Computational Management Science (2023) 20:44
https://doi.org/10.1007/s10287-023-00478-8

ORIGINAL PAPER

An ALNS-based matheuristic algorithm for a multi-product many-to-many maritime inventory routing problem

Nooshin Heidari[1] · **Ahmad Hemmati**[1]

Received: 30 May 2022 / Accepted: 25 August 2023 / Published online: 30 September 2023

Abstract

In this paper, we propose an adaptive large neighborhood search-based matheuristic algorithm to solve a multi-product many-to-many maritime inventory routing problem. The problem addresses a short sea inventory routing problem that aims to find the best route and distribution plan for multiple products with a heterogeneous fleet of vessels through a network including several producers and customers. Each port can be visited a given number of times during the planning horizon, and the stock level for each product should lie within the predefined bound limits. The problem was introduced by Hemmati et al. (Eur J Oper Res 252:775–788, 2016). They developed a mixed integer programming formulation and proposed a matheuristic algorithm to solve the problem. Although their proposed algorithm worked well in terms of running time, it suffers from disregarding a part of the solution space. In this study, we propose a new matheuristic algorithm to find better solutions by exploring the entire solution space for the same problem. In our solution methodology, we split the variables into routing and non-routing variables. Then in an iterative process, we determine the values of the routing variables with an adaptive large neighborhood search algorithm, and we pass them as input to a penalized model which is a relaxed and modified version of the mathematical model introduced in Hemmati et al. (2016). The information from solving the penalized model, including the values of the non-routing variables, is then passed to the adaptive large neighborhood search algorithm for the next iteration. Several problem-dependent operators are defined. The operators use the information they get from the penalized model and focus on decreasing the penalty values. Computational results show up to 26% improvement in the quality of the solutions for the group of instances with a large feasible solution space. We get the optimal value for the remaining instances matched with the reported results.

Keywords Maritime inventory routing problem · Many-to-many distribution network · Adaptive large neighborhood search · Matheuristic algorithm

Extended author information available on the last page of the article

1 Introduction

The Inventory Routing Problem (IRP) addresses two main logistic activities, inventory management, and vehicle routing, and is a well-studied problem in the context of vendor-managed inventory. In the IRP, the goal is to manage the flow of products across different producers and consumers to satisfy customer demands and minimize transportation costs. In IRPs, the decisions are to determine the quantity of products required to be handled at each location, the time of visiting each location, and the vehicle routes to serve different customers (Coelho et al. 2013). Various types of inventory routing problems have been studied in the literature for different modes of transportation. Two surveys by Andersson et al. (2010) and Coelho et al. (2013) have explored different aspects and variants of IRPs.

The problem we are addressing is a Short Sea Inventory Routing Problem (SSIRP), which was introduced by Hemmati et al. (2016). The problem is a multi-product, many-to-many Maritime Inventory Routing Problem (MIRP) in a continuous time framework. In this study, we propose a new method to find better solutions for the same problem. We use the same realistic-sized instances and compare the quality of the solutions with the results of the Hybrid Cargo Generating and Routing algorithm (HCGR) proposed by Hemmati et al. (2016).

Our focus here is to review different solution approaches that have been applied to similar maritime inventory routing problems. According to the literature, different mathematical models were developed to formulate MIRPs, and various commercial solvers were applied to solve the problem. (Al-Khayyal and Hwang 2007; Diz et al. 2017; Agra et al. 2017), and (Misra et al. 2020). Moreover, different exact algorithms, including but not limited to branch-and-bound (Christiansen 1999) and (Agra et al. 2013), branch-and-price (Hewitt et al. 2013), and branch-price-and-cut (Engineer et al. 2012) have been studied in the past.

Among the relevant literature, Christiansen (1999) solved a single-product MIRP with time windows by applying a branch-and-bound algorithm and using the Dantzig-Wolf decomposition approach. They decomposed the overall problem into ship routing and inventory management sub-problems. Later, Agra et al. (2013) applied a branch-and-bound approach to solve a single-product MIRP. They proposed two discrete time formulations with valid inequalities. Moreover, Engineer et al. (2012) implemented a branch-price-and-cut algorithm to solve a practical-sized single-product MIRP in an oil company. They used different cuts, including capacity cuts and other special cuts targeting fractional solutions, to find the optimal solution. Furthermore, Hewitt et al. (2013) proposed a branch-and-price algorithm to solve a MIRP and mentioned that their results are near-to-optimal. They reduced the required time for finding good solutions by developing local search schemes.

However, in addition to the aforementioned exact methods, various heuristics, metaheuristics, and matheuristics were developed to solve MIRPs. For example,

Christiansen et al. (2011) studied a multi-product MIRP in the cement industry where they proposed a constructive heuristic embedded in a Genetic Algorithm to solve realistic-sized instances within a reasonable time. Moreover, Song and Furman (2013) presented a Large Neighborhood Search (LNS) through a simple algorithmic framework to solve MIRP. Later, Agra et al. (2016) developed a local search heuristic for the stochastic MIRP. Recently, Friske and Buriol (2020) developed a solution approach including two metaheuristics: a multi-start algorithm and a large neighborhood search. They used a commercial solver to solve the reduced mixed integer problem with the solutions obtained by LNS. In addition, Friske et al. (2022) studied the use of Relax-and-Fix and Fix-and-Optimize metaheuristics over two discrete-time formulations. They evaluated the contribution of different components of the formulations to the performance of the algorithm.

As an extension of the LNS algorithm, Adaptive Large Neighborhood Search (ALNS) is known as an efficient heuristic to deal with vehicle routing problems. (Ropke and Pisinger 2006; Pisinger and Ropke 2007; Ribeiro and Laporte 2012; François et al. 2019; Yu et al. 2020; Chen et al. 2021). The effectiveness of ALNS in maritime routing problems has also been shown by different researchers. (Lianes et al. 2021; Brekkå et al. 2022).

Several published studies are relevant to our problem setting, which are discussed in more detail as follows. Christiansen (1999) formulated a MIRP in which a heterogeneous fleet of ships was planned to transport a single product through a network including producers and customers. Each port could be visited a given number of times during the planning horizon. They assumed a time window limitation for starting service at each port and the possibility of serving multiple ships at the same time. The problem was solved using branch-and-bound and applying the Dantzig-Wolf decomposition approach. In each instance, a maximum of 5 vessels, 16 ports, and 36 days as the planning horizon were considered. Later, Al-Khayyal and Hwang (2007) formulated the same problem of transporting several products on a many-to-many distribution network. The model assumed dedicated compartments for each product in each ship without time window limitations. The small-size instances were solved using a commercial solver. They mentioned the impact of the number of port visits on the solution time and the need for particular algorithms to take advantage of the structure of the model. They considered up to 3 products, 4 vessels, 4 ports, and 10 days for the planning horizon.

A few years later, Siswanto et al. (2011) proposed a set of heuristics with four groups of rules to solve the same problem with undedicated compartments. The vessels had different numbers of compartments that were not dedicated to a specific product. They provided high-quality solutions for the set of instances with at most 2 products, 3 vessels, 4 ports, and 15 days as the planning horizon. Following that, Agra et al. (2014) proposed a tightened model with valid inequalities for the problem described by Al-Khayyal and Hwang (2007). Given the continuous-time

formulation, they presented three heuristics: rolling horizon, feasibility pump, and local branching. They proposed heuristics to solve real instances with a maximum of 4 products, 2 vessels, 7 ports, and 15 days for the planning horizon. According to their results, the quality of solutions improved by combining all three heuristics.

Later, Hemmati et al. (2016) proposed the HCGR algorithm to solve the multi-product SSIRP in a continuous time framework. A heterogeneous fleet of vessels was assumed to distribute multiple products within a many-to-many distribution network. They assumed different compatibility between vessels, ports, and products. There was no limitation on the compartment and no possibility of serving multiple vessels at the same time in their problem setting. They converted the IRP into a routing and scheduling problem to solve realistic-size instances. They considered up to 3 products, 10 vessels, 16 ports, and 60 days for the planning horizon. First, they solved two mathematical models, including transportation and time window models, to find the number of products picked up and delivered from/to each port and the corresponding time scheduling. In this step, the pair of producer and customer and the time of pickup and delivery operations became fixed based on the direct transportation and operational costs. In the next step, based on the best scheduling plan, the routing problem was solved using an ALNS. The algorithm disregarded some parts of the solution space because of making a fixed pair of producers and customers. The proposed algorithm worked well in terms of running time.

Following that, Agra et al. (2017) presented discrete-time and continuous-time formulations and different valid inequalities for the problem introduced by Al-Khayyal and Hwang (2007). They defined different production/consumption rates in the discrete-time formulation for various periods. The rates were fixed during the whole planning horizon in the continuous-time formulation. They compared different proposed formulations in terms of their size, running time, and integrality gap for both discrete and continuous time. Using a commercial solver, they reported the computational results for real test instances. They found the optimal solutions for the cases with a maximum of 4 products, 2 vessels, 7 ports, and a fifteen-day planning horizon.

As mentioned earlier, the SSIRP proposed by Hemmati et al. (2016) was to find the best route and distribution plan for multiple products through a network with several producers and customers. They studied two groups of instances with different compatibility between vessels, ports, and products. The size of the instances was considerably larger compared to the literature. However, their proposed solution method could not find high-quality solutions for the instances where all the vessels were fully compatible with the ports and the products. In this paper, we propose a new solution approach to find better solutions for the problem introduced by Hemmati et al. (2016). To this end, we develop an ALNS-based Matheuristic algorithm (ALNSM) in which an ALNS algorithm determines routing variables. Moreover, the non-routing variables are taken care of by a Mixed Integer Programming (MIP) solver, given the value of routing variables. We compare our results on the test instances in Hemmati et al. (2016) and analyze the quality of the results accordingly.

The rest of the paper is organized as follows: problem characteristics are explained in Sect. 2, the proposed ALNSM algorithm is described in Sect. 3, and computational results are reported in Sect. 4, followed by the conclusion in Sect. 5.

2 Problem description

The problem introduced by Hemmati et al. (2016) addresses the distribution management of different products within a many-to-many distribution structure, including several producers and customers. It is assumed that a heterogeneous fleet of vessels is planned to transport products through the network. The vessels differ in terms of capacities, compatibility with ports and products, and transportation cost and time. There is a limitation on the number of visits for each port during the planning horizon. Furthermore, the stock level of each product should lie within the predefined minimum and maximum levels. The initial stock level and production or consumption rates for each product at each port are given. The quantity of the products to be handled at each port is limited, and the handling operation time at each port for each product is predefined. The problem is to minimize transportation and operational costs, including fixed and variable handling costs.

The following mathematical model is presented by Hemmati et al. (2016). However, for the sake of a better explanation of our solution approach, their mathematical formulation is given as follows: Ports are indexed by i and j, and vessels, products, and ports visit numbers are represented by v, k, and m/n, respectively. The m^{th} visit of port i is denoted by (i, m), and direct travel from port visit (i, m) to port visit (j, n) is represented by (i, m, j, n). N, V, and K are defined as the set of ports, the set of vessels, and the set of products, respectively. Moreover, the set of ports that vessel v can visit and the set of products k that can be transported with vessel v are denoted by N_v and K_v. Also, the set of all possible port visits, the set of possible port visits for vessel v, and the set of all possible direct travels for vessel v are represented by S^A, S^A_v, and S^X_v, respectively.

2.1 Mathematical formulation

Parameters and variables of the model are listed below:

Parameters

J_{ik}: 1/-1 if product k is produced/consumed at port i and 0 if it is not produced nor consumed at port i.

C_v: Capacity of vessel v

$\underline{Q}_{ik}$, $\overline{Q}_{ik}$: Minimum and maximum amount of product k that is allowed to be handled at port i

T^{Q}_{ik}: Required amount of time for handling one unit of product k at port i

T^{O}_{iv}: Transportation time from vessel v's origin to port i

T_{ijv}: Transportation time between port i and port j with vessel v

$\overline{T}$: Duration of the planning horizon

$\overline{M}_i$: Maximum allowed number of visits at port i

S^{O}_{ik}: Opening stock for product k at port i at the beginning of the planning horizon

R_{ik}: Production or consumption rate for product k at port i

$\underline{S}_{ik}, \overline{S}_{ik}$: Lower and upper bound for the stock level of product k at port i

$\underline{S}^{T}_{ik}, \overline{S}^{T}_{ik}$: Lower and upper bound for the stock level of product k at port i at the end of the planning horizon

C^{Q}_{ik}: Operational cost for each unit of product k at port i

C^{T}_{ijv}: Transportation cost from port i to port j with vessel v

C^{TO}_{iv}: Transportation cost from vessel v's origin to port i

C^{O}_{ik}: Fixed operational cost for product k at port i

Variables

x_{imjnv}: 1 if there is a direct movement from port visit (i, m) to port visit (j, n) with vessel v and 0 otherwise.

x^{O}_{imv}: 1 if there is a movement from the vessel v's origin to port visit (i, m) and 0 otherwise.

z_{imv}: 1 if the operation of vessel v ends at port visit (i, m) and 0 otherwise.

z^{O}_{v}: 1 if vessel v is not used and 0 otherwise.

w_{imv}: 1 if vessel v meets m^{th} visit of port i and 0 otherwise

y_{im}: 1 if m^{th} visit of port i is done and 0 otherwise.

o_{imvk}: 1 if product k is loaded/unloaded at port visit (i, m) with vessel v and 0 otherwise

l_{imvk}: Onboard quantity of product k on vessel v after port visit (i, m)

q_{imvk}: Loaded/unloaded amount of product k at port visit (i, m) with vessel v

t_{im}: Start time of handling operation at port visit (i, m)

t^E_{im}: End time of handling operation at port visit (i, m)

s_{imk}: Level of inventory for product k at the start of handling operation at port visit (i, m)

s^E_{imk}: Level of inventory of product k at the end of handling operation at port visit (i, m)

The mathematical model is defined as follows:
Objective function:

$$Min \quad Z = \sum_{v\in V}\sum_{(i,m,j,n)\in S^X_v} C^T_{ijv} x_{imjnv} + \sum_{v\in V}\sum_{(i,m)\in S^A_v} C^{TO}_{iv} x^O_{imv} + \sum_{v\in V}\sum_{(i,m)\in S^A_v}\sum_{k\in K_v} (C^O_{ik} o_{imvk} + C^Q_{ik} q_{imvk}) \tag{1}$$

Subject to
Routing constraints:

$$\sum_{(i,m)\in S^A_v} x^O_{imv} + z^O_v = 1, \qquad v \in V \tag{2}$$

$$x^O_{imv} + \sum_{(j,n,i,m)\in S^X_v} x_{jnimv} - w_{imv} = 0 \qquad v \in V, (i,m) \in S^A_v \tag{3}$$

$$w_{imv} - \sum_{(i,m,j,n)\in S^X_v} x_{imjnv} - z_{imv} = 0 \qquad v \in V, (i,m) \in S^A_v \tag{4}$$

$$\sum_{v\in V:(i,m)\in S^A_v} w_{imv} = y_{im} \qquad (i,m) \in S^A \tag{5}$$

$$y_{i(m-1)} - y_{im} \geq 0 \qquad (i,m) \in S^A : m \geq 2 \tag{6}$$

$$x^O_{imv}, w_{imv}, z_{imv} \in \{0,1\} \qquad v \in V, (i,m) \in S^A_v \tag{7}$$

$$x_{imjnv} \in \{0,1\} \qquad v \in V, (i,m,j,n) \in S^X_v \tag{8}$$

$$y_{im} \in \{0,1\} \qquad (i,m) \in S^A \tag{9}$$

$$z^O_v \in \{0,1\} \qquad v \in V \tag{10}$$

Loading and unloading constraints

$$x_{imjnv}(l_{imvk} + J_{jk}q_{jnvk} - l_{jnvk}) = 0 \qquad v \in V, (i,m,j,n) \in S^X_v, k \in K_v \tag{11}$$

$$x^O_{imv}(J_{ik}q_{imvk} - l_{imvk}) = 0 \qquad v \in V, (i,m) \in S^A_v, k \in K_v \tag{12}$$

$$\sum_{k \in K_v} l_{imvk} \leq C_v \sum_{(j,n) \in S^A_v} x_{imjnv} \qquad v \in V, (i,m) \in S^A_v \tag{13}$$

$$\underline{Q}_{ik} o_{imvk} \leq q_{imvk} \leq \overline{Q}_{ik} o_{imvk} \qquad v \in V, (i,m) \in S^A_v, k \in K_v \tag{14}$$

$$\sum_{k \in K_v} o_{imvk} \geq w_{imv} \qquad v \in V, (i,m) \in S^A_v \tag{15}$$

$$o_{imvk} \leq w_{imv} \qquad v \in V, (i,m) \in S^A_v, k \in K_v \tag{16}$$

$$l_{imvk}, q_{imvk} \geq 0 \qquad v \in V, (i,m) \in S^A_v, k \in K_v \tag{17}$$

$$o_{imvk} \in \{0,1\} \qquad v \in V, (i,m) \in S^A_v, k \in K_v \tag{18}$$

Time constraints

$$t^E_{im} \geq t_{im} + \sum_{v \in V} \sum_{k \in K_v} T^Q_{ik} q_{imvk} \qquad (i,m) \in S^A \tag{19}$$

$$t_{im} - t^E_{i(m-1)} \geq 0 \qquad (i,m) \in S^A, m \geq 2 \tag{20}$$

$$x_{imjnv}(t^E_{im} + T_{ijv} - t_{jn}) \leq 0 \qquad (i,m,j,n) \in S^X_v, v \in V \tag{21}$$

$$\sum_{v \in V} T^O_{iv} x^O_{imv} \leq t_{im} \qquad (i,m) \in S^A \tag{22}$$

$$t_{im}, t^E_{im} \geq 0 \qquad (i,m) \in S^A \tag{23}$$

Inventory constraints

$$s_{i1k} = S^O_{ik} + J_{ik} R_{ik} t_{i1} \qquad i \in N, k \in K \tag{24}$$

$$s^E_{imk} = s_{imk} - \sum_{v \in V : (i,m) \in S^A_v, k \in K_v} J_{ik} q_{imvk} + J_{ik} R_{ik} (t^E_{im} - t_{im}) \qquad (i,m) \in S^A, k \in K \tag{25}$$

$$s_{imk} = s^E_{i(m-1)k} + J_{ik} R_{ik} (t_{im} - t^E_{i(m-1)}) \qquad (i,m) \in S^A : m \geq 2, k \in K \tag{26}$$

$$\underline{S}_{ik} \leq s_{imk}, s^E_{imk} \leq \overline{S}_{ik} \qquad (i,m) \in S^A, k \in K \tag{27}$$

$$\underline{S}^T_{ik} \leq s^E_{i\overline{M}_i k} + J_{ik} R_{ik} (\overline{T} - t^E_{i\overline{M}_i})) \leq \overline{S}^T_{ik} \qquad i \in N, k \in K \tag{28}$$

$$s_{imk}, s^E_{imk} \geq 0 \qquad (i,m) \in S^A, k \in K \tag{29}$$

The objective function (1) aims to minimize the total transportation and operational costs. Equations (2) determine whether vessel v is used or not. In (3), the previous location of each port visit is defined, which can be either the origin of vessel v or another port visit of that vessel (j, n). Equations (4) state that each vessel can either end its route at a port visit (i, m) or continue to another port visit (j, n). Constraints (5) ensure that each port visit (i, m) is carried out by one of the available vessels, and equations (6) ensure that the port visits are conducted successively. Constraints (7)-(10) define binary variables while constraints (11) and (12) determine the onboard quantity of each product after each port visit. Equations (13) describe the capacity constraint for the vessels. In (14), the number of products allowed to be loaded/unloaded is limited. Equations (15) and (16) define the possibility of loading and

unloading operations with the required port visits. Constraints (17) and (18) introduce loading variables.

Constraints (19) determine the end-time operation at port visit (i, m) based on the number of products loaded/unloaded. Equations (20) state that the operation at port i cannot start before ending its previous port visit. Constraints (21) and (22) relate the beginning operation time at each port visit with the transportation time from the last port visit and its prior ending operation time. Equations (23) define time variables.

Constraints (24) determine the stock level of each product at each port at its first visit. Equations (25) and (26) define the inventory level of each product at the beginning and end of each port visit. Constraints (27) and (28) ensure that each product's stock level lies within the predefined maximum and minimum levels at each port visit and at the end of the planning horizon. Constraints (29) define stock variables.

3 ALNS-based matheuristic algorithm

We propose a matheuristic algorithm to solve a multi-product many-to-many inventory routing problem. In our algorithm, we take advantage of the power of metaheuristics to search the discrete domain and the capabilities of MIP solvers to deal with continuous variables.

In the proposed algorithm, called Adaptive Large Neighborhood Search-based Matheuristic (ALNSM), all variables in the mathematical model are divided into two main groups: routing and non-routing variables. The routing variables are taken care of by a metaheuristic algorithm known as Adaptive Large Neighborhood Search. Following that, the non-routing variables are determined by the MIP solver, given the values of the routing variables.

Considering the difficulty of generating feasible solutions for the problem, we define a penalized model where the inventory constraints are relaxed, and instead, penalty functions for the violated constraints are added to the objective function. As finding a feasible solution for the penalized model is still difficult, we initiate our algorithm with a sub-model that offers a more relaxed version of the penalized model. This sub-model, explained in Sect. 3.2, is called only once at the beginning of the algorithm to generate a feasible initial solution.

In each iteration of the ALNSM algorithm, a new solution is generated using a group of designed operators to determine routing variables. To this end, we design different operators for the penalized and the original problems. Once the search reaches the solution space of the original problem, i.e., the penalty variables become zero, it continues using operators designed solely for the original problem.

Next, the routing variables are passed to the MIP solver to calculate the objective function, determine the values of the non-routing and penalty variables, and check

the feasibility of the solution. The ALNSM algorithm proceeds with searching the solution space until it meets the stop-criterion. In order to escape from local optima, an escape-algorithm is applied which is described in Sect. 3.4. Ultimately, the best-found solution is reported. The main steps of the algorithm are summarized in Algorithm 1. The proposed penalized model and initial solution are described in Sects. 3.1 and 3.2, respectively, followed by the ALNS algorithm in Sect. 3.3.

Algorithm 1 ALNS-based matheuristic

1: **Input**: All sets and parameters, set of all operators H, set of non-penalty-related operators H'
2: generate an initial solution s (Section 3.2)
3: **if** s **is feasible for the original problem then**
4: $s_{best} \leftarrow s$
5: **repeat**
6: **if** escape condition is met (Section 3.4) **then**
7: run escape-algorithm(s, s_{best})
8: **end if**
9: $s' \leftarrow s$
10: select $h \in H$ based on the ALNS selection procedure (Section 3.3)
11: apply heuristic h to s' and return the routing variables
12: pass the routing variables to the MIP solver with the original model and determine the non-routing variables, the objective value, and check the feasibility of s'.
13: **if** s' is feasible for the original problem **then**
14: update s and s_{best} based on the acceptance method mentioned in Algorithm 2
15: **end if**
16: **until** stop condition
17: **return** s_{best}
18: **end if**
19: **if** s **is not feasible for the original problem then**
20: **repeat**
21: $s' \leftarrow s$, $s_{temp-best} \leftarrow s$
22: **if** escape condition is met (Section 3.4) **then**
23: run escape-algorithm(s, $s_{temp-best}$)
24: **end if**
25: select $h \in H'$ based on the ALNS selection procedure (Section 3.3)
26: apply heuristic h to s' and return the routing variables
27: pass the routing variables to the MIP solver with the penalized model and determine the non-routing variables, the penalty variables, the objective value, and check the feasibility of s'
28: **if** s' is feasible for the penalized model and penalty variables are not zero **then**
29: update s and $s_{temp-best}$ based on the acceptance method mentioned in Algorithm 2
30: **end if**
31: **if** s' is feasible for the penalized model and penalty variables are zero **then**
32: $s \leftarrow s'$ and go to line 4
33: **end if**
34: **until** stop condition
35: **return** " No feasible solution found."
36: **end if**

3.1 Penalized model

Taking the complexity of the problem into account for finding a feasible solution, a set of constraints in the mathematical model, i.e., inventory constraints, are relaxed, and the corresponding penalties for the violated constraints are added to the objective function. We define the stock penalty variables as follows:

g_{imk}: The amount of stock shortage for product k at port visit (i, m) at the beginning of handling operation compared to the lower bound for the stock level, $\underline{S}_{ik}$.

g^E_{imk}: The amount of stock shortage for product k at port visit (i, m) at the end "E" of handling operation compared to the lower bound for the stock level, $\underline{S}_{ik}$.

e_{imk}: The excess amount of stock for product k at port visit (i, m) at the beginning of handling operation compared to the upper bound for the stock level, $\overline{S}_{ik}$.

e^E_{imk}: The excess amount of stock for product k at port visit (i, m) at the end "E" of handling operation compared to the upper bound for the stock level, $\overline{S}_{ik}$.

g^T_{ik}: The amount of stock shortage for product k at port i at the end of planning horizon "T" compared to the lower bound for the stock level at the end of planning horizon, $\underline{S}^T_{ik}$.

e^T_{ik}: The excess amount of stock for product k at port i at the end of planning horizon "T" compared to the upper bound for the stock level at the end of planning horizon, $\overline{S}^T_{ik}$.

With the new variables defined above, the inventory constraints (27, 28) are relaxed as follows:

$$\underline{S}_{ik} - g_{imk} \le s_{imk} \le \overline{S}_{ik} + e_{imk} \qquad (i,m) \in S^A, k \in K \tag{30}$$

$$\underline{S}_{ik} - g^E_{imk} \le s^E_{imk} \le \overline{S}_{ik} + e^E_{imk} \qquad (i,m) \in S^A, k \in K \tag{31}$$

$$\underline{S}^T_{ik} - g^T_{ik} \le s^E_{i\overline{M}_i k} + J_{ik} R_{ik} (\overline{T} - t^E_{i\overline{M}_i}) \le \overline{S}^T_{ik} + e^T_{ik} \qquad i \in N, k \in K \tag{32}$$

Moreover, we define "P" as a sufficiently big number, and we update the objective function by adding the following penalty function:

$$\sum_{v \in V} \sum_{(i,m) \in S^A_v} \sum_{k \in K_v} P(g_{imk} + e_{imk} + g^E_{imk} + e^E_{imk}) + \sum_{i \in N} \sum_{k \in K} P(g^T_{ik} + e^T_{ik})$$

The main challenge here is to minimize the penalty function and find a feasible solution for the original problem. To this end, we design a set of operators to focus on this purpose described in Sect. 3.3.2.

3.2 Initial solution

Although the stock constraints in the penalized model are relaxed, the routing, loading/unloading, and time constraints should be satisfied to start with a feasible initial solution.

To make it easier to find a feasible initial solution, we start the algorithm with a solution in which each port is visited only once, and a set of constraints are satisfied. To this end, the sub-model includes routing constraints (2)–(5) and (7)–(10), loading and unloading constraints (11)–(18), and the following constraints:

$$y_{i1} = 1 \qquad i \in N \tag{33}$$

$$y_{im} = 0 \qquad i \in N, m \geq 2 \tag{34}$$

$$\sum_{v \in V} \sum_{j \in N : (i,1,j,1,v) \in S_v^X} x_{i1j1v} + \sum_{v \in V} z_{i1v} = 1 \qquad i \in N \tag{35}$$

$$\sum_{v \in V} \sum_{i \in N : (i,1,j,1,v) \in S_v^X} x_{i1j1v} + \sum_{v \in V} x^O_{j1v} = 1 \qquad j \in N \tag{36}$$

$$ku_{i1v} + 1 - ku_{j1v} \leq |N|(1 - x_{i1j1v}) \qquad v \in V, (i,1,j,1,v) \in S_v^X \tag{37}$$

$$ku_{i1v} \geq 0 \qquad v \in V, i \in N : (i,1) \in S_v^A \tag{38}$$

Constraints (33) and (34) ensure that each port is visited only once. Equations (35)-(38) along with the other mentioned constraints determine one feasible route for each used vessel.

3.3 Adaptive large neighborhood search algorithm

The core of our proposed ALNS-based matheuristic algorithm is based on an adaptive large neighborhood search framework inspired by Ropke and Pisinger (2006). In this framework, as an efficient heuristic for vehicle routing problems, different operators are defined to create new solutions. In the selection procedure, initially, all the operators have the same chance to be selected to generate new solutions. After a number of iterations called a segment, the probability of choosing different operators is updated based on their performance in the previous iterations. Therefore, the more efficient operators have a higher chance of being selected in the next iterations. The algorithm selects the operators using a roulette wheel selection principle.

Algorithm 2 presents the framework of the ALNS algorithm. The acceptance method that we used in the following framework (lines 7–12) is based on the well-known simulated annealing algorithm's acceptance criteria (Kirkpatrick et al. 1983).

Algorithm 2 Adaptive large neighborhood search

```
Generate an initial solution, s with objective function of f(s)
s_best ← s
repeat
    s' ← s
    select h ∈ H (a set of operators H) based on selection parameters
    apply heuristic h to s'
    if f(s') < f(s_best) then
        s_best ← s'
    end if
    if accept(s', s) then,
        s ← s'
    end if
    update selection parameters
until stop condition
return s_best
```

3.3.1 Solution representation

The solution is represented with a two-dimensional vector defined as follows: in the first dimension, the sequence of port visits for different vessels is defined, and in the second dimension, the corresponding visit counter for each port is specified. For example, suppose that the number of ports is 6, the number of vessels is 3, and the maximum number of visits for each port is 2. Therefore, one possible solution can be represented as follows:

1 3 0 1 2 6 0 5 3 4
2 1 0 1 1 1 0 1 2 1

The solution depicts that the first vessel makes the second port visit of port 1, and the first port visit of port 3. The second vessel makes the first port visits of ports 1, 2, and 6, and the third vessel makes the first port visit of port 5, the second port visit of port 3, and the first port visit of port 4. In this solution representation, vessels and their routes are separated by (0, 0).

3.3.2 Operators

To find the best routing plan for the fleet of vessels, we design different operators to focus on the sequence of visits performed by vessels. The main challenge is moving from the penalized model's feasible solution space to the original feasible region. To achieve this, in the first part of the algorithm, we design several operators to focus on decreasing the penalty function and moving toward the feasible region of the original problem. Once the algorithm reaches the original feasible solution space, the other operators are applied to find the best solution to the original problem. The following three principles are used in all designed operators:

1. Each used vessel should start its route by visiting one producer and end it with one customer.
2. If a given port is producing only one specific product and there is only one customer for that product, both the producer and the corresponding customer must be visited by the same vessel.
3. Changing the position of ports in a solution may result in rearranging port visit counters accordingly.

Among the following operators, the first two are specifically designed to decrease the penalty function, while the rest of them are applied to search the solution space for both penalized and original problems.

- Stock-Balanced: The Stock-Balanced operator removes one port visit with a positive stock penalty value and inserts it in a possible place in a route. Positive g_{imk} and g^E_{imk} indicate that the corresponding vessel arrives late at the port visit (i, m). Moreover, positive g^T_{ik} denotes that the last visit to port i is performed early so that the stock level of product k at the end of the planning horizon is less than the minimum required level. Similarly, the positive values for e_{imk}/e^E_{imk} and e^T_{ik} indicate a high stock level of product k at the port visit (i, m), and port i at the end of the planning horizon, respectively. The Stock-Balanced operator removes one of the port visits with a positive stock penalty value. Ports with higher penalty values have more chances to be removed. If g_{imk}, g^E_{imk}, e_{imk}, and e^E_{imk} are positive, the removed port visit will not be inserted at any place in a route of its allocated vessel later than its present point. Instead, it can be done earlier or assigned to another compatible vessel. However, positive g^T_{ik} and e^T_{ik} represent that the last visit to port i should be done later so that the stock level of product k at port i remains in a range of predefined bound limits at the end of the planning horizon. In this case, the last port visit to port i should not be done earlier than its present time. Therefore, either it can be done later or moved to another compatible vessel.
- Random-Worst-Remove-Random-Insert (RWRR): Compared to the Stock-Balanced operator, the RWRR operator selects more than one port visit with positive penalty values and inserts them in random possible places in a route. The maximum number of port visits to remove from each route is limited such that each route includes at least one producer and one customer. Among port visits with positive values, the number of possible port visits to remove is determined randomly, defining at most four. Port visits with higher penalty values get more chances to be selected and removed. Next, selected port visits with their connected ports (based on the second principle described above) are removed and inserted in a random possible place in a solution, considering the compatibility of ports and vessels. Since more than one port visit is selected to be removed, the insertion rule in the Stock-Balanced operator is not followed by this operator. Changing the position of several port visits may affect the status of other removed port visits.

- Random-Remove-Random-Insert (RRR): Unlike the RWRR, which is applied to the penalized model, RRR is used for both penalized and original problems. Herein, at most, four port visits are selected randomly to remove. Ensuring that at least one producer and one customer are assigned to each vessel, selected port visits are removed and inserted in a random possible place in a route.
- Random-Remove-First-Improvement-Insert (RRFI): Here, one port visit in a solution is selected randomly, ensuring that it does not belong to a vessel with only one producer and one customer. Next, the removed port visit is inserted in a place where the first improvement is found, i.e., a point in a solution in which the new solution is still feasible, and the objective value is less than its present value. To calculate the objective value and check the feasibility of the solution, the solver is called for each step of insertion. If there is no better possible place for insertion, the removed port visit remains in its current place in a solution.
- Swap: To diversify the search, we change the position of two port visits in a solution with a swap operator. It is necessary to make sure that selected ports are compatible with their newly assigned vessels, the logic of starting a route with one producer and ending it with one customer being satisfied, and each used vessel contains at least one producer and one customer after the swap operation.
- Route-Add: To generate solutions with fewer idle vessels, one producer and one customer are selected and successively inserted in one possible place in a solution. It is required to make sure that a common compatible vessel is used to carry a product for a producer and its customer. Therefore, it is possible to use more vessels for the transportation plan, which may create more and better feasible solutions.
- Vessel-Change: Compared to the Route-Add operator, the Vessel-Change operator generates new solutions where one used vessel is discarded, and its allocated route is assigned to the other compatible vessels. This operator helps to escape from local optima when it is necessary to reduce the number of used vessels.
- Visit-Add: In case we are allowed to visit each port more than once, the Visit-Add operator is applied. The minimum required number of visits for each port is calculated based on the initial inventory of each product at each port, the production and consumption rates of each product at each port, the stock bounds at each port for each product, and the maximum amount of each product allowed to be handled at each port. It is necessary to ensure that each port is visited at least to its minimum required number of visits. In the Visit-Add operator, extra port visits are added to the current solution to check if more port visits could lead to a better solution at a lower cost.
- Visit-Remove: Similar to the Visit-Add operator, the minimum required number of visits for each port is calculated. In this operator, the extra port visits compared to its minimum required number of visits (if any) are removed from the current solution to check the possibility of making a better new feasible solution.
- Vessel-Swap: In this operator, the routes of two vessels with common characteristics but different transportation costs are changed. The Vessel-Swap operator helps to create better feasible solutions where compatible vessels are strictly limited, and the transportation costs for compatible vessels are considerably different.

Table 1 Computational results for ALNSM and HCGR

				HCGR	ALNSM			ALNSM-HCGR
Inst. no.	$\|M\|$	$\|V\|$	$\|P\|$	Best Obj	Avg. Obj	Best Obj	Avg. Secs	Gap (%)
1	4	2	1	**327,037**	327,037	327,037	75	0.00
2	4	2	1	**367,408**	367,408	367,408	73	0.00
3	4	2	1	**396,685**	396,685	396,685	74	0.00
4	4	2	1	**412,060**	412,060	412,060	72	0.00
5	4	2	1	**328,049**	328,049	328,049	78	0.00
6	4	2	1	**210,031**	210,031	210,031	75	0.00
7	8	4	2	**694,445**	694,445	694,445	96	0.00
8	8	4	2	**723,722**	723,722	723,722	96	0.00
9	8	4	2	**739,097**	739,097	739,097	94	0.00
10	8	4	2	**655,086**	655,086	655,086	98	0.00
11	8	4	2	**695,457**	695,457	695,457	97	0.00
12	12	6	3	**949,128**	949,128	949,128	206	0.00
13	12	6	3	**865,117**	865,117	865,117	211	0.00
14	12	6	3	**989,499**	989,499	989,499	212	0.00
15	12	6	3	**905,488**	905,488	905,488	212	0.00
16	12	6	3	**1,018,776**	1,018,780	1,018,780	217	0.00
17	16	8	4	**1,232,525**	1,264,001	1,232,525	428	0.00
18	16	8	4	**1,261,802**	1,272,718	1,261,802	434	0.00
19	16	8	4	**1,277,177**	1,314,488	1,277,180	438	0.00
20	16	8	4	**1,302,173**	1,302,173	1,302,173	419	0.00
21	16	8	4	**1,317,548**	1,346,456	1,317,550	429	0.00
22	16	10	2	197,919	185,711	179,914	319	− 9.10
23	16	10	2	245,719	242,720	236,664	435	− 3.69
24	16	10	2	279,574	266,440	249,627	463	− 10.71
25	16	10	3	208,413	206,450	192,347	605	− 7.71
26	16	10	3	243,845	235,165	212,571	797	− 12.83
27	16	10	3	362,039	293,634	276,851	863	− 23.53
28	16	10	3	418,542	339,261	311,447	1053	− 25.59
29	20	10	5	**1,831,239**	1,870,533	1,831,239	724	0.00
30	20	10	5	**1,713,221**	1,732,888	1,713,221	745	0.00
31	20	10	5	**1,629,210**	1,695,540	1,629,210	754	0.00
32	20	10	5	**1,644,585**	1,701,483	1,644,585	709	0.00
33	20	10	5	**1,714,233**	1,792,897	1,714,233	711	0.00

- Farthest-Remove-Nearest-Insert (FRN): The FRN operator is designed to generate solutions with shorter distances among port visits in one single route. First, one port visit with the longest distance from its subsequent port visit is selected randomly. Port visits are arranged based on their distance in descending order to

avoid choosing the same port visit in successive iterations. Port visits with longer distances have more chances to be selected to remove. Finally, the removed port visit is inserted into a solution next to its nearest possible place.

3.4 Escape-algorithm

Although the ALNSM algorithm benefits from diversifying components, including the simulated annealing and randomness inside the operators, we also exploit the escape-algorithm to escape from local optima. In Algorithm 1, if there is no improvement after a number of iterations (m), we activate the escape-algorithm. In this iterative algorithm, two operators are randomly selected to diversify the search. The new solution in each iteration is accepted regardless of its quality. The algorithm stops once a better solution than the best-found solution is found. Otherwise, the algorithm continues until it meets the stopping criterion, which is reaching a certain number of iterations.

Algorithm 3 Escape-algorithm

Inputs: s, s_{best}
repeat
  select h from selected operators randomly
  apply heuristic h to s and store the new solution in s
  if $f(s) < f(s_{best})$ **then**
    $s_{best} \leftarrow s$
    return s, s_{best} and **Exit** the algorithm
  end if
until stop condition
return s, s_{best}

4 Computational result

A set of realistic-sized instances, as defined by Hemmati et al. (2016), is used to evaluate the performance of the ALNSM algorithm. The instances are categorized into two groups. In the first group, instances are deliberately designed to allow for problem segmentation into smaller sub-problems, enabling the reporting of optimal solutions. Conversely, the instances in the second group, indexed as 22–28 in Table 1, cannot be partitioned into smaller sub-problems, thus preventing the reporting of optimal solutions. The instances are characterized based on the number of producers and customers, the number of vessels, and the number of products for a fixed planning horizon. Furthermore, the test instances are divided into two groups based on their compatibility: with full compatibility and with limited compatibility between vessels, ports, and products. It is assumed that there is limited compatibility between vessels, ports, and products for all instances except for those indexed 22–28. In the group of instances with limited compatibility, we can only visit each

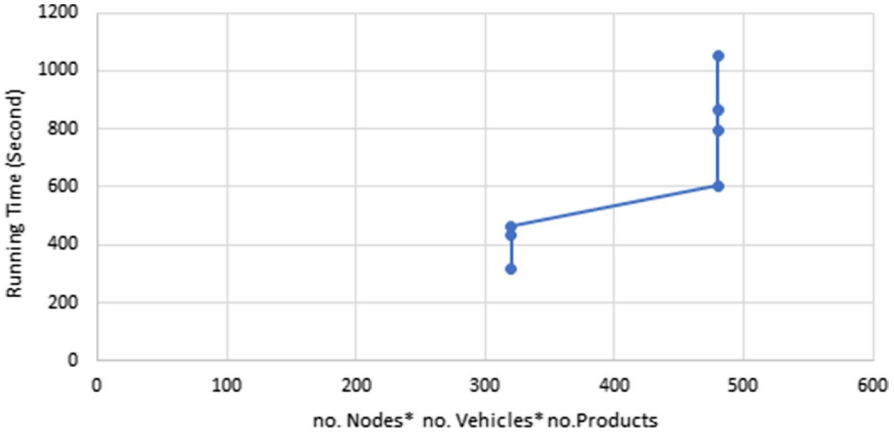

Fig. 1 With full compatibility between vessels, ports, and products

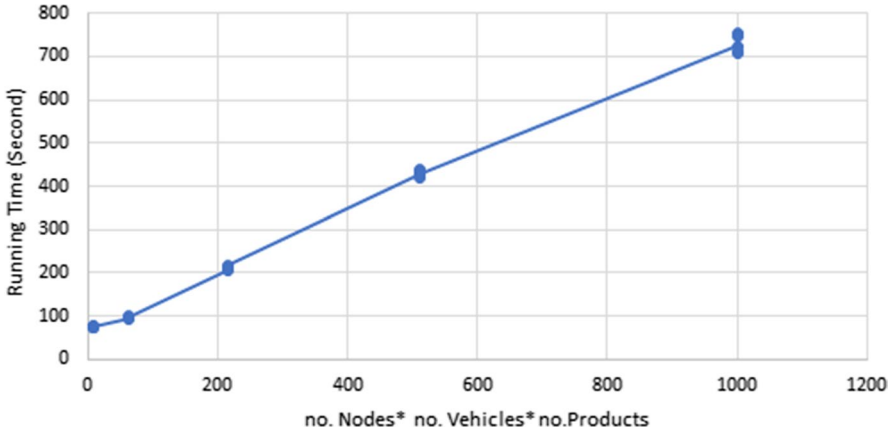

Fig. 2 With limited compatibility between vessels, ports, and products

port and carry each product with a few compatible vessels, while in the cases with full compatibility, all vessels, ports, and products are compatible.

Gurobi is used as the MIP solver in our proposed algorithm. In the ALNS algorithm, the number of iterations in each segment is 50. Moreover, in the escape-algorithm, m is defined as 100, and the stop-criterion is set at 20 iterations. The two selected operators in the escape-algorithm include swap and RRR. The ALNSM algorithm stops after 6000 iterations.

Table 1 illustrates the computational results achieved by the ALNSM and its comparison with the HCGR presented by Hemmati et al. (2016). The first four columns of Table 1 specify the instance index, the number of ports, the number of vessels, and the number of products, respectively. The next column shows the best-found solutions by the HCGR. The following three columns illustrate the

average objective values, the best-found solution, and the average running time (in seconds) achieved by running the ALNSM 10 times on each instance. The last column shows the gap between the ALNSM and the HCGR. Bold values are reported as the optimal solutions by Hemmati et al. (2016).

As the results show, our proposed algorithm finds the optimal solution for all instances with limited compatibility between vessels, ports, and products. However, for the remaining cases, our proposed algorithm outperforms the HCGR. Specifically, for instances with full compatibility, the gap between the results of the ALNSM and the HCGR is improved by up to 26%, respectively.

The ALNSM and the HCGR algorithms are tackling the problem differently. The HCGR algorithm is initiated by specifying the minimum required quantity to be loaded/unloaded from each port. This is calculated based on the given production/consumption rates, initial stocks, and the minimum and maximum bounds of stock levels. Then, the best amount of products to be loaded/unloaded from each port and the relevant time windows are determined. Here, the HCGR algorithm specifies the fixed pair of pickup and delivery ports based on their direct transportation costs and operational costs. Next, the best route of fixed pickup and delivery ports is determined using adaptive large neighborhood search. The main drawback of the HCGR algorithm is disregarding some parts of the feasible solution space and considering only the immediate impact of decisions. For example, consider a network with six ports, where ports 1 and 5 act as producers for one product, and ports 2, 3, 4, and 6 are customers. Assume that the minimum required quantities to be loaded from ports 1 and 5 are 660 and 220 units, respectively, and each customer demands a minimum of 220 units. The HCGR algorithm yields the following fixed pickup and delivery pairs:

(1, 2) with 440 units of products to be loaded from port 1 and unloaded at port 2
(1, 3) with 220 units of products to be loaded from port 1 and unloaded at port 3
(5, 4) with 220 units of products to be loaded from port 5 and unloaded at port 4
(5, 6) with 220 units of products to be loaded from port 5 and unloaded at port 6.

The above solution is obtained due to the consideration of direct transportation costs between the paired pickup and delivery ports. However, a feasible and more improved solution is possible by keeping products onboard the vessels and unloading them later. Therefore, a better feasible solution is the route of 1-2-3-4 with the loading of 660 units at port 1, unloading 220 units at ports 2, 3, and 4, and the route of 5-6 with the loading of 220 units at port 5, and unloading 220 units at port 6. Our proposed algorithm can achieve this by determining different load/unload quantities with no limitation in searching the entire solution space. In the ALNSM, routes are defined using an adaptive large neighborhood search algorithm, and a MIP solver is utilized to obtain optimal values of non-routing variables, subject to the specified routing variables. This process continues until the stop-criterion is met, reporting the best-found feasible solution.

Two groups of instances are analyzed separately to highlight the effect of problem characteristics on the running time. Figures 1 and 2 depict the relationship between the running time and problem characteristics for the first and second groups of instances, respectively.

In this context, the problem characteristic is defined as the multiplication of the number of ports, the number of vessels, and the number of products. As the results show, the required time to solve different instances is affected by the problem characteristics. Moreover, according to Fig. 1, within the group of instances with identical problem characteristics, the number of active producers and customers affects the running time. Active producers and customers handle different product types in their stock. Increasing the number of inventories in a problem leads to a corresponding increase in the running time.

5 Conclusion

In this study, we proposed an ALNS-based matheuristic algorithm to solve the multi-product many-to-many inventory routing problem introduced by Hemmati et al. (2016). The proposed ALNSM follows an iterative process where routing variables are determined using an adaptive large neighborhood search algorithm, while a MIP solver is employed to determine the optimal values of stock and time variables based on the obtained routing values. Comparing the performance of the ALNSM with the HCGR algorithm reveals that the ALNSM significantly outperforms the HCGR on different sets of realistic instances. Particularly, our algorithm excels in improving solution quality when there is full compatibility between vessels, ports, and products. In such instances, the feasible solution space is larger, necessitating more exploration to find better solutions. The ALNSM's approach of emphasizing finding optimal values for non-routing variables for each specific vessel routs allows for a more comprehensive search compared to the HCGR, where some areas of the solution space are not even explored. Our results demonstrate notable improvements of up to 26% in solution quality for this group of instances. Furthermore, for the other group of instances with limited compatibility, the ALNSM achieves the optimal solutions reported by Hemmati et al. (2016).

Funding Open access funding provided by University of Bergen (incl Haukeland University Hospital).

Declarations

Conflict of interest None of the authors have any financial or non-financial interests that are directly or indirectly related to this work.

References

Agra A, Andersson H, Christiansen M, Wolsey L (2013) A maritime inventory routing problem: discrete time formulations and valid inequalities. Networks 62(4):297–314

Agra A, Christiansen M, Delgado A, Simonetti L (2014) Hybrid heuristics for a short sea inventory routing problem. Eur J Oper Res 236(3):924–935

Agra A, Christiansen M, Hvattum LM, Rodrigues F (2016) A MIP based local search heuristic for a stochastic maritime inventory routing problem. In International conference on computational logistics, pp 18–34. Springer

Agra A, Christiansen M, Delgado A (2017) Discrete time and continuous time formulations for a short sea inventory routing problem. Optim Eng 18(1):269–297

Al-Khayyal F, Hwang S-J (2007) Inventory constrained maritime routing and scheduling for multi-commodity liquid bulk, part I: applications and model. Eur J Oper Res 176(1):106–130

Andersson H, Hoff A, Christiansen M, Hasle G, Løkketangen A (2010) Industrial aspects and literature survey: combined inventory management and routing. Comput Oper Res 37(9):1515–1536

Brekkå I, Randøy S, Fagerholt K, Thun K, Vadseth ST (2022) The fish feed production routing problem. Comput Oper Res 144:105806

Chen C, Demir E, Huang Y (2021) An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and delivery robots. Eur J Oper Res 294(3):1164–1180

Christiansen M (1999) Decomposition of a combined inventory and time constrained ship routing problem. Transp Sci 33(1):3–16

Christiansen M, Fagerholt K, Flatberg T, Haugen Ø, Kloster O, Lund EH (2011) Maritime inventory routing with multiple products: a case study from the cement industry. Eur J Oper Res 208(1):86–94

Coelho LC, Cordeau J-F, Laporte G (2013) Thirty years of inventory routing. Transp Sci 48(1):1–19

Diz GSdS, Oliveira F, Hamacher S (2017) Improving maritime inventory routing: application to a Brazilian petroleum case. Marit Policy Manag 44(1):42–61

Engineer FG, Furman KC, Nemhauser GL, Savelsbergh MW, Song J-H (2012) A branch-price-and-cut algorithm for single-product maritime inventory routing. Oper Res 60(1):106–122

François V, Arda Y, Crama Y (2019) Adaptive large neighborhood search for multitrip vehicle routing with time windows. Transp Sci 53(6):1706–1730

Friske MW, Buriol LS (2020) A multi-start algorithm and a large neighborhood search for a maritimeinventory routing problem. In: 2020 IEEE congress on evolutionary computation (CEC), pp 1–8. IEEE

Friske MW, Buriol LS, Camponogara E (2022) A relax-and-fix and fix-and-optimize algorithm for a maritime inventory routing problem. Comput Oper Res 137:105520

Hemmati A, Hvattum LM, Christiansen M, Laporte G (2016) An iterative two-phase hybrid matheuristic for a multi-product short sea inventory-routing problem. Eur J Oper Res 252(3):775–788

Hewitt M, Nemhauser G, Savelsbergh M, Song J-H (2013) A branch-and-price guided search approach to maritime inventory routing. Comput Oper Res 40(5):1410–1419

Kirkpatrick S, Gelatt CD Jr, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680

Lianes IM, Noreng MT, Fagerholt K, Slette HT, Meisel F (2021) The aquaculture service vessel routing problem with time dependent travel times and synchronization constraints. Comput Oper Res 134:105316

Misra S, Kapadi M, Gudi RD (2020) Hybrid time-based framework for maritime inventory routing problem. Ind Eng Chem Res 59(46):20394–20409

Pisinger D, Ropke S (2007) A general heuristic for vehicle routing problems. Comput Oper Res 34(8):2403–2435

Ribeiro GM, Laporte G (2012) An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. Comput Oper Res 39(3):728–735

Ropke S, Pisinger D (2006) An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp Sci 40(4):455–472

Siswanto N, Essam D, Sarker R (2011) Solving the ship inventory routing and scheduling problem with undedicated compartments. Comput Ind Eng 61(2):289–299

Song J-H, Furman KC (2013) A maritime inventory routing problem: practical approach. Comput Oper Res 40(3):657–665

Yu Z, Zhang P, Yu Y, Sun W, Huang M (2020) An adaptive large neighborhood search for the larger-scale instances of green vehicle routing problem with time windows. Complexity 1–14:2020

Authors and Affiliations

Nooshin Heidari[1] · **Ahmad Hemmati**[1]

✉ Nooshin Heidari
Nooshin.Heidari@uib.no

Ahmad Hemmati
Ahmad.Hemmati@uib.no

[1] Department of Informatics, University of Bergen, Bergen, Norway