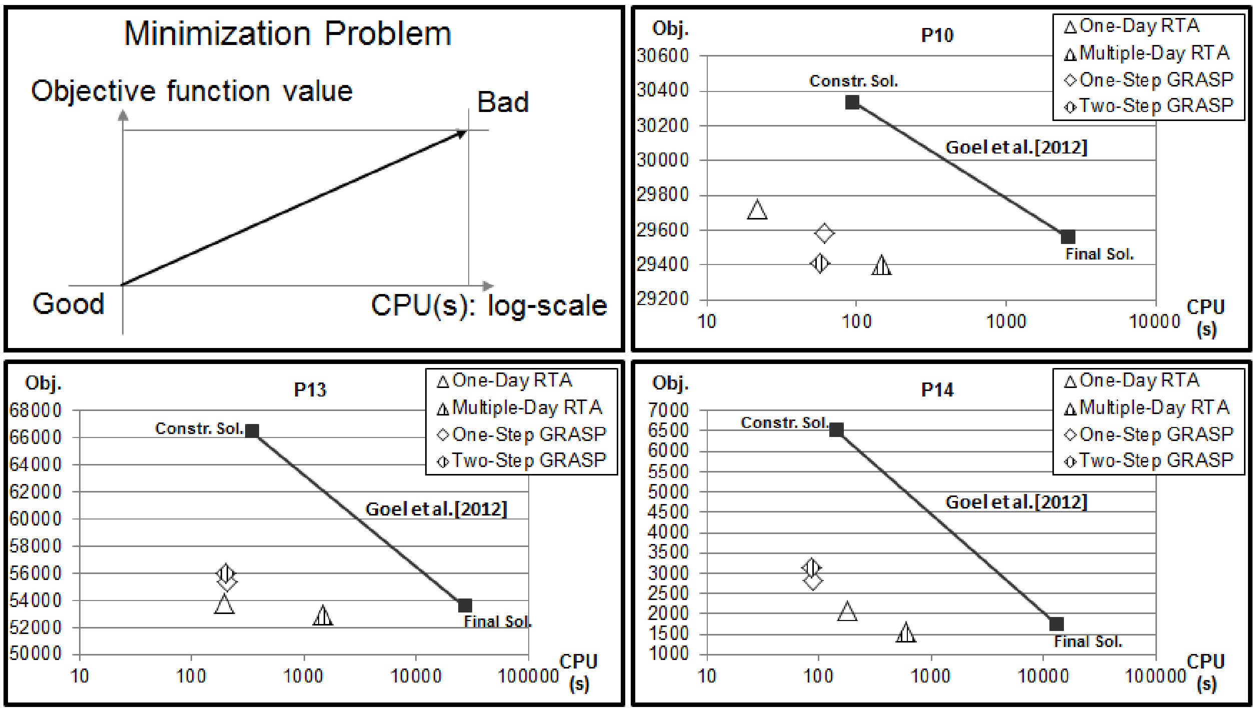

**Figure 2** **Comparison on The Three Most Challenging Cases: Construction Heuristics v.s. Goel et al. [2012]**

**Solution Improvements** We now report the results for neighborhood search algorithms. A 1000 branch-and-bound node limit was used in the solution of subproblems for all of the heuristics with the exception of the rolling time windows directional search. Due to the large size of the neighborhoods in this method, a 100 node limit was used.

A 60 second CPU time limit was also imposed for subproblems in One-Terminal Search. As mentioned previously, the ″one-terminal″ neighborhood is the full MIP problem when the selected terminal is unique within its type. Further, even when the selected terminal is not unique, if the number of terminals of its type is small, the subproblem might still be very large and difficult to solve. It is possible that in MIP subproblems the run time for each branch-and-bound node is excessively long even though no improvement in the solution is seen while approaching 1000 nodes. In order to avoid this situation, a 60 second time limit was placed on the MIP solver. This unfortunately could make it difficult to reproduce identical results for the One-Terminal Search. However, this situation is rare, and in fact across all of the results, this only occurs in one iteration of one instance and all of the remaining results for this search method terminate at the 1000 node limit within 60 seconds.

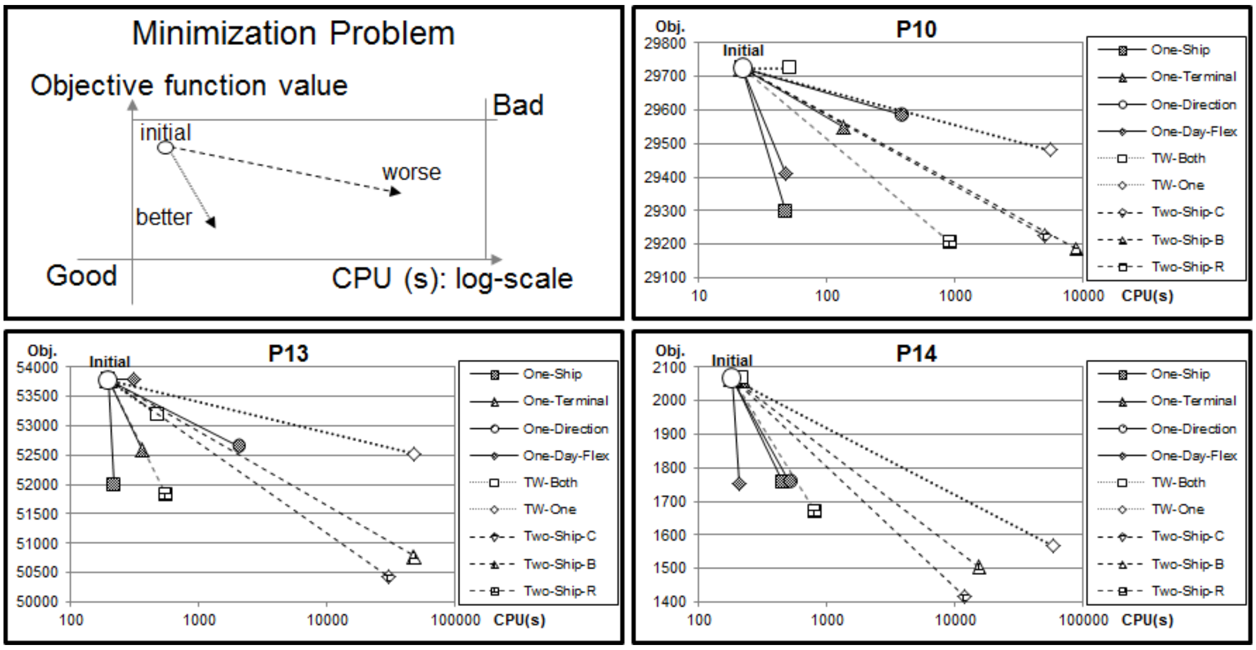

**Figure 3 Comparison on The Three Most Challenging Cases: Different Neighborhood Searches**

Even though all 14 instances are tested for all neighborhood searches starting from different initial solutions obtained by different construction heuristics, we only show the performance of these searches for the three most challenging test cases starting from the initial solutions obtained by One-Day Rolling Time Algorithm here in Figure 3. In Figure 3, the rolling time windows search is denoted as ″TW-Both″ and rolling time windows directional search is denoted as ″TW-One″; also, ″Two-Ship-C″, ″Two-Ship-B″ and ″Two-Ship-R″ represent the Two-Ship with the rule of correlation selection, benefit selection and random selection respectively.

From Figure 3, we conclude that the Singleton methods while not very computationally expensive make significant improvements to the objective values. The Two-Ship search, while the most computationally expensive, improves the solution value most significantly. Very similar results were observed starting from initial solutions obtained via other construction methods for all instances.

Given the level of performance of the Singleton methods, the natural next step is to consider combining these methods and running them in sequence, as this has the potential to produce substantial gains in solution quality with a low computational expense. Figure 4 illustrates that a sequence of Singleton methods produces solutions of similar quality to the two-ship neighborhood search with a computational cost lower by an order magnitude.

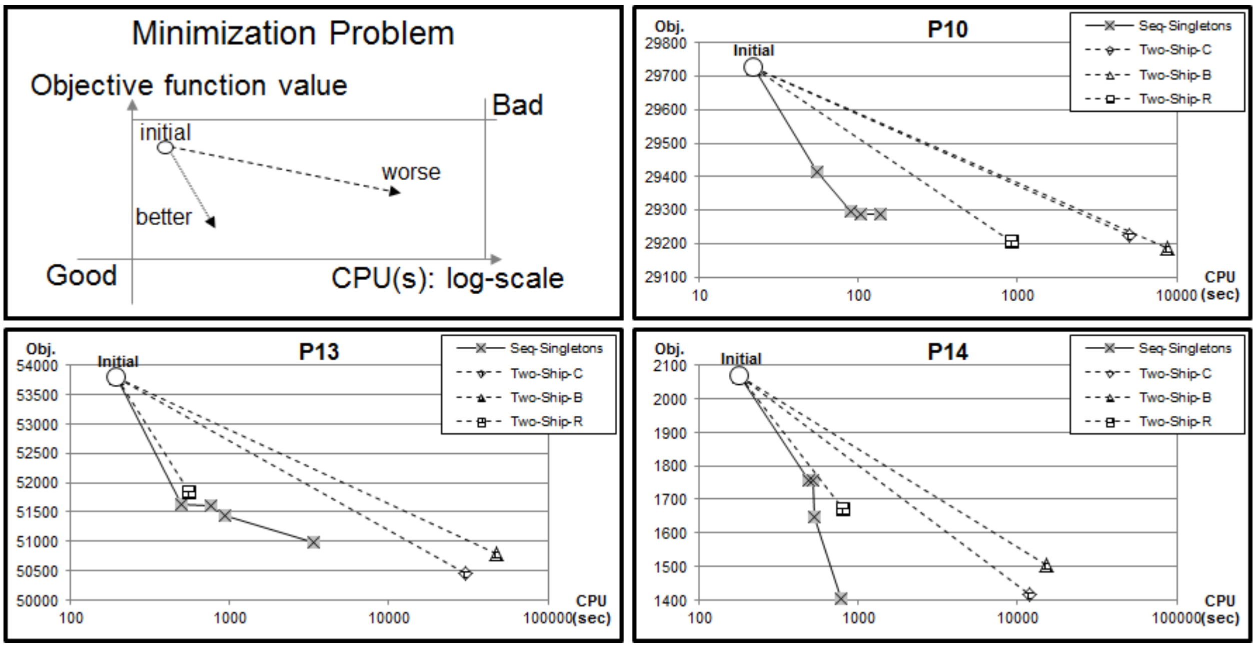

**Figure 4** **Comparison on The Three Most Challenging Cases: Sequential Singletons v.s. Two-Ship Searches**

Using the sequential Singletons as a solution-improvement scheme combined with any of the newly proposed construction heuristics can yield strictly better solutions than the benchmark set of heuristics in Goel et al. [2012] with significantly less CPU time. Results considering the three most challenging cases are shown in Figure 5.

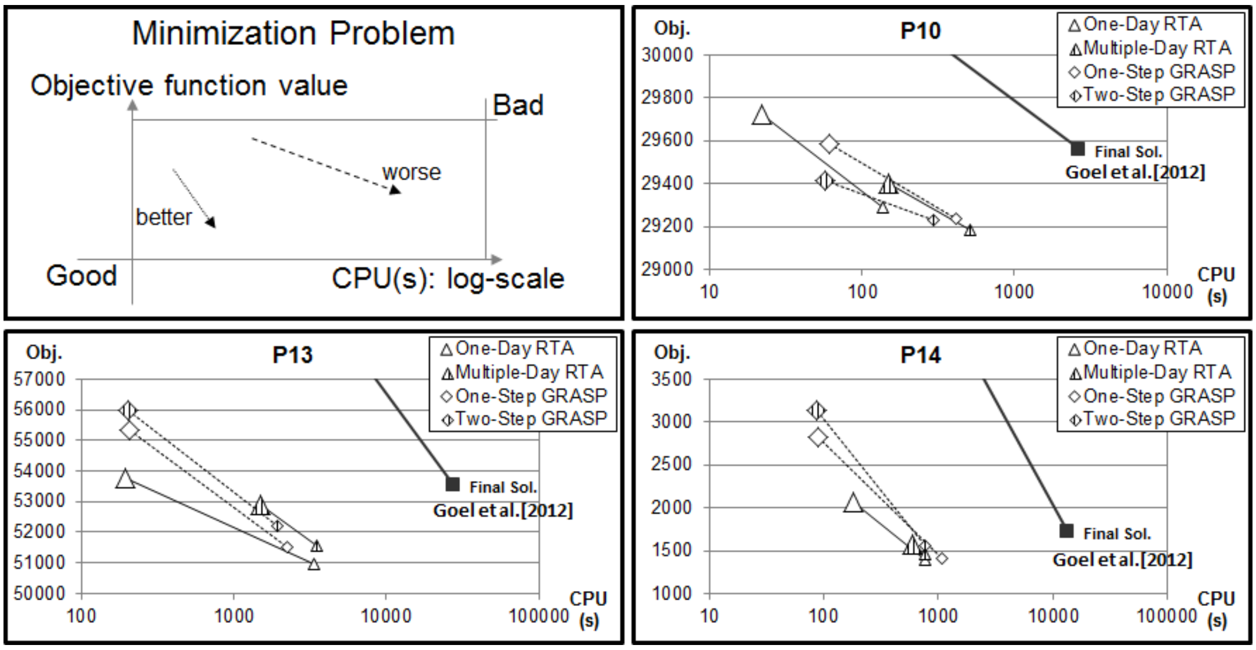

**Figure 5** **Comparison on The Three Most Challenging Cases: New Approaches v.s. Goel et al. [2012]**

A Dolan & Moré performance profile (see Dolan and Moré [2002]) is provided in Figure 6 to illustrate results for all 14 test cases. In this figure, the benchmark is based on the greedy construction heuristic, time window local search and lexicographic two-ship neighborhood search developed by Goel et al. [2012]. The figure is generated based on the idea that a technique is successful if it can find a better or equivalent solution for a case when compared to the benchmark method. If a technique is successful on a case, then its normalized CPU time is calculated. This normalized CPU time is defined as the the ratio of a technique's CPU time to the fastest successful techniques's real CPU time for a test case. In Figure 6, the X-axis represents the normalized CPU time and Y-axis represents the percentage of test cases for which a technique is successful. Thus, the step-lines reflect the performance of the various techniques. In the plot, sequential singleton search starting from different construction heuristics or a combination of construction heuristics is compared to the benchmark. All of these variations find good solutions to many instances quickly relative to the benchmark. Applying both the Multiple-Day round-trip rolling time search and One-Step GRASP and choosing the best of the two to determine the initial point performs best among the variations considered even given the added computational cost. These results indicate that the combination of multiple inexpensive construction heuristics is a good strategy. In general, these results seem to support combining a portfolio of algorithms as the best overall strategy since no individual method completely dominates the others.

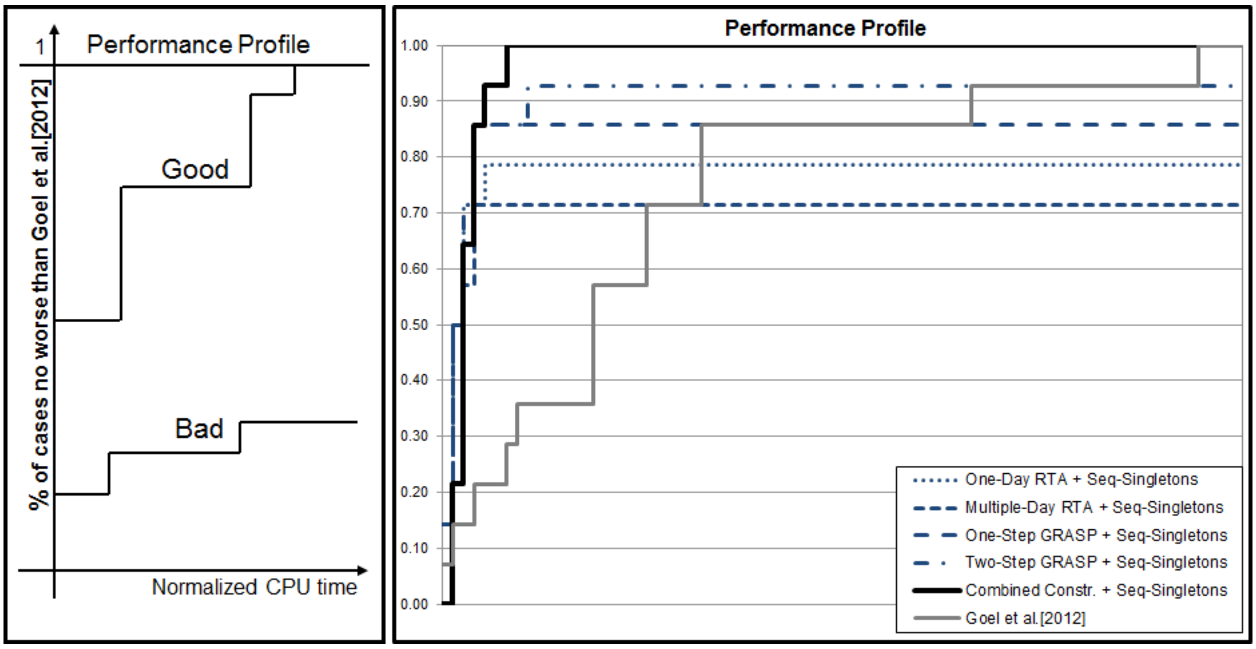

**Figure 6 Performance Profile: Relative to Goel et al. [2012]**

The optimal solution is unproven in many of the test instances. Figure 7 illustrates the corresponding performance profile based on a comparison with the benchmark method for finding the best known solutions for each case. Best known solutions are based on a combination of the results of this article and the previous effort of Goel et al. [2012]. The results show the best strategy yielding the best known solution in 87% of cases very quickly, while the benchmark method only obtains the best solution in 50% of cases and requires substantially greater CPU time. The tails associated to the two rolling time algorithms are coincident in Figure 7.

## 7. Conclusions

Computational results indicate that the proposed methods for improving lower bounds for this LNG-IRP are substantially faster than commercial optimization software. In order to evaluate solution qualities, we developed B&P algorithm using DW decomposition procedure to solve each node.

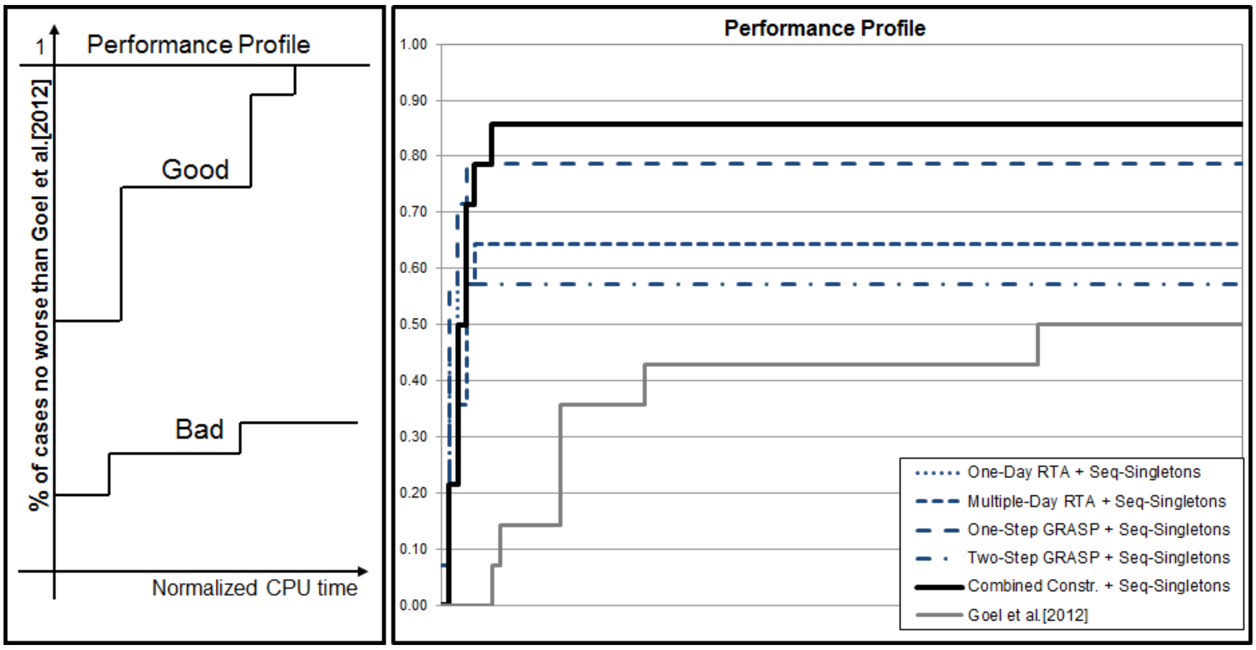

**Figure 7** **Performance Profile: Relative to Best Known Solution**

Although the algorithm fails to provide optimal solutions for the test instances, it is demonstrated as a good approach to improve the lower bounds for this LNG-IRP. The extension of this method, for example, aggregating ships into groups, could potentially further improve its efficiency.

With the aim of finding better feasible solutions, we developed a series of construction heuristics and neighborhood search methods with overall good performance. The benchmark method Goel et al. [2012] was improved upon with better solutions and significantly less CPU time, and for the most difficult cases, the speed up is by an order of magnitude. Furthermore, the results indicate that combining a portfolio of algorithms is the best overall strategy for LNG-IRP since none of these heuristics or neighborhood search methods completely dominates the others.

## References

R.K. Ahuja, O. Ergun, J. B. Orlin, and A.P. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123:75–102, 2002.

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

S.A. Araujo, M.N. Arenales, and A.R. Clark. Joint rolling-horizon scheduling of materials processing and lot-sizing with sequence-dependent setups. *Journal of Heuristics*, 13:337–358, 2007.

K. Baker. An experimental study of the effectiveness of rolling schedules in production planning. *Decision Sciences*, 8:19–27, 1977.

J.F. Bard. An analysis of a rail car unloading area for a consumer products manufacturer. *Journal of the Operational Research Society*, 48:873–883, 1997.

D. Bredström and M. Rönnqvist. Supply chain optimization in pulp distribution using a rolling horizon solution approach. *Discussion paper*, 17, 2006.

M. Christiansen. Decomposition of a combined inventory and time constrained ship routing problem. *Transportation Science*, 33(1):3–16, 1999.

M. Christiansen and K. Fagerholt. Maritime inventory routing problems. In C.A. Floudas and P.M. Pardalos, editors, *Encyclopedia of Optimization*, pages 1947–1955. Springer, 2009.

M. Christiansen, K. Fagerholt, and D. Ronen. Ship routing and scheduling: Status and perspectives. *Transportation Science*, 38(1):1–18, 2004.

M. Christiansen, K. Fagerholt, B. Nygreen, and D. Ronen. Maritime transportation. In C. Barnhart and G. Laporte, editors, *Transportation*, volume 14 of *Handbooks in Operations Research and Management Science*, chapter 4, pages 189–284. Elsevier, 2007.

M. Christiansen, K. Fagerholt, B. Nygreen, and D. Ronen. Ship routing and scheduling in the new millenium. *European Journal of Operational Research*, 228:467–483, 2013.

A. Dimitriadis, N. Shah, and C. Pantelides. Rtn-based rolling horizon algorithms for medium term scheduling of multipurpose plants. *Computers and Chemical Engineering*, 21:1061–1066, 1997.

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

O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Communication stabilized column generation. *Discrete Mathematics*, 194:229–237, 1999.

F.G. Engineer, K.C. Furman, G.L. Nemhauser, M.W.P. Savelsbergh, and J.-H. Song. A branch-price-and-cut algorithm for single product maritime inventory routing. *Operations Research*, 60 (1):106–122, 2012.

T.A. Feo and J.F. Bard. Flight scheduling and maintenance base planning. *Management Science*, 35:1415–1432, 1989.

T.A. Feo and M.G.C. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8:67–71, 1989.

P. Festa and M.G.C. Resende. Grasp: An annotated bibliography. *Essays and Surveys on Metaheuristics*, pages 325–367, 2002.

V. Goel, K.C. Furman, J.-H. Song, and A.S. El-Bakry. Large neighborhood search for lng inventory routing. *Journal of Heuristics*, 18:821–848, 2012. doi: 10.1007/s10732-012-9206-6.

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

E.E. Halvorsen-Weare and K. Fagerholt. Routing and scheduling in a liquefied natural gas shipping problem with inventory and berth constraints. *Annals of Operations Research*, 203:167–186, 2013.

E.E. Halvorsen-Weare, K. Fagerholt, and M. Rönnqvist. Vessel routing and scheduling under uncertainty in the liquefied natural gas business. *Computers and Industrial Engineering*, 64:290–301, 2013.

M. Hewitt, G. Nemhauser, M. Savelsbergh, and J.-H. Song. A branch-and-price guided search approach to maritime inventory routing. *Computers and Operations Research*, 40(5):1410–1419, 2013. doi: 10.1016/j.cor.2012.09.010.

G. Kontoravdis and J.F. Bard. A grasp for the vehicle routing problem with time windows. *ORSA Journal on Computing*, 7:10–23, 1995.

C. Mercé and G. Fontan. Mip-based heuristics for capacitated lotsizing problems. *International Journal of Production Economics*, 85:97–111, 2003.

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

D. Ronen. Cargo ships routing and scheduling: Survey of models and problems. 12:119–126, 1983.

D. Ronen. Ship scheduling: The last decade. 71:325–333, 1993.

D.M. Ryan and B.A. Foster. An integer programming approach to scheduling. In A. Wren, editor, *Computer Scheduling of Public Transport*, pages 269–280. Elsevier, 1981.

M.W.P. Savelsbergh and J.-H. Song. An optimization algorithm for the inventory routing problem with continuous moves. *Computers and Operations Research*, 35(7):2266–2282, 2008.

J.-H. Song and K.C. Furman. A maritime inventory routing problem: Practical approach. *Computers and Operations Research*, 40(3):657–665, 2013. doi: 10.1016/j.cor.2010.10.031.

L. Stauffer and T. Liebling. Rolling horizon scheduling in a rolling-mill. *Annals of Operations Research*, 69:323–349, 1997.

L.A. Wolsey. *Inter Programming.* John Wiley & Sons, New York, 1998.