Constraint Programming for LNG Ship Scheduling and Inventory Management

V. Goel[a], M. Slusky[b], W.-J. van Hoeve[c], K. C. Furman[d], Y. Shao[a]

[a] *ExxonMobil Upstream Research Company*
{vikas.goel,yufen.shao}@exxonmobil.com
[b] *Department of Mathematical Sciences, Carnegie Mellon University*
mslusky@andrew.cmu.edu
[c] *Tepper School of Business, Carnegie Mellon University*
vanhoeve@andrew.cmu.edu
[d] *ExxonMobil Research and Engineering*
kevin.c.furman@exxonmobil.com

Abstract

We propose a constraint programming approach for the optimization of inventory routing in the liquefied natural gas industry. We present two constraint programming models that rely on a disjunctive scheduling representation of the problem. We also propose an iterative search heuristic to generate good feasible solutions for these models. Computational results on a set of large-scale test instances demonstrate that our approach can find better solutions than existing approaches based on mixed integer programming, while being 4 to 10 times faster on average.

Keywords: Routing, Inventory, Constraint Programming

1. Introduction

Natural gas, which is primarily composed of methane, is a relatively clean burning fuel for a number of outlets including electricity generation and heating. Traditionally natural gas projects were developed near local markets or near major pipelines. However, liquefied natural gas (LNG) provides the ability to develop a natural gas project in an isolated area of the world and yet efficiently deliver that gas to market. The technology has been available for several decades, and it accounts for meeting a rapidly growing share of natural gas market demand. One of the most important advantages of LNG tanker ship over pipelines is the access it provides to markets overseas.

The focus of this work is on a central operational issue in the LNG industry: designing schedules for the ships to deliver LNG from the production (liquefaction) terminals to the demand (regasification) terminals. This is a complex problem because it combines ship routing with inventory management at the terminals.

A primary application of this problem is to support analysis for the design of the supply chain for new projects. Typically, new LNG projects require building not only the natural gas production and treatment facilities, but also constructing the ships and the production-side terminals, berths and storage tanks. Operationally, a delivery schedule for each customer in the LNG supply chain is negotiated on an annual basis. The supply and demand requirements of the sellers and purchasers are accommodated through the development of this annual delivery schedule, which is a detailed schedule of LNG loadings, deliveries and ship movements for the planning year.

A solution to the inventory routing problem can be used in two primary ways: (1) to analyze LNG supply chain design decisions such as LNG fleet size and composition, and terminal infrastructure during the concept and detailed design phases for new LNG projects, and (2) to develop the delivery schedules for operations of these LNG projects, quantify schedule inefficiency, and conduct "what-if" analysis to test how robust a schedule is to disruption events. Goel et al. (2012) first introduced this problem, and proposed a mixed integer programming (MIP) model as well as a MIP-based heuristic for finding good solutions. Yet, both the MIP model and the MIP heuristic suffered from scalability issues for larger instances. This was one of the main motivations for studying constraint programming in the present work as an alternative method for solving this problem.

The LNG inventory routing problem is a special case of the maritime inventory routing problem. The 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 and a production or consumption rate, therefore combining inventory management and ship routing. These are typically treated separately in much of the maritime transportation industry. In such problems, the number of visits to a port and the quantity of product to be loaded or unloaded are not predetermined. Christiansen et al. (2013) provide a recent survey in ship routing and scheduling research over the last decade. LNG inventory routing business cases and common problem

characteristics are discussed by Andersson et al. (2010).

Constraint Programming (CP) has been applied before to industrial routing and scheduling problems, and most CP solvers offer a specific modeling interface for representing these problems (Baptiste et al. 2001, 2006). In particular, it is common to represent routing problems as disjunctive scheduling problems, in which a visit to a location becomes a task to be scheduled, and the distance between two locations is modeled as a 'sequence-dependent set-up time' between tasks. CP models can be particularly effective when the schedule is subject to time windows, when additional precedence relations are imposed between tasks, and when finding good feasible solutions is more important than proving optimality of a solution. One major benefit of CP over Mixed Integer Programming (MIP) models is that CP scales much better when the time granularity is increased; the CP model and associated algorithms are dependent on the number of tasks and resources rather than the number of time points.

Our CP models will use the basic structure described above, based on disjunctive scheduling. In our case, however, the number of (optional) tasks is proportional to the number of time points, and consequently our CP models may potentially still have scalability issues. Nonetheless, our CP models are still much smaller than the corresponding MIP models. As we are mostly interested in finding good solutions in a reasonable amount of time, with provable optimality as secondary goal, the investigation of the performance of CP models on this challenging problem domain seems warranted. In fact, our experimental results will demonstrate that our CP models are competitive with a dedicated heuristic for this problem, while in some cases it can find superior solutions. Moreover, when we apply our dedicated search heuristic, our CP approach can outperform the existing MIP-based heuristic both in terms of solution quality and solving time.

This paper is organized as follows. In the next section, we present a description of the problem under consideration. In section 3, we present an overview of the CP terminology used in the paper. In sections 4 and 5 we present two CP models. Our heuristic search algorithm is presented in section 6. Computational results to highlight the effectiveness of the proposed models and algorithm over existing methods on a set of large realistic test instances are presented in section 7. Finally, conclusions and directions for future research are presented in section 8.

2. Problem Description

Every year, LNG producers develop an annual delivery schedule for each of their customers. A delivery schedule specifies in detail when each cargo will be delivered and by what ship throughout the planned year. The annual delivery schedule attempts to best accommodate the expected requirements of each party for the planned year. During the plan year, the producers and customers seek to adhere to the agreed upon delivery schedule as closely as possible.

In this paper, we address the combined inventory management and cargo routing problem for the delivery of liquefied natural gas (LNG) that was presented by Goel et al. (2012). The problem can be described as follows. There is a set of liquefaction terminals with fixed storage capacities. Each liquefaction terminal has a given production profile for the planning horizon, as well as a limited number of berths for ships to load cargo on each day. At the other end of the supply chain, there is a set of regasification (regas) terminals that have fixed storage capacities and a limited number of berths for ships to unload cargoes. Each regas terminal has an overall contractual LNG demand that has to be delivered over the planning horizon, and a given baseline consumption profile (regas rate).

The LNG is delivered using a heterogeneous fleet of ships. We restrict our attention to full-load and full-discharge problems. In other words, LNG ships are always filled to their maximum capacity prior to departing an LNG terminal and are completely discharged prior to departing regasification terminals. Due to safety and qualification restrictions, each ship can only visit a select (pre-specified) set of terminals. It is common for some LNG to boil-off during a voyage. Similar to Goel et al. (2012), we assume that we know the discharge volumes for each ship and that these amounts are independent of voyage, waiting times, and time of year. Loading and unloading an LNG ship requires one day of operation during which a berth has to be occupied. Ships are allowed to wait outside a production (regas) terminal only before loading (discharging) their cargo. Ships do not occupy a berth during this process.

The problem has the following goals: delivering the contracted amount of gas to each customer and minimizing the interruptions to operations on both sides of the supply chain during the planning horizon. Interruptions include lost production stemming from running out of storage on the production side, and stock-outs due to the lack of inventory for consumption on the demand

side.

In this paper, we assume that the travel time between terminals is an integer multiple of days and does not have seasonality. Also, we do not consider variability in the regas rates.

3. Scheduling Terminology

As mentioned before, our CP models are based on applying a resource scheduling analogy to the IRP. As we will formulate our CP model using concepts from IBM ILOG CPOptimizer, we next briefly introduce those concepts using the terminology from Laborie (2009). Specifically, we will apply 'interval variables', 'sequence variables' and 'cumul functions'.

An *interval variable* x is a decision variable that can be used to represent a task to be scheduled over time. The domain of an interval variable is a subset of $\{\perp\} \cup \{[s, e) \mid s, e \in \mathbb{Z}, s \leq e\}$. An interval variable is *fixed* if its domain is reduced to a singleton, i.e., either $x = \perp$ which reflects that x is absent, or x is present and $x = [s, e)$ for some fixed s, e. Interval variables can thus naturally represent optional tasks to be scheduled: s represents the start time, e the end time, and $e - s$ the length of the task if it is present. In our application, we use interval variables to represent the loading and discharging of ships at the terminals.

We will apply the following expressions for interval variables. Here, x represents an interval variable and A a set of interval variables.

- $\texttt{lengthOf}(x)$ returns the length of x, i.e., $e - s$ if $x = [s, e)$,
- $\texttt{presenceOf}(x)$ returns whether x is present (true) or absent (false),
- $\texttt{startOf}(x, t)$ returns the start of x if it is present, and value t if x is absent,
- $\texttt{Alternative}(x, A)$ models an exclusive alternative among elements in A. If x is present, then exactly one element of A is present and x starts and ends together with this specific element. Interval variable x is absent if and only if all elements in A are absent.

The expressions $\texttt{lengthOf}(x)$, $\texttt{presenceOf}(x)$, and $\texttt{startOf}(x, t)$ can be used in a model to define or constrain the respective attributes of x. They can also be used to impose conditions on other constraints. The expression $\texttt{Alternative}(x, A)$ represents an 'alternative resource' constraint that can

be used, for example, to identify the machine at which a task is to be executed.

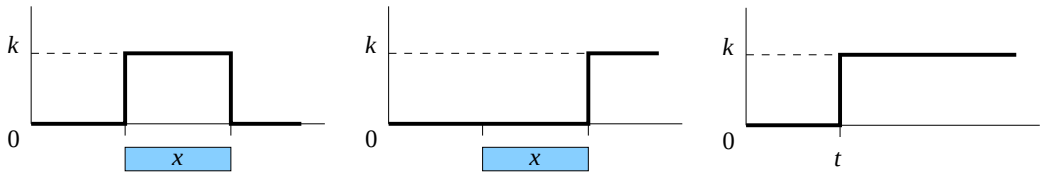
A *sequence variable* p is defined on a set of interval variables A. A value for p is a permutation of all present intervals of A. It can be used to represent a disjunctive resource, for which tasks cannot overlap in time. The no-overlap requirement also allows us to model sequence-dependent setup times between tasks. To this end, we associate each interval $a \in A$ with an integer 'type' $\theta(p, a)$, and define the transition distance between two types as a function $M : \mathbb{N} \times \mathbb{N} \to \mathbb{Z}$. That is, for two interval variables a and b, $M(\theta(p, a), \theta(p, b))$ expresses the minimal distance between the end of a and the start of b relative to the sequence variable p. For example, types may represent the color of a paint activity for a painting machine (in which case the transition time represents the cleaning time between different colors), or the geographic location of an activity in routing problems (in which case the transition time represents the distance between two activities). The no-overlap constraint can then be defined as $\texttt{NoOverlap}(p, M)$. In our application, we use sequence variables to represent the sequence of tasks, and hence the route, to be performed by a ship.

We will apply the following expressions for sequence variables. Here, p represents a sequence variable, x represents an interval variable, and A a set of interval variables.

- $\texttt{SequenceVar}(A, \theta)$ defines a sequence variable over A with type map θ,[1]
- $\texttt{First}(p, x)$ constrains x to be the first interval variable of p, if present,
- $\texttt{typeOfNext}(p, x, j_1, j_2)$ returns the type of the interval variable that is next to x in sequence p, value j_1 if x is the last interval variable of p, and value j_2 if x is absent (argument j_2 is optional and can be omitted if x is always present).

A *cumul function* can be used to represent (and constrain) the usage level of a resource over time. It aggregates individual contributions of interval variables, which are represented by *elementary* cumul functions. In this work, we apply the elementary functions `pulse`, `stepAtEnd`, and `step`.

[1]We note that both CPOptimizer and OPL do not have a compact definition for sequence variables. We therefore introduce the shorthand `SequenceVar`, to concisely represent our models.



a. `pulse`(x, k) b. `stepAtEnd`(x, k) c. `step`(t, k)

Figure 1: Example of three elementary cumul functions. Here, x is an interval variable, and k, t are integers.

These are illustrated in Figure 1. For an interval variable x, the function `pulse`(x, k) (Fig. 1.a) changes the resource level by k during the duration of x if it is present. This function is used to represent cumulative resources. The function `stepAtEnd`(x, k) (Fig. 1.b) can be used to represent resource production or consumption. It indefinitely changes the level of the resource by k at the end of x if it is present. It can also have three arguments: `stepAtEnd`$(x, k_{\min}, k_{\max})$. In this case the level of the resource is changed by a value k such that $k_{\min} \leq k \leq k_{\max}$. In this case, k is introduced as a decision variable to the problem. The 'constant' function `step`(t, k) (Fig. 1.c) changes the level of the resource by k at time point t. In our application, we use cumul functions to represent the inventory at the terminals.

Constraints on the resource level of a cumul function F can be set via the following expressions:

- `alwaysIn`(F, t_1, t_2, l, u) ensures that the level of F remains in the range $[l, u]$ for any point in the time interval $[t_1, t_2)$, or
- `alwaysEqual`(F, x, k) ensures that the level of F is equal to a given integer value k for any point between the start and end of interval variable x, if x is present.

Lastly, for a cumul function F and interval variable x, the expression `heightAtEnd`(x, F) returns the total contribution of x to F at the end of x. It returns 0 if x is not present.

4. CP1: Constraint Programming Model with Terminal Cargo Enumeration

In our first CP model, the tasks to be scheduled will correspond to the cargoes to be loaded at the production terminals, and the cargoes to be discharged at the regas terminals. We will collectively refer to these tasks as cargoes, and to both production and regas terminals simply as terminals. Each ship defines a disjunctive resource (tasks cannot overlap in time), as well as a production/consumption resource representing its inventory. Likewise, the inventory for each terminal is represented by a production/consumption resource.

Each terminal is seeking for its tasks to be executed (scheduled), while each task can be executed by at most one ship. This leads to a CP model with *optional tasks* (the cargoes) and alternative resources. Namely, we need to allocate each cargo to at most one ship, and we don't know in advance how many cargoes will be executed in total. We note that cargo sizes are determined implicitly through the assignment of cargoes to ships.

We next introduce the notation that will be used in our models.

4.1. Sets and Parameters

L	Set of production (liquefaction) terminals (integer)
R	Set of regasification terminals (integer)
J	Set of all terminals, $J = L \cup R$
T	Set of days (planning horizon), $T = \{1, 2, \dots, \lvert T \rvert\}$
D_j	Total demand for LNG over planning horizon at terminal $j \in R$
I_j	Set of possible cargoes to be loaded/discharged at terminal j
V	Set of ships (vessels)
$p_{j,t}$	Daily production rate at terminal $j \in L$ at time t
$d_{j,t}$	Daily regasification rate at terminal $j \in R$ at time t
g_j^0	Initial inventory at terminal j
γ_j	Storage capacity at terminal j
$\gamma_{v,j}$	Volume loaded/discharged by ship v at terminal j
$t_{j,i}^E, t_{j,i}^L$	i^{th} cargo at terminal j has time window $\left[t_{j,i}^E, t_{j,i}^L\right]$
$t_{j,i,v}^E, t_{j,i,v}^L$	i^{th} cargo at terminal j for ship v has time window $\left[t_{j,i,v}^E, t_{j,i,v}^L\right]$
τ_j	Minimum delay between successive cargoes at terminal j
$M(j, j')$	Travel time from terminal j to terminal j', including port time
j_v^0	Initial terminal of ship v
t_v^0	Time from which ship v is available
b_j	Number of berths at terminal j
$J_{j,v}^D$	Terminals that ship v can depart to after visiting terminal j
w_j^D	penalty for unmet demand at terminal $j \in R$
$w_{j,t}$	penalty for lost production/stockout at terminal j at time t

We also introduce the integer set $I_j := \{1, 2, \dots\}$ to represent the possible cargoes for terminal j. The size of this set depends on the production rate (resp. regasification rate) of the terminal and the time horizon. In our model, we will assume that if cargo i is scheduled, then also cargo $i - 1$ is scheduled (for $i > 1$). Therefore, the time windows for the cargoes $i \in I_j$ are preprocessed so that $t_{j,i}^E, t_{j,i}^L \leq |T|$ and $t_{j,i,v}^E, t_{j,i,v}^L \leq |T|$ for all $j, i \in I_j, v \in V$.

4.2. Variables and Cumul Functions

Interval variables:

$c_{j,i}$	i^{th} cargo at terminal j, for $i \in I_j$
$c_{j,i,v}$	i^{th} cargo at terminal j using ship v, for $i \in I_j$
a_v	unavailability of ship v before arrival at its initial destination
$slt_{j,t}$	loss in time slot t at terminal j (present if there is loss at t)
$slt2_{j,t}$	present if $slt_{j,t}$ is present, and it immediately succeeds $slt_{j,t}$

Here, loss refers to lost production for liquefaction terminals or stockout for regas terminals. The interval variable $slt2_{j,t}$ will be used in our model to represent that whenever a loss occurs, the resource must be at its bound in the next time slot.

All these interval variables except a_v are optional, i.e., they can be either present or absent, and have a fixed size of 1 (one time slot). Interval variable a_v is always present for each ship $v \in V$: it will be the first activity for ship v, with fixed length $[1, t_v^0)$.

Sequence variables and parameters:

seq_v	collects interval variables assigned to ship v
θ	type function $\theta\left(seq_v, c_{j,i,v}\right) = j$, $\theta\left(seq_v, a_v\right) = j_v^0$

Here, the type θ of an interval variable is its location, i.e., the function θ maps each interval to the location of its terminal. The type of interval a_v is the vessel's starting position j_v^0.

Integer variables:

$l_{j,t}$	loss at terminal j in time slot t

Recall that loss refers to lost production for liquefaction terminals or stockout for regas terminals.

Cumul functions:

$\tilde{p}_j$ cumulative production over time at terminal $j \in L$
$\tilde{d}_j$ cumulative regasification over time at terminal $j \in R$
$\tilde{z}_j$ cumulative volume loaded (discharged) over time at terminal $j \in J$
$\tilde{l}_j$ cumulative loss (lost production or stockout) over time at terminal $j \in J$

4.3. Constraints

We first introduce constraints to represent the cargoes at each terminal, and the ships to handle them:

$$c_{j,i} \in \left[t^E_{j,i}, t^L_{j,i} + 1\right) \quad \forall j, i \in I_j \tag{1}$$

$$c_{j,i,v} \in \left[t^E_{j,i,v}, t^L_{j,i,v} + 1\right) \quad \forall j, i \in I_j, v \tag{2}$$

$$\texttt{Alternative}(c_{j,i}, \{c_{j,i,v}\}_v) \quad \forall j, i \in I_j \tag{3}$$

$$\texttt{presenceOf}\,(c_{j,i}) \Rightarrow \texttt{presenceOf}\,(c_{j,i-1}) \quad \forall j, i \in I_j \tag{4}$$

$$\texttt{startOf}\,(c_{j,i-1}, |T|) + \tau_j \leq \texttt{startOf}\,(c_{j,i}, |T| + \tau_j) \quad \forall j, i \in I_j \tag{5}$$

$$\sum_i \texttt{pulse}\,(c_{j,i}, 1) \leq b_j \quad \forall j \tag{6}$$

Constraints (1) and (2) define the domains of the variables $c_{j,i}$ and $c_{j,i,v}$, respectively. Constraints (3) represent the assignment of terminal cargoes to ships, i.e., exactly one ship will execute cargo $c_{j,i}$, if present. Constraints (4) ensure that if the i^{th} cargo is present at terminal j, so is the $(i-1)^{th}$ cargo. Constraints (5) ensure the proper ordering, and minimum delay τ, for all consecutive cargoes at a terminal. Recall that the second argument in the function `startOf` is the value returned if the interval variable is absent. Lastly, constraints (6) ensures that the number of cargoes scheduled at each time point at a terminal does not exceed the number of berths.

The following set of constraints represent the ship schedules:

$$a_v \in \left[1, t^0_v\right) \quad \forall v \tag{7}$$

$$\texttt{lengthOf}(a_v) = t^0_v - 1 \quad \forall v \tag{8}$$

$$seq_v = \texttt{SequenceVar}\left(a_v \cup \{c_{j,i,v}\}_{j,i \in I_j}, \theta\right) \quad \forall v \tag{9}$$

$$\texttt{First}\,(seq_v, a_v) \quad \forall v \tag{10}$$

$$\texttt{typeOfNext}\left(seq_v, a_v, \phi^1\right) \texttt{ in } \left\{j^0_v, \phi^1\right\} \quad \forall v \tag{11}$$

$$\texttt{typeOfNext}\left(seq_v, c_{j,i,v}, \phi^1, \phi^2\right) \texttt{ in } J^D_{j,v} \cup \left\{\phi^1, \phi^2\right\} \quad \forall j, i \in I_j, v \tag{12}$$

$$\texttt{NoOverlap}\left(seq_v, M\right) \quad \forall v \tag{13}$$

Constraints (7) define the domains of variables a_v, while constraints (8) define the associated sizes. Constraints (9) define the sequence variable seq_v over its associated interval variables and types, for each ship v. The first activity of ship v must be a_v, as defined by constraints (10). The next two sets of constraints (11) and (12) limit the locations that can be visited after the initial location j^0_v, respectively cargo at location j. Here, ϕ^1 is a dummy value returned by `typeOfNext` for a last task completed by a ship, and ϕ^2 is a dummy value returned by `typeOfNext` for absent interval variables. The last constraint set (13) ensures that the interval variables do not overlap in time, and respect the transition times (distance) between locations.

Formally, by introducing types ϕ^1 and ϕ^2, we need to extend the definition of the transition matrix M. We do this as $M\left(j, \phi^1\right) = 1$, $M\left(j, \phi^2\right) = 0$.

The following set of constraints represent the terminal inventory and unmet demand:

$$slt_{j,t} \in [t, t+1) \quad \forall j \in J, t \in T \tag{14}$$

$$\tilde{p}_j = \sum_t \texttt{step}\left(t+1, p_{j,t}\right) \quad \forall j \in L \tag{15}$$

$$\tilde{d}_j = \sum_t \texttt{step}\left(t+1, d_{j,t}\right) \quad \forall j \in R \tag{16}$$

$$\tilde{z}_j = \sum_v \sum_{i \in I_j} \texttt{stepAtEnd}\left(c_{j,i,v}, \gamma_{v,j}\right) \quad \forall j \in J \tag{17}$$

$$\tilde{l}_j = \sum_t \texttt{stepAtEnd}\left(slt_{j,t}, 1, l_{j,t}\right) \quad \forall j \in J \tag{18}$$

$$\texttt{alwaysIn}\left(\texttt{step}\left(1, g^0_j\right) + \tilde{p}_j - \tilde{z}_j - \tilde{l}_j, 1, |T|, 0, \gamma_j\right) \quad \forall j \in L \tag{19}$$

$$\texttt{alwaysIn}\left(\texttt{step}\left(1, g^0_j\right) - \tilde{d}_j + \tilde{z}_j + \tilde{l}_j, 1, |T|, 0, \gamma_j\right) \quad \forall j \in R \tag{20}$$

Constraints (14) define the domain of the interval variable $slt_{j,t}$ that represents the loss at terminal j in time slot t. Constraints (15), (16), and (17) define cumul functions for the daily production, daily demand, and loading/discharge of cargo at the terminals. Constraints (18) represent the cumul function for the loss over time at terminal j. These are all added

together, with the starting inventory g_j^0, to form the overall inventory level of each terminal; in constraints (19) and (20) these levels are constrained to be in the range $[0, \gamma_j]$ for each terminal j. Note that the production and regasification terminals are handled separately.

In principle, the constraints (14)-(20) are sufficient to define the inventory aspect of the problem. However, the constraint propagation related to the interval variables representing the loss can be improved by adding the following redundant 'complementarity' constraints. These constraints represent that whenever a loss occurs, the resource must be at its lower bound (for regasification terminals) or its upper bound (for liquefaction terminals). To this end, we introduce a new optional interval variable $slt2_{j,t}$ that will represent that during time slot $t+1$ following a loss in slot t, the resource level is at its respective bound:

$$slt2_{j,t} \in [t+1, t+2) \quad \forall j \in J, t \in T \tag{21}$$

$$\texttt{presenceOf}\left(slt_{j,t}\right) = \texttt{presenceOf}\left(slt2_{j,t}\right) \quad \forall j \in J, t \in T \tag{22}$$

$$\texttt{alwaysEqual}\left(\texttt{step}\left(1, g_j^0\right) + \tilde{p}_j - \tilde{z}_j - \tilde{l}_j, slt2_{j,t}, \gamma_j\right) \quad \forall j \in L \tag{23}$$

$$\texttt{alwaysEqual}\left(\texttt{step}\left(1, g_j^0\right) - \tilde{d}_j + \tilde{z}_j + \tilde{l}_j, slt2_{j,t}, 0\right) \quad \forall j \in R \tag{24}$$

Constraints (21) define the domain and size of the interval variable $slt2_{j,t}$. Constraints (22) ensure that variables $slt_{j,t}$ and $slt2_{j,t}$ are both present or absent. Constraints (23) and (24) ensure that the resource level of j is equal to γ_j for liquefaction terminals and 0 for regasification terminals whenever a loss occurs at time slot t.

Lastly, we state the objective function as follows:

$$\begin{aligned}\text{minimize} \quad \sum_{j \in R} w_j^D \cdot \texttt{max}\left(0, D_j - \sum_{v, i \in I_j} \gamma_{v,j} \cdot \texttt{presenceOf}\left(c_{j,i,v}\right)\right) \\ + \sum_{j \in J} \sum_{t} w_{j,t} \cdot \texttt{heightAtEnd}\left(slt_t, \tilde{l}_j\right)\end{aligned}$$

Here, the first term represents the penalty for under-delivery, weighted by w_j^D for regasification terminal $j \in R$. The second term represents the penalty for lost production and stockout, weighted by $w_{j,t}$ for terminal $j \in J$ and time point $t \in T$.

5. CP2: Constraint Programming Model with Ship Visit Enumeration

In the previous section, we defined cargoes from the perspective of the terminals. Here, we present an alternative CP model following from a 'dual' perspective, that of the ships. For each ship, we enumerate a maximal set of terminal visits that it can execute within the planning horizon. Each visit to a terminal corresponds to a task in a scheduling problem. The terminals correspond to resources. Each ship is seeking for its tasks to be executed. The constraint that a ship should travel from a production terminal to a regas terminal and vice versa is implicitly captured by restricting the domains of the variables. Cargo sizes are determined implicitly based on the ship for the associated task.

5.1. Sets and Parameters

We will re-apply the sets and parameters from Section 4.1. In addition, we need the following:

I_v	set of possible terminal visits for ship v
J_v	set of terminals compatible with ship v
$J_{v,i}$	if visit $i = 1$, $J_{v,i} = \{j_v^0\}$.
	if i is odd ≥ 1 and j_v^0 is a production terminal, $J_{v,i} = J_v \cap L$
	if i is odd ≥ 1 and j_v^0 is regas terminal, $J_{v,i} = J_v \cap R$
	if i is even and j_v^0 is a production terminal, $J_{v,i} = J_v \cap R$
	if i is even and j_v^0 is a regas terminal, $J_{v,i} = J_v \cap L$
$t_{v,i}^E$	earliest time i^{th} terminal visit for ship v can take place
t_v^{min}	minimum travel time between any production and any regas terminal at which ship v is allowed

5.2. Variables and Cumul Functions

The main change from the model in Section 4 is the definition of the interval variables. We now apply the following interval variables:

$c_{v,i}$	i^{th} terminal visit for ship v, for $i \in I_v$
$c_{v,i,j}$	i^{th} terminal visit for ship v at terminal j, for $i \in I_v$, $j \in J_{v,i}$

All these interval variables are optional and have a fixed size of 1 (one time slot). In addition, the model will apply interval variables $slt_{j,t}$ and $slt2_{j,t}$ as defined in Section 4. The model will also apply sequence variable seq_v

from Section 4. Consequently, we redefine type function θ to operate on the interval variables $c_{v,i,j}$, as $\theta\left(seq_v, c_{v,i,j}\right) = j$. Lastly, the model will apply the cumul functions as defined in Section 4.

5.3. Constraints

We first present the constraints that represent the terminal visits for each ship, and their allocation to the terminals:

$$c_{v,i} \in \left[t^E_{v,i}, t^L_{v,i}\right) \quad \forall v, i \in I_v \tag{25}$$

$$c_{v,i,j} \in \left[t^E_{v,i}, t^L_{v,i}\right) \quad \forall v, i \in I_v, j \in J_{v,i} \tag{26}$$

$$\texttt{Alternative}\left(c_{v,i}, \left\{c_{v,i,j}\right\}_{j \in J_{v,i}}\right) \quad \forall v, i \in I_v \tag{27}$$

$$\texttt{presenceOf}\left(c_{v,i}\right) \Rightarrow \texttt{presenceOf}\left(c_{v,i-1}\right) \quad \forall v, i \in I_v \tag{28}$$

$$\texttt{startOf}\left(c_{v,i-1}, |T|\right) + t^{min}_v \leq \texttt{startOf}\left(c_{v,i}, |T|\right) \quad \forall v, i \in I_v \tag{29}$$

$$\sum_{v,i} \texttt{pulse}\left(c_{v,i,j}, 1\right) \leq b_j \quad \forall j \tag{30}$$

Constraints (25) and (26) define the domain of interval variables $c_{v,i}$ and $c_{v,i,j}$, respectively. Constraints (27) represent the assignment of ship visits to terminals, i.e., visit $c_{v,i}$ will be assigned to exactly one terminal $j \in J_{v,i}$, if present. Constraints (28) make sure that if visit $i+1$ takes place, so does visit i for ship v. Constraints (29) ensure that visits i and $i-1$ respect a minimum time interval. Lastly, constraints (30) represent the berth capacity.

The following set of constraints represent the ship schedules:

$$seq_v = \texttt{SequenceVar}\left(\left\{c_{v,i,j}\right\}_{i \in I_v, j \in J_{v,i}}, \theta\right) \quad \forall v \tag{31}$$

$$\texttt{NoOverlap}\left(seq_v, M\right) \quad \forall v \tag{32}$$

Since in this model the interval variables $c_{v,i,j}$ are defined from the ship's perspective, these constraints are more compact than the corresponding set of constraints (7)-(13).

The constraints representing the terminal inventory and unmet demand are equivalent to those from Section 4, apart from the substitution of $c_{v,i,j}$ in

the cumul function definition of $\tilde{z}_j$:

$$slt_{j,t} \in [t, t+1) \quad \forall j \in J, t \in T \tag{33}$$

$$\tilde{p}_j = \sum_t \texttt{step}\left(t+1, p_{j,t}\right) \quad \forall j \in L \tag{34}$$

$$\tilde{d}_j = \sum_t \texttt{step}\left(t+1, d_{j,t}\right) \quad \forall j \in R \tag{35}$$

$$\tilde{z}_j = \sum_v \sum_{i \in I_j} \texttt{stepAtEnd}\left(c_{v,i,j}, \gamma_{v,j}\right) \quad \forall j \in J \tag{36}$$

$$\tilde{l}_j = \sum_t \texttt{stepAtEnd}\left(slt_{j,t}, 1, l_{j,t}\right) \quad \forall j \in J \tag{37}$$

$$\texttt{alwaysIn}\left(\texttt{step}\left(1, g_j^0\right) + \tilde{p}_j - \tilde{z}_j - \tilde{l}_j, 1, |T|, 0, \gamma_j\right) \quad \forall j \in L \tag{38}$$

$$\texttt{alwaysIn}\left(\texttt{step}\left(1, g_j^0\right) - \tilde{d}_j + \tilde{z}_j + \tilde{l}_j, 1, |T|, 0, \gamma_j\right) \quad \forall j \in R \tag{39}$$

Similar to the previous model, we can apply the complementarity constraints (21)-(24) to improve the propagation of the loss variables.

The objective function is stated as:

$$\text{minimize} \quad \sum_{j \in R} w_j^D \cdot \texttt{max}\left(0, D_j - \sum_{(v,i)|j \in J_{v,i}} \gamma_{v,j} \cdot \texttt{presenceOf}\left(c_{v,i,j}\right)\right) + \sum_{j \in J} \sum_t w_{j,t} \cdot \texttt{heightAtEnd}\left(slt_t, \tilde{l}_j\right)$$

6. Cumulative Volume Lower Bound Based Search

In this section, we present a heuristic method to generate good feasible solutions for models CP1 and CP2. The proposed algorithm is based on the insight that the overall objective for the problem will be minimized if we maximize loadings at the production terminal, and discharges at the regas terminals such that the loads and discharges take place at regular intervals. Such a schedule will minimize the likelihood of violating inventory constraints, and maximize the likelihood of delivering the contractual volumes to the regas terminals.

Let $z_{j,t}$ represent the cumulative volume loaded (discharged) at production (regas) terminal j until t. Let $l_{j,t}$ and $g_{j,t}$ represent the loss and inventory,

respectively, at terminal j at time t. We have

$$g_{j,t} = g_j^0 + \sum_{t' \leq t} (p_{j,t'} - l_{j,t'}) - z_{j,t} \quad \forall j \in L, t \in T \tag{40}$$

If the total lost production over the entire planning horizon at production terminal j is λ_j, we have

$$\sum_{t' \leq t} l_{j,t'} \leq \lambda_j \quad \forall j \in L, t \in T. \tag{41}$$

Together with the inventory capacity constraints $g_{j,t} \leq \gamma_j, \forall j \in L, t \in T$, it follows that

$$z_{j,t} \geq g_j^0 - \gamma_j + \sum_{t' \leq t} p_{j,t'} - \lambda_j \quad \forall j \in L, t \in T \tag{42}$$

In the context of our constraint programming models, inequality (42) can be modeled as follows

$$\tilde{z}_j \geq \texttt{step}\left(1, g_j^0 - \gamma_j - \lambda_j\right) + \tilde{p}_j \quad \forall j \in L \tag{43}$$

Observe that the time dependency is implicit in the CP formulation, since $\tilde{z}_j$ and $\tilde{p}_j$ are cumul functions over the time horizon T.

Similarly, if λ_j is the total stockout over the planning horizon for regas terminal j, we can show that

$$z_{j,t} \geq -g_j^0 + \sum_{t' \leq t} d_{j,t'} - \lambda_j \quad \forall j \in R, t \in T \tag{44}$$

Inequality (44) can be modeled in our CP models as follows

$$\tilde{z}_j \geq \texttt{step}\left(1, -g_j^0 - \lambda_j\right) + \tilde{d}_j \quad \forall j \in R \tag{45}$$

As we will demonstrate with Theorem 1, constraints (43) and (45) not only limit the total losses at terminal j to be no greater than λ_j, but also force the model to limit the search to solutions where loadings and discharges at a terminal take place at "regular" intervals by enforcing a lower limit on the volume loaded or discharged at each terminal at each time period. Figure 2 illustrates the cumulative lower bound implied by constraint (43) on the

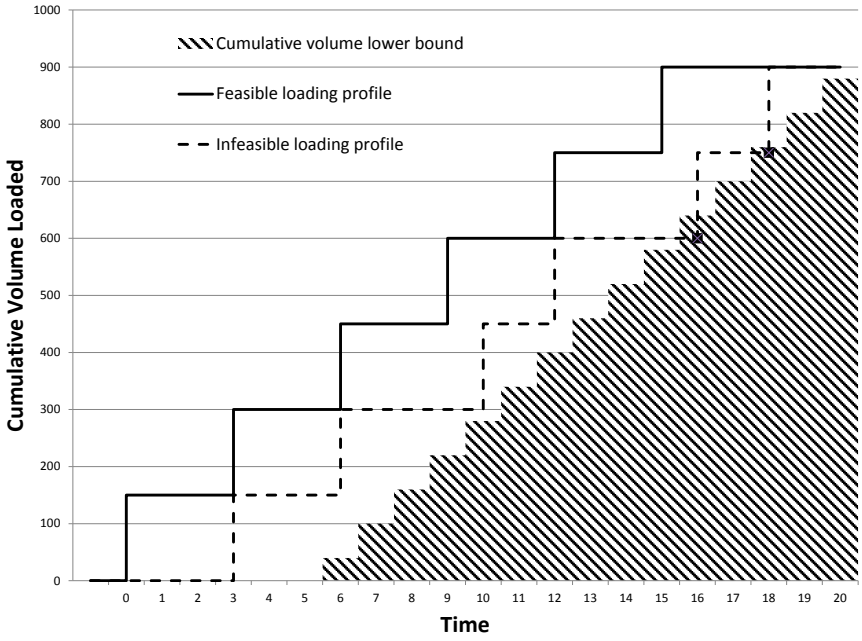


Figure 2: A feasible and an infeasible loading schedule based on constraint (43) for illustrative example

volume loaded at a production terminal for a notional example with $p_{j,t} = 60, g_j^0 = 0, \gamma_j = 320, \lambda_j = 0$. The shaded region represents the infeasible space defined by the constraint. This constraint will restrict the search to solutions where the cumulative function representing volume loaded at this terminal does not fall below curve shown. The figure also shows two loading profiles that each load 900 units with a different schedule. One of the loading profiles is feasible for constraint (43), while the other one is infeasible since it intersects with the infeasible space twice.

The tightness of constraints (43) and (45) can be controlled by adjusting (increasing) the parameter λ, which represents the vector of lost production at production terminals and stockouts at regas terminals. In the rest of this paper, we refer to λ as the loss vector. The term loss at a terminal represents the lost production for production terminals, and stockout for regas terminals.

Theorem 1 shows that for given loss vector λ^*, constraints (43) and (45) when applied to CP1 or CP2 restrict the feasible region to solutions where the lost production or stockout at any terminal at any time is no greater

than that in the solution defined by λ^*.

Theorem 1. *Let λ^* represent the loss vector for a solution to model CP1 (or CP2). There does not exist a feasible solution to CP1 (or CP2) with loss vector λ^s such that $\exists j \in (L \cup R), \lambda_j^s > \lambda_j^*$ and that satisfies constraints (43) and (45) as defined for $\lambda = \lambda^*$.*

Proof. We prove the theorem for CP1 by contradiction. The proof for CP2 is similar. Suppose there exists a solution to CP1 that satisfies constraints (43) and (45) defined for $\lambda = \lambda^*$. Further, suppose that the solution has a loss vector λ^s, such that for terminal $j \in L, \lambda_j^s > \lambda_j^*$ (the proof for the case where $j \in R$ is similar).

Let $\lambda_j^s = \lambda_j^* + \epsilon$ where $\epsilon > 0$. Let $\tilde{z}_j^s$ and $\tilde{l}_j^s$ represent the values for the cumulative functions $\tilde{z}_j$ and $\tilde{l}_j$, respectively, in the solution under consideration.

From (43) we have

$$\tilde{z}_j^s \geq \texttt{step}\left(1, g_j^0 - \gamma_j - \lambda_j^*\right) + \tilde{p}_j$$

Equivalently, we have

$$\tilde{z}_j^s \geq \texttt{step}\left(1, g_j^0\right) - \texttt{step}\left(1, \gamma_j\right) - \texttt{step}\left(1, \lambda_j^*\right) + \tilde{p}_j \tag{46}$$

Let us define the cumulative function $\tilde{g}_j^s$ to represent the inventory in tank for terminal j in the solution under consideration. By definition

$$\tilde{g}_j^s = \texttt{step}\left(1, g_j^0\right) + \tilde{p}_j - \tilde{z}_j^s - \tilde{l}_j^s \tag{47}$$

Combining (46) and (47), we get

$$\tilde{g}_j^s + \tilde{l}_j^s \leq \texttt{step}\left(1, \gamma_j\right) + \texttt{step}\left(1, \lambda_j^*\right) \tag{48}$$

We use the notation $F|_t$ to indicate the value of a cumul function F at time t. Let

$$t^s = \min\left\{t \mid \tilde{l}_j^s\Big|_t = \lambda_j^s\right\}. \tag{49}$$

By definition of $\tilde{l}_j^s$, $\tilde{l}_j^s\Big|_{|T|} = \lambda_j^s$. Hence, such t^s exists.

Considering (48) at t^s, we have

$$\tilde{g}_j^s\big|_{t^s} + \tilde{l}_j^s\Big|_{t^s} \leq \gamma_j + \lambda_j^*$$

Since $\tilde{l}_j^s\Big|_{t^s} = \lambda_j^s = \lambda_j^* + \epsilon$, we get

$$\tilde{g}_j^s\big|_{t^s} \leq \gamma_j - \epsilon \tag{50}$$

By definition of t^s, we have $\tilde{l}_j^s\Big|_{t^s-1} < \tilde{l}_j^s\Big|_{t^s}$. Thus, from (18) we get $\texttt{presenceOf}\,(slt_{j,t^s}) = True$. Then from (22) we get $\texttt{presenceOf}\,(slt2_{j,t^s}) = True$. Next, from (23) we get

$$g_j^0 + \left(\tilde{p}_j - \tilde{z}_j - \tilde{l}_j\right)\Big|_{t^s} = \gamma_j$$

Combining this with (47), we get

$$\tilde{g}_j^s\big|_{t^s} = \gamma_j \tag{51}$$

(50) and (51) are contradictory since $\epsilon > 0$. □

The proposed cumulative volume lower bound based search is a four phase algorithm. In phase 0, the algorithm finds a first feasible solution by solving the model in full space until a feasible solution is generated. At each iteration of phases 1 and 2, the algorithm limits the search to solutions such that the total loss at any terminal is smaller than a specified maximum value. This maximum loss at any terminal is set to be no greater than the loss at that terminal in the incumbent solution. In phase 1, we set the maximum loss at each terminal to be a fraction of the total loss at the terminal in the incumbent solution. This fraction is chosen to be the same for all terminals. At each iteration in phase 2 on the other hand, we reduce the maximum loss for only one terminal at a time. Therefore phase 1 seeks to reduce the losses across all terminals simultaneously while phase 2 involves seeking reduction in loss at one terminal at a time. Finally, in phase 3 the algorithm continues to search in the region where it found the best solution until a stopping criterion is met.

As mentioned above, in phases 1 and 2 we seek to find progressively better solutions by monotonically decreasing the total loss at each terminal. If at any iteration we are unable to find a solution in the reduced space defined by the maximum loss vector, we backtrack by choosing a less aggressive value for the maximum loss vector. This is achieved by setting the maximum loss vector to be a bigger fraction of the loss vector in the incumbent solution.

Note that the overall objective involves minimizing the total unmet de-

mand at the regas terminals in addition to minimizing loss production and stockouts. Progressively tightening constraints (43) and (45) by choosing monotonically increasing values for λ ensures that the lost production and stockouts are monotonically decreasing, but not the overall objective. Hence, we introduce constraint (52) into the model at each iteration to exclude solutions that have a worse objective value than that of the incumbent (represented by Φ).

$$\sum_{j \in R} w_j^D \cdot \texttt{max}\left(0, D_j - \sum_{v,i \in I_j} \gamma_{v,j} \cdot \texttt{presenceOf}\left(c_{j,i,v}\right)\right) + \sum_{j \in J} \sum_{t} w_{j,t} \cdot \texttt{heightAtEnd}\left(slt_t, \tilde{l}_j\right) \geq \Phi \quad (52)$$

The cumulative volume lower bound based search is presented in Algorithm 1. This algorithm can be applied to both models CP1 and CP2. We refer to the selected model simply as CP. Let $CP'(\lambda, \Phi)$ represent the model obtained by adding constraints (43), (45) and (52) to model CP, where λ and Φ represent the loss vector and the objective value, respectively, in the incumbent solution.

In algorithm 1, we use the following notation.

Φ^{inc}	Objective value of the incumbent solution
λ^{inc}	Loss vector in the incumbent solution
λ^{max}	Maximum loss vector used to define the search space
λ^{prev}	Maximum loss vector used in last search that generated a new incumbent
ξ_k^{ρ}	Scale factor used in phase ρ to calculate λ^{max} based on λ^{inc}
k^{max}	Number of scale factors to consider in phases 1 and 2

7. Computational Results

In this section, we present computational results to compare the performance of the CP models and the proposed search algorithm with the MIP based approaches of Goel et al. (2012), which represent the state of the art. All results have been obtained on a 64-bit Windows PC with two Intel Xeon quad core processors and 24 GB RAM. All MIP models are solved using CPLEX 11.1 while all CP models are solved using CPOptimizer 12.1. All runs on constraint programming models involved partitioning the search into

Algorithm 1 Cumulative Volume Lower Bound Based Search

{Phase 0}
Generate initial feasible solution to (CP)
Update λ^{inc}, Φ^{inc}; $\lambda^{prev} := \lambda^{inc}$

{Phase 1}
$k := 1$
while $k \leq k^{max}$ **do**
  $\lambda^{max} := \lambda^{inc} \cdot \xi_k^1$
  solve $CP'\left(\lambda^{max}, \Phi^{inc}\right)$
  if found solution **then**
    update $\lambda^{inc}, \Phi^{inc}$
    $\lambda^{prev} := \lambda^{max}$
  else
    $k := k + 1$
  end if
end while

{Phase 2}
for all terminals j **do**
  $k := 1$
  while $k \leq k^{max}$ **do**
    $\lambda_j^{max} := \lambda_j^{inc} \cdot \xi_k^2$
    for all terminals $j' \neq j$ **do**
      $\lambda_{j'}^{max} := \lambda_{j'}^{inc}$
    end for
    solve $CP'\left(\lambda^{max}, \Phi^{inc}\right)$
    if found solution **then**
      update $\lambda^{inc}, \Phi^{inc}$
      $\lambda^{prev} := \lambda^{max}$
    else
      $k := k + 1$
    end if
  end while
end for

{Phase 3}
$\lambda^{max} := \lambda^{prev}$
solve $CP'\left(\lambda^{max}, \Phi^{inc}\right)$

two phases. The first search phase involved branching on the cargo interval variables ($c_{j,i,v}$ for model CP1 and $c_{v,i,j}$ for model CP2), while the second search phase involved branching on the remaining variables.

Table 1 reports the specifications for the instances used in the study. These instances reflect realistic data. Instances P1-P14 are based on the instances presented by Goel et al. (2012). The data on some of these instances have minor differences compared to the instances presented by Goel et al. (2012) since we assume fixed regas rates and non-seasonal travel times in this paper. The remaining instances are newer and in most cases, larger instances. Table 1 reports the number of production terminals ($|P|$), regas terminals ($|R|$), ships ($|V|$); the number of continuous and integer variables, and the number of constraints in the MIP model; the number of variables and constraints, and the memory usage for the constraint programming models CP1 and CP2 at the end of a 5 hour solve. We also report the best-known solution (BKS) for each instance. The table clearly emphasizes the growth in the size of the MIP model with instance size. Note that it is not meaningful to directly compare the sizes of the constraint programming models with those of the MIP model since the variable and constraint constructs in constraint programming internally contain much heavier data structures than the MIP model. However, it is clear that CP2 is significantly smaller in size than CP1 in terms of the number of variables in constraints. This explains the scalability of CP2 compared to CP1 in terms of memory usage.

Tables 2 and 3 compare the performance of MIP and CP-based approaches in solving these instances. Specifically, we compare the performance of solving the MIP model using CPLEX (referred to as MIP) with that of solving CP1 and CP2 with CPOptimizer. All these methods are run to a 5 hour CPU limit. We also present the performance of the MIP-based heuristic (referred to as MIP-Heur) presented by Goel et al. (2012). We use the lexicographic ordering method presented by those authors. This algorithm is run to completion.

Table 2 presents the first feasible (FF) and the time to first feasible (TFF) solution for each of the four methods. Clearly we can observe that first feasible solutions generated by MIP are fairly poor, especially for the larger instances. First feasible solutions generated by the CP models are of similar or better quality than MIP-Heur. For 3 of the largest instances, CP1 is unable to find a feasible solution. The time to first feasible for the CP models (for all instances in the case of CP2, and for the smaller instances in the case of CP1) is also comparable or smaller than the time to first feasible for MIP-

				MIP			CP1			CP2			BKS
Instance	$\|P\|$	$\|R\|$	$\|V\|$	Cont. vars.	0-1 vars.	Constr.	Vars.	Constr.	Memory	Vars.	Constr.	Memory	
P1	2	1	6	2192	12812	6504	4328	4503	42.5 MB	2595	3087	30.3 MB	1722
P2	1	3	8	2924	16980	8635	6936	6563	67.2 MB	3575	4372	38.7 MB	1087
P3	2	1	10	2192	21029	9303	7330	7507	72.7 MB	2761	3447	32.4 MB	8029
P4	3	1	13	2922	27547	12196	9836	9940	93.8 MB	3974	5256	48.0 MB	0
P5	1	4	15	3655	31449	14269	13874	11558	134.4 MB	4719	6011	54.2 MB	2736
P7	1	6	6	5117	18335	10812	6768	6744	63.2 MB	5618	6142	53.9 MB	1510
P10	1	4	27	3655	95788	32744	38905	29158	338.7 MB	7967	11695	108.5 MB	32362
P11	1	5	14	4398	34438	15625	13752	11569	134.1 MB	5906	7655	69.2 MB	345
P12	1	4	18	3655	37752	16491	15258	12222	136.7 MB	4502	5514	50.5 MB	631
P13	1	8	40	6579	132995	47364	54588	39332	503.3 MB	10077	13454	140.6 MB	1438
P14	1	10	69	8041	232856	79425	215290	132118	1.9 GB	15375	22089	203.1 MB	62090
P16	1	5	29	4398	81558	30217	29630	21783	280.9 MB	6848	9526	95.5 MB	1009
P18	1	13	56	10262	320988	101387	128152	85333	1.2 GB	19112	25932	272.3 MB	6253
P20	1	18	100	13927	476426	152261	337155	206676	3.0 GB	26783	37401	379.9 MB	7354
P22	1	8	32	6597	141134	47954	37443	28588	354.6 MB	12552	17572	166.1 MB	3263
P23	1	8	32	6597	382592	108836	50535	41680	470.6 MB	39012	53686	599.2 MB	2914
P24	1	9	42	7330	179916	60320	71428	49573	690.0 MB	11364	15025	150.4 MB	3426
P25	1	9	42	7330	277351	84931	79929	58251	747.6 MB	12734	16499	178.0 MB	2907

Table 1: Instance specifications. For each instance, we report the number of production terminals $|P|$, the number of regasification terminals $|R|$, the fleet size $|V|$. For the MIP model we report the number of continuous variables, binary variables, and constraints. For the CP models we report the number of variables and constraints as well as the memory usage at the end of a 5-hour solving process. The last column reports the best known solution (BKS).

	FF				TFF			
Instance	MIP	MIP-Heur	CP1	CP2	MIP	MIP-Heur	CP1	CP2
P1	39726	1797	1781	1774	3	2	1	1
P2	75063	1296	1411	1567	5	2	1	1
P3	75670	8389	8166	8085	5	2	1	1
P4	83815	400	655	647	12	2	1	1
P5	114396	3773	3878	4475	32	3	1	1
P7	41966	4195	1510	1510	4	2	1	1
P10	188535	32715	32929	32670	48	6	6724	2
P11	96776	505	570	570	15	3	5	1
P12	105989	656	631	676	16	4	24	1
P13	183048	8176	5025	5543	101	11	1138	7
P14	494236	70474	inf	63352	2395	18	inf	4
P16	138856	2676	4412	4443	43	7	223	1
P18	320962	11382	inf	10266	821	22	inf	12
P20	525058	19596	inf	18123	1905	34	inf	22
P22	214438	13591	6163	6847	70	10	327	2
P23	214438	7135	3658	3862	590	22	1117	132
P24	245312	10350	5601	5319	266	12	2001	2
P25	245312	7743	5932	5633	344	17	2133	2

Table 2: First feasible solutions generated by MIP and CP based approaches. For each instance we report the objective value of the first feasible solution (FF) found by MIP, MIP-Heur, CP1, and CP2. We also report the solving time (in seconds) to the first feasible solution (TFF) for MIP, MIP-Heur, CP1, and CP2.

Heur. This is significant considering that MIP-Heur uses a problem specific heuristic to generate the first feasible solution.

Table 3 presents the best feasible (BF), and the time to best feasible (TBF) solution for each of the four methods. The best feasible solutions generated by the CP models are comparable to, and in some cases significantly better than those generated by MIP-Heur. For the smaller instances, the time to best feasible for the CP models is significantly higher than MIP-Heur. For the larger instances, MIP-Heur and CP approaches have mixed

	BF				TBF			
Instance	MIP	MIP-Heur	CP1	CP2	MIP	MIP-Heur	CP1	CP2
P1	1743	1743	1743	1743	17901	60	558	117
P2	1087	1132	1157	1157	16369	45	12418	15334
P3	8029	8077	8029	8029	282	40	13137	3256
P4	0	0	14	0	7522	79	17387	1135
P5	2736	3628	3017	2736	10108	115	16394	3757
P7	1622	1919	1510	1510	17630	1453	379	1
P10	37076	32520	32620	32437	18001	825	15810	3986
P11	6269	375	360	345	6459	2332	1382	6075
P12	5481	631	631	631	17753	540	24	16837
P13	14621	1656	4475	1438	9439	27076	5699	11885
P14	356087	62090	inf	62962	13570	15916	inf	17241
P16	10405	1762	3991	1738	10633	3164	17620	17910
P18	320962	7441	inf	6253	821	25297	inf	17813
P20	283826	11167	inf	11655	2450	81299	inf	16515
P22	87606	5612	5532	3263	109	3386	17804	16330
P23	135069	4599	3308	3232	821	9121	17351	14623
P24	121513	5600	5550	3499	5069	4728	16759	17801
P25	113970	4678	5311	3341	5980	8816	17885	16581

Table 3: Best feasible solutions generated by MIP and CP based approaches. For each instance we report the objective value of the best feasible solution (BF) found by MIP, MIP-Heur, CP1, and CP2. We also report the solving time (in seconds) to the best feasible solution (TBF) for MIP, MIP-Heur, CP1, and CP2.

performance in terms of getting to their best feasible solutions faster.

In summary, it can be concluded from Tables 2 and 3 that within the time limits provided, the CP models have significantly better performance than solving the model as a MIP when all sets of models are solved simply using commercials solvers. Furthermore, by formulating the problem as a CP model and using a commercial CP solver we can generate solutions of comparable or better quality than the specialized heuristic (MIP-Heur) presented by Goel et al. (2012).

Next, we evaluate the performance of the CP-based search algorithm presented in this paper. We compare the performance of this method against MIP-Heur since that is the best algorithm among the existing methods. In the results presented, the sub-problems solved in phases 1 and 2 of the proposed algorithm are terminated based on optimality, infeasibility, branch limit and/or a solution limit, whichever comes first. At the beginning of phase 2, we order the set of terminals in decreasing order of the corresponding losses in the incumbent solution. In phase 2, we iterate through the set of terminals in this sorted order starting from the terminal that has the largest total loss in the solution found at the end of phase 1. This enables us to search early in the regions with the maximum potential for improvement. For instances where the first feasible solution has an objective value greater than 10,000, we choose $k^{max} = 5$. We choose $\xi^1 = \xi^2 = [0.9, 0.95, 0.97, 0.99, 0.999]^T$ for these instances. For all other instances we choose $k^{max} = 4$ and $\xi^1 = \xi^2 = [0.9, 0.95, 0.97, 0.99]^T$. This approach sets the desired accuracy level at 0.1% for instances with a large objective value, and at 1% for instances with a relatively smaller objective value.

We report solutions for the proposed search algorithm based on 2 different configurations of the termination criteria. CP-Heur-nx-ksol refers to the configuration of the algorithm where sub-problems are solved with a branch limit of $(n \cdot \sum_v |I_v|)$ and a solution limit of k. Similarly, CP-Heur-nx refers to the configuration of the algorithm where sub-problems are solved with a branch limit of $(n \cdot \sum_v |I_v|)$ but without a solution limit.

Since the CP-based algorithms are terminated with an ad hoc time-limit whereas MIP-Heur has a convergence based stopping criterion, we have designed a methodology to compare the quality of solutions obtained by the proposed search algorithm, and its speed in generating the solutions. For each instance, we define $BCS_{a,b}$ as the best common solution obtained by algorithms a and b. Specifically, $BCS_{a,b} = \max(BF_a, BF_b)$. The speed-up of algorithm a versus b is defined in terms of the times taken by the two algorithms to get a solution at least as good as $BCS_{a,b}$.

Similarly, for each instance we define $MCTT_{a,b}$ as the Maximum Common Total Time elapsed for algorithms a and b. Specifically, $MCTT_{a,b} = \min(TBF_a, TBF_b)$. We quantify the quality of solutions obtained by algorithms a and b as the difference in the best solutions obtained by these algorithms at this common time relative to the best known solution for the instance. Specifically, we report the gap in quality of solutions as $\frac{100 \cdot (S_a - S_b)}{BKS}$, where S_a and S_b represent the best solutions obtained by algorithms a and b

	CP2	CP-Heur-4x-1sol	CP-Heur-32x
P1	0.31	7.2	9
P2	0.0	1.18	0.39
P3	11	11	11
P4	0.07	1.8	0.24
P5	0.53	25.5	25.5
P7	916	916	916
P10	1.10	3.88	1.29
P11	384.67	76.93	72.13
P12	2.82	0.78	0.97
P13	2.30	1.21	6.35
P14	0.46	3.56	6.11
P16	0.25	15.84	2.25
P18	22.09	3.51	1.09
P20	2.02	1.97	6.08
P22	260.67	2.65	1.83
P23	50.42	48.58	47.54
P24	1300	1300	1300
P25	5.51	33.74	10.31
Geo mean	4.55	10.5	7.86

Table 4: Speed-up of CP based algorithms relative to MIP-Heur in reaching a common solution

respectively at $MCTT_{a,b}$.

In Table 4 we report the speed-ups for each of the CP based methods relative to MIP-Heur. Speed-ups greater than 1 indicate that the CP based method is faster than MIP-Heur in getting to a solution of similar quality. Simply solving CP2 using CPOptimizer achieves an average speed-up of factor 4.55. Our dedicated search algorithm achieves a speed-up of factor 7-10 on average.

In Table 5 we report the % gap in solution objective for each of the CP based methods relative to MIP-Heur at the maximum common total time defined above. Gaps are evaluated relative to the best known solution for the instance. A positive gap indicates that the CP based method finds a better solution than MIP-Heur at the given time limit. An "inf" value indicates

	% gap relative to BKS		
	CP2	CP-Heur-4x-1sol	CP-Heur-32x
P1	-1.34	0.41	0.93
P2	-24.2	4.14	-1.1
P3	0.31	inf	inf
P4	-350	244	-393
P5	3.76	34.32	27.81
P7	inf	inf	inf
P10	0.21	0.32	0.47
P11	4.35	39.13	42.03
P12	0	-0.32	0
P13	17.59	45.55	101.11
P14	-1.83	2.51	13.17
P16	-1.78	66.2	65.21
P18	20.28	11.23	4.65
P20	16	26.52	57.33
P22	12.2	61.26	23.14
P23	46.6	47.53	25.29
P24	43.67	54.79	50.55
P25	30.44	15.65	60.92

Table 5: % Gap in solution quality of specified algorithm and MIP-Heur relative to best known solution (BKS) after a common solve time

that MIP-Heur cannot find any solution by the time (which coincides with MCTT) the CP based method terminates after successfully finding a solution. Clearly, we can see that at MCTT, solving CP2 with CPOptimizer, and the two configurations of the proposed search algorithm provide significantly better solutions than MIP-Heur for the larger instances.

8. Conclusions

In this paper, we have addressed the problem of optimal LNG inventory routing. Most of the previous work in this area has used a mixed integer programming based approach. We have presented two CP models, based on a disjunctive scheduling representation, as an alternative approach to

solving this problem. In addition, we have proposed an iterative heuristic search algorithm to generate good feasible solutions. We have shown that when using leading commercial solvers, the CP-based approach generates good solutions much faster than a MIP based approach. Specifically, given the same amount of time, the CP model when solved using a commercial solver has been shown to generate solutions of significantly higher quality than a specialized MIP based heuristic. Alternatively, for a given solution quality, the CP model when solved using a commercial solver is on average 4.55 times faster than the specialized MIP based heuristic. The CP based iterative heuristic is able to achieve even further improvements in speed (up to a factor 10.5 on average) and solution quality over the MIP based specialized heuristic.

References

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.

P. Baptiste, C. Le Pape, and W. Nuijten. *Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems*. Kluwer Academic Publishers, 2001.

P. Baptiste, P. Laborie, C. Le Pape, and W. Nuijten. Constraint-based scheduling and planning. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, chapter 22. Elsevier, 2006.

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

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.

P. Laborie. Ibm ilog cp optimizer for detailed scheduling illustrated on three problems. In *Proceedings of CPAIOR*, volume 5547 of *LNCS*, pages 148–162. Springer, 2009.