Rapid prototyping of parallel primal heuristics for domain specific MIPs: Application to maritime inventory routing

Lluís-Miquel Munguía[a,e], Shabbir Ahmed[b], David A. Bader[a],
George L. Nemhauser[b], Yufen Shao[c], Dimitri J. Papageorgiou[d]

[a] *College of Computing, Georgia Institute of Technology, Atlanta GA 30332*
[b] *School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta GA 30332*
[c] *ExxonMobil Upstream Research Company, Houston, TX 77098*
[d] *Corporate Strategic Research, ExxonMobil Research and Engineering Company, Annandale NJ 08801*
[e] *Corresponding author: lluis.munguia@gatech.edu*

Abstract

Parallel Alternating Criteria Search (PACS) relies on the combination of computer parallelism and Large Neighborhood Searches to attempt to deliver high quality solutions to any generic Mixed-Integer Program (MIP) quickly. While general-purpose primal heuristics are widely used due to their universal application, they are usually outperformed by domain-specific heuristics when optimizing a particular problem class. In this paper, we focus on the fast development of domain-specific parallel primal heuristics. Our approach entails specializing PACS to better adapt to the target problem structure. We showcase its application to two classes of the Maritime Inventory Routing Problem, an important application of MIPs to real world problems. We computationally compare the proposed modified framework with state-of-the art specialized algorithms and MIP solvers. Results show the effectiveness of our approach, and how the modular nature of PACS can provide an excellent platform for the rapid prototyping of parallel domain-specific heuristics.

Keywords: Parallel Computing, Primal Heuristics, Discrete Optimization, Maritime Inventory Routing, Large Neighborhood Search

1. Introduction

Mixed Integer Programming (MIP) [1] algorithms can solve a large variety of planning and operational problems in transportation [2], energy [3], production [4], and finance [5]. Formally, MIPs can be described as:

$$\min\{c^T x | Ax = b, l \leqslant x \leqslant u, x_i \in \mathbb{Z}, \forall i \in \mathcal{I}\} \tag{MIP}$$

where $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $\mathcal{I} \subseteq \{1, \ldots, n\}$ is the subset of integer variable indices. The decision vector x is bounded by $l \in \overline{\mathbb{R}}^n$ and $u \in \overline{\mathbb{R}}^n$, where $\overline{\mathbb{R}}$ is the extended set of real numbers $\mathbb{R} \cup \{-\infty, \infty\}$.

Primal heuristics are an essential component of MIP algorithms whose sole purpose is to find high quality solutions quickly. Though they rarely guarantee successfully finding such solutions, their role is of capital importance in providing solutions early in the search. In addition to producing valid upper bounds for the problem, high quality feasible solutions can help fathom a significant part of the solution space during the search, thus accelerating MIP solving.

Most MIP solvers based on Branch-and-Cut [6] are general-purpose tools that can be applied to any optimization problem that can be modeled as a MIP. These algorithms incorporate the use of primal heuristics as part of their core functionality. Due to their importance, a vast catalogue of primal heuristics [7, 8] has been developed and incorporated in all general-purpose MIP solvers. Because of their use as an integrated component, most heuristics must be of universal application, and cannot rely on any knowledge that may be applicable to only a handful of problem classes. However, general-purpose MIP solvers may prove to be inadequate for solving particular applications. Problem-specific algorithms capable of exploiting additional structural properties have been shown to outperform their general-purpose counterparts. Thus when solving difficult instances, it may be necessary to use specialized heuristics.

Parallel Alternating Criteria Search [9](PACS) is a parallel distributed-memory primal heuristic designed for finding solutions to any generic MIP. PACS relies on Large Neighborhood Search (LNS) techniques in order to find solution refinements very quickly. LNS's entail solving carefully restricted sub-MIPs derived from the original problem. The advantage is that this reduced problem is significantly easier to optimize quickly. It has also been proven to be a very successful technique for dealing with large MIPs. The heuristic proves to be very effective at finding high quality solutions to MIPs belonging to all kinds of applications, since it does not rely on assumptions regarding the underlying structure of the problem. The approach is competitive or better than state-of-the art MIP solvers for more than 90% of the instances in the MIPLIB2010 library [10].

In this paper, we seek to bridge the performance gap between general purpose and domain-specific heuristics. We investigate the suitability of PACS as a platform available for rapidly prototyping a specialization that better addresses the specific structure of the targeted MIP. Its application is showcased using a uniform set of real-world Maritime Inventory Routing Problem (MIRP)

instances. When specialized to better exploit the internal structure of MIRPs, PACS can be substantially more effective. Improvements in performance are made possible with two main novel contributions: new definitions of MIRP-specific search neighborhoods and a modified objective function.

The remainder of the paper is organized as follows: Section 2 introduces the PACS heuristic. In Section 3 we introduce the mathematical models of the Maritime Inventory Routing problem as well as details of the specialization. Section 4 presents computational experiments and results on standard instances from the literature. Finally, Section 5 provides some concluding remarks. Additionally, auxiliary Section 6 provides a review of the nomenclature used in the formulation.

2. Parallel Alternating Criteria Search

In PACS, two auxiliary MIP subproblems derived from the original problem are iteratively solved to attain a first feasible solution, and to improve it with respect to the original objective. The first of the auxiliary MIPs is a Feasibility MIP (FMIP), and poses the problem of finding a feasible starting solution as an optimization problem. Two vectors of continuous variables Δ^+ and Δ^- of size m (corresponding to the m constraints) are introduced. A decision vector is feasible to a MIP if and only if it can be extended to a solution of value 0 to the associated FMIP. Rather than directly solving FMIP, its difficulty is reduced by fixing a given subset $\mathcal{F}$ of the integer variables and optimizing the remaining. In turn, a second auxiliary problem, referred to as an Optimality MIP (OMIP), is aimed at improving a partially feasible vector with respect to the original objective. In OMIP, the same auxiliary slack variables are introduced in each constraint. In order to ensure that the optimal solution to OMIP remains at most as infeasible as the input solution $\hat{x}$, an additional constraint that limits the amount of slack is added, where the degree of infeasibility is bounded by $\sum_{i=1}^{m}(\hat{\Delta}_i^+ + \hat{\Delta}_i^-)$.

By iteratively solving carefully restricted neighborhoods of both auxiliary MIPs, the heuristic will hopefully converge (although its convergence is not guaranteed) to a high quality feasible solution. By construction, infeasibility decreases monotonically after each iteration. On the other hand, the solution quality may fluctuate with respect to the original objective. Figure 1 depicts the expected behavior of the algorithm.

$$\min \sum_{i=1}^{m} \Delta_i^+ + \Delta_i^-$$

s.t.

$$\begin{aligned} & Ax + I_m\Delta^+ - I_m\Delta^- = b \\ & x_j = \hat{x}_j, \forall j \in \mathcal{F} \\ & l \leqslant x \leqslant u \\ & x_j \in \mathbb{Z}, \forall j \in \mathcal{I} \\ & \Delta^+ \geqslant 0, \Delta^- \geqslant 0 \end{aligned} \tag{FMIP}$$

$$\min \quad c^T x$$

s.t.

$$\begin{aligned} & Ax + I_m\Delta^+ - I_m\Delta^- = b \\ & \sum_{i=1}^{m}(\Delta_i^+ + \Delta_i^-) \leqslant \sum_{i=1}^{m}(\hat{\Delta}_i^+ + \hat{\Delta}_i^-) \\ & x_j = \hat{x}_j, \forall j \in \mathcal{F} \\ & l \leqslant x \leqslant u \\ & x_j \in \mathbb{Z}, \forall j \in \mathcal{I} \\ & \Delta^+ \geqslant 0, \Delta^- \geqslant 0 \end{aligned} \tag{OMIP}$$

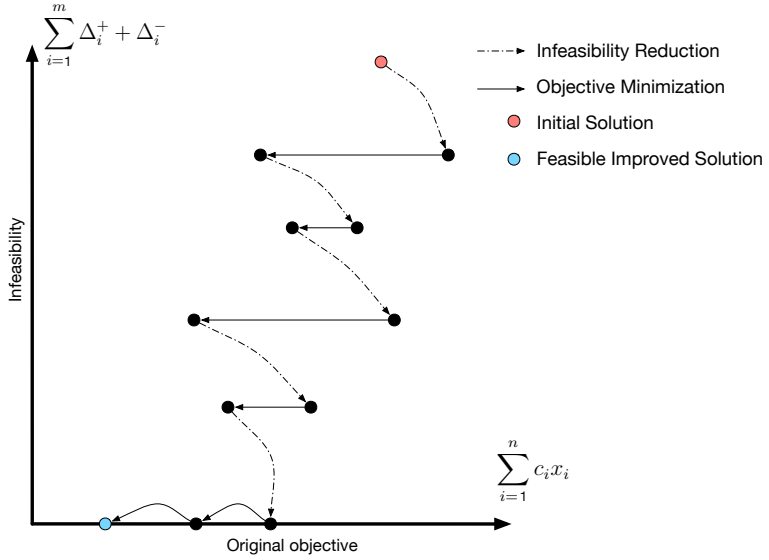


Figure 1: Transition to a high quality feasible solution

2.1. Parallelization of Alternating Criteria Search

We introduce parallelism by generating and solving a diversified set of large neighborhood searches in each step of the process, which are solved simultaneously. The improved solutions found in parallel are then combined by solving an additional sub-MIP, in which the variables that have the same value across the different solutions are fixed. Figure 2 depicts an example for a simple 0-1 knapsack instance. Firstly, the Feasibility MIP is derived from the original problem instance.

Next, two subproblems characterized by different fixings are solved in parallel. In a final step, the variables with coinciding values are fixed and a feasible solution is found.

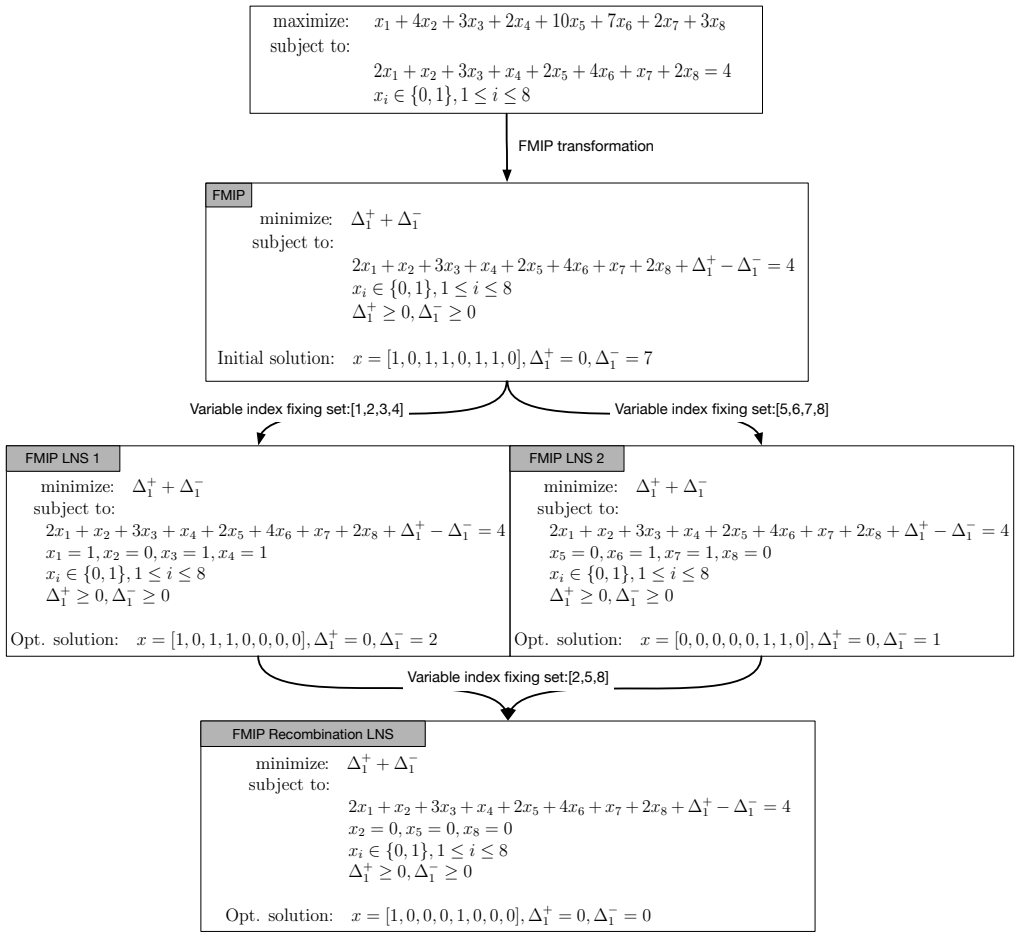


Figure 2: Example depicting a feasibility improvement iteration for a 0-1 knapsack sample instance

2.2. Specializing PACS

PACS is a heuristic designed for general purpose MIPs, and as such, none of its components leverages the MIP structure of the input problem. In order to increase the effectiveness of the heuristic, we introduce a set of problem-specific algorithmic improvements which substitute its generic counterparts. The proposed changes include the use of new LNS neighborhood definitions

as well as a modification to the objective function in FMIP and OMIP. With the latter measure, we hope to improve the rate of convergence to a first feasible solution.

2.2.1. Increasing the effectiveness of LNS

In Large Neighborhood Search, the set of variables to be fixed adjusts the difficulty of the sub-MIP to be optimized, as well as the effectiveness of the MIP solver at finding high quality solutions quickly. PACS uses a simple, generic, yet intuitive variable fixing scheme to be able to tackle any kind of problem structure. Greater effectiveness in finding solutions can be achieved by considering the internal structure of the problem in variable fixing. Numerous examples can be found in the literature for most common MIP applications. All high performing variable selection schemes intend to preserve the integrity and cohesiveness of substructures within the problem, as this is the key for decomposing it effectively. A successful variable fixing must identify which variables must be changed in order to find a better solution, free them, and leave the remainder fixed. Additionally, a variable fixing scheme must not be too restrictive, or very few improvements will be found as a result. Conversely, if not enough variables are fixed, the search space might be too large to find any improvements in a small amount of time. Neighborhood diversification is another key property, since parallel efficiency depends on it.

2.2.2. Improving the rate of convergence to a first feasible solution

As introduced so far, one of the primary objectives of PACS is to find a first feasible solution. Although the convergence of the heuristic is not guaranteed, the system in place has proven to be substantially effective when optimizing most problems of the MIPLIB2010 library [9]. When solving problems where feasibility is a challenge however, PACS is likely to stall and fail at finding any solution at all. Our second algorithmic modification is geared towards improving the heuristic's effectiveness at converging to a first feasible solution and reducing the probability of stalling.

We propose to use an objective penalization system in order to effectively enforce an order in which constraints are satisfied. To this end, the objective of FMIP is modified with an additional vector of weights $\Lambda \in [0, \infty)^m$ to bias the objective function, which becomes $\sum_{i=1}^{m} \Lambda_i(\Delta_i^+ + \Delta_i^-)$. The main goal of the measure is to achieve feasibility in the most critical constraints first. As a result, infeasibility is driven to secondary constraints instead, which should be easier to repair in the future.

Constraint satisfaction priorities are completely problem dependent. As shown in the experimental results, however, they can be instrumental in increasing the effectiveness of the heuristic when chosen appropriately.

3. Application to Maritime Inventory Routing

Currently, global seaborne logistics are the most utilized form of freight transportation and account for the vast majority of world trade. In most cases, the optimization of the costs related to these shipping operations is vital to their economic viability. As a result, these problems are ideal examples of real-world application of MIPs. In the petrochemicals sector alone, maritime inventory routing problems are solved regularly to assist business users make tactical and operational decisions. Maritime inventory routing problems (MIRP) are often formulated as MIPs and attempt to capture important practical details present in maritime shipping, such as inventory management at ports and vessels, as well as vessel routing and scheduling. Despite being an extremely flexible and faithful mathematical model, MIRP instances are very challenging to solve to provable optimality. In some cases, it is challenging to find a single feasible solution.

3.1. Related Work

Our primary focus is on methods for finding primal solutions to maritime inventory routing problems. Papageorgiou et al. [11] provide a thorough literature review. Papageorgiou et al. [12] conduct a comprehensive comparison of the state-of-the art in primal heuristics and exact methods used for solving this class of problems.

Maritime inventory routing problems have been applied to a broad range of applications, such as cement manufacturing [13], calcium carbonate slurry shipping [14], oil supply in stochastic scenarios [15], and routing and inventory management of vacuum gas oil [16]. MIRP instances present a real challenge to most commercial MIP solvers, and it can even be challenging to find a good feasible solution. Many construction heuristics have been proposed for MIRPs or similarly constructed problems, such as the ones presented in [17, 18, 19]. These are specialized algorithms designed to provide a first feasible solution at the beginning of the optimization. Construction heuristics usually rely on greedy procedures, multi-start local searches or solving restricted subproblems. Additional specialized heuristics for similar LNG routing problems are presented in [20].

A large number of works have proposed solution methods based on some variation of Large Neighborhood Search (LNS). LNS heuristics circumvent the complexity of the original problem by solving derived subproblems obtained by restricting a subset of the variables. A fully featured MIP solver is then used to optimize the subproblem, which delivers a solution valid to the original problem. LNS approaches differ in how the search neighborhood is defined. In the context of MIRP instances, a widely used LNS strategy entails fixing the variables related to a subset of the vessels. Different variations of this approach are used in [17, 21, 22, 23, 24, 12], and differ in how the subset of vessels is selected.

Song and Furman [24] apply LNS techniques in combination with a branch-and-cut algorithm to find improved solutions to an arc-flow model very similar to the one used in this section. On the same instances, Engineer et al. [25] introduce an alternative column generation formulation, and solve it using branch-cut-and-price in combination with several newly introduced classes of cuts. Hewitt et al. [21] follows by applying a branch-and-price guided search (a method previously used for Fixed-Charge Multicommodity Network Flow problems [26]) in order to find high quality solutions quickly. This approach uses a small amount of parallelism (four processors) to speed up the algorithm.

An alternative decomposition approach very popularly applied to MIRPs is based on rolling horizon techniques. In a rolling horizon heuristic, the planning horizon is subdivided into smaller overlapping subhorizons. Each subdivision can be characterized with a tractable subproblem, which can be consecutively solved in a limited amount of time. Rolling horizon heuristics are introduced in [27, 28, 29, 19]. A variation is the fix-and-relax heuristic proposed by Uggen et al. [30], in which all posterior integer decision variables not included in the subhorizon are relaxed, and left to be continuous. Another significant departure is the approximate dynamic programming approach proposed by Papageorgiou et al. [22], in which the MIRP instance is formulated as a dynamic programming problem and interpreted as a sequence of vessel dispatching problems. Outside of the realm of heuristics, Goel et al. [31] introduce a constraint programming method based on a disjunctive scheduling representation.

Most of the aforementioned works focus on large MIRPs with large planning horizons. Papageorgiou et al. [23] focuses on an operational MIRP with a much smaller horizon, a more detailed model and more challenging from the perspective of feasibility. They overcome the added complexity by designing a two-stage algorithm, in which decisions are first made among loading/discharging

regions. In a second step, more detailed routing decisions are made at the ports within each of the regions.

To the best of our knowledge, the works of Asokan et al. [32] are the only attempt at introducing parallel heuristics for LNG inventory routing problems, which are a special class of MIRPs. In this work, the authors parallelize LNS heuristics introduced in [17].

Parallel MIP solvers such as GUROBI [33], CPLEX [34], and ParaSCIP [35] are the only alternative parallel algorithms currently available. In addition to high quality solutions, MIP solvers also provide a lower bound. Studies have suggested that parallelizing the branch-and-bound search may not scale well to a large number of cores [36], and this behavior is reflected in some of the aforementioned distributed-memory implementations. Another disadvantage is the fact that MIP solvers are general algorithms and they usually do not exploit the underlying network structure that characterizes MIRPs.

3.2. A time-space discretization of MIRPs

In the remainder of this section, we introduce the arc-based discrete-time formulations used in the modeling of MIRPs. Further, we perform a preliminary analysis of the performance of PACS when applied to the problems at hand. To conclude, we showcase a set of problem-specific modifications in order to further improve the performance of the parallel algorithm.

MIRPs model deep-sea vessel routing with inventory tracking at every port-vessel pair throughout a time-space network. A depiction of the model used is shown in Figure 3. Given a set $\mathcal{T}$ of time periods and a set of $\mathcal{J}$ of ports, the network consists of a set of nodes $\mathcal{N}_{s,t}$, which symbolize the state of the ports through different time periods. Additionally, a source node n_s and a sink node n_t are used to symbolize the entrance/exit of vessels to the system. A set of directed arcs $\mathcal{A}$ model the travel of vessels between ports. More concretely, each vessel v uses a set of dedicated arcs $\mathcal{A}^v$. For a particular pair of node n and vessel v, we may define the forward star $\mathcal{FS}_n^v$ as the set of outgoing arcs associated with a vessel v leaving from node n. Conversely, the reverse star $\mathcal{RS}_n^v$ denotes the set of incoming arcs.

The MIRPLIB library [11] consists of a set of MIRP instances inspired by real-world problems. The library is composed of multiple instance classes, which differ in the operational resolution and the planning horizon. Group 1 instances feature an operational MIRP with planning horizons of 45 and 60 periods, multiple ports per region and split pickups and deliveries. Group 1 instances

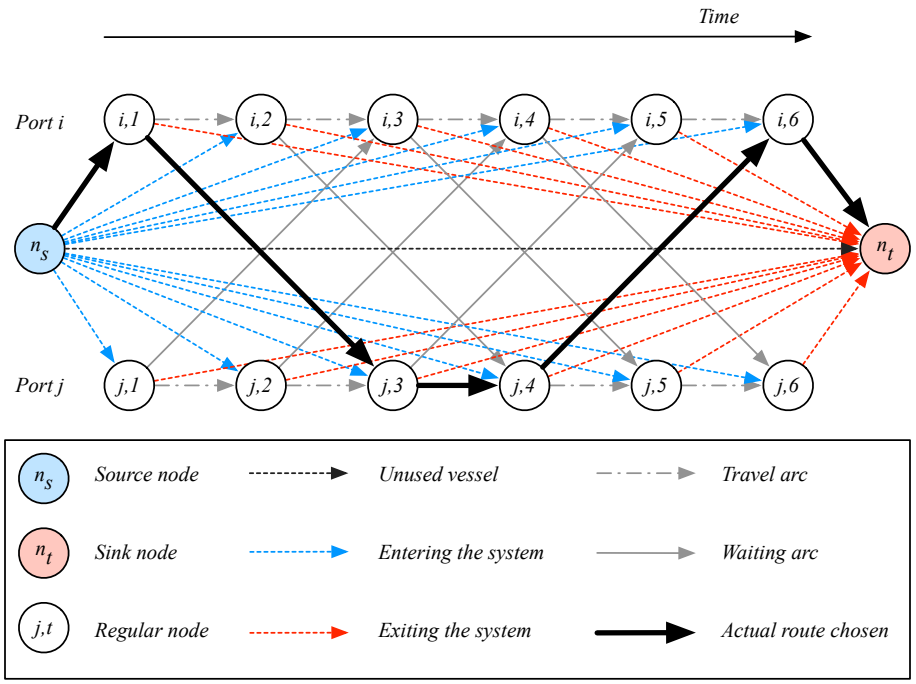


Figure 3: Time-space horizon modeling of MIRP

present a challenge from the perspective of feasibility, and most commercial MIP solvers struggle to find a single feasible solution. In contrast, group 2 instances offer simplified models with planning horizons greater than 60 time periods, but only involving one port per region and never split pickups and deliveries. Feasibility is trivial for group 2 instances, and the challenge is rather to find high quality feasible solutions quickly.

3.2.1. Group 1 instances: the challenge of feasibility

In group 1 instances, the objective is to maximize the revenue obtained when product is delivered to a port and to minimize the expenses incurred by the transportation costs, penalties for buying/selling product to the spot market as well as the delays in their loading/unloading. The model features two sets of binary decision variables: $x \in \{0,1\}^{|\mathcal{A}|}$ and $z \in \{0,1\}^{|\mathcal{N}| \times |\mathcal{V}|}$. x_a^v takes value 1 if vessel v uses a travel arc a. The binary variable z_n^v indicates whether vessel v loads/discharges at node n. The constraints ensure the coherency of the model as follows: constraints (1b) guarantee the conservation of vessel flow for each triplet of port, time, and vessel. Constraints (1c) and

(1d) ensure the inventory is balanced at each vessel and port throughout timesteps and maintained within limits. Constraints (1e) guarantee that the number of loads/discharges at a given pair of port and time does not exceed the number of available berths. The coupling constraints (1f) ensure a vessel can only load/discharge at a node if it is present. Constraints (1g) and (1h) require the vessels to travel at capacity from a loading region to a discharging region and empty when traveling in the opposite direction. The amount of product that a port can buy/sell in the spot market is restricted by (1i) and (1j). A thorough explanation of each of the symbols is provided in the Appendix.

$$\max \sum_{n\in\mathcal{N}}\sum_{v\in\mathcal{V}} R_n f_n^v - \sum_{v\in\mathcal{V}}\sum_{a\in\mathcal{A}^v} C_a^v x_a^v - \sum_{v\in\mathcal{V}}\sum_{n\in\mathcal{N}} (t\epsilon_z) z_n^v - \sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}} P_{j,t}\alpha_{j,t} \tag{1a}$$

$$s.t. \sum_{a\in\mathcal{FS}_n^v} x_a^v - \sum_{a\in\mathcal{RS}_n^v} x_a^v = \begin{cases} 1 & \text{if } n = n_s \\ -1 & \text{if } n = n_t \\ 0 & \text{if } n \in \mathcal{N} \end{cases}, \qquad \forall n \in \mathcal{N}_{s,t}, \forall v \in \mathcal{V} \tag{1b}$$

$$s_{j,t} = s_{j,t-1} + \Delta_j(d_{j,t} - \sum_{v\in\mathcal{V}} f_n^v - \alpha_{j,t}), \qquad \forall n = (j,t) \in \mathcal{N} \tag{1c}$$

$$s_t^v = s_{t-1}^v + \sum_{n=(j,t)\in\mathcal{N}} \Delta_j f_n^v, \qquad \forall t \in \mathcal{T}, \forall v \in \mathcal{V} \tag{1d}$$

$$\sum_{v\in\mathcal{V}} z_n^v \leqslant B_j, \qquad \forall n = (j,t) \in \mathcal{N} \tag{1e}$$

$$z_n^v \leqslant \sum_{a\in\mathcal{RS}_n^v} x_a^v, \qquad \forall n = (j,t) \in \mathcal{N}, \forall v \in \mathcal{V} \tag{1f}$$

$$s_t^v \geqslant Q^v x_a^v, \qquad \forall v \in \mathcal{V}, \forall a = ((j_1,t),(j_2,t')) \in \mathcal{A}^v : \quad j_1 \in \mathcal{J}^P, j_2 \in \mathcal{J}^C \cup \{n_t\} \tag{1g}$$

$$s_t^v \leqslant Q^v(1 - x_a^v), \qquad \forall v \in \mathcal{V}, \forall a = ((j_1,t),(j_2,t')) \in \mathcal{A}^v : \quad j_1 \in \mathcal{J}^C, j_2 \in \mathcal{J}^P \cup \{n_t\} \tag{1h}$$

$$\sum_{t\in\mathcal{T}} \alpha_{j,t} \leqslant \alpha_j^{\max}, \qquad \forall j \in \mathcal{J} \tag{1i}$$

$$0 \leqslant \alpha_{j,t} \leqslant \alpha_{j,t}^{\max}, \qquad \forall j \in \mathcal{J}, \forall t \in \mathcal{T} \tag{1j}$$

$$F_{j,t}^{\min} z_{j,t}^v \leqslant f_{j,t}^v \leqslant F_{j,t}^{\max} z_{j,t}^v, \qquad \forall n = (j,t) \in \mathcal{N}, \forall v \in \mathcal{V} \tag{1k}$$

$$S_{j,t}^{\min} \leqslant s_{j,t} \leqslant S_{j,t}^{\max}, \qquad \forall n = (j,t) \in \mathcal{N} \tag{1l}$$

$$0 \leqslant s_t^v \leqslant Q^v, \qquad \forall v \in \mathcal{V}, \forall t \in \mathcal{T} \tag{1m}$$

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

$$z_n^v \in \{0,1\}, \qquad \forall n = (j,t) \in \mathcal{N}, \forall v \in \mathcal{V} \tag{1o}$$

Figure 4: Group 1 MIRP formulation

3.2.2. Group 2 instances: the challenge for optimality

Group 2 instances feature long-horizon deterministic routing with simplified operational constraints. The objective remains to minimize the transportation costs and penalties caused by buying/selling product from/to the spot market. The only integer decision variables are x_a^{vc}, which determine the number of vessels belonging to vessel class vc that take arc a. Similar to group 1 instances, constraints (2b) ensure the conservation of flow for each triplet of port, time, and vessel class. Inventory restrictions for each pair of port and time are specified in (2c), and constraints (2d) ensure that the number of vessels that attempt to load/discharge is limited by the number of berths available. This formulation does not require inventory tracking of vessels, since vessels are required to travel at capacity from a loading region to a discharging region and empty when traveling in the opposite direction. In contrast to group 1 instances, there are no constraints that limit the amount of stockout, although stockout is penalized in the objective.

$$\max \sum_{vc \in \mathcal{VC}} \sum_{a \in \mathcal{A}^{vc}} -C_a^{vc} x_a^{vc} + \sum_{j \in \mathcal{J}} \sum_{t \in \mathcal{T}} -P_{j,t} \alpha_{j,t} \tag{2a}$$

$$s.t. \sum_{a \in \mathcal{FS}_n^{vc}} x_a^{vc} - \sum_{a \in \mathcal{RS}_n^{vc}} x_a^{vc} = \begin{cases} 1 & \text{if } n = n_s \\ -1 & \text{if } n = n_t \\ 0 & \text{if } n \in \mathcal{N} \end{cases}, \qquad \forall n \in \mathcal{N}_{s,t}, \forall vc \in \mathcal{VC} \tag{2b}$$

$$s_{j,t} = s_{j,t-1} + \Delta_j (d_{j,t} - \sum_{vc \in \mathcal{VC}} \sum_{a \in \mathcal{FS}_n^{vc,inter}} Q^{vc} x_a^{vc} - \alpha_{j,t}), \qquad \forall n = (j,t) \in \mathcal{N} \tag{2c}$$

$$\sum_{vc \in \mathcal{VC}} \sum_{a \in \mathcal{FS}_n^{vc,inter}} x_a^{vc} \leqslant B_j, \qquad \forall n = (j,t) \in \mathcal{N} \tag{2d}$$

$$\alpha_{j,t} \geqslant 0, \qquad \forall n = (j,t) \in \mathcal{N} \tag{2e}$$

$$s_{j,t} \in [S_{j,t}^{\min}, S_{j,t}^{\max}], \qquad \forall n = (j,t) \in \mathcal{N} \tag{2f}$$

$$x_a^{vc} \in \{0,1\}, \qquad \forall vc \in \mathcal{VC}, \forall a \in \mathcal{A}^{vc,inter} \tag{2g}$$

$$x_a^{vc} \in \mathbb{Z}_+, \qquad \forall vc \in \mathcal{VC}, \forall a \in \mathcal{A}^{vc} \backslash \mathcal{A}^{vc,inter} \tag{2h}$$

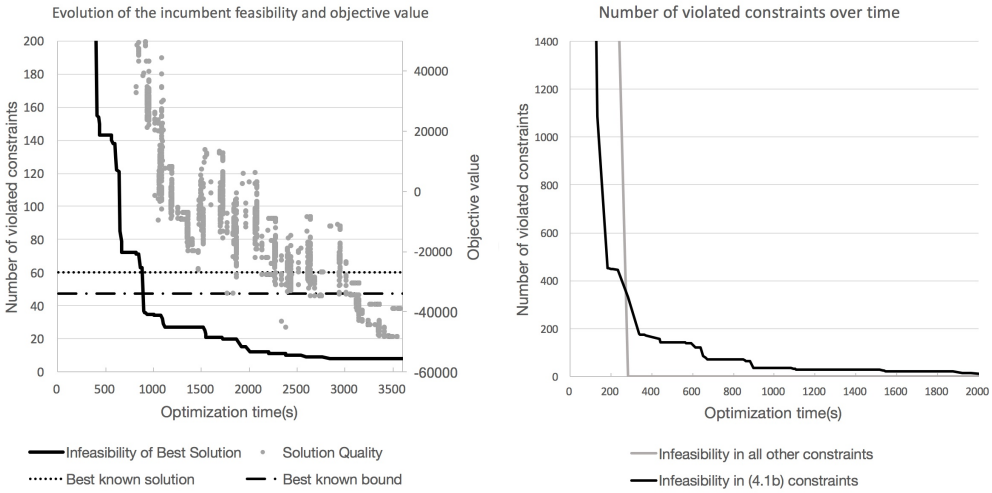
Figure 5: Group 2 MIRP formulation

3.3. Applying PACS to MIRP instances

One of our primary focuses is on the application of PACS to group 1 instances. Our computational experiments presented in Subsection 4 indicate that the algorithm struggles considerably to

converge to feasible solutions. The reason for this is an apparent stalling of the heuristic, and its inability to repair all infeasibilities. We illustrate the issue in Figure 6(a), where the performance of the heuristic is depicted for a particular problem instance. Specifically, we display the number of violated constraints in the incumbent solution as a function of time. The objective value of all the solutions found by the heuristic are also plotted on the secondary axis. As seen in the chart, the infeasibility of the incumbent solution is reduced significantly at the beginning of the optimization. Meanwhile, the quality of the solutions found also converges to a value significantly lower than the best known bound for the problem. Since not all constraints are enforced, it is possible to obtain infeasible solutions with a better objective than the best bound. PACS is unable to repair all infeasibilities. Figure 6(b) provides further information regarding the nature of the infeasible constraints. We determine that the algorithm fails in solutions featuring a few unsatisfied flow conservation constraints. While we illustrate the issue with a particular example, the same issue can be extended to most of the instances in the set, as detailed in Subsection 4.3.

We apply both algorithm modifications introduced in the previous section, which attempt to



(a) (b)

Figure 6: Application of PACS to a group 1 MIRP instance, with 4 loading ports, 9 discharging ports, 17 vessels, and a horizon of 60 timesteps. (a) Depicts the evolution of the infeasibility of the incumbent, and objective value of found solutions as a function of time. (b) Shows a breakdown of the infeasibility found in the incumbent by constraint type as a function of time.

increase the effectiveness of the algorithm at finding a first feasible solution. The first proposed modification is to the objective of FMIP, with the intent of preventing infeasibility from accumulating in constraints that may be very hard to repair. Secondly, we introduce MIRP-specific variable fixing schemes, which may be more effective at finding quality solutions than their generic counterparts.

3.3.1. Objective penalizations

Finding feasible solutions for group 1 instances is challenging due to the small tolerances imposed in the inventory constraints of ports and vessels. A feasible vessel schedule must ensure enough product is transported to satisfy the consumption demand at every port and to avoid stockout. Vessels must carry the bulk of the product, since constraints $(1i) \cup (1j)$ severely limit the amount of product that can be bought or sold on the spot market. This is not the case for group 2 instances. In the latter, no upper bound on the spot market is imposed.

In the spirit of group 2 instances, our intent is to prioritize the satisfiability of the vessel schedule over the inventory management at ports. In the modified Group1FMIP, we apply a large penalty Λ to all variables except the ones representing the constraints regulating port inventory limits (the set $(1i) \cup (1j)$). A priori, the generated solutions will provide feasible routing schedules for each vessel at the expense of violating many stock deficiency/excess constraints. As optimization advances, we hope the latter can be gradually fixed by systematically re-adjusting feasible vessel trips. The excess of slack $\alpha_{j,t}$ is also minimized in OMIP, since it is present in the original objective. Therefore, the amount of infeasibility is minimized in both auxiliary MIP models.

As seen in Group1FMIP, constraints are divided into two separate submatrices. A^{α} represents the submatrix related to the constraints regulating the excess slack $\alpha_{j,t}$, the set $(1i) \cup (1j)$. Let k be the cardinality of such set. In turn, $\hat{A}$ contains the submatrix of the remaining constraints. Δ variables associated with the constraints in $\hat{A}$ are penalized with Λ in the objective, in order to prioritize the satisfiability of the constraints in A^{α}.

$$\min \sum_{i=1}^{m-k} \Lambda(\Delta_i^+ + \Delta_i^-) + \sum_{i=1}^{k} \Delta_i^{+\alpha} + \Delta_i^{-\alpha}$$

s.t.

$$\begin{aligned}
&\hat{A}x + I\Delta^+ - I\Delta^- = \hat{b} \\
&A^\alpha x + I\Delta^{+\alpha} - I\Delta^{-\alpha} = b^\alpha \\
&x_i = \hat{x}_i, \forall i \in \mathcal{F} \\
&l \leqslant x \leqslant u \\
&x_i \in \mathbb{Z}, \forall i \in \mathcal{I} \\
&\Delta^+ \geqslant 0, \Delta^- \geqslant 0 \\
&\Delta^{+\alpha} \geqslant 0, \Delta^{-\alpha} \geqslant 0
\end{aligned} \tag{Group1FMIP}$$

3.3.2. A MIRP-specific variable fixing scheme

We propose to use two kinds of MIRP-specific variable fixing schemes that decompose the problems into complementary substructures. The first is a variation of a vessel decomposition approach, and the latter is a modification of a time-window variable fixing scheme. Both incorporate randomness in order to satisfy the need for parallel diversified search neighborhoods.

A prevalent large neighborhood search algorithm in the literature is the so called k-opt search, which entails fixing the variables belonging to all but a subset of vessels. This is a widely used strategy due to its simplicity and effectiveness. We specify ours with two key elements. We incorporate an input parameter ρ, which determines the proportion of vessels to be fixed. In addition, we incorporate randomness in the selection of vessels. The latter is necessary in order to leverage computer parallelism. In the proposed algorithm, a diversified set of large neighborhood searches is generated by selecting different subsets of variables to fix. By solving them simultaneously, we hope to increase the chances of finding solution improvements. Pseudocode is provided in Algorithm 1.

While the k-opt search decomposes the problem by vessel, the second strategy we propose seeks to decompose the problem by time. The time-window selection scheme involves establishing a time window between two timesteps and fixing all travel arcs outside of it. The generated neighborhood allows the improvement of a solution by modifying the travel schedules of all vessels within the allowed time window. Travel arcs related to all ports and vessels are left unfixed at the same time.

Algorithm 1 Random vessel selection neighborhood

Input: Fraction of variables to fix ρ, $0 < \rho < 1$

Output: Set of integer indices $\mathcal{F}$

function VesselSelectionFixing(ρ)
    $\mathcal{F}$ =set of all integer variable indices $\mathcal{I}$
    while $|\mathcal{F}| > \rho \cdot |\mathcal{I}|$ **do**
        i:= random vessel $i \in \mathcal{V}$
        Remove from $\mathcal{F}$ all variable indices related to vessel i
    end while
    return $\mathcal{F}$
end function

Pseudocode is provided in Algorithm 2.

The time-window selection scheme as presented can become terribly ineffective at finding solution improvements if a small parameter ρ is selected, or if the problem features a large number of vessels and ports. In such cases, the produced variable selection may encompass a small time window. Any vessel trip longer than this time window will have a fraction of its active travel variables fixed and won't be rerouted when solving the associated LNS.

We present a modification of the standard time-window variable selection in Algorithm 3, in which a subset of the ports are fixed for each vessel in order to allow larger time windows. In order to determine which subset of ports remains fixed, we define the auxiliary concept of a port

Algorithm 2 Random time-window selection neighborhood

Input: Fraction of variables to fix ρ, $0 < \rho < 1$

Output: Set of integer indices $\mathcal{F}$

function TimeWindowSelectionFixing(ρ)
    $\mathcal{F}$ =set of all integer variable indices $\mathcal{I}$
    $offset = 0$
    t:= random time $t \in \mathcal{T}$
    while $|\mathcal{F}| > \rho \cdot |\mathcal{I}|$ **do**
        Remove from $\mathcal{F}$ all variable indices related to time $t - offset$, for vessels $v \in \mathcal{V}$ and ports $p \in \mathcal{J}$
        Remove from $\mathcal{F}$ all variable indices related to time $t + offset$, for vessels $v \in \mathcal{V}$ and ports $p \in \mathcal{J}$
        $offset = offset + 1$
    end while
    return $\mathcal{F}$
end function

span. The port span of a vessel v between two time steps t_1 and t_2 in a solution x is the set of ports traversed by v during the time window in x. An example depicting multiple examples of port spans is shown in Figure 7. By its definition, the produced set contains only the ports visited by v within the time window. The proposed constrained time-window selection scheme incorporates the definition of a port span in order to include only the relevant ports for each vessel in the variable selection, while the remaining are fixed.

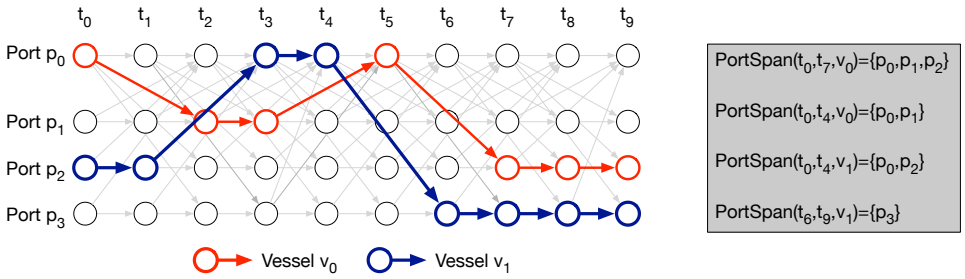


Figure 7: The port span of a vessel v between two time steps t_1 and t_2 in a solution x is the set of ports traversed by v during the time window in x.

The proposed algorithm also prioritizes the optimization of the port and time pairs with the largest excess/deficiency of product. For this purpose, all port-time pairs (p, t) are ranked by the value of their associated $\alpha_{p,t}$ variable. A single pair (p', t') is selected randomly among the pairs with the largest amount. Then, a time-window is defined using t' as its epicenter. For each vessel v, the algorithm proceeds to unfix the variables within the time window belonging to the port span and p'. The size of the time window is incremented gradually until the desired amount of variables is selected.

With (p', t') as the epicenter of the time-window, we intend to rectify the deficiency of stock at port p' and time t' by rerouting some vessel to p' before the timestep t'. By using the port span, we force non-relevant ports to remain fixed in hopes of producing an easier subproblem with a larger time-window that contains the high quality solutions.

If feasibility has already been achieved, pairs (p, t) are ranked by their absolute contribution to the original objective instead. Modifications in high ranking pairs will hopefully have a more significant impact in improving the solution.

Each of the proposed variable fixing strategies provides a substantially different set of search neighborhoods, which add to the diversity of the approach. We incorporate both by allowing each

parallel thread to choose randomly among both strategies, as shown in Algorithm 4.

Algorithm 3 Constrained time-window selection neighborhood

Input: Fraction of variables to fix ρ, $0 < \rho < 1$, input solution x

Output: Set of integer indices $\mathcal{F}$

function ConstrainedTimeWindowSelectionFixing(ρ)
  Generate all pairs (p, t)
  Rank all pairs by inventory excess/deficit $\alpha_{j,t}$
  Select pair (p', t') randomly among the pairs with most excess/deficit
  $offset = 0$
  $\mathcal{F}$ =set of all integer variable indices $\mathcal{I}$
  while $|\mathcal{F}| > \rho \cdot |\mathcal{I}|$ **do**
    $t_1 := t' - offset$
    $t_2 := t' + offset$
    for every vessel $v \in \mathcal{V}$ **do**
      P:=PortSpan(t_1, t_2, v, x)
      $P := P \cup p'$
      for all ports $p \in P$ **do**
        Remove from $\mathcal{F}$ all variable indices related to vessel i and port p between $[t_1, t_2]$
      end for
    end for
    $offset := offset + 1$
  end while
  return $\mathcal{F}$
end function
function PortSpan(t_1, t_2, v, x)
  $P := \emptyset$
  for each $t \in \{t_1, \ldots, t_2\}$ **do**
    for each $p \in \mathcal{P}$ **do**
      if vessel v is traversing p at time t **then**
        $P := P \cup p$
      end if
    end for
  end for
end function

Algorithm 4 Variable fixing selection algorithm

Input: Fraction of variables to fix ρ, $0 < \rho < 1$, input solution x

Output: Set of integer indices $\mathcal{F}$

```
1: function HybridVariableFixing(ρ)
2:     Generate random integer n
3:     if n%2 = 0 then
4:         F:=VesselSelectionFixing(ρ)
5:     else
6:         F:=ConstrainedTimeWindowSelectionFixing(ρ)
7:     end if
8:     return F
9: end function
```

4. Experimental Results

In this section, we evaluate the performance and behavior of the MIRP-specific Parallel Alternating Criteria Search (MIRPpacs) when solving instances in the MIRPLIB library [11]. Out of the 100 instances currently present, there are 28 categorized as group 1 instances, while the remaining belong to group 2. MIRPLIB group 2 instances feature a planning horizon of 120, 180 or 360 periods and are classified in three difficulty categories [12]. A group 2 instance is declared easy if at least one commercial MIP solver (in default settings) is able to close more than 90% of the gap (as defined in Section 4.1) in half an hour. On the other hand, if no MIP solver is able to close more than 10% of the optimality gap, the instance is labeled as hard. Of the 72 group 2 instances, the library contains 21 easy instances, 25 medium instances, and 26 hard instances. The difficulty of a problem instance is typically dependent on the number of ports, vessels, and time periods. Most easy instances contain fewer than 5 discharging ports and no more than 180 time periods, while all but 2 hard instances have more than 8 discharging ports and at least 180 time periods. MIRPpacs is implemented in C++, using CPLEX 12.7.2 as a backbone solver. We compare our framework against the state-of-the-art general purpose MIP solver CPLEX 12.7.2, and the default version of PACS. All of our computations are performed on an 8-node computing cluster, each with two Intel Xeon X5650 6-core processors (96 cores in total) and 24 GB of RAM memory.

4.1. Evaluation of primal solution quality

We evaluate the quality of primal solutions in terms of the primal gap and primal integral, as described in [37]. Given a solution x for a MIP with an optimal solution $\hat{x}$, the primal gap $\gamma(x) \in [0, 1]$ of x is defined as:

$$\gamma(x) = \begin{cases} 0 & \text{if } |c^T\hat{x}| = |c^T x| = 0 \\ 1 & \text{if } c^T\hat{x} \cdot c^T x < 0 \\ \frac{|c^T\hat{x} - c^T x|}{\max\{|c^T\hat{x}|, |c^T x|\}} & \text{else.} \end{cases} \tag{3}$$

Given a time limit $t_{\max}$, we define the primal gap function $p : [0, t_{\max}] \mapsto [0, 1]$ as:

$$p(t) = \begin{cases} 1 & \text{if no solution is found by time } t \\ \gamma(x(t)) & \text{with } x(t) \text{ being the incumbent solution} \\ & \text{at point } t, \text{ else.} \end{cases} \tag{4}$$

The primal gap function is monotonically decreasing and measures the progress of the optimization towards the optimal solution. The primal integral is defined as:

$$P(t) = \int_{t=0}^{T} p(t)\mathrm{d}t \tag{5}$$

The primal integral $P(t)$ captures the notion of how early solutions are found. Both $p(t)$ and $P(t)$ are considered powerful metrics when evaluating the performance of finding high quality primal solutions.

In addition to the primal gap, the optimality gap Γ captures the difference between the best found upper and lower bounds, z_{UB} and z_{LB}:

$$\Gamma = \frac{z_{UB} - z_{LB}}{z_{UB}} \ . \tag{6}$$

4.2. Tuning of parameters and nondeterminism

MIRPpacs and PACS require two kinds of input parameters that regulate the difficulty and the solution time of each LNS within the heuristic. The parameter pair $[\rho, t]$ determine the percentage of variables to be fixed and the LNS time limit (measured in seconds). For group 1 instances, the pair $[0.7, 5]$ is selected. In turn, group 2 instances are solved using $[0.2, 5]$, $[0.5, 5]$ and $[0.7, 5]$, for Easy, Medium and Hard instances respectively. CPLEX is set in its parallel distributed-memory

nondeterministic setting in order to utilize all 96 available cores and with a focus on primal solutions (the emphasis on Hidden Feasible solutions setting). Settings are set to default, otherwise. All the compared parallel algorithms are of nondeterministic nature, and we repeat each of the experiments five times. Unless otherwise noted, the performance charts presented in this section display the average among all runs.

The following set of figures and tables evaluate the quality of the solutions provided by the different methods when solving group 1 instances. To our knowledge, the only heuristic developed for this specific problem set was presented by Papageorgiou et al. [23]. The construction heuristic was based on a construction phase in which multiple starts were tested to obtain a solution to a coarse-grained, aggregate version of the problem. This was followed by solution polishing at each regional level and local search. A Gurobi-based implementation was used, and its performance results are presented under the label CH+LS. While the heuristic is sequential, it can take advantage of 12-core shared-memory parallelism.

4.3. Group 1 MIRP instances

In Figure 8(a), we show the evolution of the average primal gap as a function of time. As stated in its definition, a primal gap of 100% is assigned if no solution for a particular instance is found. At first glance, standard PACS is only able to find solutions for a small subset of instances. As a result it scores a comparatively higher average primal gap throughout the optimization. The reason for its poor performance has already been analyzed in section 3.3, and it is one of the main motivating factors behind the development of MIRPpacs. CPLEX is a full-fledged MIP solver. As such, it is not afflicted by the same stalling problem and performs slightly better. CH+LS is able to find solutions for significantly more instances than CPLEX, despite using only a fraction of the cores. The algorithmic modifications introduced in MIRPpacs certainly make a difference in comparison to its generic counterpart. The specialized heuristic becomes effective at finding high quality solutions for most instances, and this is reflected in a much improved average gap right from the beginning of the optimization.

The same differences are reflected in Figure 8(b), in which the primal integral is depicted instead. The differences are quite significant after one hour of optimization, as MIRPpacs shows an average primal integral that is 2.5 times smaller than CPLEX.

Figure 9(a) plots the percentage of group 1 instances for which a feasible solution is found

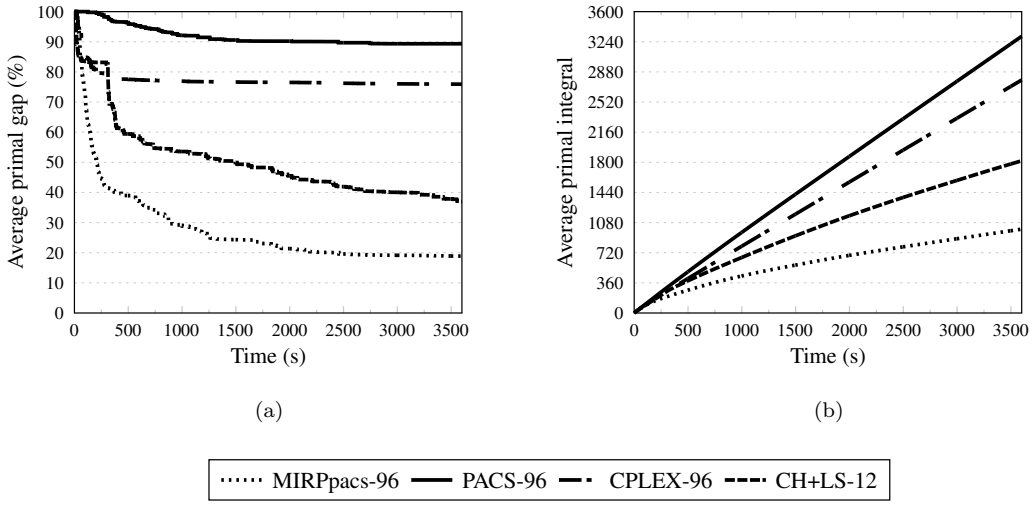


Figure 8: (a) Average primal gap and (b) average primal integral for group 1 instances

by each of the compared methods. MIRPpacs is able to find feasible solutions for 85% of the instances, CH+LS is able to do so for 69%, while CPLEX stalls at 25% and standard PACS at

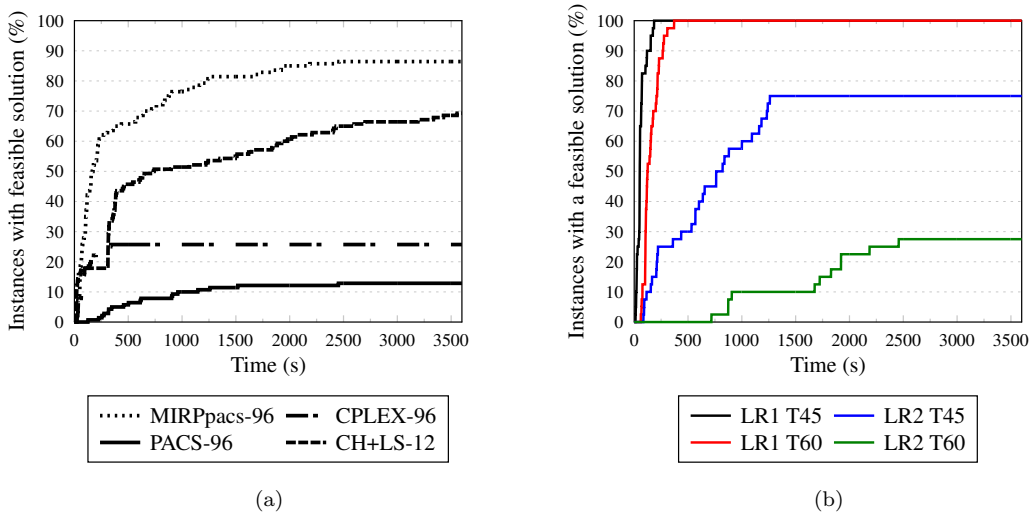


Figure 9: (a) Percentage of instances for which a feasible solution is found. (b) Percentage of instances for which MIRPpacs finds a feasible solution, broken down by instance subclass

18%. In Figure 9(b), the performance of MIRPpacs is shown after group 1 instances are subdivided by the number of time periods (45 vs. 60) and by the number of loading regions (1 vs. 2). Note that a region may contain multiple ports. Results show that instances with a single region are significantly easier than their multiple region counterparts, as MIRPpacs is able to find solutions for 100% of the single loading region instances in less than 730 seconds. Increasing the number of regions raises the complexity. When 45 period instances are considered, solutions are found for 75% of the cases. Performance drops significantly for instances with two regions and 60 ports.

Table 1 provides further details about the variability in the performance of the two best non-deterministic approaches. We analyze the statistical results of the multiple executions in terms of the average, standard deviation, minimum, maximum, and median of the primal gap and primal

Table 1: Group 1: Performance comparison of non-deterministic approaches

		Parallel Alternating Criteria Search			**CPLEX (Opportunistic Mode)**		
Time Cutoff(s)		180	600	3600	180	600	3600
Avg.	P. Gap(%)	53.02	37.10	18.89	80.96	77.36	75.90
	P. Integral	137.16	311.19	1001.33	160.31	489.16	2782.76
	Count (%)	52.14	67.86	86.43	22.14	25.71	25.71
Std dev	Primal Gap	9.51	4.40	3.84	1.74	1.62	2.05
	Primal Integral	10.10	29.63	128.12	1.17	9.06	60.65
	Count (%)	9.94	3.19	1.60	1.60	1.60	1.60
min	Primal Gap	42.10	31.88	13.41	78.44	74.72	72.51
	Primal Integral	125.98	274.80	837.57	158.92	475.78	2679.98
	Count (%)	42.86	64.29	85.71	21.43	25.00	25.00
median	Primal Gap	54.13	37.32	19.45	81.21	77.86	76.58
	Primal Integral	136.45	310.89	1005.48	160.22	490.63	2804.07
	Count (%)	50.00	67.86	85.71	21.43	25.00	25.00
max	Primal Gap	63.24	42.13	22.56	82.32	78.37	77.17
	Primal Integral	149.93	346.80	1158.71	161.75	496.56	2818.57
	Count (%)	64.29	71.43	89.29	25.00	28.57	28.57

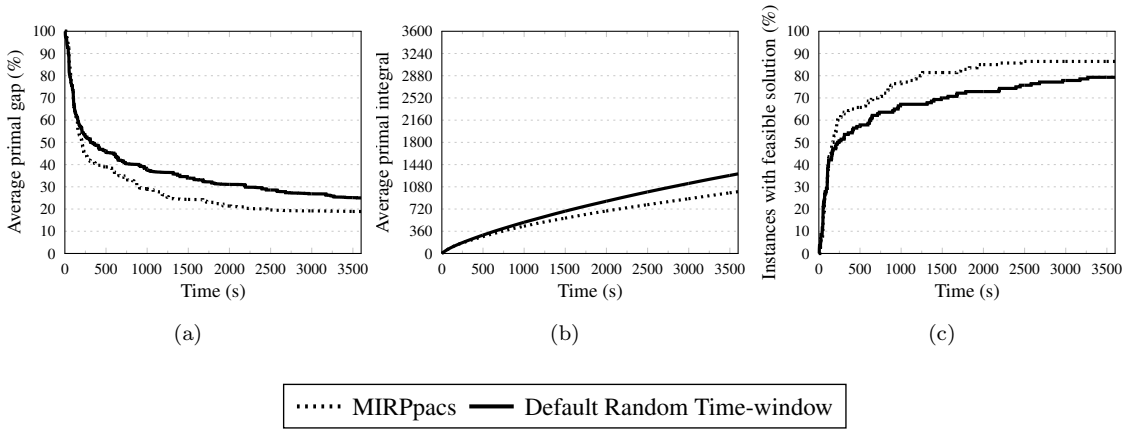


Figure 10: (a) Percentage of instances for which a feasible solution is found. (b) Percentage of instances for which MIRPpacs finds a feasible solution, broken down by instance subclass

integral for different time cutoffs. MIRPpacs shows more variability than CPLEX, especially at early stages of the optimization. However, the relative distance between both diminishes with time. Even when the worst run of MIRPpacs is compared with the best run of CPLEX, the parallel heuristic displays a better primal gap after 180 seconds than the one achieved by CPLEX after one hour.

In Section 3.3.2, we presented a modification of the standard time-window variable fixing approach. In the proposed variant, a subset of the ports are fixed for each vessel with the objective of allowing larger time windows. In addition, our variable fixing scheme prioritizes optimizing the tuples of port and time with the largest amount of stock deficiency first. In Figure 10, we evaluate the impact of the proposed modifications by comparing MIRPpacs to a variant of the heuristic in which the standard time-window variable fixing strategy (Algorithm 2) is used instead.

Our proposed modification allows MIRPpacs to find feasible solutions for 7% more instances. After one hour of optimization, the version using the standard time-window displays a primal integral that is 28% larger, and an average primal gap that is 32% higher.

4.4. Group 2 MIRP instances

We incorporate a state-of-the-art MIRP-specific heuristic to the comparison, such as the rolling horizon heuristic as introduced in Papageorgiou et al. [12]. As explained in Subsection 3.1, the

rolling horizon heuristic is a very common form of time decomposition applied to MIRPs. After the authors in the aforementioned work compared multiple state-of-the-art primal algorithms for MIRPs, the provided heuristic proved to be one of the best performing construction heuristics. While it is a sequential algorithm, it takes advantage of the shared-memory parallelism provided by the underlying MIP solver. The data displayed in the following charts is the one presented in the original work. The rolling horizon heuristic (RHH) was run using a single computing node with 8 parallel cores. With the purpose of eliminating the discrepancies between computer systems, we also report its performance after normalizing the CPU times according to the performance metrics in Passmark [38]. Normalized times are calculated as $T_{\text{norm}} = \frac{T_{\text{orig}} \cdot S_{\text{orig}}}{S_{\text{norm}}}$, where T_{orig} is the original time reported by the authors, while S_{norm}[1] and S_{orig}[2] are the CPU scores of the processors used in our experiments and the other authors in the comparison respectively.

Figure 11(a) shows the evolution of the average primal time as a function of time for all compared methods. When all instances are considered, MIRPpacs proves to be the best performing algorithm. It is able to achieve an average primal gap of 20% in less than 168 seconds, while the next algorithm

[1] $S_{\text{norm}} = 7605$ (Intel Xeon X5650)
[2] $S_{\text{orig}} = 14403$ (Intel Xeon E5-2687W)

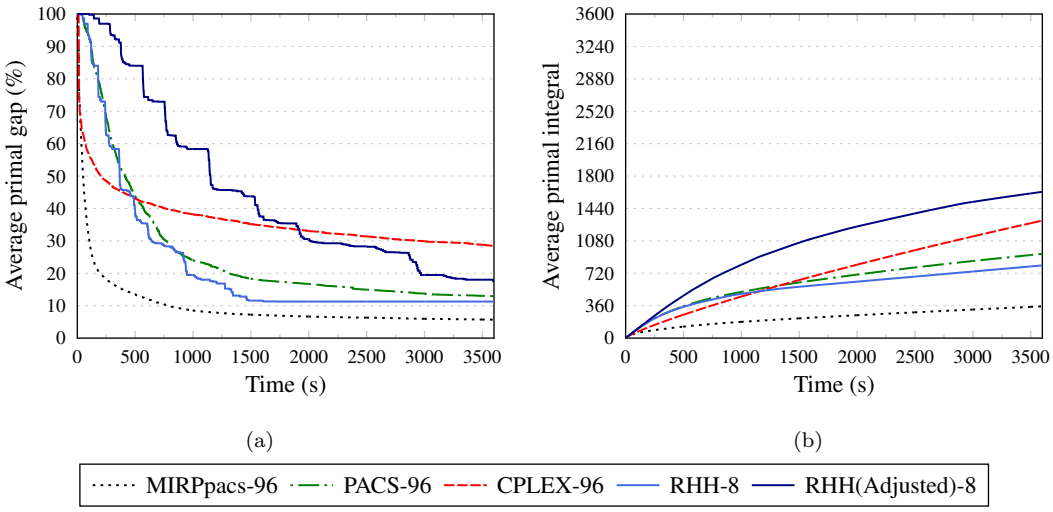


Figure 11: (a) Average primal gap and (b) average primal integral for all group 2 instances.

to achieve the same requires 5 times as much time. It also requires less than 600s to find solutions with an average gap of less than 10%. The mark of 10% is not surpassed by any other method. The commendable performance of standard PACS is also worth noting, as it performs similarly to the non-normalized version of the rolling horizon heuristic despite being a heuristic for general purpose MIPs. CPLEX seems to follow a different pattern to the aforementioned heuristics, as it is able to perform on the level of MIRPpacs at the beginning of the optimization. However, it is surpassed by most heuristics after 500 seconds and ends as the worst performing contender after 2000 seconds. The rolling horizon heuristic proves to be a better performer than most parallel methods despite using only 8 cores.

The primal integral is plotted in Figure 11(b). Similarly reflected as in the previous plot, MIRPpacs shows a significantly lower average primal integral. Precisely, it is 2.5 times better than the next best contender after 3600 seconds. CPLEX shows a slight advantage at the beginning of the optimization versus PACS and RHH. The advantage is neutralized after 1200 seconds.

In Figure 12, group 2 instances are split by difficulty category. When only small instances are

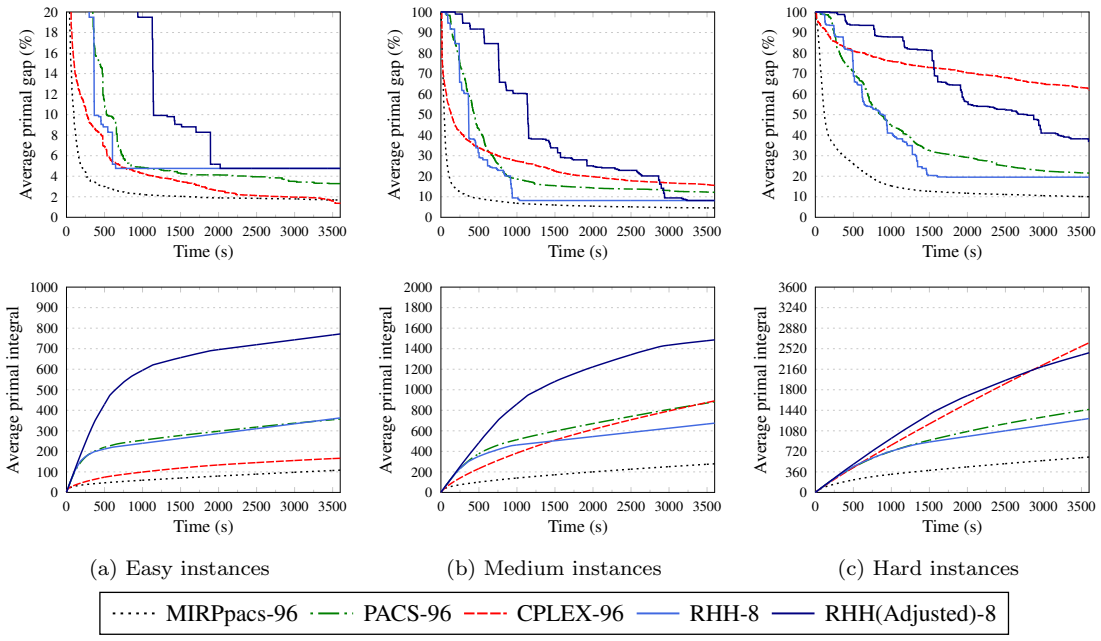


Figure 12: Average primal gap and average primal integral for (a) easy, (b) medium, and (c) hard group2 instances.

considered, a full featured MIP solver is able provide high quality solutions after other methods stagnate. CPLEX is able to outperform all methods at the end of the optimization and obtain the best average gap. However, MIRPpacs presents a lower primal integral due to the fact that it converges to high quality solutions much earlier in the search. The advantage of the primal heuristics gradually increases as the problem size grows. RHH shows better performance than PACS for both medium and hard instances, though PACS uses 12 times as many cores. The combination of domain-specific improvements and parallelism allow MIRPpacs to be particularly effective for hard instances in comparison to its general purpose counterpart.

Table 2 provides details regarding the effects of nondeterminism on the variability between experiments. MIRPpacs has a similar variability to the one displayed by CPLEX. The standard deviation on the primal gap decreases as the optimization advances. The trend seen in group 1 instances is maintained, as MIRPpacs achieves a better average primal gap in less than 180 seconds than CPLEX is able to achieve after 1 hour of optimization. As a result MIRPpacs shows a primal integral that is 3.7 times better on average.

At the time of this writing, primal heuristics remain the best available option for handling

Table 2: Group 2: Performance comparison of non-deterministic approaches

		Parallel Alternating Criteria Search			**CPLEX (Opportunistic Mode)**		
Time Cutoff(s)		180	600	3600	180	600	3600
Avg.	P. Gap(%)	20.75	12.17	5.70	51.19	41.76	27.98
	P. Integral	75.48	140.56	352.36	111.26	302.07	1305.22
Std dev	Primal Gap	3.84	3.71	2.23	3.29	2.56	2.24
	Primal Integral	5.75	18.79	86.46	3.55	12.52	67.22
min	Primal Gap	16.53	8.10	3.45	47.75	38.49	25.08
	Primal Integral	68.77	119.81	263.15	107.54	288.25	1226.04
median	Primal Gap	20.49	11.72	5.28	50.87	41.79	28.14
	Primal Integral	75.39	138.20	339.69	110.98	301.13	1305.10
max	Primal Gap	25.82	17.15	8.95	55.51	44.75	30.56
	Primal Integral	82.97	166.18	476.49	116.09	318.04	1391.54

MIRP instances of practical size, as they significantly outperform state-of-the-art MIP solvers in the process. In turn, specialized heuristics will always outperform their general purpose counterparts.

5. Conclusions

Parallel Alternating Criteria Search proves to be an effective framework when solving Maritime Inventory Routing Problems, but it can be significantly improved by tailoring a few of the key components of the algorithm. Firstly, we introduce specific objective penalizations, with the intent of improving the discovery of a first feasible solution. Secondly, we introduce new definitions of MIRP-specific variable fixing schemes, in order to improve the effectiveness of the large neighborhood search. The new specialized parallel heuristic is able to significantly outperform state-of-the-art MIP solvers and domain specific heuristics. The advantage increases considerably when solving hard instances featuring long horizon periods, and a large number of ports and vessels.

The specialization process described in this paper can be easily adapted to other MIP problem classes as well. When targeting a specific problem, we preliminarily recommend running the heuristic in its vanilla form. If feasible solutions are not found, it is highly likely that all the infeasibility is accumulated in a single constraint class. The next step is to penalize the related Δ variables and run Parallel Alternating Criteria Search again. Our recommendation is then to iterate on penalizing the constraint sets stalling the heuristic until feasible solutions are found. In order to improve the convergence to a high quality feasible solution, the generic variable fixing scheme can be readily exchanged for a problem specific counterpart.

Parallel Alternating Criteria Search is an excellent platform that can be used as is, or for a rapid prototyping of heuristics for particular MIP domains. It can be applied as a standalone heuristic or embedded in an exact branch-and-bound algorithm. We hope this work will motivate researchers to apply Parallel Alternating Criteria Search to their own MIP domains. The framework is readily available for download [39].

6. Nomenclature

6.1. Indices and sets

$t \in \mathcal{T}$	set of time periods with $T = \|\mathcal{T}\|$
$v \in \mathcal{V}$	set of vessels
$vc \in \mathcal{VC}$	set of vessel classes
$j \in \mathcal{J}^P$	set of production ports
$j \in \mathcal{J}^C$	set of consumption ports
$j \in \mathcal{J}$	set of all ports:$\mathcal{J} = \mathcal{J}^P$
$n \in \mathcal{N}$	set of regular nodes or port-time pairs: $\mathcal{N} = \{n = (j,t) : j \in \mathcal{J}, t \in \mathcal{T}\}$
$n \in \mathcal{N}_{s,t}$	set of all nodes, including the source node n_s and a sink node n_t
$a \in \mathcal{A}$	set of all arcs
$a \in \mathcal{A}^v$	set of arcs associated with vessel $v \in \mathcal{V}$
$a \in \mathcal{A}^{vc}$	set of arcs associated with vessel class $vc \in \mathcal{VC}$
$a \in \mathcal{FS}_n^v$	set of all outgoing arcs associated with node $n = (j,t) \in \mathcal{N}_{s,t}$ and vessel $v \in \mathcal{V}$
$a \in \mathcal{FS}_n^{vc}$	set of all outgoing arcs associated with node $n = (j,t) \in \mathcal{N}_{s,t}$ and vessel class $vc \in \mathcal{VC}$
$a \in \mathcal{RS}_n^v$	set of all outgoing arcs associated with node $n = (j,t) \in \mathcal{N}_{s,t}$ and vessel $v \in \mathcal{V}$
$a \in \mathcal{RS}_n^{vc}$	set of all outgoing arcs associated with node $n = (j,t) \in \mathcal{N}_{s,t}$ and vessel class $vc \in \mathcal{VC}$
$\mathcal{FS}_n^{vc,inter}$	set of all outgoing interregional arcs for node $n = (j,t) \in \mathcal{N}_{s,t}$ and vessel class $vc \in \mathcal{VC}$

6.2. Problem data

$\alpha_{j,t}^{\max}$	upper bound on the amount of product that can be bought/sold at the spot market at port $j \in \mathcal{J}$ and time $t \in \mathcal{T}$
$\alpha_j^{\max}$	upper bound on the cumulative amount of product that can be bought/sold at the spot market at port $j \in \mathcal{J}$ over the entire planning horizon
B_j	number of berhs available at port $j \in \mathcal{J}$
C_a^v	cost for vessel $v \in \mathcal{V}$ to traverse arc $a \in \mathcal{A}^v$
C_a^{vc}	cost for vessel class $vc \in \mathcal{VC}$ to traverse arc $a \in \mathcal{A}^v$
$d_{j,t}$	number of units produced or consumed at port $j \in \mathcal{J}$ in time period $t \in \mathcal{T}$
Δ_j	an indicator parameter taking value $+1$ if $j \in \mathcal{J}^P$, and -1 otherwise
ϵ_z	nonnegative cost parameter associated with attempting to load or discharge at a port

$F_{j,t}^{\min}(F_{j,t}^{\max})$	minimum (maximum) amount of product that can be loaded or discharged at port $j \in \mathcal{J}$ from a single vessel in time period $t \in \mathcal{T}$
$P_{j,t}$	nonnegative penalty parameter associated with one unit of lost production or stock-out at port $j \in \mathcal{J}$ in time period $t \in \mathcal{T}$
Q^v	capacity of vessel $v \in \mathcal{V}$
Q^{vc}	capacity of vessel class $vc \in \mathcal{VC}$
R_n	the unit sales revenue for product discharged at port-time pair $n = (j,t) \in \mathcal{N}$
$S_{j,t}^{\min}(S_{j,t}^{\max})$	lower bound (capacity) at port $j \in \mathcal{J}$ in time period $t \in \mathcal{T}$
$s_{j,0}$	initial inventory at port $j \in \mathcal{J}$
s_0^v	initial inventory on vessel $v \in \mathcal{V}$

6.3. Problem decision variables

$\alpha_{j,t}$	(continuous) amount of product that port $j \in \mathcal{J}$ purchases or sells to the spot market in time period $t \in \mathcal{T}$
f_n^v	(continuous) amount loaded or discharged at port-time pair $n = (j,t) \in \mathcal{N}$ from vessel $v \in \mathcal{V}$
$s_{j,t}$	(continuous) number of units of inventory at port $j \in \mathcal{J}$ available at the end of period $t \in \mathcal{T}$
s_t^v	(continuous) number of units of inventory on vessel $v \in \mathcal{V}$ available at the end of period $t \in \mathcal{T}$
x_a^v	(binary) takes value 1 if vessel $v \in \mathcal{V}$ uses arc a incident to node $n = (j,t) \in \mathcal{N}$
x_a^{vc}	(integer) takes value 1 if vessel class $vc \in \mathcal{VC}$ uses arc a incident to node $n = (j,t) \in \mathcal{N}$
z_n^v	(binary) takes value 1 if vessel $v \in \mathcal{V}$ attempts to load or discharge product at node $n = (j,t) \in \mathcal{N}$

7. Acknowledgements

This research has been supported in part by ExxonMobil Upstream Research Company, the National Science Foundation, the Office of Naval Research and the Air Force Office of Scientific Research.

Bibliography

[1] G. L. Nemhauser, L. A. Wolsey, Integer programming and combinatorial optimization, Wiley, Chichester. GL Nemhauser, MWP Savelsbergh, GS Sigismondi (1992). Constraint Classification for Mixed Integer Programming Formulations. COAL Bulletin 20 (1988) 8–12.

[2] M. SteadieSeifi, N. Dellaert, W. Nuijten, T. V. Woensel, R. Raoufi, Multimodal freight transportation planning: A literature review, European Journal of Operational Research 233 (1) (2014) 1 – 15. doi:https://doi.org/10.1016/j.ejor.2013.06.055.
URL http://www.sciencedirect.com/science/article/pii/S0377221713005638

[3] R. Baños, F. Manzano-Agugliaro, F. Montoya, C. Gil, A. Alcayde, J. Gómez, Optimization methods applied to renewable and sustainable energy: A review, Renewable and Sustainable Energy Reviews 15 (4) (2011) 1753 – 1766. doi:https://doi.org/10.1016/j.rser.2010.12.008.
URL http://www.sciencedirect.com/science/article/pii/S1364032110004430

[4] Y. Pochet, L. A. Wolsey, Production planning by mixed integer programming, Springer Science & Business Media, 2006.

[5] D. Bertsimas, C. Darnell, R. Soucy, Portfolio construction through mixed-integer programming at grantham, mayo, van otterloo and company, Interfaces 29 (1) (1999) 49–66.

[6] A. Land, A. Doig, An automatic method of solving discrete programming problems, Econometrica: Journal of the Econometric Society (1960) 497–520.

[7] T. Berthold, Primal heuristics for mixed integer programs, Master's thesis, TU Berlin (2006).

[8] M. Fischetti, A. Lodi, Heuristics in mixed integer programming, Wiley Encyclopedia of Operations Research and Management Science.

[9] L.-M. Munguía, S. Ahmed, D. A. Bader, G. L. Nemhauser, Y. Shao, Alternating criteria search: a parallel large neighborhood search algorithm for mixed integer programs, Computational Optimization and Applicationsdoi:10.1007/s10589-017-9934-5.
URL https://doi.org/10.1007/s10589-017-9934-5

[10] T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. E. Bixby, E. Danna, G. Gamrath, A. M. Gleixner, S. Heinz, A. Lodi, H. Mittelmann, T. Ralphs, D. Salvagnin, D. E. Steffy, K. Wolter, MIPLIB 2010, Mathematical Programming Computation 3 (2) (2011) 103–163.

[11] D. J. Papageorgiou, G. L. Nemhauser, J. Sokol, M.-S. Cheon, A. B. Keha, Mirplib–a library of maritime inventory routing problem instances: Survey, core model, and benchmark results, European Journal of Operational Research 235 (2) (2014) 350–366.

[12] D. J. Papageorgiou, M.-S. Cheon, S. Harwood, F. Trespalacios, G. L. Nemhauser, Recent progress using matheuristics for strategic maritime inventory routing, in: Modeling, Computing and Data Handling Methodologies for Maritime Transportation, Springer, 2018, pp. 59–94.

[13] M. Christiansen, K. Fagerholt, T. Flatberg, Øyvind Haugen, O. Kloster, E. H. Lund, Maritime inventory routing with multiple products: A case study from the cement industry, European Journal of Operational Research 208 (1) (2011) 86 – 94. doi:http://dx.doi.org/10.1016/j.ejor.2010.08.023.
URL http://www.sciencedirect.com/science/article/pii/S0377221710005606

[14] S. Dauzère-Pérès, A. Nordli, A. Olstad, K. Haugen, U. Koester, P. O. Myrstad, G. Teistklub, A. Reistad, Omya hustadmarmor optimizes its supply chain for delivering calcium carbonate slurry to european paper manufacturers, Interfaces 37 (1) (2007) 39–51.
URL http://www.jstor.org/stable/20141461

[15] A. Agra, M. Christiansen, A. Delgado, L. M. Hvattum, A maritime inventory routing problem with stochastic sailing and port times, Computers & Operations Research 61 (2015) 18 – 30. doi:http://dx.doi.org/10.1016/j.cor.2015.01.008.
URL http://www.sciencedirect.com/science/article/pii/S0305054815000210

[16] K. C. Furman, J.-H. Song, G. R. Kocis, M. K. McDonald, P. H. Warrick, Feedstock routing in the exxonmobil downstream sector, Interfaces 41 (2) (2011) 149–163.
URL http://www.jstor.org/stable/23016240

[17] V. Goel, K. C. Furman, J.-H. Song, A. S. El-Bakry, Large neighborhood search for lng inventory routing, Journal of Heuristics 18 (6) (2012) 821–848.

[18] M. Stålhane, J. G. Rakke, C. R. Moe, H. Andersson, M. Christiansen, K. Fagerholt, A construction and improvement heuristic for a liquefied natural gas inventory routing problem, Computers & Industrial Engineering 62 (1) (2012) 245–255.

[19] Y. Shao, K. C. Furman, V. Goel, S. Hoda, A hybrid heuristic strategy for liquefied natural gas inventory routing, Transportation Research Part C: Emerging Technologies 53 (2015) 151–171.

[20] F. Mutlu, M. K. Msakni, H. Yildiz, E. Sönmez, S. Pokharel, A comprehensive annual delivery program for upstream liquefied natural gas supply chain, European Journal of Operational Research 250 (1) (2016) 120–130.

[21] M. Hewitt, G. Nemhauser, M. Savelsbergh, J.-H. Song, A branch-and-price guided search approach to maritime inventory routing, Computers & Operations Research 40 (5) (2013) 1410 – 1419. doi:http://dx.doi.org/10.1016/j.cor.2012.09.010.
URL http://www.sciencedirect.com/science/article/pii/S0305054812002183

[22] D. J. Papageorgiou, M.-S. Cheon, G. Nemhauser, J. Sokol, Approximate dynamic programming for a class of long-horizon maritime inventory routing problems, Transportation Science 49 (4) (2014) 870–885.

[23] D. J. Papageorgiou, A. B. Keha, G. L. Nemhauser, J. Sokol, Two-stage decomposition algorithms for single product maritime inventory routing, INFORMS Journal on Computing 26 (4) (2014) 825–847.

[24] J.-H. Song, K. C. Furman, A maritime inventory routing problem: Practical approach, Computers & Operations Research 40 (3) (2013) 657–665.

[25] F. G. Engineer, K. C. Furman, G. L. Nemhauser, M. W. P. Savelsbergh, J.-H. Song, A branch-price-and-cut algorithm for single-product maritime inventory routing, Operations Research 60 (1) (2012) 106–122. doi:10.1287/opre.1110.0997.

[26] M. Hewitt, G. Nemhauser, M. W. P. Savelsbergh, Branch-and-price guided search for integer programs with an application to the multicommodity fixed-charge network flow problem, INFORMS Journal on Computing 25 (2) (2013) 302–316. doi:10.1287/ijoc.1120.0503.

[27] F. Al-Khayyal, S.-J. Hwang, Inventory constrained maritime routing and scheduling for multicommodity liquid bulk, part i: Applications and model, European Journal of Operational Research 176 (1) (2007) 106–130.

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

[29] A. Agra, M. Christiansen, A. Delgado, L. Simonetti, Hybrid heuristics for a short sea inventory routing problem, European Journal of Operational Research 236 (3) (2014) 924–935.

[30] K. T. Uggen, M. Fodstad, V. S. Nørstebø, Using and extending fix-and-relax to solve maritime inventory routing problems, Top 21 (2) (2013) 355–377.

[31] V. Goel, M. Slusky, W.-J. van Hoeve, K. C. Furman, Y. Shao, Constraint programming for lng ship scheduling and inventory management, European Journal of Operational Research 241 (3) (2015) 662–673.

[32] B. V. Asokan, K. C. Furman, V. Goel, Y. Shao, G. Li, Parallel large-neighborhood search techniques for lng inventory routing, Submitted for publication.

[33] Gurobi optimizer, `http://www.gurobi.com` (2015).

[34] IBM CPLEX optimizer, `http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/` (2015).

[35] Y. Shinano, T. Achterberg, T. Berthold, S. Heinz, T. Koch, ParaSCIP: A parallel extension of SCIP, in: Competence in High Performance Computing 2010, Springer, 2012, pp. 135–148.

[36] T. Koch, T. Ralphs, Y. Shinano, Could we use a million cores to solve an integer program?, Mathematical Methods of Operations Research 76 (1) (2012) 67–93.

[37] T. Berthold, Measuring the impact of primal heuristics, Operations Research Letters 41 (6) (2013) 611–614.

[38] P. S. P. Ltd, Passmark software, [Online; accessed 2017-08-05] (2017).
URL `http://passmark.com/`

[39] L.-M. Munguia, Pacs, https://bitbucket.org/llmunguia/parallel-alternating-criteria-search, [Online] (2017).
URL `https://bitbucket.org/llmunguia/parallel-alternating-criteria-search`