A Multi-Start Algorithm and a Large Neighborhood Search for a Maritime Inventory Routing Problem

1st Marcelo W. Friske
Informatics Department
Federal University of Rio Grande do Sul
Porto Alegre, Brazil
mwfriske@inf.ufrgs.br

2nd Luciana S. Buriol
Informatics Department
Federal University of Rio Grande do Sul
Porto Alegre, Brazil
buriol@inf.ufrgs.br

***Abstract*—Maritime Inventory Routing Problem (MIRP) is a challenging combinatorial problem in which decision solutions evolve vessels route and schedule, besides the inventory management at the ports along a limited planning horizon. Common solution approaches for solving MIRPs use matheuristics combining heuristic elements and mathematical programming for obtaining high-quality solutions in relatively short processing time. The performance of such approaches can be compromised if the problem size grows considerably. To provide an alternative approach for solving larger MIRPs, we propose a multi-start metaheuristic capable of producing several solutions for the problem in short processing time. Additionally, we developed a large neighborhood search for improving a subset of the generated solutions, which are then used to build a reduced mixed-integer problem which is solved by a mathematical solver. Computational results demonstrated that our algorithm can obtain solutions in short processing time, compared to the current solutions methods, and it can solve larger problem instances, in which no solutions were known.**

***Index Terms*—Maritime inventory routing problem, multi-start algorithm, large neighborhood search, mixed integer program.**

I. Introduction

Heuristics usually are used for solving combinatorial problems in which exact approaches cannot provide solutions in reasonable processing times. On the other hand, building even a feasible solution for highly constrained problems with heuristic approaches can be a hard task. In such cases, it is preferable to use hybrid strategies, such as *matheuristics*, which combines heuristic and exact methods to obtain high-quality solutions. However, even using matheuristics, the computational time for some problems can be prohibitive, limiting the size of the instances that can be solved efficiently.

In this work, we propose a matheuristic composed of a multi-start metaheuristic, a large neighborhood search (LNS), and a reduced mixed-integer problem (RMIP) for solving a Maritime Inventory Routing Problem (MIRP) described by [1]. In essence, the MIRP aims to define the route and schedule of a set of vessels, including the amount of product(s) loaded and unloaded at each visited port. Additionally, it must manage the inventory at ports, keeping its level between lower and upper limits along the planning horizon. Several constraints can be incorporated into the problem considering real scenarios, leading to a very hard combinatorial problem.

Solution approaches for MIRPs presented in the literature usually are related to hybrid methods, combining heuristic and exact methods such as rolling-horizon heuristics [2], [3] and relax-and-fix [4], [5]. In some cases, exact approaches are proposed to examine the quality of the formulation [6], [7], but the problems usually consider some simplifications and small instances. Pure heuristic approaches, i.e. heuristic or metaheuristics that do not use an exact approach as black-box are less frequently found. In [8] is presented a parameterized constructive heuristic for a MIRP related to the cement industry. The heuristic iteratively selects the best vessel to perform a visit between a pair of ports The vessel is selected considering different ranks with different weights value. For defining the best weight combination, a genetic algorithm was proposed. A similar approach was proposed by [9] for solving a MIRP related to the oil industry, where tankers transfer oil from offshore platforms to inland terminals. In that work, a multi-start algorithm is proposed and either vessels and ports can be selected at random from a set of rankings. Additionally, two route improvements and a local search procedure were proposed. The last one uses a mathematical solver to solve a residual model. The work of [10] considers a MIRP in which total fuel consumption should also be minimized. Due to the complexity of the non-linear mathematical program proposed, [10] solves the problem using a particle swarm optimization for composite particle, validating results with basic particle swarm optimization and genetic algorithms.

The MIRP studied in this work was proposed by [1] and the most effective results were found by [11], which developed a two-stage decomposition approach similar to a benders decomposition. Other works such as [5], [12]–[14] also proposed solution methods for this MIRP variant. The common point of these works is that a mathematical solver is indispensable for the solution approach, either if it is an exact method or a hybrid approach. Although our solution approach uses a mathematical solver, it can be dispensable as it works as a final improvement proceadure. Also, even though [1] made available instances with up to 360 time periods, the solution approaches of [5], [12]–[14] tested instances considering only planning horizons with 45 and 60 time periods.

The present work was carried out with the support of CNPq, Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil and FAPERGS, project PqG 17/2551-0001201-1

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

Multi-start metaheuristics are useful in solving hard combinatorial optimization problems where it is difficult to define an efficient and effective neighborhood structure for using local search methods to improve a unique generated solution [15]. The basic algorithm consists of generating a set of solutions from random starting points to obtain diversification. Optionally, improvement procedures can be applied to each solution generated. For a comprehensive review of multi-start methods, we recommend the work of [15].

The large neighborhood search (LNS) method improves one or more solutions obtained by the multi-start algorithm. This method explores a sample of a solution neighborhood, as exploring the whole neighborhood is not practical. The LNS iteratively uses a destroy and repair methods to gradually improve the solution. The destroy method removes part of the solution (usually turning it infeasible), while the repair method is responsible to rebuild the solution in a different way. For a survey on LNS, we refer to the work of [16]. Some works on MIRPs already proposed LNS for solving the problem [4], [13], [17], but using a mathematical solver for it. This method is known as fix-and-optimize, and fixing a subset of the model variables is equivalent to the destroy method, and the repair method corresponds to solving the partially fixed problem.

This work aims to evaluate our metaheuristic as an alternative solution approach for a MIRP variant in which only solver-based methods were proposed. We also tested long planning horizon instances in which no attempts to solve them were found in the literature. The remainder of the work is organized as follows: Section II presents the problem definition and shortly describes the mathematical model. The solution approach and its components are presented in sections III, IV, and V, respectively. Computational results are presented in Section VI, followed by the conclusions in Section VII.

II. Problem Description

In this section, we describe the MIRP proposed by [1]. Due to the size restrictions, we do not present the problem formulation, which is found in the referred paper.

The problem consists of transporting a single product between a set of ports $\mathcal{J}$ using a fleet of heterogeneous vessels $\mathcal{V}$, mananging the ports inventory along a discrete planning horizon $\mathcal{T}$. Set $\mathcal{J}$ is divided into two subsets, each one corresponding to the type of port: loading (or production) ports $\mathcal{J}^{\mathrm{P}}$ and discharging (or consumption) ports $\mathcal{J}^{\mathrm{C}}$. Each port $i \in \mathcal{J}$ has an inventory storage limit $S_i^{\max}$ and a minimum level of product inventory $S_i^{\min}$. Also, a berth limit B_j imposes the maximum number of vessels that can operate (load or discharge) at the port in the same time period. The production (or consumption) rate in port i is denoted by D_{it} and can vary at each time period $t \in \mathcal{T}$. The inventory of port i at time t is denoted by variable s_{it}, while its inventory at the beginning of the planning horizon is given by parameter s_{j0}. Discharging ports $i \in \mathcal{J}^{\mathrm{C}}$ pay a revenue R_{it} per product unit unloaded by the vessels in each time period. The vessel set $\mathcal{V}$ is divided into vessel classes representing different characteristics, such as load capacity Q_v, traveling time T_{vij} between ports i and j and traveling costs C_{vij}. Each vessel $v \in \mathcal{V}$ has its initial load s_0^v, initial port j_0^v, and first time available t_0^v at the planning horizon given as input parameters.

Besides the routing and inventory constraints common to other MIRPs, there are specific characteristics of the problem. This MIRP variant is denoted as deep-sea, where traveling times are much higher than operating times. Thus, the problem imposes that vessels departing from a loading port to a discharging port must be at full capacity, while they must be empty when performing the inverse path. When a vessel v starts to operate at the port i in time t, the amount f_{it}^v operated must lie between a minimum $F_i^{\min}$ and a maximum $F_i^{\max}$ operation capacity. The port inventory constraints are allowed to be violated considering certain limits. In this case, variable $\alpha_{it},\ i \in \mathcal{J}, t \in \mathcal{T}$ is used to define a quantity of product that can be bought from or sold to a simplified spot market to keep the port inventory at its limit. Besides a penalization value incurred in the objective function, α_{it} is limited by parameters $\alpha_{it}^{\max}$ and $\alpha_i^{\max}$. The first one imposes a limit for each port and time period, while the second defines the limit for each port in the whole planning horizon. The objective is to maximize the revenue obtained in discharging operations minus the vessels traveling and operating costs, and minus the penalization costs of using spot markets.

III. The Multi-start Algorithm

The multi-start algorithm builds a set S of n solutions using a constructive greedy heuristic with randomness. For explaining the heuristic, the following notation is adopted:

- $\mathcal{R}_i$ - Set of ports corresponding to the geographical region of port i.
- t_i^{viol} - is the earlier time period t in which an inventory violation occurs at port i, i.e. $t_i^{\mathrm{viol}} = \min\{t \in \mathcal{T} : s_{it} < S_i^{\min},\ i \in \mathcal{J}^{\mathrm{C}}\}$, and $t_i^{\mathrm{viol}} = \min\{t \in \mathcal{T} : s_{it} > S_i^{\max}, i \in \mathcal{J}^{\mathrm{P}}\}$;
- *Vessel voyage* - Sequence of vessel actions starting by departing from port $i \in \mathcal{J}$, and ending at port $j \in \mathcal{J}$, such that i and j are different type ports. Optionally, a port $i' \in \mathcal{R}_i$ can be visited after port i, and a port $j' \in \mathcal{R}_j$ can be visited after port j. A vessel voyage ends after finishing its operation at port j (or j'), such that no more operations can be performed, i.e., if $j \in \mathcal{J}^{\mathrm{P}}$ the vessel is at full capacity, and if $j \in \mathcal{J}^{\mathrm{C}}$ the vessel is empty.

The constructive algorithm iteratively builds a concatenation of voyages for each vessel until there is no inventory violation at the ports, i.e. $t_i^{\mathrm{viol}} > |\mathcal{T}|, i \in \mathcal{J}$. The multi-start algorithm is described in Fig. 1 and the details are explained below.

A. Initializing Solution s - line 3

This function puts each vessel $v \in \mathcal{V}$ in its initial port j_v^0 and its first time available t_v^0 given as parameter input. We assume that vessels always operate at the initial port, as they start at full capacity ($s_0^v = Q_v$) if $j_v^0 \in \mathcal{J}^{\mathrm{C}}$ and start empty ($s_0^v = 0$) if $j_v^0 \in \mathcal{J}^{\mathrm{P}}$. The algorithm decides if vessel v will also operate at a port $j' \in \mathcal{R}_{j_v^0}$ after operated at j_v^0. In this case it is necessary to split the vessel capacity between ports j_v^0 and

1: **Input:** n, p^{CP}, p^{2nd}
2: **while** $|S| < n$ **do**
3: *Init*(s, p^{2nd}) ▷ Randomness
4: **while** there is a $j \in \mathcal{J}$ with t_j^{viol} **do**
5: j = *urgentPort*()
6: i = *counterpartPort*(j, p^{CP}) ▷ Randomness
7: v = *selectBestVessel*(i, j, p^{2nd}) ▷ Randomness
8: *UpdadeSolution*(s, v, i, j)
9: *CompleteSolution*(s)
10: $S = S \cup s$

Fig. 1. Multi-Start Algorithm

j'. This decision procedure is detailed in Section III-E1. For now, assume that after the end of the initialization function, all vessels $v \in \mathcal{V}$ operated at its initial port j_v^0 (and possibly at a port $j' \in \mathcal{R}_{j_v^0}$), completing its first *vessel voyage*.

Different vessels may start the route at the same port, but in different time periods. Thus, the orded in which each vessel is initialized in the solution may affects either the port inventory levels and the other vessels operations at the same port. For obtaining more variability in the solutions, this order randomly defined by the algorithm for each solution $s \in S$

After a vessel operation is implemented at port $j \in \mathcal{J}$, the inventory violation time t_j^{viol} is updated, incresing its value.

B. Selecting the Urgent Port - line 5

The urgent port j is the port with the earliest inventory violation time, i.e., $j = \arg\min_{i \in \mathcal{J}}\{t_i^{\mathrm{viol}}\}$. It can be a loading or a discharging port, and in case of a tie, the discharging port is selected. If the tie occurs between ports of the same type, the one with the highest inventory capacity is selected.

C. Selecting the Counterpart Port - line 6

A counterpart port i is selected for supplying the demand of urgent port j. The candidate ports are always of a different type of port j. They are ranked in increasing order considering t_i^{viol} and the distance to the urgent port j. The port with the minor sum of its position in the two ranks is selected as the counterpart port, and in case of a tie, the port with the earliest inventory violation time is selected. This strategy is the same as proposed by [8]. Additionally, to avoid producing myopic solutions, we defined a probability p^{CP} of selecting a counterpart port at random between all candidate ports.

D. Selecting the Best Vessel - line 7

This function evaluates each vessel $v \in \mathcal{V}$ to perform the voyage between counterpart port i and urgent port j. For the explanation, we assume that i is a loading port, and j is a discharging port. If v is currently at a loading port $l \neq i$ in the partial solution, it does not need to visit port i, as v will be already loaded and can travel directly to port j. Otherwise, v needs to travel from its current position to the counterpart port i. In such case, it is also evaluated if a port $i' \in \mathcal{R}_i$ should be visited after port i and before departing to urgent port j (see Sec. III-E1 for details). After visiting and operating at port i (and possibly at port i'), vessel v travels at full capacity to urgent port j. It is also evaluated if a port j', such that $\mathcal{R}_{j'} = \mathcal{R}_j$ can be visited after j. Note that when evaluating each vessel, the optional ports i' and j' can be different.

Sometimes a vessel can arrive at a port only after t_i^{viol}. In such cases, spot market variables α_{it} are used to keep the inventory at the bound limits ($s_{it} \leq S_i^{\max}$ for loading ports, $s_{it} \geq S_i^{\min}$ for discharging ports) until one time period before the vessel can arrive and operate at the port. As the use of spot market variables is limited, if they are not sufficient to keep the inventory of ports at the bound limits until the vessel arrives, auxiliary variables $\beta_{it}, i \in \mathcal{J}, t \in \mathcal{T}$ are used. They are equivalent to α_{it}, but not bounded and highly penalized in the objective function. The costs of using α_{it} and β_{it} variables are considered penalization costs of the voyage.

After vessels were evaluated, the best one is selected according to one of the following criteria, chose at random:

1) *Lowest cost*: Considers the traveling, operations and possible penalization costs minus the profit obtained by the vessel in the voyage;
2) *Highest cost*: The opposite of the previous criteria;
3) *Smallest number of waiting times*: Considers the total number of time periods in which a vessel needs to wait in the ports along the voyage;
4) *Closest arrival time*: Calculates the difference between the arrival time and the inventory violation time at the urgent port. The lower the value, the better;
5) *Earliest arrival time*: Selects the vessel which can first arrive at urgent port j to operate;
6) *Smallest number of operations*: Selects the vessel which needs the lowest number of operations at the ports to finish the voyage;
7) *Lowest cost/capacity ratio*: considers the vessel voyage cost divided by its capacity Q_v;
8) *Lowest penalization*: in cases when no vessel can arrive at urgent port j before inventory violation time, selects the vessel with the lowest penalization cost associated.

Criteria {1, 3, 4, 6, 7, 8} aims to improve solutions quality, while criteria {2,5} aims to provide variability on the generated solutions.

While the constructive algorithm iterates, it is possible that a vessel is able to operate in a port at a time period earlier than an already implemented operation. In such cases, for defining a feasible operation it is necessary to check the inventory of the port at the last time period in which operation took place. For example, consider that an operation was implemented in port $j \in \mathcal{J}^{\mathrm{C}}$ at time $t' = 9$, such that $s_{jt'} = 250$. The port inventory capacity is $S_j^{\max} = 300$. Now suppose that vessel v arrives in port j at time $t = 4$ and $s_{jt} = 100$. Considering only the inventory at time t, vessel v can discharge 200 units without violating the inventory capacity of port j. However, at time t' the maximum amount that can be discharged is only 50 units. Therefore, if some discharging operation occurs before time t', the value to be discharged must be at most the inventory space available at time t'. The following variable parameter defines the maximum value that can be operated

at port i in time t without causing an inventory violation in future time periods:

$$f_{it}^{\max} = \begin{cases} \min_{t' \in \mathcal{T}: t' \geq t} \{s_{it'}\} & \text{if } i \in \mathcal{J}^{\mathrm{P}} \\ \min_{t' \in \mathcal{T}: t' \geq t} \{S_i^{\max} - s_{it'}\} & \text{if } i \in \mathcal{J}^{\mathrm{C}} \end{cases} \quad (1)$$

E. Specific proceadures

Two problem specific functions are used internally in $Init$ and $selectBestVessel$ functions. For explaining them, we consider a discharging port $i \in \mathcal{J}^{\mathrm{C}}$. The procedure for a loading port is analogous.

1) Deciding between operating at one or two ports: When evaluating a vessel v to discharge at port i, the algorithm must decide between discharging all its cargo in port i or spliting the vessel load with a second port $i' \in \mathcal{R}_i$.

Such decision is used to handle two issues: i) no other vessel except v can arrive at port i' before $t_{i'}^{\mathrm{viol}}$. Therefore, it is preferable that v visits port i' to avoid a major probability of building an infeasible solution; ii) vessel v needs to wait too many time periods to discharge in port i due to inventory constraints. Although there are no costs for a vessel to wait at a port, such a situation should be avoided as it can compromise the next voyages. Thus, it is possible that dividing a vessel load between two ports may result in lesser waiting time periods, and the voyage can finish earlier.

The algorithm ranks all candidate ports $i' \in \mathcal{R}_i$ (including port i) considering three criteria: inventory violation, end time of the operation, and voyage cost. Firstly it is evaluated if only vessel v can arrive at a candidate port i' before $t_{i'}^{\mathrm{viol}}$. If this is the case, then v must visit i'. If there is no candidate port in such a situation (or more than one), the algorithm selects the port i' in which vessel v ends its voyage earlier in the planning horizon. If there is a tie between two or more candidate ports considering this criterion, the port i' that implies in the lowest voyage cost is selected.

If the best-ranked port is different from i, the vessel capacity is divided between the ports such that the maximum amount is operated in port i at time t, and the remaining is operated at port i', respecting the minimum amount $F_{i'}^{\min}$. Otherwise, if $i' = i$, the total vessel capacity is operated at port i. Besides the ranking, we defined a probability $p^{2\mathrm{nd}}$ of randomly selecting a port $i' \in \mathcal{R}_i$ to provide more variability in the solutions.

2) Defining operation values and times: Supose that vessel v arrives in port i at time t, and needs discharge the amount f. This operation is subject to several problem constraints, such as inventory, operation, and berth limits. Firstly, if berth occupation in i at time t is equal to berth limit B_i, then vessel v must wait at the port until a time $t' \geq t$ in which a berth is available. There are three options for a vessel to discharge in a port: the first and most preferable is to discharge f in port i at time t without waiting at the port, i.e., $f_{it}^v = f$. This is possible only if i has capacity to recieve the amount f, and there is sufficient inventory space at the time t, i.e., $f \leq F_i^{\max}$ and $f \leq S_i^{\max} - s_{it}$. In the second option, the vessel needs to wait one or more times while $f > S_i^{\max} - s_{it}$ before discharging f. The third option occurs when $f > F_i^{\max}$. In such a case, it is mandatory to a vessel to operate multiple time periods, splitting the amount f.

F. Updating Solution - line 8

This procedure implements in solution s the voyage of the selected vessel v between ports i and j, updating their inventory according to the amount operated and also the inventory violation time t_i^{viol} and t_j^{viol}.

G. Complete the Solution - line 9

After the end of the while loop, there is no port with an inventory violation, and a feasible solution is obtained (unless some auxiliary variable β_{it} is positive). The *CompleteSolution* function aims to obtain a better solution quality, verifying for each discharging port if there is a vessel that can discharge in the port before the end of the planning horizon. If such a case exists and the operation provides a revenue greater than the voyage and operation costs, the voyage is performed.

IV. Large Neighborhood Search

The large neighborhood search is responsible for improving solutions given by the multi-start algorithm and possibly remove the infeasibility when some variable β_{it} is positive. The LNS partially destroys a solution s by removing $u\%$ of the vessels at random. The route of each removed vessel is then rebuilt in a different way, generating a candidate solution s'. If solution s' is accepted, it becomes the current solution s. The procedure is repeated until it reaches a maximum number of iterations $maxIt^{\mathrm{LNS}}$, or the candidate solution is not accepted for $\max^{\mathrm{NA}}$ consecutive iterations. The best solution found is returned at the end of the LNS.

The route of vessels is rebuilt with the same constructive heuristic of the multi-start algorithm, given in lines 4-9 of Fig. 1. The unique difference is that the constructive heuristic in the LNS considers only the vessels removed from the solution. The LNS is performed on a set $S^{\mathrm{LNS}} \subseteq S$ which is composed of m solutions, such that $\lceil \frac{m}{2} \rceil$ are the best solutions obtained by the multi-start algorithm (considering different objective function values) and $\lfloor \frac{m}{2} \rfloor$ are solutions chosen at random from set S. This configuration is chosen to provide both exploitation and exploration of the solutions.

A. Acceptance Criteria

A candidate solution s' is accepted if its objective value is better than the value of the current solution s. The algorithm also accepts worse solutions than the current one under some conditions. To do this, we use the temperature parameter used in simulated annealing in a similar way as [18], in which the probability of a worse solution being selected decreases as the temperature downs. More precisely, the probability is given by $p(S) = e^{-\frac{f(S') - f(S)}{T}}$ where T is the temperature. The initial temperature is defined by the value of the initial solution (provided by the multi-start algorithm) multiplied by a factor $0 < f^{\mathrm{start}} < 1$, and decreased at each iteration by a cool rate $0 < c < 1$, until reaching the final temperature

defined based on the initial solution objective and a constant $0 < f^{\text{end}} < 1$, such that $f^{\text{end}} < f^{\text{start}}$.

B. Backtracking

For avoiding the LNS getting stuck in local minima and for intensifying the search in promising solution spaces, if a new best solution is not found after max^{reset} iterations, the current solution is set to the best solution found so far. The current temperature is set to half of the starting temperature to provide more exploration of the search space from the best solution.

V. Building Reduced MIP

After LNS was performed on the m solutions of set S^{LNS}, such solutions are converted to a reduced mixed integer program. We used the fixed-charge network flow model presented in [13] for the same MIRP, which provide a thighter formulation than the time-space network model of [1]. In the reduced MIP only the variables (and respective constraints) that assume a positive value in the corresponding metaheuristic solution are inserted in the model. The exceptions are port inventory variables s_{it}, spot market variables α_{it}, and auxiliary variables β_{it}, which are added for all $i \in \mathcal{J}, t \in \mathcal{T}$. The reduced model is then solved by a mathematical solver for t^{RMIP} seconds for trying to obtain a better solution than the best one provided by the LNS. As the size of the reduced model (number of columns and rows) is smaller than the original model, it can be solved to optimality in a short processing time depending on the value of parameter m.

Although solving RMIP is usually quick, adding only the variables and constraints corresponding to the LNS solutions can be restrictive, so that the solution found by the solver is the same found by the LNS. Thus, we defined "adjacent" variables of the solution that are also included in the model. They cover the operation and waiting variables of the FCNF model. Fig. 2 illustrates the adjacent variables according to a vessel v route solution visiting a port i.

In the example of Fig. 2, vessel v arrives in port i at time $t = 1$, wait for three time periods, and then operates at time $t = 3$ to after depart to another port. The adjacent variables corresponds to the waiting (w) and operation (o) variables between the arrival time $t = 1$ of vessel v in port i until the departing time $t = 3$. They enable the solver to decide if a vessel can improve its route by changing possible time periods in which the operation(s) take(s) place and the value which is operated. Note that in the example of Fig. 2, vessel v still must operate at the time period $t = 3$, as the traveling arcs variables x_{ijvt} representing the depart from port i in time periods with $t < 3$ are not included in the model.

VI. Computational Experiments

In this section, we evaluate the performance of our algorithm and its three components: the multi-start, the large neighborhood search and the reduced mixed integer program model. The objective is to verify the capacity of our algorithm to produce good solutions for the MIRP presented, either for instances with short and long planning horizons.

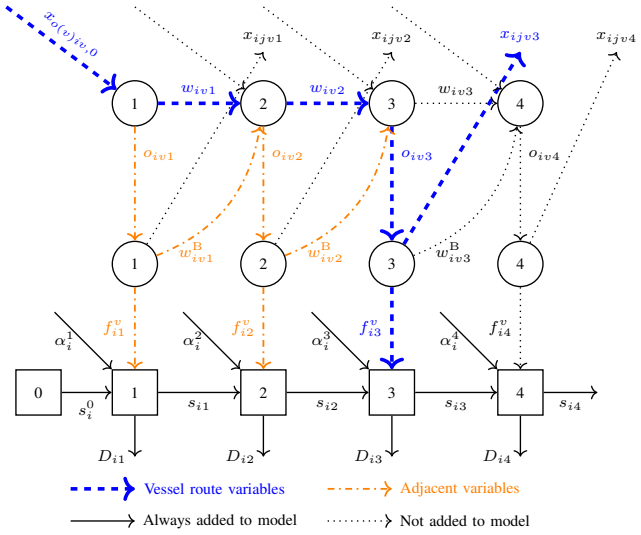

Fig. 2. Example of a MIRP FCNF model solution and the adjacent variables.

The experiments were carried out in an Intel Core i7 930 computer running at 2.8 GHz with 12 GB RAM. The algorithm was implemented in C++, and CPLEX 12.7.1 solver was used to solve the reduced MIP model using a single core. The number of repetitions per run of the algorithm is set to 10, and the average values considering objective function and processing time were considered.

A. Benchmark Instances

The tests were performed using the "Group 1" instances available at MIRPLIB[1]. The benchmark set consists of 14 instances, such that the planning horizon of each instance can be changed and set up to $|\mathcal{T}| = 360$ days.

B. Parameters Calibration

The multi-start algorithm and the LNS parameters must be defined. They were calibrated using the *irace* package[2] for obtaining the best combination of its values. Some parameters have not been calibrated and a fixed value is set for them. We also study the value variation of the fixed parameters individually. For defining the training instances for *irace*, we modified six distinct instances from the test set, defining a planning horizon with size $|\mathcal{T}| = 30$ and changing the x and y coordinates of the ports at random. More precisely, we selected instances LR1_1_DR1_3_VC1_V7a, LR1_1_DR1_4_VC3_V11a, LR1_2_DR1_3_VC3_V8a, LR2_22_DR2_22_VC3_V10a, and LR2_22_DR3_333_VC4_V14a as training instances. In the calibration test, we do not consider the results from the RMIP, i.e., the output is the best solution found after solving the LNS for m solutions. The evaluated parameter values, the best one defined by the calibration and the fixed-parameter values are presented in Table I.

[1]https://mirplib.scl.gatech.edu/
[2]http://iridia.ulb.ac.be/irace/

TABLE I
RESULTS FROM THE PARAMETER CALIBRATION

Parameter	Evaluated values	Choosen Value
p^{2nd}	[0%,10%,20%,30%,40%,50%]	0%
p^{CP}	[0%,10%,20%,30%,40%,50%]	50%
f^{start}	{0.9,0.9999}	0.95
f^{end}	{0.0001,0.1}	0.09
c	{0.0001,0.1}	0.01
max^{reset}	[500,1000,2500,5000]	500
n	Fixed	500
m	Fixed	4
$maxIt^{LNS}$	Fixed	25000
$\max^{\text{NA}}$	Fixed	$2(max^{\text{reset}})$
u	Fixed	20%

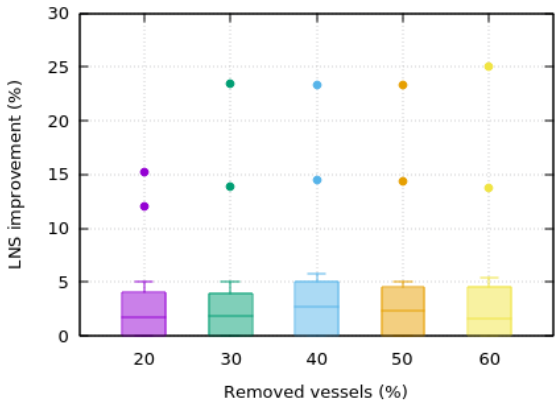

Fig. 3. Improvement of the multi-start solutions by LNS varying the values of parameter u

The parameter t^{RMIP} is defined according to two characteristics of the instance: the number of loading regions (one or two), and the size of the planning horizon. For the smallest instances corresponding to those with one loading region and $|\mathcal{T}| = 45$, t^{RMIP} is set to 60 seconds. For the instances with two loading regions and the same planning horizon, the value is doubled. For the remaining sizes of $\mathcal{T}$, the time is increased proportionally to the number of time periods.

C. Individual Parameters Analysis

We study how varying separately parameters u, n and m affects the solution quality and processing time of the proposed method. For these tests, we considered the 14 instances with $|\mathcal{T}| = 45$.

1) Evaluating Parameter u: We first analyze how varying the percentage of vessels removed from the current solution in each LNS iteration affects the solution quality. We performed tests with values $u = \{20, 30, 40, 50, 60\}$. The remaining parameters are fixed to the values defined in the last column of Table I. Fig. 3 presents the box plot corresponding to the improvement percentage of the multi-start solution by the LNS varying parameter u. Thus, it considers only the insntaces in which multi-start algorithm obtained a feasible solution.

As observed in Fig. 3, the improvement percentage slight varies as parameter u increases considering the median value. The outliers corresponds to two instances in which there is a more siginificant improvement of the solution. Using a binomial statical test with confidence level of 0.95 we cannot affirm that incrasing the value of parameter u provides better LNS solutions. The processing time increases proportionally to the percentege u increase due to the major number of vessesls that are evaluated in each iteration. Considering the runs in which the multi-start solution is infeasible, the LNS was able to remove such infeasibility in {26.3, 28.9, 36.8, 42.1, 47.3} percent of the cases. Although the feasibility increases with the increase of u (the larger parameter u, the larger the search space) it is not possible to confirm such hipotesys using the same previous mentioned statical test. We decided to use $u = 40$ for the remaining tests, as it provided the best median value considering the improvement of multi-start solutions.

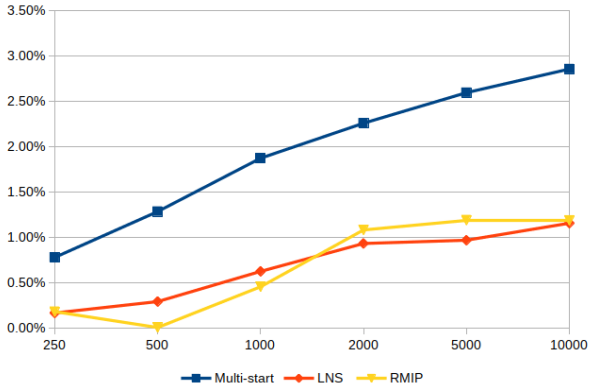

Fig. 4. Average improvement of the solution quality by varying parameter n, considering the multi-start, the LNS, and the RMIP.

2) Evaluating Parameter n: The next parameter evaluated was the number of solutions generated by the multi-start algorithm. In this section and the subsequent ones, the results of the RMIP are also considered. We tested the values $n = \{100, 250, 500, 1000, 2000, 5000, 10000\}$ and the results are presented in Fig. 4, showing for each parameter value the average increase of the solution quality in relation to the base case $n = 100$, considering the multi-start algorithm, the large neighborhood search and the reduced MIP.

The results of Fig. 4 confirm the tendency that the greater the number of generated solutions, the greater the improvement of the solution quality in the multi-start algorithm. Considering the LNS and the RMIP, we observed in several runs of the algorithm that they provided worse solutions with $n = 10000$ than with $n = 100$, which contributed to a smaller increase in the solution quality of such components. Although not shown in Fig. 4, the results demonstrated an approximate increase of 10% in the number of feasible solutions between the minimum and maximum value of n considering the multi-start and the LNS algorithm. For the RMIP, this increase was below 4%, demonstrating that the final algorithm solution is less affected varying parameter n.

We have chosen the value parameter $n = 1000$ for the remainder experiments, as it presented a good balance between solution quality and feasible solutions.

3) Evaluating Parameter m: The last evaluated parameter corresponds to the number of multi-start solutions improved by the LNS and then used to build the RMIP. The objective is to verify whether it is valid to increase the m parameter to get a better solution, as it drastically increases the processing time of the LNS. We evaluate the parameter with values $m = \{1, 4, 8, 16, 32\}$. The results are shown in Fig. 5, presenting for each parameter the average deviation GAP from the best-known solutions considering the feasible solutions found. The lines in the figure represent the average number of instances in which no feasible solution was found.

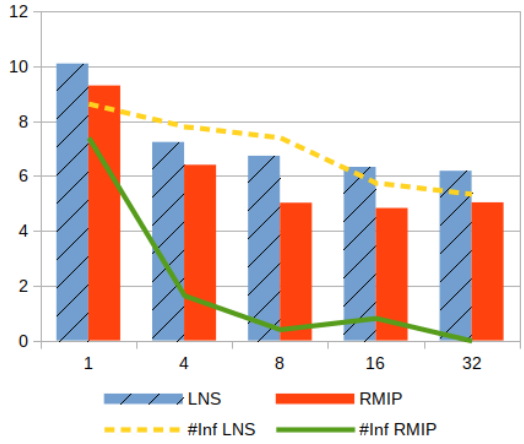

Fig. 5. Average GAP deviation and average number of instances with no feasible solution found for different values of parameter m, considering LNS and the RMIP results.

Fig. 5 demonstrates a considerably average imporvement in the RMIP solutions varying m until value 8, while the average GAP of the LNS solutions decreases less than 0.5% as m is doubled. The limited time for running RMIP may explain the slight difference between the average GAP varying parameter m from 8 to 32.

The average number of infeasible instances tends to decrease as m increases, reaching no infeasible solutions with RMIP when $m = 32$. As the processing time increases proportionally to the number m, this parameter should be small, otherwise larger instances will take a long time to be solved. Thus, we define $m = 8$ for the remaining tests.

D. Comparative Results

We compare our algorithm to the best-known values presented by [5], [11], [13], which solutions for $|\mathcal{T}| = \{45, 60\}$ are reported. We performed the tests using the parameter values defined in the previous sections. Table II summarizes the results, presenting for each $|\mathcal{T}|$, the average GAP deviation from the best-known solution, the average number of instances with no feasible solution found, the average processing time, and the average reduction in the processing time. For calculating the processing time reduction, the time was normalized using the *PassMark Software*[3]. The last line of the table presents the average results.

[3]https://www.cpubenchmark.net/

TABLE II
COMPARATIVE WITH THE BEST KNOWN VALUES CONSIDERING SOLUTION QUALITY AND PROCESSING TIME

$\|\mathcal{T}\|$	*GAP(%)*	*Inf.*	*Time(s)*	*Time reduction (%)*
45	5.1	1.1/14	711.2	79.4%
60	8.0	3.3/14	936.4	91.7%
Avg	6.5		660.7	85.5%

We can observe from Table II that our approach is not much robust considering solution quality, as the average GAP from the best-known solution is higher, varying from 0.0% to 25.1% depending on the instance. Although not shown in Table II, no feasible solution was found in the 10 repetitions of just one instance with $|\mathcal{T}| = 60$. Considering processing time, our algorithm provided the final solution in much shorter processing time than the other approaches. We then test if the results can be improved if the processing time is increased. Thus, we set the parameters u, n, m with the highest values tested in Section VI-C and defined $t^{RMIP} = 3600$ seconds. The results presented an average GAP of 3.9% (varying from -0.7% to 13.5%), while they're still one instance in which no feasible solution was found in the 10 runs of the algorithm. Considering individual runs, the algorithm could find new best-known values for five instances. Although the average processing time was 38.4% faster than the average processing times reported by [5], [11], [13], it is much higher for the small instances, due to the LNS and the RMIP, in which the last usually reaches the time limit of 3600 seconds. Such results demonstrated that the algorithm's performance tends to increase if the parameters u, n, m values are also increased. However, as the processing time required is higher for small instances, it may be more interesting to use higher parameter values for testing larger instances, and still obtaining a lower processing time than other approaches for solving this MIRP.

E. Long Planning Horizon Tests

We ran our algorithm in the instances with long planning horizon $|\mathcal{T}| = \{120, 180, 360\}$. For these tests, we perform five repetitions of the algorithm per instance. As this is the first attempt to solve these instances, we evaluate the quality of the solution considering the average GAP to the lower bound obtained by solving the linear relaxation of the model proposed by [13].

Table III presents the average results for each size of the planning horizon, showing the number of feasible solutions found and GAP (only for the feasible instances) obtained at the end of the LNS and the end of RMIP. The last column presents the average total time of the algorithm.

As expected, the difficulty of obtaining feasible solutions, and the average GAP increases as the planning horizon's size increases. The multi-start algorithm and the LNS were responsible for 23% and 63% of the processing time on average, respectively. Observe that the processing times increased more than 430% between the tests with $|\mathcal{T}| = 180$ and $|\mathcal{T}| = 360$. This can be explained due to the bottleneck in updating the port's inventory. Every time a vessel operates at a port, the port

TABLE III
LNS AND RMIP RESULTS FOR THE LONG PLANNING HORIZON INSTANCES.

	LNS		*RMIP*		*Total*
	#Feasible	*GAP(%)*	*#Feasible*	*GAP(%)*	*time (s)*
$\|\mathcal{T}\| = 120$	7.0/14	16.4	6.0/14	21,2	1,353.4
$\|\mathcal{T}\| = 180$	5.2/14	19.7	5.2/14	27.0	2,166.6
$\|\mathcal{T}\| = 360$	2.2/14	13.5	2.4/14	13.2	9,512.4

inventory needs to be updated in each time period from the time of the operation until the end of the planning horizon. As the number of operations performed increases with the size of the planning horizon, further inventory updates are required. The RMIP presented worse results considering feasibility and solution quality in general, as the number of variables grows proportionally to the planning horizon size. Thus, the time limit defined for the CPLEX is too restrictive and may be increased to obtain better results with RMIP.

It is important to say that we are not sure if there are feasible solutions for all instances in all long planning horizons tested, in particular, the instances in which spot market variables are necessary to build feasible solutions. This occurs because the value $\alpha_i^{\max}, i \in \mathcal{J}$ representing the total amount of product that can be bought from or sold to a spot market in the planning horizon is the same independently of the size of such planning horizon. Thus, the higher the planning horizon, the more difficult to obtain a feasible solution.

VII. CONCLUSION

In this work, we proposed a solution approach for a maritime inventory routing problem in which obtaining feasible solutions is a challenge. It is composed of two metaheuristics: a multi-start algorithm and a large neighborhood search, which uses a random constructive algorithm. Also, a reduced mixed-integer program is built considering a solution set obtained by the LNS and solved with the CPLEX solver. Computational experiments were performed to analyze the influence of the solution quality and processing times varying some parameter values. The solution approach was compared with the ones that obtained the best-known solutions, and although not outperforming them considering solution quality, the processing time is much shorter. We also performed the tests with long planning horizon instances, in which no attempt to solve them was presented in the literature, and our algorithm was able to obtain feasible solutions for some of these instances. Differently from other works that proposed solution approaches for the MIRP variation studied here, our solution method is independent of a mathematical solver, but its performance can be increased when the reduced mixed-integer program is solved. A next step to improve our multi-start algorithm is to use memory mechanisms that can guide the search based on previously built solutions.

REFERENCES

[1] D. J. Papageorgiou, G. L. Nemhauser, J. Sokol, M. S. Cheon, and A. B. Keha, "MIRPLib - A library of maritime inventory routing problem instances: Survey, core model, and benchmark results," *European Journal of Operational Research*, vol. 235, no. 2, pp. 350–366, 2014.

[2] A. Agra, M. Christiansen, A. Delgado, and L. Simonetti, "Hybrid heuristics for a short sea inventory routing problem," *European Journal of Operational Research*, vol. 236, no. 3, pp. 924–935, 2014.

[3] J. G. Rakke, M. Stålhane, C. R. Moe, M. Christiansen, H. Andersson, K. Fagerholt, and I. Norstad, "A rolling horizon heuristic for creating a liquefied natural gas annual delivery program," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 5, pp. 896–911, 2011.

[4] K. T. Uggen, M. Fodstad, and V. S. Nørstebø, "Using and extending fix-and-relax to solve maritime inventory routing problems," *Top*, vol. 21, no. 2, pp. 355–377, 2013.

[5] M. W. Friske and L. S. Buriol, "A relax-and-fix algorithm for a maritime inventory routing problem," in *International Conference on Computational Logistics*. Springer, 2017, pp. 270–284.

[6] Y. Jiang and I. E. Grossmann, "Alternative mixed-integer linear programming models of a maritime inventory routing problem," *Computers & Chemical Engineering*, vol. 77, pp. 147–161, 2015.

[7] F. G. Engineer, K. C. Furman, G. L. Nemhauser, M. W. P. Savelsbergh, and J.-H. Song, "A Branch-Price-and-Cut Algorithm for Single-Product Maritime Inventory Routing," *Operations Research*, vol. 60, no. 1, pp. 106–122, feb 2012.

[8] M. Christiansen, K. Fagerholt, T. Flatberg, Ø. Haugen, O. Kloster, and E. H. Lund, "Maritime inventory routing with multiple products: A case study from the cement industry," *European Journal of Operational Research*, vol. 208, no. 1, pp. 86–94, 2011.

[9] A. d. L. Stanzani, V. Pureza, R. Morabito, B. J. V. d. Silva, D. Yamashita, and P. C. Ribas, "Optimizing multiship routing and scheduling with constraints on inventory levels in a brazilian oil company," *International Transactions in Operational Research*, vol. 25, no. 4, pp. 1163–1198, 2018.

[10] A. De, S. K. Kumar, A. Gunasekaran, and M. K. Tiwari, "Sustainable maritime inventory routing problem with time window constraints," *Engineering Applications of Artificial Intelligence*, vol. 61, pp. 77–95, 2017.

[11] D. J. Papageorgiou, A. B. Keha, G. L. Nemhauser, and J. Sokol, "Two-stage decomposition algorithms for single product maritime inventory routing," *INFORMS Journal on Computing*, vol. 26, no. 4, pp. 825–847, nov 2014.

[12] C. E. Andrade, S. Ahmed, G. L. Nemhauser, and Y. Shao, "A hybrid primal heuristic for finding feasible solutions to mixed integer programs," *European Journal of Operational Research*, vol. 263, no. 1, pp. 62 – 71, 2017.

[13] M. W. Friske and L. S. Buriol, "Applying a relax-and-fix approach to a fixed charge network flow model of a maritime inventory routing problem," in *Computational Logistics*, R. Cerulli, A. Raiconi, and S. Voß, Eds. Cham: Springer International Publishing, 2018, pp. 3–16.

[14] L.-M. Munguía, S. Ahmed, D. A. Bader, G. L. Nemhauser, Y. Shao, and D. J. Papageorgiou, "Tailoring parallel alternating criteria search for domain specific mips: Application to maritime inventory routing," *Computers & Operations Research*, vol. 111, pp. 21 – 34, 2019.

[15] R. Martí, J. M. Moreno-Vega, and A. Duarte, "Advanced multi-start methods," in *Handbook of metaheuristics. International Series in Operations Research & Management Science*. Springer, 2010, vol. 146, pp. 265–281.

[16] D. Pisinger and S. Ropke, "Large neighborhood search," in *Handbook of metaheuristics*. Springer, 2010, pp. 399–419.

[17] V. Goel, K. C. Furman, J. H. Song, and A. S. El-Bakry, "Large neighborhood search for LNG inventory routing," *Journal of Heuristics*, vol. 18, no. 6, pp. 821–848, dec 2012.

[18] S. Ropke and D. Pisinger, "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows," *Transportation science*, vol. 40, no. 4, pp. 455–472, 2006.