Computers & Operations Research 40 (2013) 1410–1419

Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

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

A branch-and-price guided search approach to maritime inventory routing ☆

Mike Hewitt [a,*], George Nemhauser [b], Martin Savelsbergh [c], Jin-Hwa Song [d,1]

[a] Department of Industrial and Systems Engineering, Rochester Institute of Technology, Rochester, NY 14623, USA
[b] H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0205, USA
[c] School of Mathematical and Physical Sciences, University of Newcastle, Callaghan, NSW 2308, Australia
[d] Global Technology, SK Innovation, Seoul, Korea

ARTICLE INFO

Available online 13 October 2012

Keywords:
Inventory routing
Integer programming
Heuristic search

ABSTRACT

We apply branch-and-price guided search to a real-world maritime inventory routing problem, in which the inventory of a single product, which is produced and consumed at multiple sites, and its transport, which is done with a heterogeneous fleet of vessels, is managed over a finite horizon. Computational experiments demonstrate that branch-and-price guided search quickly produces solutions that are near-optimal and of better quality than those produced by a state-of-the-art, commercial integer programming solver that is given much more time. We also develop local search schemes to further reduce the time needed to find high quality solutions and present computational evidence of their efficacy.

1. Introduction

Coordinating inventory management and vehicle routing decisions presents opportunities and challenges to both practitioners and researchers. For practitioners, simultaneously deciding how much and how to transport products from suppliers to consumers (as in vendor managed resupply) or between the multiple levels of a vertically integrated supply chain can yield greater efficiencies. For researchers, integrating inventory management and vehicle routing into a single problem provides a new opportunity for two groups of experts, especially since straightforward extensions of known methods for the individual problems do not show much promise.

As noted in [1], unlike the literature on the vehicle routing problem (VRP), wherein the majority of papers consider one of a core set of academically interesting models (e.g., the Capacitated VRP, Periodic VRP, VRP with Time Windows, etc.), almost all papers on the inventory routing problem (IRP) are motivated by an industrial application and introduce models that represent the specific decision environment. As such, papers on the IRP tend to resemble more closely the papers on what is sometimes called a *rich* VRP, as discussed in [12], and each paper introduces its own new "twist". For a survey of the different incarnations of the IRP, we refer the reader to [1].

A few characteristics often seen in inventory routing with maritime transportation (MIRP) lead to problems that are very challenging to solve: (1) multiple production and consumption sites, (2) planning horizons that are very long (often in months), and (3) vessel capacities that depend on location. The latter is a result of different draft limits at ports (see for example, [18,**?**]). These characteristics contribute to the fact that real-life instances of integer programming formulations of the MIRP tend to be large and difficult to solve; and are frequently beyond the capabilities of state-of-the art, commercial solvers.

Many approaches for solving the MIRP are integer programming-based, using either an arc-based, compact formulation in a branch-and-cut framework [18], or a path-based, extended formulation in a branch-and-price-and-cut framework [4,10,8]. These path-based extended formulations, in which a path represents the route of a vessel visiting a number of production and consumption facilities, show similarities to the extended formulations that have proven to be successful for the VRP [7]. However, the inventory management aspects of the problem require a greater degree of coordination amongst the paths, because determining whether a path is feasible typically requires more information. Thus, most extended formulations for the MIRP are not simple set-covering problems, but instead either include variables that model load and discharge decisions [5], or extend the definition of a column to include load and discharge decisions [4,8]. An interesting observation, see [8], is that while computational studies have shown that path-based extended formulations for the VRP yield small root node gaps, of the order of 5–15%, path-based extended formulations for the MIRP can yield very large gaps, often upwards of 100%.

☆ Research supported in part by funding from ExxonMobil Research and Engineering Company.

* Corresponding author.
E-mail address: mrheie@rit.edu (M. Hewitt).

[1] Work performed while Jin-Hwa Song was at ExxonMobil Research and Engineering Company.

0305-0548/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2012.09.010

Our focus, too, is on a MIRP motivated by an industrial application. In the taxonomy of variants of the IRP presented in [1], the problem considered is a single product, finite horizon IRP with deterministic supply/demand, a heterogeneous fleet, continuous routing, and a many-to-many producer, consumer topology. More specifically, we continue the investigation of the real-life maritime inventory routing problem presented in [18], which designs and implements a branch-and-cut algorithm with the primary goal of producing high-quality solutions. Ref. [8] complements that work by developing an extended formulation and a branch-and-price algorithm with the primary goal of computing strong bounds. Our emphasis is again on producing high-quality solutions, but doing so in a short amount of time.

Our algorithm is based on a new extended formulation for the MIRP, one that is not designed to produce a strong dual bound, but to speed up the search for high-quality primal solutions. The idea of using an extended formulation in this manner was first presented in [14], where it was successfully applied to the multi-commodity fixed charge network flow problem. The algorithm for solving this extended formulation is referred to as branch-and-price guided search (BPGS). One of the strengths of BPGS is that most of its components are not problem-specific and thus BPGS can easily be applied to new problems.

Therefore, we first demonstrate how BPGS can be applied to solve the MIRP and present a set of computational experiments that show that high-quality solutions can indeed be produced quickly. Then, we enhance BPGS with problem-specific local search schemes that enable it to find even better solutions in even less time. We note that many of these local search schemes can be applied to other variants of the IRP.

The remainder of this paper is organized as follows. In Section 2, we describe the maritime inventory routing problem motivating this research, highlight some of its special characteristics, and present a mixed-integer programming formulation. In Section 3, we introduce BPGS. In Section 4, we discuss how BPGS can be applied to the maritime inventory routing problem of interest, and study its effectiveness. In Section 5, we explore problem-specific local search schemes to enhance BPGS and show their efficacy. Finally, in Section 6, we present our conclusions and future research objectives related to this work.

2. Problem description

In this section, we provide a brief description of the real-life maritime inventory routing problem, focusing on the motivating structural elements. A complete and more detailed description can be found in [18].

Vessels transport a single product from supply ports in Europe to consumption ports in the US so as to maximize profits. Because supply and consumption ports are on different sides of the Atlantic ocean, a vessel first loads product at one or more supply ports, then crosses the ocean, and then discharges product at one or more consumption ports. Revenues are generated by the price collected for the product at the consumption ports and costs are incurred because of the price paid for the product at the supply ports and charges for the vessels used to transport the product from supply ports to consumption ports.

We consider a planning horizon of D days and index days by $d=1,\ldots,D$. One of the challenges in this problem is the length of the planning horizon, often 35–45 days, as this can lead to integer programming formulations with many integer variables. We denote the set of supply ports by S, the set of consumption ports by C, and the set of all ports by P. In our problem, $S\cap C=\emptyset$. Because each supply port can yield product at a different rate each day, and each consumption port can consume product at a different rate each day, we denote this rate for port p on day d as δ_{pd}. Note that $\delta_{pd}\geq 0$ (≤ 0) $\forall p\in S$ (C). The capacity that each port has for storing product can also vary by day. Thus, we denote the maximum amount of product that port p can have inventory at the end of day d by I_{pd}^{max}. Also, each port may have a lower bound on how much product it must have on hand at the end of day d. We denote this by I_{pd}^{min}. When loading (discharging) occurs at a port, the nature of the product requires that neither too little nor too much can be loaded (discharged). Specifically, if a vessel loads (discharges) from port p on a given day d, then F_{pd}^{min} and F_{pd}^{max} denote the smallest and largest quantity of product that can be loaded (discharged) respectively. Lastly, we denote the price of product that is loaded or discharged at port p by q_p, where $q_p<0$ when p is a supply port and thus the vessel is loading or purchasing product, and $q_p>0$ when p is a consumption port and thus the vessel is discharging or selling product.

A heterogeneous set of vessels, V, that can have different capacities, cost, speed, etc., is used to transport product from supply ports to consumption ports. We denote the maximum amount of product a vessel can have on board at any point in time by I_v^{max}. Each vessel $v\in V$ is leased and therefore is available for a specific time period. Using t_{SC}^{min}, the fewest number of days to travel from any supply port to any consumption port, we convert a vessel's lease period into a range of days, $[e_S^v,l_S^v]$, during which it may load product at supply ports, and a range of days, $[e_C^v,l_C^v]=[e_S^v+t_{SC}^{min},D]$ during which it may discharge product at consumption ports. Berthing rules dictate that at most one vessel can load (discharge) at a port on a single day.

We model a vessel's product loading and discharging, and routing decisions on a time–space network with node set $\mathcal{N}^0$ and arc set $\mathcal{A}$. Because production and consumption rates are quoted by day and travel times are quoted in days, our node set is based on a discretization of time into days. Thus, our time-space network contains nodes of the form (p,d) for all $p\in P$ and $d\in\{1,\ldots,D\}$. We represent the set of these nodes by $\mathcal{N}$. To get the full node set, $\mathcal{N}^0$, we append to this network a source node, $(s,0)$ and a sink node $(t,D+1)$, from which the vessel will begin and end its route. While each vessel travels on its own time–space network, we describe the network structure in general, highlighting where properties vary by vessel.

The time to travel between ports is quoted in days and the arc set, $\mathcal{A}$, of our network includes arcs of the form $a=((p,d),(p',d+t_{pp'}))$, where $t_{pp'}$ is the time in days to travel from port p to port p'. A vessel may idle at a port without loading or discharging. We model this by including in $\mathcal{A}$ arcs of the form $a'=((p,d),(p,d+1))$. To model the beginning of vessel v's voyage, we include in $\mathcal{A}$ arcs of the form $((s,0),(p,d)),\forall p\in S,d=e_S^v,\ldots,l_S^v$, and to model when the voyage ends, we include arcs of the form $((p,d),(t,D+1)),\forall p\in C,d=e_S^v+t_{SC}^{min},\ldots,D$. Lastly, to model that each vessel need not be used, we include the arc $((s,0),(t,D+1))$. Associated with each arc $a\in\mathcal{A}$ and vessel $v\in V$ is a cost c_a^v which represents the cost of vessel v moving on arc a. For arcs that model moving from port p to port p', this cost reflects the cost of travel between those two ports. Arcs that model idling at a port also have a cost, although it is typically much smaller than the cost of traveling to another port. Finally, all arcs that represent departing from the source node ($a=((s,0),(p,d))$) or arriving at the sink node ($a=((p,d),(t,D+1))$) have cost 0. We illustrate the time–space network for a single vessel v in Fig. 1.

We next describe the mixed-integer programming (MIP) model of this problem. We let the binary variable x_a^v indicate whether vessel $v\in V$ takes arc $a\in\mathcal{A}$, the binary variable y_{pd}^v indicate whether vessel $v\in V$ loads (discharges) at port p on day d, and the continuous variables f_{pd}^v represent the quantity that is loaded (discharged). Lastly, the continuous variables I_{pd} represent the total amount of product in inventory at port p on day d and I_d^v

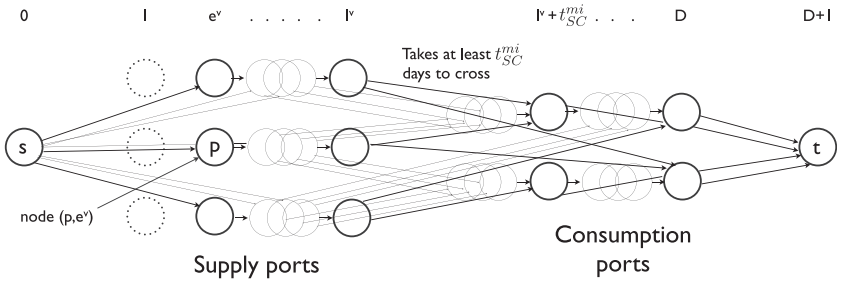

Fig. 1. Time–space network for vessel v.

represent the total amount of product on board vessel v on day d. With these variables, the MIP formulation of the MIRP is

$$\max \quad \sum_{v\in V}\sum_{p\in P}\sum_{d=1}^{D} q_p f_{pd}^v - \sum_{v\in V}\sum_{a\in\mathcal{A}} c_a^v x_a^v$$

$$\text{s.t.} \quad \sum_{a\in\delta^+(n)} x_a^v - \sum_{a\in\delta^-(n)} x_a^v = \alpha_n \quad \forall v\in V,\ \forall n\in\mathcal{N}^0, \tag{1}$$

$$y_{pd}^v \le \sum_{a\in\delta^-(n)} x_a^v \quad \forall v\in V,\ \forall n=(p,d)\in\mathcal{N}, \tag{2}$$

$$\sum_{v\in V} y_{pd}^v \le 1 \quad \forall (p,d)\in\mathcal{N}, \tag{3}$$

$$F_{pd}^{min} y_{pd}^v \le f_{pd}^v \le F_{pd}^{max} y_{pd}^v \quad \forall v\in V,\ \forall (p,d)\in\mathcal{N}, \tag{4}$$

$$I_{d-1}^v + \sum_{p\in S} f_{pd}^v - \sum_{p\in C} f_{pd}^v = I_d^v \quad \forall v\in V,\ \forall d=1,\ldots,D, \tag{5}$$

$$I_{pd-1} + \delta_{pd} - \sum_{v\in V} f_{pd}^v = I_{pd} \quad \forall (p,d)\in\mathcal{N}: p\in S, \tag{6}$$

$$I_{pd-1} + \delta_{pd} + \sum_{v\in V} f_{pd}^v = I_{pd} \quad \forall (p,d)\in\mathcal{N}: p\in C, \tag{7}$$

$$x_a^v \in \{0,1\} \quad \forall v\in V,\ \forall a\in\mathcal{A}, \tag{8}$$

$$y_{pd}^v \in \{0,1\} \quad \forall v\in V,\ \forall (p,d)\in\mathcal{N}, \tag{9}$$

$$f_{pd}^v \ge 0 \quad \forall v\in V,\ \forall (p,d)\in\mathcal{N}, \tag{10}$$

$$I_d^v \in [0, I_v^{max}] \quad \forall v\in V,\ \forall d=1,\ldots,D, \tag{11}$$

$$I_{pd} \in [I_p^{min}, I_p^{max}] \quad \forall (p,d)\in\mathcal{N}. \tag{12}$$

The objective is to maximize total profit, which equals the revenue from product delivered minus the cost of product picked up and the cost of transportation. Constraints (1) ensure flow balance of each vessel v in the time–space network, where α_n represents whether node n is a source ($\alpha_n=1$), sink ($\alpha_n=-1$), or intermediate ($\alpha_n=0$) node for a vessel, $\delta^+(n)=\{(n,n')\in\mathcal{A}\}$, and $\delta^-(n)=\{(n',n)\in\mathcal{A}\}$. Constraints (2) ensure that a vessel does not load (discharge) at port p on day d unless it is at that port on that day. Constraints (3) model a berthing rule that at most one vessel may load (discharge) from a port on a given day. Constraints (4) ensure that when loading or discharging occurs, the amount loaded or discharged falls within the appropriate bounds. Constraints (5) update the inventory on board vessel v on day d to reflect both the amount on board on day $d-1$ and the amount loaded and discharged on day d. Constraints (6) and (7) update the inventory at port p on day d to reflect the amount in inventory on day $d-1$, the amount produced (consumed) by the port on day d, and the amount loaded (discharged) by vessels on that day. We model existing inventory at port p at the beginning of the planning horizon with the quantity I_{p0}.

The real-life problem actually is more complicated than the one just described. For example, the number of days each vessel can idle is limited, it is possible to pay to increase a vessel's capacity, and the amount of product a vessel can have on board when arriving at, or departing from, a port can be limited because of draft limits at the port. However, for brevity, and because these constraints do not impact the solution approach(es), we omit how we model these practical considerations. These constraints are, however, properly taken into account in our computational experiments.

We next summarize BPGS. A more detailed description is given in [14].

3. Branch-and-price guided search

The idea motivating BPGS is that by adding simple constraints to an integer program we may be able to solve it quickly. We can then design a search procedure that produces primal solutions by adding different sets of constraints to the integer program and solving the resulting, easier to solve problem. Specifically, for an integer program P given by

$$\begin{aligned}\max \quad & cx+dy\\ \text{s.t.} \quad & Ax+By=b,\\ & x \text{ real, } y \text{ integer,}\end{aligned}$$

with optimal value V_P, and a given integer matrix N and integer vector r, both of appropriate dimension, we define *restriction* $P_N(r)$ of P as

$$\begin{aligned}\max \quad & cx+dy\\ \text{s.t.} \quad & Ax+By=b,\\ & Ny\le r,\\ & x \text{ real, } y \text{ integer,}\end{aligned}$$

with optimal value $V_N(r)$. Let $S_P=\{(x,y)|Ax+By=b, x \text{ real, } y\text{integer}\}$, the set of feasible solutions to P, and $R=\{r|r=Ny \text{ for some } (x,y)\in S_P\}$, the set of vectors associated with feasible solutions to problem P. We have that $V_P\ge V_N(r)\ \forall r\in R$ and $V_P=V_N(r^*)$ where $r^*=Ny_P^*$ for an optimal solution (x_P^*,y_P^*) to P. Thus, a strategy for finding high quality solutions to P is to search R for elements (r's) that will induce a restriction $P_N(r)$ that has a solution of high quality, and solve that restriction. A major advantage of this strategy is that it will produce a feasible solution to P each time a restriction is solved.

Next, we assume that we know the set R and build a model that extends the formulation of P to both choosing a vector $r\in R$ and solving the resulting restriction $P_N(r)$. Abusing notation and recalling that elements of the set R are vectors, we next let R represents the matrix with columns corresponding to those elements (vectors) and define the master problem MP

$$\begin{aligned}\max \quad & cx+dy\\ \text{s.t.} \quad & Ax+By=b,\\ & Ny-Rz\le 0,\end{aligned}$$

$$1z = 1,$$
$$x \text{ real}, y \text{ integer}, z \text{ binary},$$

where the binary variables z in *MP* represent the choice of vector r for which the restriction $P_N(r)$ should be solved.

Because R is likely to be too large to enumerate, we solve *MP* with a branch-and-price approach [3]. With $\overline{R} \subseteq R$, we define *RMLP* as

$$\begin{aligned} \max \quad & cx + dy \\ \text{s.t.} \quad & Ax + By = b, \\ & Ny - \overline{R}z \le 0 (\pi), \\ & 1z = 1 (\alpha), \\ & x \text{ real}, y \text{ real}, z \text{ real}, \end{aligned} \tag{13}$$

with dual variables $\pi \ge 0$ and α unrestricted. The resulting pricing problem, with r now representing a vector of variables, is then

$$\begin{aligned} -\alpha + \max \quad & \pi r \\ \text{s.t.} \quad & Ax + By = b, \\ & Ny - r = 0, \\ & x \text{ real}, y \text{ integer}, r \text{ integer}. \end{aligned}$$

We present pseudo-code of our branch-and-price approach for solving *MP* in Algorithm 1. Because the algorithm begins with $\overline{R} = \emptyset$,*RMLP* is infeasible the first time it is solved, at which point we use Farkas' lemma to derive the necessary dual variables for pricing. Standard techniques from branch-and-price can be used to produce a dual bound on V_{MP}, the optimal value of *MP*, without completely enumerating R. While Algorithm 1 (*BP*) will solve *MP* to optimality, it only uses solutions to the pricing problem to solve linear programs and does not take full advantage of the fact that they can induce restrictions.

Algorithm 1. Branch-and-price algorithm for solving MP (BP).

```
1: set R̄ = ∅
2: while MP has not been solved do
3:    select an unevaluated node
4:    while MLP associated with node has not been solved do
5:       solve RMLP associated with node
6:       solve the pricing problem to find an improving vector r
      to add to R̄
7:    end while
8: branching if necessary
9: end while
```

This observation inspires our first enhancement, which we call a primal construction (PC) process. A PC process solves restrictions $P_N(r)$ induced by solutions to the pricing problem. To find high-quality solutions quickly (and because there may be limited time for executing a PC process), we try to make intelligent choices regarding which restriction to solve next. We have different methods for making these choices, including examining solutions to *RMLP* and choosing the most recent solution to the pricing problem. We choose which method to use each time a PC process is executed using a biased roulette-wheel approach, as suggested in [13,15].

The paradigm of solving restrictions $P_N(r)$ to produce primal solutions also provides simple and effective methods for creating local search neighborhoods of a known solution. Observe that $\tilde{r} \ge r^{BEST} = Ny_P^{BEST}$ implies $V_N(\tilde{r}) \ge V_N(r^{BEST})$. This suggests that solving $P_N(\tilde{r})$ with an $\tilde{r} \ge r^{BEST}$ with well-chosen elements i such that $\tilde{r}_i > r_i^{BEST}$ may yield an improving solution. We have various schemes for constructing such an $\tilde{r}$. Thus, our second enhancement, which we call a primal improvement (PI) process, repeatedly chooses one of these schemes using a biased roulette-wheel approach, applies the chosen scheme to create a vector $\tilde{r}$, and then solves the resulting restriction $P_N(\tilde{r})$. The schemes are as follows.

Table 1
Process summary.

Name	Description
BP	Branch-and-price tasks, including managing branch-and-bound tree, solving pricing problems, and solving linear relaxations
PC	Solves $P_N(r)$ for $r \in \overline{R}$
PI	Solves $P_N(\tilde{r})$ for $\tilde{r}$ created with local search schemes. Note $\tilde{r}$ is not necessarily in $\overline{R}$

The first scheme, which we call *Augment-best*, creates $\tilde{r}$ by choosing individual entries of r^{BEST} to increment with the choice based on metrics related to those entries, such as the corresponding values in the primal and dual solutions to *RMLP*. Specifically, we begin with $\tilde{r} = r^{BEST}$ and then choose entries i and values δ such that we will set $\tilde{r}_i = \tilde{r}_i + \delta$.

The next two schemes use the concept of path-relinking [9], i.e., by combining the structural information from two good solutions we may produce a restriction that yields an even better solution. Given r^{BEST} and another restriction vector r, we create $\tilde{r}$ by setting $\tilde{r}_i = \max\{r_i, r_i^{BEST}\}$. The first of these two schemes, which we call *Priced-r*, chooses a vector r that is a solution to the pricing problem and the second, which we call *Solution-r*, chooses a vector r that is associated with a known solution.

With these local search ideas care has to be taken to ensure that the norm of the vector $\tilde{r}$ is not too large, as the resulting restriction may take too long to solve. Thus, when creating a vector $\tilde{r}$ in one of these schemes we ensure that $\|\tilde{r}\|_1 \le \gamma$ where γ is an algorithm parameter.

We summarize the three processes that make up branch-and-price guided search in Table 1. Because these processes are only loosely coupled, they may be executed in parallel, which we do. Further details regarding the execution of these processes, including how they are executed in parallel, can be found in [14]. Even though BPGS optimally solves a problem when given enough time, its main strength and practical value is that it finds high-quality solutions quickly.

4. BPGS for MIRP

To apply BPGS to the MIRP, we need to choose the matrix N to define the structure of $P_N(r)$. We choose the matrix N so that for a given vector r, the restriction $P_N(r)$ forces certain ports to be closed on certain days. More specifically, we add constraints

$$\sum_{v \in V} y_{pd}^v \le r_{pd} \quad \forall (p,d) \in \mathcal{N},$$

which force port p to be closed on day d when $r_{pd} = 0$. With this choice of N, *MP* is obtained by adding the constraints

$$\sum_{v \in V} y_{pd}^v \le \sum_{i=1}^{|R|} r_{pd}^i z_{r_i} \quad \forall (p,d) \in \mathcal{N},$$

where r^i denotes the ith element of the set R, and a component r_{pd}^i indicates whether port p is open on day d ($r_{pd}^i = 1$) or not ($r_{pd}^i = 0$), and z_{r_i} is a binary variable that identifies the restriction (or port schedule).

To solve *MP* with a branch-and-price approach, BPGS solves a pricing problem containing binary variables ρ_{pd} for $(p,d) \in \mathcal{N}$, with

non-negative objective function coefficients π_{pd} for $(p,d) \in \mathcal{N}$ representing the value of port p being open on day d. Thus, the pricing problem solved by BPGS is

$$-\alpha + \max \sum_{(p,d) \in \mathcal{N}} \pi_{pd} \rho_{pd}$$
$$\text{s.t.} \sum_{v \in V} y^v_{pd} = \rho_{pd} \quad \forall (p,d) \in \mathcal{N},$$
$$\rho_{pd} \in \{0,1\} \quad \forall (p,d) \in \mathcal{N}$$

and constraints (1)–(12) from the definition of MIRP (with r_{pd} replaced by ρ_{pd}). While the pricing problem optimizes over the same feasible region as the original problem, its objective function is significantly different as all the objective function coefficients are non-negative and there are no objective coefficients associated with the routing variables. In particular, the objective function encourages the variables ρ_{pd} to take on the value 1 in solutions to the linear programming relaxation. In addition to having an objective function that is likely to make the pricing problem easier to solve, the pricing problem does not have to be solved to optimality for the scheme to work.

While the objective function encourages the variables ρ_{pd} to take on the value 1, the solution with $\rho_{pd} = 1 \ \forall p \in P, d \in D$, which is undesirable as it would induce a restriction that is equivalent to the original problem, is not feasible. When all ρ_{pd} variables are equal to 1 in a feasible solution, it must also be the case that each port is visited by some vessel on each day (due to constraints (14) and (2)). This is not possible when the number of vessels is less than the total number of ports. At the same time, certain instance characteristics may lead to solutions to the pricing problem that are sparse (few variables ρ_{pd} equal to 1). For example, with instances where vessels begin their voyages at different periods of the horizon, many travel times between supply (consumption) ports are greater than one day, and travel times across the ocean are quite long, routing constraints (constraint (1)) will limit the number of ports that can be visited during the horizon. Similarly, for instances where vessel capacity is sufficiently low, the fact that a vessel must load (discharge) at least a certain amount when visiting a port (constraint (4)) should also lead to few ports being visited during the horizon.

In the context of the MIRP, interpretations of the local search schemes are: *Augment-best* opens ports on days other than when they are open in the best-known solution, and when we view a solution to the MIRP as inducing a schedule of when ports are open, *Priced-r* and *Solution-r* combine two port schedules by opening a port on a day if it is open in either schedule on that day.

We implemented BPGS in C++, with CPLEX 11.2 used as the MIP/LP solver. We dedicate one processor to solving *MP* with branch-and-price (a BP process), one processor to solving restrictions $P_N(r)$ induced by solutions to the pricing problem (a PC process), and two processors to solving restrictions $P_N(\tilde{r})$ created with the local search schemes (PI processes). Message passing between processes was implemented via a combination of MPI [11] and text files.

Table 2
Primal comparison of BPGS with CPLEX.

Instance	CPLEX Time to Best	CPLEX Primal	CPLEX Gap	BPGS Time to Best	BPGS Primal	BPGS Gap	BPGS Gap CPLEX	CPLEX Time to Beat or Tie
6-3-4-1	39,430.66	2922.28	3.68	1157.00	2948.71	2.81	0.90	
6-3-4-2	40,271.81	1178.32	8.19	1624.00	1194.22	6.95	1.33	
6-3-4-3	42,480.63	4209.85	7.92	1414.00	4183.02	8.51	−0.64	37,043.50
6-3-4-4	41,210.05	3472.73	9.49	1550.00	3390.00	11.65	−2.44	38,675.83
6-3-4-5	40,338.02	2891.47	7.14	530.00	2864.17	8.02	−0.95	9691.47
6-3-4-6	208.62	6340.02	0.99	768.00	6403.72	0.00	0.99	
6-3-4-8	42,050.96	5388.14	0.00	372.00	5233.00	2.88	−2.96	16,586.04
6-3-4-9	39,591.90	6076.65	0.00	581.00	6076.65	0.00	0.00	31,741.39
6-3-4-10	43,080.80	6960.61	1.85	674.00	6979.07	1.59	0.26	
6-4-3-1	40,004.23	1627.10	44.83	796.00	2448.81	16.97	33.56	
6-4-3-2	40,570.58	2561.35	21.38	901.00	2721.97	16.45	5.90	
6-4-3-3	43,165.29	2322.72	29.24	1809.00	2845.60	13.31	18.38	
6-4-3-4	32,156.32	4486.74	11.75	1580.00	4564.98	10.21	1.71	
6-4-3-5	40,666.43	2312.42	15.11	1032.00	2312.42	15.11	−0.00	39,210.55
6-4-3-6	397.03	6157.74	1.47	1196.00	6201.56	0.77	0.71	
6-4-3-7	1740.89	6034.51	0.00	593.00	5975.12	0.98	−0.99	1740.89
6-4-3-8	144.10	5988.16	0.02	743.00	5989.47	0.00	0.02	
6-4-3-9	25,809.83	5650.78	0.41	1538.00	5595.24	1.39	−0.99	22,818.89
6-4-3-10	19,315.47	6413.11	1.45	460.00	6413.11	1.45	−0.00	16,435.10
6-4-4-1	40,058.15	4280.32	17.19	1683.00	4331.79	16.20	1.19	
6-4-4-2	42,060.20	4838.48	13.22	1059.00	5060.37	9.25	4.38	
6-4-4-3	42,638.38	3801.57	11.27	1738.00	3732.66	12.88	−1.85	42,348.23
6-4-4-4	42,838.99	4328.37	13.23	1564.00	4298.10	13.83	−0.70	42,812.88
6-4-4-5	41,297.21	4630.38	4.95	1175.00	4615.19	5.27	−0.33	41,297.21
6-4-4-6	758.50	5696.00	1.35	1574.00	5724.94	0.85	0.51	
6-4-4-7	26,774.81	5746.15	2.39	431.00	5638.73	4.21	−1.91	12,489.40
6-4-4-8	42,496.05	5161.62	3.08	1053.00	5164.94	3.02	0.06	
6-4-4-9	40,774.80	6410.63	1.53	1269.00	6445.34	0.99	0.54	
6-4-6-1	21,300.04	5752.86	6.59	1833.00	5789.33	5.99	0.63	
6-4-6-2	42,146.15	8244.99	3.44	1272.00	7940.05	7.01	−3.84	40,419.97
6-4-6-3	41,181.80	5911.45	6.69	1582.00	5865.72	7.41	−0.78	41,181.80
6-4-6-4	42,243.41	5910.19	14.78	1264.00	6638.00	4.28	10.96	
6-4-6-5	41,174.80	7629.73	5.91	1435.00	7312.02	9.83	−4.35	36,622.41
6-6-4-1	43,074.91	5392.97	13.94	1802.00	5651.87	9.81	4.58	
6-6-4-2				1627.00	4703.47	17.96		
6-6-4-3	41,138.15	5033.46	13.84	1790.00	5280.15	9.62	4.67	
6-6-4-4	42,386.52	3993.97	32.82	1269.00	5117.80	13.91	21.96	
6-6-4-5	42,226.93	7498.46	5.00	1493.00	7596.70	3.75	1.29	
Average	33,762.25		9.08	1216.61		7.24	2.48	29,444.72

Because BPGS solves integer programs, we chose to benchmark it against the commercial solver CPLEX (version 11.2). Because a parallel version of CPLEX was not available at the time of experimentation, CPLEX was run on a single processor for 12 h (with an optimality tolerance of 1% and all other settings at their defaults). BPGS was allowed to run for 30 min (with restrictions $P_N(r)$ being solved with an optimality tolerance of 1% and a time limit of 1 min, and pricing problems solved with an optimality tolerance of 5% and a time limit of 5 min). To search neighborhoods of potentially different sizes, we provide different values for γ to the two local search processes. The first process uses $\gamma = 0.15 * |N|$, while the second uses $\gamma = 0.3 * |N|$. All experiments were performed on a machine with 8 Intel Xeon CPUs running at 2.66 GHz with 32 GB RAM. Unless otherwise noted, computation times are reported in seconds.

For our experiments, we use the instances from [8]. They observed that the instances with six vessels, and three or more load and discharge ports were the most difficult to solve, and, in particular, proved the most challenging with respect to finding primal solutions. We focus on those instances in our experiments.

The results are reported in Table 2. Because the dual bounds produced by CPLEX for these instances are often weak and do not provide a good measure of the quality of primal solutions, we report optimality gaps computed with the dual bounds produced by the branch-and-cut-price approach presented in [8]. We report the instance name in the form # *Vessels-# Load Ports-# Discharge Ports-Instance #* (Instance), the time CPLEX needed to find its best primal solution (CPLEX Time to Best), the value of the best primal solution found by CPLEX in 12 h (CPLEX Primal), the quality of the best primal solution produced by CPLEX (CPLEX Gap) calculated as $100 \times$ (Engineer et al. Dual $-$CPLEX Primal)/(Engineer at al. Dual), the time BPGS needed to find its best primal solution (BPGS Time to Best), the value of the best primal solution found by BPGS in 30 min (BPGS Primal), the quality of the best primal solution produced by BPGS (BPGS Gap) calculated as $100 \times$ (Engineer et al. Dual $-$BPGS Primal)/(Engineer et al. Dual), a comparison of the quality of primal solutions produced by BPGS and CPLEX calculated as $100 \times$ (BPGS Primal $-$CPLEX Primal)/(BPGS Primal), and when able, the time CPLEX needed to find an equal or better solution than BPGS could find in 30 min (CPLEX Time to Beat or Tie).

What we see is that BPGS is in fact quickly producing high-quality solutions, only 7% away from optimal, on average. In 30 min, BPGS is able to produce a solution that is within 5% of optimal for 16 of the 39 instances, and within 1% of optimal for seven instances. We also see that BPGS is superior to CPLEX with

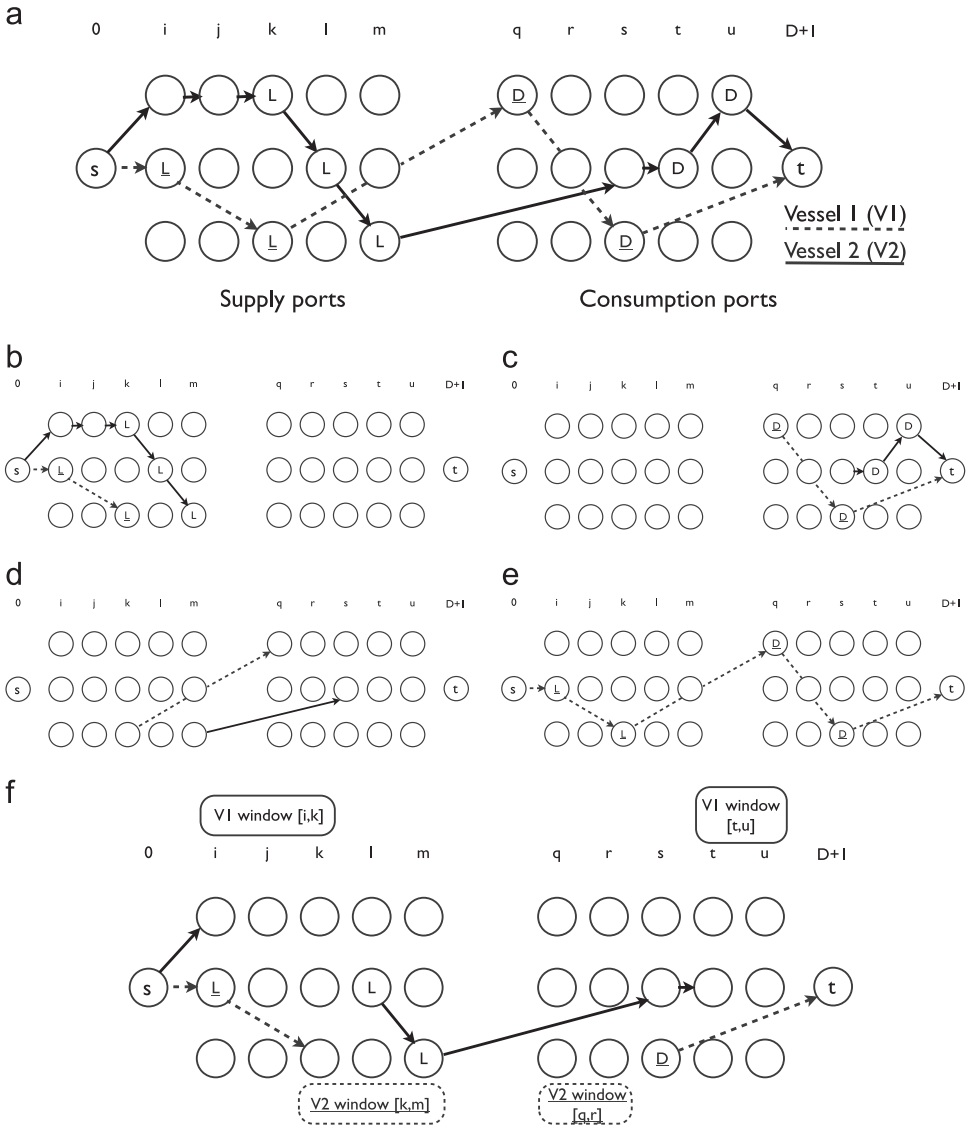

Fig. 2. Local search schemes: (a) solution, (b) fix supply, (c) fix consumption, (d) fix crossing, (e) fix vessel, and (f) fix window.

respect to producing primal solutions. On average, BPGS is able to produce in 30 min a solution that is 2.48% better than what CPLEX can produce in 12 h; CPLEX is able to find a better solution for only 15 of the 39 instances, and it takes CPLEX over 9 h, on average, to do so. We note that for one instance (6-6-4-2), CPLEX was not able to produce a primal solution in 12 h.

A more detailed analysis shows that 51.95% of solution improvement is due to the *Augment-best* scheme, 17.78% is due to the *Priced-r* scheme, and 30.27% is due to the *Solution-r* scheme. This is notably different from the application of BPGS to the multi-commodity fixed charge network flow problem discussed in [14], where nearly 90% of solution improvement was due to *Augment-best*. We also note that, averaging across all instances, the pricing problem is solved 45.17 times, and to optimality 67.23% of those times.

Even though the results in Table 2 show that BPGS does well, in the next section we show that we can do better by incorporating problem-specific techniques.

5. Enhancing BPGS for MIRP

Other than choosing the matrix N using our knowledge of the underlying problem, BPGS uses no problem-specific techniques. It is well-known, however, that exploiting problem structure can lead to more powerful solution techniques. Therefore, in this section, we explore the use of problem-specific local search schemes, but again define neighborhoods as the set of solutions to a restricted integer program and searching that neighborhood by solving that integer program. Similar ideas can be found in [6,17,2,16,13,18].

The problem-specific local search schemes explored are based on the following five questions:

- What are the best ocean transit and deliveries for vessels whose product pickups are known?
- What are the best pickups and ocean transit for vessels whose product deliveries are known?
- What are the best pickups and deliveries for vessels whose ocean transits are known?
- What are the best schedules for one or more vessels if the schedules of the other vessels are known?
- What are the best activities for vessels during a time period when activities outside that time period are known?

In each of these questions, parts of the solution are assumed known and we are seeking to optimally complement that partial solution. This can be implemented by identifying for each vessel v two disjoint sets of arcs $\mathcal{A}_1^v \subset \mathcal{A}$ and $\mathcal{A}_0^v \subset \mathcal{A}$, and two disjoint sets of nodes $\mathcal{N}_1^v \subset \mathcal{N}$ and $\mathcal{N}_0^v \subset \mathcal{N}$, and then create an instance of MIRP with the extra constraints

$$x_a^v = 0 \quad \forall a \in \mathcal{A}_0^v,\ \forall v \in V,$$

$$x_a^v = 1 \quad \forall a \in \mathcal{A}_1^v,\ \forall v \in V,$$

$$y_{pd}^v = 0 \quad \forall (p,d) \in \mathcal{N}_0^v,\ \forall v \in V,$$

$$y_{pd}^v = 1 \quad \forall (p,d) \in \mathcal{N}_1^v,\ \forall v \in V.$$

Thus we are solving instances of the MIRP where some decisions regarding the routing of vessels and the timing of loads or discharges are prescribed. Note we are not fixing the quantities a vessel loads or discharges.

Given a solution to *MIRP*, with $x_a^v = \bar{x}_a^v$ and $y_{pd}^v = \bar{y}_{pd}^v$, we use five schemes for choosing the sets $\mathcal{A}_1^v, \mathcal{A}_0^v, \mathcal{N}_1^v, \mathcal{N}_0^v$, each corresponding to one of the questions above. We illustrate these schemes in Fig. 2(b)–(f). In Fig. 2(a) we depict the current solution and in Fig. 2(b)–(f) we depict the portion of the solution that is fixed in the instance of the MIRP that we solve to search for an improving solution.

- *Fix supply*: This scheme fixes the route each vessel takes through the supply ports and when and where it loads product. Thus, to fix the route of vessel v through supply ports, we set $\mathcal{A}_0^v = \{a \in \mathcal{A} : \bar{x}_a^v = 0$ and $a = ((p_1,d_1),(p_2,d_2))$, with $p_1 \in S, p_2 \in S\}$ and $\mathcal{A}_1^v = \{a \in \mathcal{A} : \bar{x}_a^v = 1$ and $a = ((p_1,d_1),(p_2,d_2))$, with $p_1 \in S, p_2 \in S\}$. To fix when and where loading occurs, we set $\mathcal{N}_0^v = \{(p,d) \in \mathcal{N} : \bar{y}_{pd}^v = 0$ and $p \in S\}$, and $\mathcal{N}_1^v = \{(p,d) \in \mathcal{N} : \bar{y}_{pd}^v = 1$ and $p \in S\}$. This scheme fixes the last supply port it visits, but does not fix the first consumption port it visits. We illustrate this scheme in Fig. 2(b).
- *Fix consumption*: This scheme fixes the route each vessel takes through the consumption ports and when and where it discharges product. Thus, to fix the route of vessel v through consumption ports, we set $\mathcal{A}_0^v = \{a \in \mathcal{A} : \bar{x}_a^v = 0$ and $a = ((p_1,d_1),(p_2,d_2))$, with $p_1 \in C, p_2 \in C\}$ and $\mathcal{A}_1^v = \{a \in \mathcal{A} : \bar{x}_a^v = 1$ and $a = ((p_1,d_1),(p_2,d_2))$, with $p_1 \in C, p_2 \in C\}$. To fix when and where discharging occurs, we set $\mathcal{N}_0^v = \{(p,d) \in \mathcal{N} : \bar{y}_{pd}^v = 0$ and $p \in C\}$, and, $\mathcal{N}_1^v = \{(p,d) \in \mathcal{N} : \bar{y}_{pd}^v = 1$ and $p \in C\}$. This scheme fixes when and where a vessel arrives at the set of consumption ports, but does not fix the supply port from which it departs for the set of consumption ports. We illustrate this scheme in Fig. 2(c).
- *Fix crossing*: This scheme only fixes when and where each vessel crosses from the set of supply ports to the set of consumption ports. Thus, for each vessel v we set $\mathcal{A}_0^v = \{a \in \mathcal{A} : \bar{x}_a^v = 0$ and $a = ((p_1,d_1),(p_2,d_2))$, with $p_1 \in S, p_2 \in C\}$, $\mathcal{A}_1^v = \{a \in \mathcal{A} : \bar{x}_a^v = 1$ and $a = ((p_1,d_1),(p_2,d_2))$, with $p_1 \in S, p_2 \in C\}$, $\mathcal{N}_0^v = \emptyset$, and $\mathcal{N}_1^v = \emptyset$. We illustrate this scheme in Fig. 2(d).

Each of these three schemes fully specifies an instance of the MIRP to solve. For the next two schemes, additional information is required as these schemes define a family of possible instances of the MIRP to solve. To choose an instance from this family, we use information from the linear programming relaxation of our extended formulation, *RMLP*. Recall that *RMLP* contains the constraints $\sum_{v \in V} y_{pd}^v \leq \sum_{i=1}^{|\bar{R}|} r_{pd}^i z_{r_i}$ for each node (p,d) in the time–space network. Given a solution to *RMLP* with variable values z_{RMLP}^*, we interpret the quantity $(\bar{R} z_{RMLP}^*)_{pd}$, the amount port p is used on day d in the solution to *RMLP*, as an indication of whether node (p,d) should appear in a high-quality solution to MIRP. Specifically, the higher the value $(\bar{R} z_{RMLP}^*)_{pd}$, the more we believe that port p should be visited by some vessel on day d in a high-quality solution. Also, recall that we associate a dual $\pi_{pd} (\geq 0)$ with these constraints and that the objective of the pricing problem is to maximize $\sum_{(p,d) \in \mathcal{N}} \pi_{pd} r_{pd}$. Therefore, the higher the value of the dual variable π_{pd}, the more we believe that port p should be visited by some vessel on day d in a high-quality solution.

- *Fix vessel*: Similar to the heuristics presented in [16,18], this scheme chooses a subset of vessels, $\overline{V} \subset V$, and then fixes all routing variables and variables that represent the location and timing of loads or discharges for those vessels. Specifically, we set $\mathcal{A}_0^v = \{a \in \mathcal{A} : \bar{x}_a^v = 0$ and $v \in \overline{V}\}, \mathcal{A}_1^v = \{a \in \mathcal{A} : \bar{x}_a^v = 1$ and $v \in \overline{V}\}, \mathcal{N}_0^v = \{(p,d) \in \mathcal{N} : \bar{y}_{pd}^v = 0$ and $v \in \overline{V}\}, \mathcal{N}_1^v = \{(p,d) \in \mathcal{N} : \bar{y}_{pd}^v = 1$ and $v \in \overline{V}\}$. To choose a subset of vessels, $\overline{V} \subset V$, of fixed size k, we assign each vessel a score s_v and then randomly draw k vessels from V with a bias towards those with a high score. We use three different scoring mechanisms for vessels, and each time we execute the

Fix Vessel scheme, we use the same mechanism for all vessels. The three mechanisms are as follows:

- *Dual-based*: $s_v = \sum_{(p,d) \in \mathcal{N}: \overline{y}^v_{pd} = 1} \pi_{pd}$.
- *LP-based*: $s_v = \sum_{(p,d) \in \mathcal{N}: \overline{y}^v_{pd} = 1} (\overline{R}z^*_{RMLP})_{pd}$.
- *Profit-based*: $s_v = \sum_{p \in P} \sum_{d=1}^{D} q_p \overline{f}^v_{pd} - \sum_{a \in \mathcal{A}} c^v_a \overline{x}^v_a$, where $\overline{f}^v_{pd}$ is the value of the variable f^v_{pd} in the current solution.

The first two schemes are based on our interpretation of the quantities $(\overline{R}z^*_{RMLP})_{pd}$ and π_{pd} and assess whether vessel v is visiting the right nodes in the current solution. The higher the sums $\sum_{(p,d) \in \mathcal{N}: \overline{y}^v_{pd} = 1} \pi_{pd}$ and $\sum_{(p,d) \in \mathcal{N}: \overline{y}^v_{pd} = 1} (\overline{R}z^*_{RMLP})_{pd}$, the more we believe the vessel's route currently visits the right ports on the right days and thus the route need not be re-optimized. The last scheme considers the profit (or loss) earned by each vessel in the current solution. The greater profit a vessel earns (as measured by $\sum_{p \in P} \sum_{d=1}^{D} q_p \overline{f}^v_{pd} - \sum_{a \in \mathcal{A}} c^v_a \overline{x}^v_a$), the more we believe that vessel's route need not be re-optimized.

- *Fix window*: We choose for each vessel v two windows $[a^v_S, b^v_S]$ and $[a^v_C, b^v_C]$ with the first occurring during the load window $[e^v_S, l^v_S]$ and the second occurring during the discharge window $[e^v_C, l^v_C]$. Once these windows are defined for a vessel, we fix the routing, load, and discharge decisions for that vessel that occur outside of those windows. Specifically, we set

  - $\mathcal{A}^v_0 = \{a \in \mathcal{A} : \overline{x}^v_a = 0$ and $a = ((p_1,d_1),(p_2,d_2))$, with $p_1 \in S, p_2 \in S, d_1 \notin [a^v_S, b^v_S]\} \cup \{a \in \mathcal{A} : \overline{x}^v_a = 0$ and $a = ((p_1,d_1),(p_2,d_2))$, with $p_1 \in C, p_2 \in C, d_1 \notin [a^v_C, b^v_C]\}$.
  - $\mathcal{A}^v_1 = \{a \in \mathcal{A} : \overline{x}^v_a = 1$ and $a = ((p_1,d_1),(p_2,d_2))$, with $p_1 \in S, p_2 \in S, d_1 \notin [a^v_S, b^v_S]\} \cup \{a \in \mathcal{A} : \overline{x}^v_a = 1$ and $a = ((p_1,d_1),(p_2,d_2))$, with $p_1 \in C, p_2 \in C, d_1 \notin [a^v_C, b^v_C]\}$.
  - $\mathcal{N}^v_0 = \{(p,d) \in \mathcal{N} : \overline{y}^v_{pd} = 0$ and $p \in S, d \notin [a^v_S, b^v_S]\}$ $\cup \{(p,d) \in \mathcal{N} : \overline{y}^v_{pd} = 0$ and $p \in C, d \notin [a^v_C, b^v_C]\}$.
  - $\mathcal{N}^v_1 = \{(p,d) \in \mathcal{N} : \overline{y}^v_{pd} = 1$ and $p \in S, d \notin [a^v_S, b^v_S]\}$ $\cup \{(p,d) \in \mathcal{N} : \overline{y}^v_{pd} = 1$ and $p \in C, d \notin [a^v_C, b^v_C]\}$.

We use the same procedure for choosing the windows $[a^v_S, b^v_S]$ and $[a^v_C, b^v_C]$ and present it in Algorithm 2.

Algorithm 2. *Generate window.*

Require: A range of days $[e,l]$
Require: A list of scores $[s_e, s_{e+1}, \ldots, s_l]$ that estimate the value of the actions not taken by the vessel on each day
Require: A range of possible window lengths $[l_w, u_w]$
  Randomly draw the length L of the window to generate from $[l_w, u_w]$.
  for $d = 1$ to $l - L$ **do**
    set $score_d = \sum_{i=d}^{d+L} s_i$ $score_d$ represents our estimate of the potential of the window $[d, d+L]$.}
  end for
  Randomly draw the beginning day of the window a from $[1, l-L]$ with a bias towards days with a high score $score_a$.
  return $[a, a+L]$

We use two different mechanisms to evaluate the actions not taken by a vessel on a day, and each time we execute the *Fix Window* scheme we use the same mechanism for all vessels. The two mechanisms are as follows:

- *Dual-based*: $s^v_i = \sum_{p \in P: \overline{y}^v_{pi} = 0} \pi_{pi}$.
- *LP-based*: $s^v_i = \sum_{p \in P: \overline{y}^v_{pi} = 0} (\overline{R}z^*_{RMLP})_{pi}$.

Note that for both of these mechanisms, for each day i, we are summing our metric $(\pi_{pi}, (\overline{R}z^*_{RMLP})_{pi})$ of whether port p should be visited on day i over the ports the vessel does *not* visit on that day in the current solution. Thus, we interpret $\sum_{p \in P: \overline{y}^v_{pi} = 0} \pi_{pi}$ and $\sum_{p \in P: \overline{y}^v_{pi} = 0} (\overline{R}z^*_{RMLP})_{pi}$ as metrics of the missed opportunities for vessel v on day i. The greater those sums, the more we believe there is a chance for vessel v to do better on that day than what it does in the current solution.

With our mechanism for choosing an individual vessel's window defined, we next define the complete *Fix Window* scheme in Algorithm 3.

Algorithm 3. *Fix window.*

Require: The current solution, $(\overline{x}, \overline{y})$
Require: Dual (π_{pd}) and primal information $((\overline{R}z^*_{RMLP})_{pd})$ from a solution to *RMLP*
Require: A range of possible window lengths $[l_w, u_w]$
Require: The scoring mechanism, Dual-based or LP-based to use
  for $v \in V$ **do**
  Set $[a^v_S, b^v_S] = WindowGenerate([e^v_S, l^v_S], [s_{e^v_S}, \ldots, s_{l^v_S}], [l_w, u_w])$
  Set $[a^v_C, b^v_C] = WindowGenerate([e^v_C, l^v_C], [s_{e^v_C}, \ldots, s_{l^v_C}], [l_w, u_w])$
  end for
  Construct sets $\mathcal{A}^v_0, \mathcal{A}^v_1, \mathcal{N}^v_0, \mathcal{N}^v_1$ as described above and solve the instance of the MIRP

We choose which of these schemes to execute next with a biased roulette-wheel approach (similar to the approach used in [13]) based on the total solution improvement the scheme has yielded so far. For the schemes that have multiple scoring mechanisms, we treat the scheme and scoring mechanism as an individual scheme. Thus, we randomly draw the next scheme to use from the list [*Fix Supply*, *Fix Consumption*, *Fix Crossing*, *Fix Vessel: Dual-based*, *Fix Vessel: LP-based*, *Fix Vessel: Profit-based*, *Fix Window: Dual-based*, *Fix Window: LP-based*] with a bias based on the past performance of each of these schemes.

To computationally study the effectiveness of these problem-specific (PS) local search schemes, we repeat the experiments of Section 4, only instead of having two PI processes execute the schemes *Augment-best*, *Priced-r*, and *Solution-r*, we have one PI process execute only those schemes and the other process execute only the schemes discussed in this section. When executing the scheme *Fix Vessel* we optimize the decisions for two vessels. Thus, because of all the instances in our study involve six vessels, when executing this scheme we choose a subset of vessels of size $k = 4$. When executing the scheme *Fix Window* we consider windows whose length is in the range $[l_w, u_w] = [3,8]$. For the instances we have experimented with, the load window $[e^v_S, l^v_S]$ for a vessel is typically 10–12 days long and the discharge window $[e^v_C, l^v_C]$ is typically 10–20 days long.

The results are reported in Table 3. Specifically, we report the time needed to find the best primal solution (PS Time to Best), the value of that best primal solution (PS Primal), a comparison of the quality of primal solutions produced with those produced by CPLEX (PS Gap CPLEX) calculated as $100 \times$ (PS Primal $-$ CPLEX Primal)/ (PS Primal), the quality of the best primal solution produced as measured against the dual bound from [8] (PS Gap) calculated as $100 \times$ (Engineer et al. Dual $-$ PS Primal)/(Engineer et al. Dual), a comparison of the quality of primal solutions produced with those produced without the use of problem-specific local search schemes (PS Gap BPGS) calculated as

Table 3
Effectiveness of problem-specific local search schemes.

Instance	PS Time to Best	PS Primal	PS Gap CPLEX	PS Gap BPGS	PS Gap	CPLEX Time to Beat or Tie PS
6-3-4-1	418.00	3033.85	3.68	2.81	0.00	
6-3-4-2	1254.00	1194.22	1.33	0.00	6.95	
6-3-4-3	298.00	4424.51	4.85	5.46	3.22	
6-3-4-4	1035.00	3431.37	−1.21	1.21	10.57	41,197.81
6-3-4-5	1329.00	2898.25	0.23	1.18	6.92	
6-3-4-6	327.00	6403.72	0.99	0.00	0.00	
6-3-4-8	1290.00	5388.14	0.00	2.88	0.00	16,586.04
6-3-4-9	351.00	6076.65	0.00	0.00	0.00	31,741.39
6-3-4-10	756.00	7019.86	0.84	0.58	1.02	
6-4-3-1	1126.00	2509.25	35.16	2.41	14.92	
6-4-3-2	1326.00	2813.47	8.96	3.25	13.64	
6-4-3-3	956.00	2993.39	22.41	4.94	8.81	
6-4-3-4	913.00	4856.55	7.61	6.00	4.47	
6-4-3-5	1276.00	2478.75	6.71	6.71	9.00	
6-4-3-6	1309.00	6249.46	1.47	0.77	0.00	
6-4-3-7	276.00	6034.51	0.00	0.98	0.00	1,740.89
6-4-3-8	584.00	5988.16	0.00	−0.02	0.02	144.00
6-4-3-9	260.00	5600.31	−0.90	0.09	1.30	22,818.89
6-4-3-10	436.00	6413.11	−0.00	0.00	1.45	19,317.26
6-4-4-1	1159.00	4772.72	10.32	9.24	7.67	
6-4-4-2	1575.00	5353.36	9.62	5.47	3.99	
6-4-4-3	1061.00	4174.39	8.93	10.58	2.57	
6-4-4-4	1589.00	4690.91	7.73	8.37	5.96	
6-4-4-5	753.00	4731.50	2.14	2.46	2.88	
6-4-4-6	311.00	5771.26	1.30	0.80	0.05	
6-4-4-7	430.00	5886.73	2.39	4.21	0.00	
6-4-4-8	548.00	5146.18	−0.30	−0.36	3.37	42,501.75
6-4-4-9	401.00	6443.28	0.51	−0.03	1.03	
6-4-6-1	473.00	6096.83	5.64	5.04	1.00	
6-4-6-2	1245.00	8455.46	2.49	6.10	0.97	
6-4-6-3	1472.00	6194.42	4.57	5.31	2.22	
6-4-6-4	1491.00	6785.29	12.90	2.17	2.16	
6-4-6-5	645.00	7905.64	3.49	7.51	2.51	
6-6-4-1	1323.00	5881.40	8.30	3.90	6.15	
6-6-4-2	767.00	5309.68	100.02	11.42	7.39	
6-6-4-3	1025.00	5492.01	8.35	3.86	6.00	
6-6-4-4	1356.00	5629.27	29.05	9.09	5.31	
6-6-4-5	1378.00	7785.81	3.69	2.43	1.35	
	908.47		5.76	3.60	3.72	22,006.00

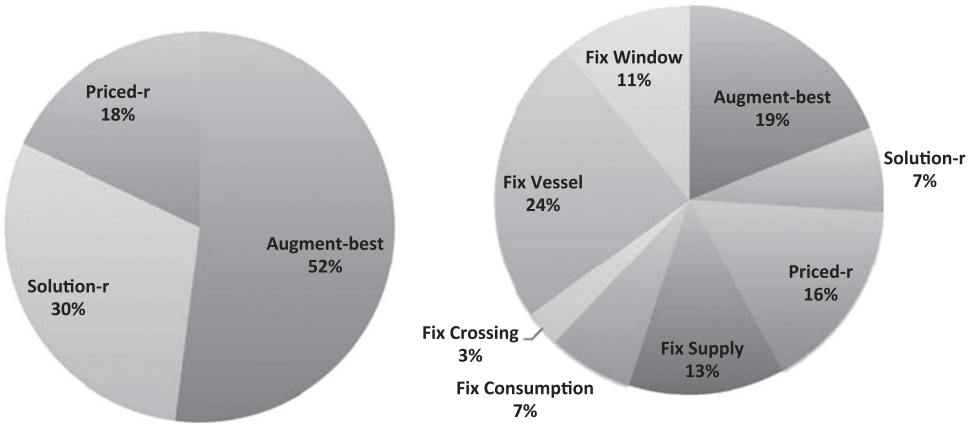

Fig. 3. Effectiveness of local search schemes: (a) BPGS and (b) BPGS with schemes.

$100 \times$ (PS Primal − BPGS Primal)/(PS Primal), and when able, the time CPLEX needed to find an equal or better solution (CPLEX Time to Beat or Tie PS).

We see that the problem-specific local search schemes are in fact effective, as, on average they enable BPGS to find better solutions (3.6% better) in less time (308.14 fewer seconds) than BPGS can find without using them. For the 23 instances for which BPGS could not find a solution that is within 5% of optimal using the problem-specific local search schemes lead to an average solution improvement of 5.43%. Using the bound produced by [8], we see that using these problem-specific schemes leads to solutions that are on average only 3.72% away from optimal. We also note that CPLEX, when given 12 h, is only able to produce an equivalent or better solution for eight of the 39 instances, and a solution that is at least 1% better for only one of those instances.

In Fig. 3(a) and (b), we report the percentage of total solution improvement that can be attributed to each of the local search schemes. In the pie chart on the left, we report the percentages for

BPGS, and in the pie chart on the right, we report the percentages for BPGS enhanced with problem-specific local search schemes. We see that all local search schemes contribute, but that *Fix Vessel* appears to be most effective.

Regarding scoring mechanisms, we note that all the mechanisms for choosing vessels (Dual-based, LP-based, and Profit-based) and windows (Dual-based, LP-based) to re-optimize, are effective, and that, on average, the LP-based mechanisms contribute the most towards total solution improvement.

6. Conclusions

We have applied BPGS to a maritime inventory routing problem. Our computational experiments further validate the potential of BPGS to be a near-black-box method for quickly producing high-quality solutions to meaningful instances of complex, real-life problems. We have also shown that with a larger time investment, it is possible, by exploiting problem knowledge in a more involved way, to enhance BPGS and to find better solutions in even less time. While designed for this maritime inventory routing problem, many of the problem-specific local search schemes can be applied to other variants of the IRP.

There are several avenues for future research. Regarding the MIRP, the structure of the problem suggests that the use of an extended formulation inspired by lot-sizing problems. In lot-sizing, it has been shown that using variables that represent both when product is produced and when it will be consumed can lead to a significantly stronger formulation. A similar idea may be used for the MIRP. Also, unlike many extended formulations, adding valid inequalities to the extended formulation solved by BPGS does not affect the pricing problem. Thus, developing such inequalities for the MIRP is another area we intend to explore. Regarding BPGS, we are interested in understanding whether there is benefit in embedding multiple different restrictions in $Ny \leq r$, or whether there is benefit in using different extended formulations based on different restrictions in different processors.

References

[1] Andersson H, Hoff A, Christiansen M, Hasle G, Løkketangen A. Industrial aspects and literature survey: combined inventory management and routingComputers & Operations Research 2010;37:1515–36.
[2] Archetti C, Speranza MG, Savelsbergh MWP. An optimization-based heuristic for the split delivery vehicle routing problem. Transportation Science 2008;42:22–31.
[3] Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MWP, Vance PH. Branch-and-price: column generation for huge integer programsOperations Research 1998;46:316–29.
[4] Christiansen M. Decomposition of a combined inventory and time constrained ship routing problem. Transportation Science 1999;33:3–16.
[5] Christiansen M, Nygreen B. Robust inventory ship routing by column generation. In: Desaulniers G, Desrosiers J, Solomon MM, editors. Column generation. US: Springer; 2005. p. 197–224.
[6] De Franceschi R, Fischetti M, Toth P. A new ILP-based refinement heuristic for vehicle routing problems. Mathematical Programming B 2006;105:471–99.
[7] Desrochers M, Desrosiers J, Solomon M. A new optimization algorithm for the vehicle routing problem with time windows. Operations Research 1992;40:342–54.
[8] Engineer FG, Furman K, Nemhauser GL, Savelsbergh MWP, Song, J-H. A branch-price-and-cut algorithm for a maritime inventory routing problem. Operations Research, http://dx.doi.org/10.1287/opre.1110.0997, in press.
[9] Glover F. A template for scatter search and path relinking. Lecture notes in computer science, vol. 1363; 1998. p. 13–54.
[10] Gronhaug R, Christiansen M, Desaulniers G. A branch-and-price method for a liquefied natural gas inventory routing problem. Transportation Science 2010;44:400–15.
[11] Gropp W. Using MPI: portable parallel programming with the message passing interfaceThe MIT Press; 1999.
[12] Hartl R, Hasle G, Janssens G. Special issue on rich vehicle routing problems. Central European Journal of Operations Research 2006;14:103–4.
[13] Hewitt M, Nemhauser G, Savelsbergh M. Combining exact and heuristic approaches for the capacitated fixed charge network flow problem. INFORMS Journal on Computing 2009;22:314–25.
[14] Hewitt M, Nemhauser G, Savelsbergh M. Branch-and-price guided search for integer programs with an application to the multicommodity fixed charge network flow problem. INFORMS Journal of Computing, http://dx.doi.org/10.1287/ijoc.1120.0503, in press.
[15] Pisinger D, Ropke S. A general heuristic for vehicle routing problems. Computers and Operations Research 2007;34:2403–35.
[16] Savelsbergh M, Song J-H. An optimization algorithm for inventory routing with continuous moves. Computers and Operations Research 2008;35:2266–82.
[17] Schmid V, Doerner KF, Hartl RF, Savelsbergh MWP, Stoecher W. A hybrid solution approach for ready-mixed concrete delivery. Transportation Science 2008;43:70–85.
[18] Song J-H, Furman KC. A maritime inventory routing problem: practical approach. Computers & Operations Research, http://dx.doi.org/10.1016/j.cor.2010.10.031, in press.