Solving a Supply-Delivery Scheduling Problem with Constraint Programming

Katherine Giles and Willem-Jan van Hoeve

Tepper School of Business, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213, USA
kgiles@tepper.cmu.edu, vanhoeve@andrew.cmu.edu

Abstract. We describe a constraint programming approach for a supply-delivery problem in the petrochemical industry, in which barges transport liquid material from supplier locations to downstream processing plants. The problem is to design a pickup-and-delivery route for each barge such that given minimum and maximum inventory levels at each location are met for a given fleet size. This optimization problem is part of a larger planning system to determine the fleet size, negotiate pickup windows and quantities, and design operational schedules. We evaluate our model on representative supply networks provided by BP North America, and contrast our results with those obtained by a mixed-integer programming approach.

1 Introduction

In the chemical processing industry, often material has to be processed by multiple plants in order to convert it into the final product. We consider such supply-delivery problem in the petrochemical industry, in which liquid material needs to be transported from supplier locations to downstream processing plants. The liquid material is transported by water, using so-called "tows" that consist of one power unit and two barges. We focus on the design of pickup-and-delivery schedules for the tows with the aim of satisfying minimum and maximum inventory levels at each of the supply locations and processing plants. The high-level objective is to minimize total cost, which is determined by the fleet size, the violation of time window constraints, and the violation of inventory (lower and upper) capacity constraints.

As a specific application, we study the supply-delivery problem that is operated by BP (formerly known as British Petroleum) in North America. BP employs the optimization problem described above within a larger planning system to determine the target fleet size, and to negotiate pickup windows and quantities. In addition, the solution to the optimization problem provides a basis for the operational schedule. Upon the start of the project, BP employed a mixed-integer programming (MIP) optimization model which had several drawbacks. First, the MIP model had difficulty finding optimal (or even good feasible solutions), already for relatively small instances. Second, larger instances posed challenges with respect to memory issues. Third, given the large computation

times, it was difficult to use the MIP model in the desired strategic planning context. In order to find solutions of sufficient quality, this approach required intensive manual interaction: The model would be solved in several iterations, in each of which the problem parameters (such as the fleet size or pickup windows) would be adjusted.

Given these challenges, and driven by the scheduling aspects of the problem, we therefore present an alternative optimization model based on constraint programming (CP). In particular, we will represent the problem as a constraint-based scheduling problem, using activities and resources. Our main finding is that the CP model scales much better than the MIP formulation that is currently in place. For certain representative instances considered, CP can find optimal solutions in a fraction of the time it takes MIP. The CP model is therefore much better suited than the existing MIP model to provide solutions in a broader planning context.

The remainder of the paper is structured as follows: In Section 2 we provide a brief review of the most relevant literature. We then give a detailed description of our problem in Section 3. This is followed by the constraint programming model in Section 4. We provide an evaluation of our model in Section 5, and conclude in Section 6.

2 Related Work

Our problem can be viewed as a variant of the maritime inventory routing problem. The basic maritime inventory routing problem (see [3]) 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. Christiansen and Fagerholt [4] provide a recent survey in ship routing and scheduling research.

A recent paper by Goel et al. [5] introduces a constraint programming model for a maritime inventory routing problem in the context of liquefied natural gas (LNG) tanker scheduling. It follows the approach of representing the routing problem as a constraint-based scheduling model, which has been successfully applied before to many industrial routing and scheduling problems [2, 1]. In particular, the routing problem is represented as a disjunctive scheduling problem, 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. In addition, the same tasks can be associated with resource constraints to model the inventory levels over time. The effectiveness of CP scheduling models for such complex inventory routing problems was one of the main motivations for considering CP for our problem.

Our problem does differ considerably from that in [5], and from the standard maritime inventory routing problem. First, we consider multiple products instead of a single commodity. Furthermore, the attributes of the three location types

(suppliers, plants, and customers) are not simply pickup and delivery nodes; the plants both consume and produce products, necessitating both pickup and delivery visits. Lastly, our vessels have two barges, each of which can carry a separate product.

3 Problem Description

We will next describe the specific application at BP in detail [6]. The supply network is defined by a set of suppliers S, a set of plants P, a set of customer locations C, and a set of products F; see Figure 1 for a schematic representation of a typical instance with four suppliers, two plants, and two customer locations. The suppliers provide 'feed' to the plants (to be picked up in specified time windows), which then convert this into different final products. In the example of Figure 1, the set of products is $F = \{\text{feed, product 1, product 2, product 3}\}$. For convenience, we define $F^- := F \setminus \{\text{feed}\}$ to be the set of final products only. The final products are shipped from plants to customer locations to meet the demand (or to avoid inventory overflow at the plant). In addition, some plants can directly meet the demand of a product instead of shipping it to a customer location. For example, in Figure 1, Plant 2 only produces final product 1, whose demand is met exclusively by prescheduled pickups. Lastly, some plants are connected to a pipeline which provides an alternate supply of feed (for example, Plant 1 in Figure 1). Both pipeline deliveries and prescheduled pickups are external events; no tows will be used for these activities.

We are given a discrete planning horizon (in days) $H = \{1, 2, \ldots, \hat{H}\}$, where $\hat{H}$ represents the end of the horizon. We are also given a set of tows $T = \{1, 2, \ldots, M\}$, where M is the maximum number of tows. Each tow consists of two barges and a power unit, and has a total capacity of 2Λ metric ton. The specific attributes (input data) for the suppliers, plants, customers, and tows are given in Table 1. The goal is to design a pickup-and-delivery schedule for the tows such that (ideally) all time window constraints and all inventory capacity constraints are respected.

While the structure provided by this network is very generic, our case has the following specific elements:[1]

- In our model, we do not explicitly convert feed into final products. Instead, the conversion is modeled implicitly by specifying daily production and consumption amounts. (Note that other materials are required for the conversion as well.)
- Suppliers are contractually bound to have sufficient feed available for any scheduled tow pickup that can occur over the scheduling horizon. We therefore do not need to represent the daily production of feed, nor the inventory levels, at each supplier location. (Our model is easily extended to handle this, however.)

[1] Via private communication with Norman Jerome, BP Americas.

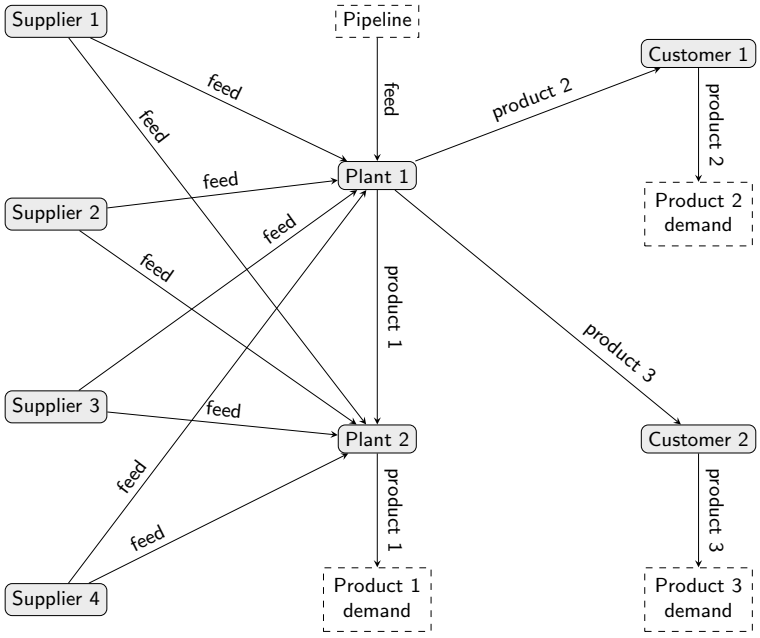

Fig. 1. Petrochemical supply-delivery network.

- The most important operational concern is to have sufficient feed available at the plants (i.e., the lower limit is typically binding). The upper capacity of the feed inventory at the plants is typically not binding.
- The maximum inventory capacity of the final products at the plants is typically not reached.
- The consumption rate of the final products at the customer locations is not part of our problem description. At the same time, the maximum inventory level is practically never binding for customer locations.

Given these considerations, as our specific objective we chose to minimize the total amount of feed underflow at the plant locations. However, we do present a generic CP model that can accommodate different objectives as well.

4 Constraint Programming Model

We next present our constraint programming model for the base problem: Given a fixed number of tows, find a supply-delivery schedule that minimizes the total feed underflow at the plants. We use the optimization modeling system AIMMS to express our model, using 'activities' and 'resources' for the constraint-based scheduling formulation [2]. We will first review the relevant AIMMS syntax, and then provide the details of the model.

Table 1. Attributes of suppliers, plants, customers, and tows.

W_s	Set of pickup windows for supplier $s \in S$,
$l_{i,s}$	Start date of pickup window $i \in W_s$ for $s \in S$,
$u_{i,s}$	End date of pickup window $i \in W_s$ for $s \in S$,
$V^{\text{pu}}_{p,f}$	Set of prescheduled visits to pick up product $f \in F^-$ for plant $p \in P$,
V^{del}_{p}	Set of prescheduled visits to deliver feed for plant $p \in P$,
$d^{pu}_{p,f,i}$	Date of visit $i \in V^{\text{pu}}_p$ for $p \in P$ and product $f \in F^-$,
$d^{del}_{p,i}$	Date of visit $i \in V^{\text{del}}_p$ for $p \in P$,
$a^{\text{pu}}_{p,f,i}$	Amount of $f \in F^-$ picked up at visit $i \in V^{\text{pu}}_{p,f}$ for $p \in P$,
$a^{\text{del}}_{p,i}$	Amount of feed delivered at visit $i \in V^{\text{del}}_p$ for $p \in P$,
$I^{\text{init}}_{f,i}$	Initial inventory of product $f \in F$ at location $i \in P \cup C$,
$I^{\text{min}}_{f,i}$	Minimum inventory of product $f \in F$ at location $i \in P \cup C$,
$I^{\text{max}}_{f,i}$	Maximum inventory of product $f \in F$ at location $i \in P \cup C$,
$\Pi_{f,p,d}$	Production amount of product $f \in F$ at plant $p \in P$ on date $d \in H$,
$\Gamma_{p,d}$	Consumption amount of feed at plant $p \in P$ on date $d \in H$,
$D_{i,j}$	Distance (in days) from location i to location j $(i, j \in S \cup P \cup C)$,
l_t	First destination for tow $t \in T$,
Λ	Barge capacity (500 metric ton in our data case).

4.1 AIMMS Syntax for Constraint-Based Scheduling

The activities and resources in AIMMS provide an interface to advanced scheduling constraints, in particular those available in IBM ILOG CP Optimizer [7]; readers familiar with the IBM ILOG CP Optimizer scheduling interface will recognize the similarities. Activities correspond to tasks to be executed over the time horizon. They have a start time, end time, and duration. Each activity also has a *schedule domain* which defines the range of possible dates for the start and end time. In addition, an activity can be *optional*, which means that its presence will be a decision variable. Activities can impact one or more resources; this is modeled at the resource level. In AIMMS, an activity A defines the following decision variables:

A.`Begin`	the start time of A,
A.`End`	the end time of A,
A.`Length`	the duration of A,
A.`Present`	the presence of A (with Boolean domain).

Several useful functions on activities are provided. For activity A and $d \in H$ we have:

- `ActivityBegin(`A`,`d`)`: Returns d if A is absent, and A.Begin if A is present.

Resources can be declared in two ways in AIMMS: Sequential or parallel. A sequential resource maintains a unary resource level (either 0 or 1), which means

that at most one task can be active at a time. The definition of a sequential resource includes the following attributes:

- Resource name and index set (possibly empty),
- *Activities:* List of activities that influence the resource,
- *Group set:* Set of groups in which the activities are divided,
- *Group definition:* Maps activities to group set elements,
- *Group transition:* Represents the transition/setup time between pairs of group elements,
- *First activity:* Reference to the first activity in the sequence.

Sequential resources have several useful associated functions. For sequential resource R, activity A, and $l, d \in H$, we have:

- `BeginOfNext(`R`,`A`,`l`,`d`)`: Returns d if A is absent, l if A is present and scheduled as last activity on R, and B.`Begin` if A is present and not scheduled as last activity on R, and B is the next activity of A scheduled on R.
- `EndOfNext(`R`,`A`,`l`,`d`)`: Returns d if A is absent, l if A is present and scheduled as last activity on R, and B.`End` if A is present and not scheduled as last activity on R, and B is the next activity of A scheduled on R.

For sequential resource R, activity A, and group elements l, l', we have:

- `GroupOfNext(`R`,`A`,`l`,`l'`)`: Returns l' if A is absent, l if A is present and scheduled as last activity on R, and the group of B on R if A is present and not scheduled as last activity on R, and B is the next activity of A scheduled on R.

For parallel resources, multiple tasks can be active simultaneously, as long as the activity level of the resource is within a given lower and upper bound. The activity level represents the cumulative resource value over time, and is influenced by the activities. The definition of a parallel resource includes the following attributes:

- Resource name and index set (possibly empty),
- *Activities:* List of activities that influence the resource,
- *Level range:* $\{L..U\}$ specifies that the activity level must be between L and U at each time point,
- *Initial level:* Specifies the initial activity level,
- *Begin change:* Specifies for each activity A by how much the activity level is changed at A.`Begin`,
- *End change:* Specifies for each activity A by how much the activity level is changed at A.`End`.

A useful function for a resource R and a time point $d \in H$ is the following:

- $R(d)$.`ActivityLevel`: Returns the activity level of R at time d.

Table 2. Variables for Modeling the Pickup and Delivery at the Locations.

$b^1_{t,i,f,p}$	amount of $f \in F^-$ picked up by barge 1 for $t \in T$ at plant p for visit $i \in N_p$,
$b^2_{t,i,f,p}$	amount of $f \in F^-$ picked up by barge 2 for $t \in T$ at plant p for visit $i \in N_p$,
$b^{\text{pu}}_{t,i,f,p}$	total amount of $f \in F^-$ picked up for $t \in T$ at plant p for visit $i \in N_p$,
$b^{\text{dem}}_{t,l,i,f}$	total amount of $f \in F^-$ delivered for $t \in T$, location $l \in L_f$, and $i \in N_l$,
$c_{p,d}$	amount of feed consumed at plant $p \in P$ on date $d \in H$,
$s_{p,d}$	amount of feed shortage (underflow) at plant $p \in P$ on date $d \in H$,
$g^{\text{del}}_{p,i}$	amount of feed delivered at plant $p \in P$ for visit $i \in V^{\text{del}}_p$,
$g_{t,p,i}$	amount of feed delivered with $t \in T$ at plant $p \in P$ for visit $i \in N_p$.

This function uses 'overloading' of the resource name; for example, if we define a resource indexed over i as $R(i)$, we can access the activity level at time $d \in H$ via $R(i,d)$.`ActivityLevel`.

We remark that the description above is limited to those concepts that are relevant to our paper. For a complete description of the scheduling functionality in AIMMS we refer to [8].

4.2 Modeling the Location Visits and Inventory Levels

There are three different types of locations: Suppliers, who are solely pickup nodes, customers, who are exclusively delivery nodes, and plants, which both produce and consume product and, as such, are both pickup and delivery nodes.

Location Visits Each possible visit by a tow to a location is indexed by a master set $N := \{1, 2, \dots\}$, for which the upper value is the maximum number of pickup windows for all the suppliers. That is, each tow cannot make more than $|N|$ visits to a location. One of the drivers of the computational efficiency of our model, however, is defining an auxiliary set $N_l \subseteq N$, with the maximum number of individual tow visits varying by location $l \in S \cup P \cup C$. For a supplier $s \in S$, the maximum number is equal to the number of pickup windows $|W_s|$, whereas for plants and customers, the maximum number can be adjusted by the user through the graphical interface, if desired. Restricting the set of location visits also restricts many of the index domains, so keeping its cardinality low greatly improves computational performance.

Integer Variables Due to the multiple products and split load functionality, we require additional variables to represent the amount of material picked up and delivered at each visit. We let $L_f \subseteq P \cup C$ be the set of locations that serve as demand point for final product $f \in F^-$. The list of variables we will use to represent the pickup and delivery at the locations is given in Table 2.

As we must distinguish between barge capacity and tow capacity, we introduce the following constraints, for $t \in T, p \in P, i \in N_p, f \in F^-$:

$$0 \leq b^1_{t,i,f,p} \leq \Lambda,$$
$$0 \leq b^2_{t,i,f,p} \leq \Lambda,$$
$$b^{\text{pu}}_{t,i,f,p} = b^1_{t,i,f,p} + b^2_{t,i,f,p}.$$

In addition, we ensure that each barge can only contain one product:

$$\sum_{f \in F^-} (b^1_{t,i,f,p} > 0) \leq 1 \qquad \text{for } t \in T, p \in P, i \in N_p,$$
$$\sum_{f \in F^-} (b^2_{t,i,f,p} > 0) \leq 1 \qquad \text{for } t \in T, p \in P, i \in N_p.$$

To model the feed shortage, we define the next constraints, for $p \in P, d \in H$:

$$0 \leq c_{p,d} \leq \Gamma_{p,d}, \tag{1}$$
$$0 \leq s_{p,d} \leq \Gamma_{p,d}, \tag{2}$$
$$c_{p,d} + s_{p,d} = \Gamma_{p,d}. \tag{3}$$

Unlike products, each tow picks up a "full load" (two barges) upon a supplier visit. Therefore, to model the delivery of feed with tows, we introduce the constraint $0 \leq g_{t,p,i} \leq 2\Lambda$, for $t \in T, p \in P, i \in N_p$. The other constraints for this purpose will be given in Section 4.5, as they depend on the definition of plant visit activities.

Activities The key activities in our model are the possible visits that each tow can make to each location. We define the following optional activities representing the i-th visit of tow t to the respective locations:

$$\texttt{VisitSupplier}(t, s, i) \text{ for } t \in T,\ s \in S,\ i \in N_s,$$
$$\texttt{VisitPlant}(t, p, i) \quad \text{for } t \in T,\ p \in P,\ i \in N_p,$$
$$\texttt{VisitCust}(t, c, i) \quad \text{for } t \in T,\ c \in C,\ i \in N_c.$$

Each of these activities has a fixed duration of one day, and a uniquely defined schedule domain. For $\texttt{VisitSupplier}(t, s, i)$, the schedule domain includes only those days that are part of the pickup windows. $\texttt{VisitPlant}(t, p, i)$ and $\texttt{VisitCust}(t, c, i)$ have schedule domains that begin either the day the first prescheduled tow is due to arrive, or the earliest day a tow can arrive with delivery of feed or product, derived from the travel time from the closest supplier/plant production node.

We also define (fixed) activities to represent the daily production or consumption, as well as the prescheduled visits, at each location:

$$\texttt{PlantProduction}(p, f, d) \text{ for } p \in P,\ f \in F^-,\ d \in H,$$
$$\texttt{PlantConsumption}(p, d) \quad \text{for } p \in P,\ d \in H,$$
$$\texttt{PreSchedPickUp}(p, f, i) \quad \text{for } p \in P,\ f \in F^-,\ i \in V^{\text{pu}}_{p,f},$$
$$\texttt{PreSchedDelivery}(p, i) \quad \text{for } p \in P,\ i \in V^{\text{del}}_{p}.$$

Each of these activities has a fixed duration of one day, and must be present. The activities $\texttt{PlantProduction}(p, f, d)$ and $\texttt{PlantConsumption}(p, d)$ must take place on day d, i.e., their associated start time variable is fixed to d. The start time variables of activities $\texttt{PreSchedPickUp}(p, f, i)$ and $\texttt{PreSchedDelivery}(p, i)$ are fixed to $d^{\text{pu}}_{p,f,i}$ and $d^{\text{del}}_{p,i}$, respectively. Therefore, these activities can be viewed as constants; they are necessary to model the resource levels, but their associated variables are fixed to given values.

We note that the activity $\texttt{PlantConsumption}(p, f, d)$ reduces the inventory level by $c_{p,d}$; any consumption that would reduce the inventory level below the lower bound is considered $s_{p,d}$, or underflow, as defined by constraints (1)-(3). A similar overflow variable could be modeled for $\texttt{PlantProduction}(p, f, d)$, but in this application, we found it more efficient to set hard upper bounds on product inventory levels and force the plants to ship product to customers to relieve any excess.

Location Resources We incorporate the above production, consumption, tow visits, and prescheduled events in defining resources to represent the inventory levels of the various products at each location, as follows:

Parallel resource: $\texttt{PlantProductInventory}(p, f)$ for $p \in P, f \in F^-$
Activities: $\texttt{PlantProduction}(p, f, d)$ for $d \in H$, $\texttt{VisitPlant}(t, p, i)$
$\texttt{PreSchedPickup}(p, f, i)$ for $t \in T$, $i \in N_p$
Level range: $\{I^{\min}_{f,p} \,..\, I^{\max}_{f,p}\}$
Initial level: $I^{\text{init}}_{f,p}$
End change: $\texttt{PlantProduction}(p, f, d)$: $\mathit{\Pi}_{f,p,d}$
$\texttt{VisitPlant}(t, p, i)$: $-b^{\text{pu}}_{t,i,f,p}$
$\texttt{PreSchedPickup}(p, f, i)$: $-a^{\text{pu}}_{p,f,i}$

Parallel resource: $\texttt{PlantFeedInventory}(p)$ for $p \in P$
Activities: $\texttt{PlantConsumption}(p, d)$ for $d \in H$, $\texttt{VisitPlant}(t, p, i)$,
$\texttt{PreSchedDelivery}(p, i)$, for $t \in T$, $i \in N_p$
Level range: $\{I^{\min}_{\text{feed},p} \,..\, I^{\max}_{\text{feed},p}\}$
Initial level: $I^{\text{init}}_{\text{feed},p}$
End change: $\texttt{PlantConsumption}(p, d)$: $-c_{p,d}$
$\texttt{VisitPlant}(t, p, i)$: $g_{t,p,i}$
$\texttt{PreSchedDelivery}(p, i)$: $g^{\text{del}}_{p,i}$

Parallel resource: $\texttt{CustomerProductInventory}(c, f)$
Activities: $\texttt{VisitCust}(t, c, i)$, for $t \in T$, $i \in N_c$
Level range: $\{0 \,..\, I^{\max}_{f,p}\}$
Initial level: $I^{\text{init}}_{f,c}$
End change: $\texttt{VisitCust}(t, c, i)$: $b^{\text{dem}}_{t,c,i,f}$

To ensure that at most one tow can visit each plant or supplier at a time, we introduce the following sequential resources:

Sequential resource: $\texttt{PlantVisits}(p)$ for $p \in P$
Activities: $\texttt{VisitPlant}(t, p, i)$, for $t \in T$, $i \in N_p$

Sequential resource: $\texttt{SupplierVisits}(s)$ for $s \in S$
Activities: $\texttt{VisitSupplier}(t, s, i)$, for $t \in T$, $i \in N_s$

The supplier visits are special in that the set of possible visits N_s is defined by the set of pickup windows W_s for supplier $s \in S$, which is not the case for plants. Therefore, we distribute these possible visits over the tows by adding the following constraints to ensure that at most one tow can visit a supplier in each pickup time window:

$$\sum_{t \in T} \texttt{VisitSupplier}(t, s, i).\texttt{Present} \leq 1 \qquad \text{for } s \in S, i \in N_s. \tag{4}$$

We recall that the specific pickup time windows are represented by the schedule domain of $\texttt{VisitSupplier}(t, s, i)$, which is defined as $\{l_{i,s}, \ldots, u_{i,s}\}$ for $t \in T$, $s \in S$, $i \in N_s$.

4.3 Modeling the Tows

For each tow, we define a unary resource for the sequence of visits (the route) and parallel resources for the inventory levels of each product carried by the tow. We assume that during each supplier pickup visit, both barges are filled with feed to full capacity (this was given as a requirement).

We first define additional activities $\texttt{TowFirstAct}(t)$ for $t \in T$, which represent the starting location for each tow. As the scheduling process is dynamic, this activity accounts for tows *en route* at the beginning of the planning horizon. These prescheduled tows, even if not active for the entire schedule horizon, are the minimum number of total tows that can be hired. Any additional tows are not prescheduled, and begin at a depot with a one-day travel time to any location.

We can now define the sequential resource representing the tow's route as follows:

Sequential resource: $\texttt{RouteSeq}(t)$ for $t \in T$
Activities: $\texttt{VisitSupplier}(t, s, i)$, $\texttt{VisitPlant}(t, p, i)$
$\texttt{VisitCust}(t, c, i)$, $\texttt{TowFirstAct}(t)$
Group set: $S \cup P \cup C$
Group definition: $\texttt{VisitSupplier}(t, s, i)$: s,
$\texttt{VisitPlant}(t, p, i)$: p,
$\texttt{VisitCust}(t, c, i)$: c,
$\texttt{TowFirstAct}(t)$: l_t
Group transition: $\texttt{(i,j)}$: $D_{i,j}$ for $i, j \in S \cup P \cup C$
First activity: $\texttt{TowFirstAct}(t)$

As tows carry several types of inventory, including both feed and final products, each of which is modeled as a different resource, it is necessary to use variables rather than parameters upon any plant visit. Hence, while the tow feed inventory level increases by a constant amount upon visiting a supplier, on any plant visit activity, a tow may or may not be delivering feed. Consequently, to model the tow inventory for feed, we define:

Parallel resource: $\texttt{TowFeedInv}(t)$
Activities: $\texttt{VisitPlant}(t,p,i)$,$\texttt{VisitSupplier}(t,s,i)$
Level range: $\{0\,..\,2\Lambda\}$
Begin change: $\texttt{VisitPlant}(t,p,i)$: $-g_{t,p,i}$
End change: $\texttt{VisitSupplier}(t,s,i)$: 2Λ

We similarly model the tow inventory for each final product as follows:

Parallel resource: $\texttt{TowProdInv}(t,f)$ for $t \in T, f \in F^-$
Activities: $\texttt{VisitPlant}(t,p,i)$,$\texttt{VisitCust}(t,c,i)$
Level range: $\{0\,..\,2\Lambda\}$
End change: $\texttt{VisitPlant}(t,p,i)$: $b^{\text{pu}}_{t,i,f,p} - b^{\text{dem}}_{t,p,i,f}$
$\texttt{VisitCust}(t,c,i)$: $-b^{\text{dem}}_{t,c,i,f}$

4.4 Additional Sequencing Constraints

There are two broad categories of additional sequencing constraints, both of which prune the search tree by limiting the options $\texttt{RouteSeq}(t)$ has after processing each activity. Neither category contains constraints that are strictly required to model the problem; however, their inclusion improves solver performance.

Visit Sequencing We introduce the following constraints for the plant visits, for $t \in T, p \in P, i \in N_p \setminus \{1\}$:

$$\texttt{VisitPlant}(t,p,i)\texttt{.Present} \Rightarrow \texttt{VisitPlant}(t,p,i-1)\texttt{.Present} \tag{5}$$

$$\texttt{ActivityBegin(VisitPlant}(t,p,i)\texttt{,}\hat{H}\texttt{)} > \texttt{EndOfNext(RouteSeq}(t)\texttt{,VisitPlant}(t,p,i-1)\texttt{,}\hat{H}-1\texttt{,}\hat{H}-1\texttt{)} \tag{6}$$

Constraint (5) is a symmetry-breaking constraint that ensures we schedule the visits in order of N_p. Constraint (6) ensures we cannot schedule two consecutive visits of tow t at plant p, which prevents idling. We define similar constraints for the customer visits, for all $t \in T, c \in C, i \in N_c \setminus \{1\}$:

$$\texttt{VisitCust}(t,c,i)\texttt{.Present} \Rightarrow \texttt{VisitCust}(t,c,i-1)\texttt{.Present} \tag{7}$$

$$\texttt{ActivityBegin(VisitCust}(t,c,i)\texttt{,}\hat{H}\texttt{)} > \texttt{EndOfNext(RouteSeq}(t)\texttt{,VisitCust}(t,c,i-1)\texttt{,}\hat{H}-1\texttt{,}\hat{H}-1\texttt{)} \tag{8}$$

Location Sequencing In addition to defining the order of visits, our model also restricts the locations a tow can visit after a given activity. As visits to suppliers leave a tow with no available capacity, we introduce the following constraints that make sure we visit a plant after a supplier, for $t \in T, s \in S, i \in N_s$:

$$(\texttt{BeginOfNext(RouteSeq}(t)\texttt{,VisitSupplier}(t,s,i)\texttt{,}\hat{H}\texttt{,}\hat{H}\texttt{)} \neq \hat{H}) \Rightarrow \texttt{GroupOfNext(RouteSeq}(t)\texttt{,VisitSupplier}(t,s,i)\texttt{)} \in P \tag{9}$$

Likewise, we introduce a constraint to model that we visit a demand location after picking up some product at a plant. We then define, for $t \in T, p \in P, i \in N_p$:

$$(\texttt{BeginOfNext(RouteSeq}(t)\texttt{,VisitPlant}(t,p,i)\texttt{,}\hat{H}\texttt{,}\hat{H}\texttt{)} \neq \hat{H}) \tag{10}$$
$$\wedge\, (b^{\text{pu}}_{t,i,f,p} > 0) \Rightarrow (\texttt{GroupOfNext(RouteSeq}(t)\texttt{,VisitPlant}(t,p,i)\texttt{)} \in L_f \setminus \{p\})$$

It is also possible to deliver final products at multiple locations in sequence. Since we must first empty both barges before we can pick up new feed from a supplier (a given requirement), we introduce the following constraints for $t \in T$, $p \in P, i \in N_p$:

$$(\texttt{(BeginOfNext(RouteSeq}(t)\texttt{,VisitPlant}(t,p,i)\texttt{,}\hat{H}\texttt{,}\hat{H}\texttt{)} \neq \hat{H})\ \wedge \tag{11}$$
$$\bigvee_{f \in F^-} (\texttt{TowProdInv}(t, f\texttt{,VisitPlant}(t,p,i)\texttt{.Begin).ActivityLevel} > 0)) \Rightarrow$$
$$\texttt{GroupOfNext(RouteSeq}(t)\texttt{,VisitPlant}(t,p,i)\texttt{)} \in \cup_f L_f,$$

and for $t \in T, c \in C, i \in N_c$:

$$(\texttt{(BeginOfNext(RouteSeq}(t)\texttt{,VisitCust}(t,c,i)\texttt{,}\hat{H}\texttt{,}\hat{H}\texttt{)} \neq \hat{H})\ \wedge \tag{12}$$
$$\bigvee_{f \in F^-} (\texttt{TowProdInv}(t, f\texttt{,VisitCust}(t,c,i)\texttt{.Begin).ActivityLevel} > 0)) \Rightarrow$$
$$\texttt{GroupOfNext(RouteSeq}(t)\texttt{,VisitCust}(t,c,i)\texttt{)} \in \cup_f L_f.$$

In practice, these constraints can be refined to operate on subsets of locations and products (and their associated activities). For example, in our case, customer sites only receive one type of final product. Note that the model does not direct a tow's route after both barges are empty, to allow the tows to "choose" to visit a supplier to pick up more feed, or a plant to pick up more final product.

4.5 Linking Pickup and Delivery Amounts with Visits

We introduce the following constraints to model the delivery amount of feed at the plants, for $t \in T, p \in P, i \in N_p$:

$$(\texttt{VisitPlant}(t,p,i)\texttt{.Present} = 0) \Rightarrow (g_{t,p,i} = 0) \tag{13}$$
$$(\texttt{VisitPlant}(t,p,i)\texttt{.Present} = 0) \Rightarrow (\sum_{f \in F^-} b^{\text{dem}}_{t,p,i,f} = 0) \tag{14}$$
$$(\texttt{VisitPlant}(t,p,i)\texttt{.Present} = 0) \Rightarrow (\sum_{f \in F^-} b^{\text{pu}}_{t,i,f,p} = 0) \tag{15}$$
$$(\texttt{VisitPlant}(t,p,i)\texttt{.Present} = 1) \Rightarrow \tag{16}$$
$$(\texttt{TowFeedInv}(t, \texttt{VisitPlant}(t,p,i)\texttt{.End)}\texttt{.ActivityLevel} = 0)$$

Constraints (13), (14) and (15) state that no feed or product can be delivered, or product can be picked up, when the plant is not visited. Constraint (16) states that partial deliveries are not allowed (this is a given requirement).

Similar constraints can be defined for the customer locations:

$$(\texttt{VisitCust}(t,c,i).\texttt{Present} = 0) \Rightarrow (\sum_{f \in F^-} b^{\text{dem}}_{t,c,i,f} = 0) \tag{17}$$

$$(\texttt{VisitCust}(t,c,i).\texttt{Present} = 1) \Rightarrow (\sum_{f \in F^-} b^{\text{dem}}_{t,c,i,f} > 0) \tag{18}$$

$$(\texttt{VisitCust}(t,c,i).\texttt{Present} = 1) \Rightarrow \\ (\texttt{TowProdInv}(t, \texttt{VisitCust}(t,c,i).\texttt{End}).\texttt{ActivityLevel} = 0) \tag{19}$$

Constraint (17) ensures that we cannot deliver any product when the customer is not visited. Constraint (18) on the other hand states that we must deliver a product when the customer is visited, and constraint (19) makes sure that no partial delivery occurs.

4.6 Objective

The objective function is deceptively simple. In our implementation of the model, we chose to minimize the total shortage of feed at the plants, as that was the most pressing issue apparent:

$$\min \sum_{p \in P, d \in H} s_{p,d}. \tag{20}$$

As we will see in our experimental evaluation, the specific instances we were given permit solutions in which there is no feed shortage, i.e., given a sufficient number of tows, our solutions satisfy all pickup window, underflow, and overflow constraints. However, solving the problem as a constraint optimization problem with objective (20) proved much more computationally efficient than solving the associated constraint satisfaction problem in which feed shortage is not allowed.

We do note, however, that the objective can easily be adapted to include other terms, depending on the application at hand.

5 Evaluation

We implemented our CP model in AIMMS 4.20, using IBM ILOG CPLEX and CP Optimizer (12.6.3) as MIP and CP solver, respectively. We performed an evaluation on the supply network given in Figure 1. We consider two representative cases over the same 92-day schedule horizon. Both cases have seven tows en route and 20 total tows available; additionally, both have similar production and demand profiles. They differ in the initial location and availability of the tows as well as in some of the dates of supplier pickup windows.

We compare our CP model with the MIP model that is currently in use at BP [6], as described in the introduction. The objective of the MIP model is to minimize total cost, which is a weighted sum of the number of tows in use, the violation of inventory upper and lower capacity constraints (overflow and

Table 3. Comparing CP and MIP solutions on two representative cases.

	CP				MIP	
	Case 1		Case 2		Case 1	Case 2
	7 tows	8 tows	7 tows	8 tows		
Variables	7,042	7,593	7,000	7,545	466,564	466,816
Constrains	3,387	3,836	3,371	3,822	509,068	509,320
Number of tows	7	8	7	8	8	7
Total underflow (mt)	0	0	0	0	200	17.72
Total overflow (mt)	0	0	0	0	0	11.83
Time window violations	0	0	0	0	0	5
Optimality gap					18.40%	13.69%
Solving time (s)	385	313	1,140	170	3,600	3,600

underflow), and violation of pickup time windows. The tows account for the largest relative cost in the objective.

The MIP model is given the option to use all 20 tows available, which, together with minimizing the constraint violations, allows to find some feasible solutions early in the solving process. The CP model, on the other hand, uses a fixed number of tows; in the reported experiments we use the minimum required number of tows (seven) as well as eight tows.

The results are presented in Table 3. The table first shows for each model the number of variables and constraints. In addition, in order to compare CP with MIP, we report for each solution the number of tows used, the total underflow and overflow, and the number of time window violations, i.e., visits to suppliers outside the defined pickup windows. The last two rows indicate the optimality gap (for MIP) and total solving time. Both MIP models were unable to solve the cases optimally within a time limit of 3,600s, whereas all CP models were solved optimally, in some cases within a couple minutes. Furthermore, the solutions provided by CP satisfy all problem constraints, whereas the solutions found by MIP use more tows for Case 1 (8 tows instead of 7), and violate various constraints.

Aimms not only offers an optimization modeling language, but also comes with visualization tools to build an end-user interface. The use of the model as a planning tool is facilitated by a graphical user input page which allows the user to vary the maximum number of tow and location visits, as discussed earlier, and, most importantly, the maximum number of tows. The input page accompanies several additional visualizations, including the inventory profiles at the locations, as well as the routing schedules for the tows. Figures 2 and 3 provide an illustration; they depict the inventory profiles for Plant 1 (feed) and Plant 2 (product 1) and the routing sequence for eight tows, in an optimal solution.

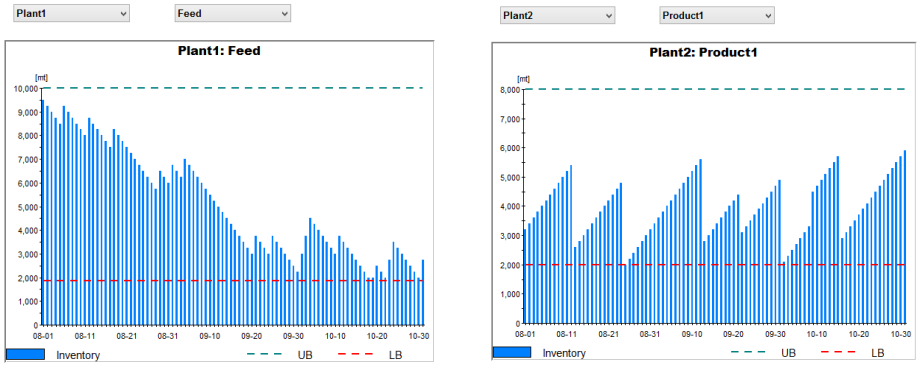

Fig. 2. Inventory levels for Plant 1's feed (left) and Plant 2's product 1 (right).

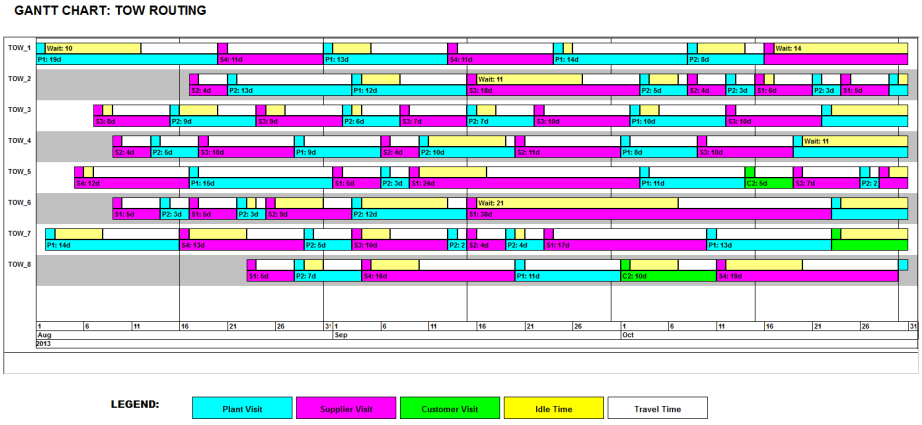

Fig. 3. Gantt chart representing the routes for the tows.

6 Conclusion

We introduced a constraint programming model for a maritime inventory routing problem in the petrochemical sector. Our model is based on a constraint-based scheduling formulation, and relies on activities to represent possible vessel visits, and parallel resources to represent the inventory levels over time. In addition, we utilize sequential resources to represent the vessel routes. As a case study we considered a barge scheduling problem from BP, with supplier, plant, and customer locations. We compared the performance of our CP model to an existing MIP formulation. The CP model was able to find optimal solutions with a small fleet size that satisfied all problem constraints, while the MIP formulation had scalability issues, and was not able to return solutions of similar quality even when given much more computation time.

References

1. 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.
2. P. Baptiste, C. Le Pape, and W. Nuijten. *Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems*. Kluwer Academic Publishers, 2001.
3. 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.
4. 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.
5. V. Goel, M. Slusky, W.-J. van Hoeve, K. Furman, and Y. Shao. Constraint Programming for LNG Ship Scheduling and Inventory Management. *European Journal of Operational Research*, 241(3):662–673, 2015.
6. N. Jerome. Description of a MIP Supply Delivery Scheduling Problem, 2014. Unpublished manuscript.
7. 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.
8. M. Roelofs and J. Bisschop. *AIMMS – The Language Reference*. AIMMS, 2016.