An adaptive large neighborhood search heuristic for the multi-port continuous berth allocation problem

Martin-Iradi, Bernardo; Pacino, Dario; Ropke, Stefan

Published in:
European Journal of Operational Research

Link to article, DOI:
10.1016/j.ejor.2024.02.003

Publication date:
2024

Document Version
Publisher's PDF, also known as Version of record

European Journal of Operational Research 316 (2024) 152–167

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Production, manufacturing, transportation and logistics

An adaptive large neighborhood search heuristic for the multi-port continuous berth allocation problem

Bernardo Martin-Iradi *, Dario Pacino, Stefan Ropke

DTU Management, Technical University of Denmark, Akademivej Building 358, 2800 Kgs. Lyngby, Denmark

ARTICLE INFO

Keywords:
OR in maritime industry
Container terminal
Berth allocation problem
Speed optimization
Heuristics

ABSTRACT

In this paper, we study a problem that integrates the vessel scheduling problem with the berth allocation into a collaborative problem denoted as the multi-port continuous berth allocation problem (MCBAP). This problem optimizes the berth allocation of a set of ships simultaneously in multiple ports while also considering the sailing speed of ships between ports. Due to the highly combinatorial character of the problem, exact methods struggle to scale to large-size instances, which points to exploring heuristic methods. We present a mixed-integer problem formulation for the MCBAP and introduce an adaptive large neighborhood search (ALNS) algorithm enhanced with a local search procedure to solve it. The computational results highlight the method's suitability for larger instances by providing high-quality solutions in short computational times. Practical insights indicate that the carriers' and terminal operators' operational costs are impacted in different ways by fuel prices, external ships at port, and the modeling of a continuous quay.

1. Introduction

The liner shipping industry is one of the major forms of international freight transportation. According to the report by UNCTAD (2020), seaborne trade and container throughput continued growing steadily until 2019. Despite the Covid disruption during 2020, maritime trade is projected to recover and expand by 4.3% in 2021. The report also highlights that the world fleet is increasing, not only in the number of ships (more than 3% in 2021) but also in size. The share of the total capacity carried by mega-vessels increased from 6% to 40% in the last ten years.

This increase in demand, together with IMO's goal of reducing shipping emissions by 50% by 2050 (IMO, 2018), requires container terminals to increase capacity and improve the efficiency and sustainability of their operations. The current growth of the vessel fleet and size directly impacts one of the most critical container terminal operations, namely the berth allocation (Steenken et al., 2004). Mathematically, this problem is denoted as the Berth Allocation Problem (BAP), which aims to assign incoming ships to berthing positions. The BAP can assume the quay to be discrete or continuous. In the discrete version, the quay is divided into positions where each can be occupied by one ship at a time. In the continuous BAP, ships can berth at any point in the quay while respecting a safe distance from other ships. Furthermore, the BAP can be dynamic or static. The static BAP assumes all the ships to be already at the port when the planning is done, whereas, in the dynamic version, ships can arrive at the port at different times during the planning period. It should be noted that the dynamic BAP is still a deterministic problem. The term *dynamic* refers to the different arrival times of each ship and not to the nature of the problem (Cordeau et al., 2005) like in, for example, vehicle routing problems. Fig. 1 shows an example solution of the continuous and dynamic BAP. Typically, each ship has a fixed time window defined by its expected berthing start and finish time. If the ship arrives before its berthing start time or if the quay is occupied, it must wait at the port, resulting in *waiting time*. Similarly, when a ship exceeds its expected finish time, a *delay* is incurred. As a result, the *service time* of a ship is defined as the entire time that it spends at the port, i.e., its handling time or berthing period plus its waiting time. The authors refer to Sections 3–5 for further problem details.

Terminals optimize their berth allocation to minimize their operational costs and the time ships need to spend at the port, including waiting time, handling time, and any delays. Due to the fierce competition between container terminals, they do not tend to share more information than is strictly required and do the planning independently from other terminals. One potential problem is that if congestion arises in a port, the affected ships can easily propagate delays to the following ports in their routes. One way to reduce the delay is for vessels to

* Corresponding author.
E-mail addresses: bmair@dtu.dk (B. Martin-Iradi), darpa@dtu.dk (D. Pacino), ropke@dtu.dk (S. Ropke).

https://doi.org/10.1016/j.ejor.2024.02.003
Received 24 January 2023; Accepted 5 February 2024
Available online 6 February 2024

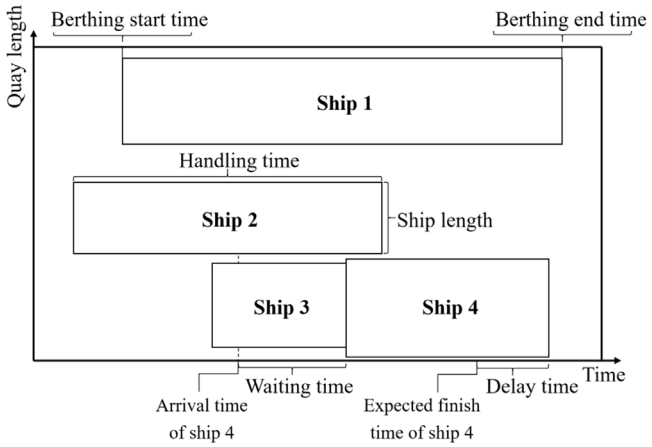


Fig. 1. Example solution of the continuous and dynamic BAP for a port terminal with four vessels.

speed up when sailing between ports. However, sailing faster results in higher fuel consumption. This type of decision-making can be addressed by shipping line companies (i.e., carriers) in the Vessel Scheduling Problem (VSP). The goal of the VSP is to optimize the sailing speeds between consecutive ports in the vessel's route (i.e., voyage legs). Most VSP studies aim to minimize the vessels' fuel consumption, turnaround time at the port, and the number of vessels needed to ensure a given route frequency. However, the VSP has its limitations. One of them is the simplistic way of modeling the berthing times of ships at port. Whereas some studies model a simplified version of berth allocation, most do not include it. Not integrating the BAP into the VSP can lead to an unrealistic or even infeasible berth allocation and, as a result, delays that ships can propagate.

A problem that integrates the berth allocation with the vessels' speed optimization was first introduced by Venturini et al. (2017) as the Multi-port Berth Allocation Problem (MBAP). This problem selects a set of ships and a set of ports that are part of their routes and simultaneously optimizes the berth allocation at all the ports, together with the sailing time between ports. Venturini et al. (2017) studied the version of the problem with a discrete set of berthing positions. The problem involves the joint optimization of carrier and terminal operations and relies on the a priori agreement of the vessels and ports involved. Martin-Iradi et al. (2022b) showed that this type of collaboration could generate cost savings for the players involved (i.e., shipping carriers and terminal operators) but also benefit the environment as fuel emissions can be reduced significantly.

As mentioned earlier in this section, the main difference between the continuous and the discrete BAP is the flexibility in the berthing positions. The set of berthing positions in the discrete BAP corresponds to a subset of those from the continuous BAP. Therefore, one can argue that modeling the quay as continuous can lead to a more resource-efficient plan, as the optimal solution of the continuous BAP is equal to or better than that of the discrete BAP. However, this potential increase in solution quality comes at the expense of higher complexity, as the solution space becomes considerably larger.

Martin-Iradi et al. (2022a) studied the MBAP with a continuous quay (MCBAP) and highlighted the additional complexity, as the method proposed cannot scale to large instances. This scalability issue is addressed in our study, where we employ heuristic methods that can tackle large real-world instances.

This paper makes the following four contributions:

1. We define a new mixed-integer problem (MIP) formulation for the MCBAP.
2. We present an instance generator for the MCBAP based on real-world port data, and define a set of benchmark instances that are made publically available.
3. We implement an adaptive large neighborhood search (ALNS) method tailored to the MCBAP and enhance it with a Local Search (LS) procedure based on ejection chains.
4. We show the viability of the ALNS method on real-size instances where it is able to find high-quality solutions faster than baseline commercial solvers.

The remainder of this paper is structured as follows. Section 2 comprises an extensive literature review of the MBAP together with other collaborative problems that include berth allocation or vessel scheduling. Section 3 describes the MCBAP in detail and presents the MIP formulation. The solution method is described in Section 4. Section 5 includes the instance generator's details and the computational study. The conclusions and further research is summarized in Section 6.

2. Literature review

One of the most important problems in a container terminal is the BAP, which has been studied extensively for over two decades. A survey of most of these studies is compiled in surveys by Carlo et al. (2014) and Bierwirth and Meisel (2015). Lim (1998) presented one of the first formulations of the problem and showed that it is NP-hard. Due to the additional hardness involving the BAP variant with a continuous quay, the use of heuristic methods has been predominant in the literature. The first studies of the continuous BAP were by Kim and Moon (2003) and Imai et al. (2005), where they presented MIP formulations to the problem and solved it using heuristic and metaheuristic algorithms such as simulated annealing. Cordeau et al. (2005) covered both the discrete and continuous BAP and solved them using a taboo search. Guan and Cheung (2005) presented both a composite heuristic and a tree search exact method and showed that both outperformed commercial solvers. De Oliveira et al. (2012) addressed the continuous BAP combining a clustering search with a simulated annealing metaheuristic, and Mauri et al. (2016) employed an ALNS, the same metaheuristic framework as in this paper, to solve both the discrete and continuous BAP. A hybrid variant between the continuous and discrete BAP was studied in Kordić et al. (2016), where ships can only berth in a subset of positions.

One of the main integrated problems studied is the berth allocation and quay crane assignment problem (Iris & Lam, 2018). Iris et al. (2017) present a mixed integer problem formulation with additional enhancements and implement an ALNS heuristic to solve it, whereas Cheimanoff et al. (2022) uses a variable neighborhood search heuristic.

The VSP has also attracted significant attention in the literature. Dulebenets et al. (2019) present a comprehensive survey about the problem and highlight the potential of collaboration and information sharing as one of the future research directions. To the best of our knowledge, Fagerholt (2001) presented the first formulation of the VSP. Negotiating the port calls with the terminal operator (Dulebenets, 2018) indicates that carriers and terminal operators can achieve significant savings. A collaborative version of the VSP is presented by Dulebenets (2019), where terminal operators offer different port call durations and handling rates, leading to win-win situations. Fagerholt et al. (2010) aim at minimizing fuel consumption by optimizing the speed in a shipping route and modeling it as a shortest path problem. The authors discretize the possible arrival times at each port to approximate the non-linear relation between fuel consumption and sailing speed. Du et al. (2011) and Sun et al. (2018) integrate vessel speed optimization and berth allocation by considering ships within a certain sailing distance from the port.

In the last decade, together with the increased access to data, the study of problems that require collaboration between different

stakeholders (e.g., carriers and terminal operators) has become more relevant. Wang et al. (2015) present two collaborative mechanisms that encourage sharing accurate information between carriers and terminal operators. Lalla-Ruiz et al. (2016) study the discrete BAP and present a cooperative search based on a grouping strategy where group members can only share information within the group. The collaborative berth allocation problem (CBAP) was introduced by Dulebenets et al. (2018) where a terminal planning its berth allocation can divert excessive demand to other terminals. Hellsten et al. (2020) present an ALNS heuristic for the port scheduling problem (PSP), where the aim is to schedule feeder vessels in multi-terminal ports. Collaboration has also been studied in disruption management. Lyu et al. (2022) present a formulation for re-planning the berth allocation and quay crane assignment and propose a heuristic method to solve it. Guo et al. (2022) study the berth assignment and allocation problem, which integrates the BAP with the berth assignment and line clustering problem. The first formulation of the MBAP was first introduced by Venturini et al. (2017). It solved a dynamic and discrete BAP in multiple ports while optimizing ships' sailing speed between ports. Martin-Iradi et al. (2022b) presented a branch-and-price method for the same problem and conducted a study of the collaboration mechanism using cooperative game theory. Martin-Iradi et al. (2022a) extended the branch-and-price method to the MBAP with a continuous quay, the same problem of this study, and showed that exact methods are competitive for small and medium size instances but struggle to scale for larger instances. Recently, Yu et al. (2022) presented a genetic algorithm to solve a problem that integrates the BAP with speed optimization and vessel service differentiation to address both vertical and horizontal collaborations.

3. Problem description

The MCBAP integrates operational aspects concerning terminal operators and shipping carriers. We consider a set of ships and a set of terminals, each of them in a different port, to optimize their operations. Each ship visits all or a subset of the ports as a part of its route. The ships may visit the ports in different orders. The aim of the problem is to determine the berthing position and time of the ships at each of the terminals visited. Each terminal has a limited berthing space, given by the length of the quay. The service time required to load and unload the vessel is denoted as handling time and depends on the berthing position. We assume that it increases linearly with the deviation from an ideal position. Similar to most BAP studies, the berthing time and positions of ships are subject to a set of restrictions. Ships have a time window to be serviced also known as a port call, this is planned in advance and helps the operator to allocate berthing capacity and avoid excessive congestion. To allow for delays, the end of the time window is not strict but delays are penalized as they require the use of unexpected resources such as more worker hours.

It is well known that the relation between sailing speed and fuel consumption is non-linear. In fact, this relation is often approximated with a cubic function as in Eq. (1) (Martin-Iradi et al., 2022b; Venturini et al., 2017)

$$F(s) = (\frac{s}{s_d})^3 F_d \tag{1}$$

where s is the sailing speed, s_d is the design speed of the ship, and F_d is the fuel consumption at the design speed. For our formulation, we discretize the set of possible sailing speeds and assume ships will sail the distance between ports at one of those speeds. Given the set of feasible sailing speeds, we can compute the corresponding set of fuel consumption rates. This assumption ensures a linear formulation of the problem.

Fig. 2 shows an example graphical representation of the problem, highlighting the main operational aspects of a ship (i.e., ship 1). The ship berths strictly after its earliest start time and departs towards the next port in the route as soon as the loading and unloading operations conclude. The chosen speed (i.e., *slow*) and the distance between ports directly determines the travel time and arrival to the next port (i.e., port 2). At the time of arrival the quay is occupied and ship 1 needs to wait until a berthing position is available. Due to the late berthing start and the handling time, the ship's service time exceeds the expected finish time and incurs in a delay.

3.1. MIP formulation

We present a new MIP formulation for the MCBAP. This formulation is based on the one for the continuous BAP from Kim and Moon (2003) and the one for the discrete MBAP from Venturini et al. (2017):
Sets and parameters:

N	Set of all ships berthing at any of the ports.
$N^* \subseteq N$	Set of ships that we are optimizing.
$\bar{N} \subseteq N$	Set of external ships which are considered fixed.
P	Set of ports.
S	Set of speeds.
L_p	Length of quay in port $p \in P$.
$P_i \subseteq P$	Set of ports planned to be visited by ship $i \in N^*$ sorted in visiting order.
$C_i = \{1, \dots, c_i\}$	Set of port calls for ship $i \in N$, one for each port visit. c_i is the last port visit, and the value is equal to the number of port calls.
ρ_i^c	The port $p \in P$ corresponding to port visit $c \in C_i$ for ship $i \in N$.
$N_p \subseteq N$	Set of ships that visit port $p \in P$.
$C_i^p \subseteq C_i$	Port call positions of ship $i \in N$ visiting port $p \in P$.
$x_0^{i,c}$	The ideal berthing position for ship $i \in N^*$ at port visit $c \in C_i$ measured at the leftmost position of the ship.
$h_0^{i,c}$	Handling time at the ideal berthing position for ship $i \in N^*$ at port visit $c \in C_i$.
EST_i^c	The earliest start time of berthing for ship $i \in N^*$ at port visit $c \in C_i$.
EFT_i^c	The expected finish time of berthing for ship $i \in N^*$ at port visit $c \in C_i$.
LFT_i^c	The latest finish time of berthing for ship $i \in N^*$ at port visit $c \in C_i$.
β	The relative increase in handling time per unit of distance from the ideal berthing position.
$\Delta^{p,p'}$	Distance between ports $p, p' \in P$.
Θ_s	Travel time per unit of distance at speed $s \in S$.
Γ_s^i	Fuel consumption per unit of distance at speed $s \in S$ for ship $i \in N^*$.
l_i	Length of ship $i \in N$.
F	Fuel cost in USD per tonne.
H	Cost handling time in USD per hour.
D	Cost of delay time in USD per hour.
I	Cost of waiting time in USD per hour.
U	Cost penalty of exceeding the latest finish time in USD per hour.

Decision variables:

$x_i^c \in \mathbb{R}^+$	the leftmost position of ship $i \in N$ at the quay for port visit $c \in C_i$.
$y_i^c \in \mathbb{R}^+$	the start time of berthing of ship $i \in N$ at port visit $c \in C_i$.
$v_{i,s}^c \in \mathbb{B}$	1 if speed $s \in S$ is chosen by ship $i \in N^*$ to sail between port visits c and $c+1$; $c \in C_i \backslash \{c_i\}$.
$d_i^c \in \mathbb{R}^+$	delay over EFT_i^c for ship $i \in N^*$ at port visit $c \in C_i$.
$u_i^c \in \mathbb{R}^+$	delay over LFT_i^c for ship $i \in N^*$ at port visit $c \in C_i$.

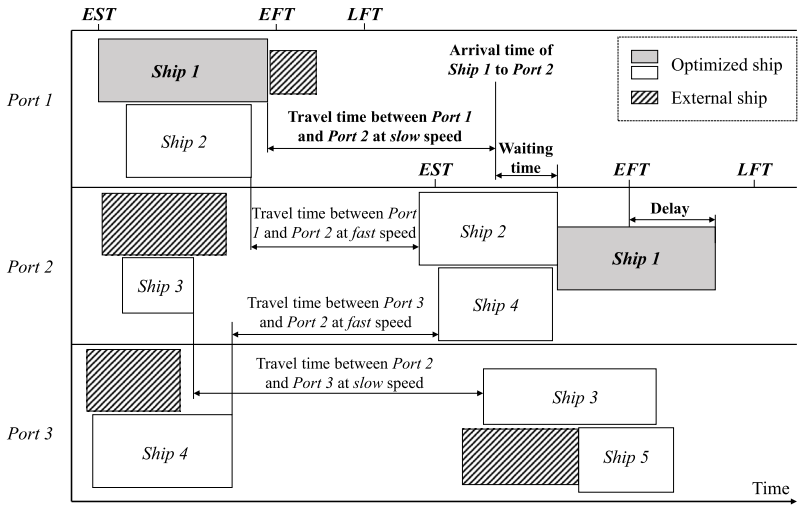


Fig. 2. Example representation of a solution for the MCBAP with five ships visiting three ports. The bolded elements refer to the timeline of operations for ship 1, where EST, EFT and LFT denote the earliest start time, the expected finish time, and the latest finish time of the ship at the ports. In this example, to travel between ports we assume only two different speeds (*slow* and *fast*).

Auxiliary variables:

$\sigma_{i,j}^{c,c'} \in \mathbb{B}$ 1 if ship i is positioned left of vessel j in the quay space at port visit $c \in C_i^p$ and port visit $c' \in C_j^p$ at port $p \in P$, 0 otherwise; $i, j \in N_p, i \neq j$.

$\delta_{i,j}^{c,c'} \in \mathbb{B}$ 1 if ship i finishes berthing before vessel j starts berthing at port visit $c \in C_i^p$ and port visit $c' \in C_j^p$ at port $p \in P$, 0 otherwise; $i, j \in N_p, i \neq j$.

$r^{i,c} \in \mathbb{R}^+$ distance between ideal and actual berthing position of ship $i \in N^*$ at port visit $c \in C_i$.

Dependent variables:

$a_i^c \in \mathbb{R}^+$ arrival time of ship $i \in N^*$ at port visit $c \in C_i$.

$h_i^c \in \mathbb{R}^+$ handling time of ship $i \in N^*$ at port visit $c \in C_i$.

$$\min \sum_{i \in N^*} \left(\sum_{c \in C_i} I(y_i^c - a_i^c) + H(h_i^c) + D(d_i^c) + U(u_i^c) + \sum_{c \in C_i \setminus \{c_i\}} F(v_i^c \Gamma_s^i \Delta^{\rho_i^c, \rho_i^{c+1}}) \right) \quad (2)$$

$$x_i^c + l_i \leq L^p, \quad \forall i \in N_p, c \in C_i^p, p \in P \quad (3)$$

$$x_i^c + l_i \leq x_j^{c'} + L^p \left(1 - \sigma_{i,j}^{c,c'}\right), \quad \forall p \in P, i, j \in N_p, i \neq j, c \in C_i^p, c' \in C_j^p \quad (4)$$

$$y_i^c + h_i^c \leq y_j^{c'} + M \left(1 - \delta_{i,j}^{c,c'}\right), \quad \forall p \in P, i, j \in N_p, i \neq j, c \in C_i^p, c' \in C_j^p \quad (5)$$

$$\sigma_{i,j}^{c,c'} + \sigma_{i,j}^{c',c} + \delta_{i,j}^{c,c'} + \delta_{i,j}^{c',c} \geq 1, \quad \forall i, j \in N_p, i < j, c \in C_i^p, c' \in C_j^p, c < c', p \in P \quad (6)$$

$$y_i^c + h_i^c + \sum_{s \in S} v_{i,s}^c \Theta_s \Delta^{\rho_i^c, \rho_i^{c+1}} = a_i^{c+1}, \quad \forall i \in N^*, c \in C_i \setminus \{c_i\} \quad (7)$$

$$a_i^c \leq y_i^c, \quad \forall i \in N^*, c \in C_i \quad (8)$$

$$EST_i^c \leq y_i^c, \quad \forall i \in N^*, c \in C_i \quad (9)$$

$$y_i^c + h_i^c - EFT_i^c \leq d_i^c \quad \forall i \in N^*, c \in C_i \quad (10)$$

$$y_i^c + h_i^c - LFT_i^c \leq u_i^c \quad \forall i \in N^*, c \in C_i \quad (11)$$

$$\left(1 + \beta r^{i,c}\right) h_0^{i,c} = h_i^c, \quad \forall i \in N^*, c \in C_i \quad (12)$$

$$x_i^c - x_0^{i,c} \leq r^{i,c}, \quad \forall i \in N^*, c \in C_i \quad (13)$$

$$x_0^{i,c} - x_i^c \leq r^{i,c}, \quad \forall i \in N^*, c \in C_i \quad (14)$$

$$\sum_{s \in S} v_{i,s}^c = 1, \quad \forall i \in N^*, c \in C_i \setminus \{c_i\} \quad (15)$$

$$y_i^c, x_i^c \geq 0 \quad \forall i \in N, c \in C_i \quad (16)$$

$$a_i^c, h_i^c, d_i^c, u_i^c, r^{i,c} \geq 0 \quad \forall i \in N^*, c \in C_i \quad (17)$$

$$v_{i,s}^c \in \{0,1\} \quad \forall i \in N^*, c \in C_i \setminus \{c_i\} \quad (18)$$

$$\sigma_{i,j}^{c,c'}, \delta_{i,j}^{c,c'} \in \{0,1\} \quad \forall i, j \in N_p, i \neq j, c \in C_i^p, c' \in C_j^p, p \in P \quad (19)$$

The set of external ships $\bar{N}$ is considered fixed. Therefore, the corresponding set of decision variables $x_i^c, y_i^c, h_i^c, r^{i,c}$ for ships $i \in \bar{N}$ are constant and given as input to the problem.

The objective function (2) minimizes the operational costs of the carriers and terminal operators. This is measured as a weighted sum of the waiting time cost, handling time cost, delay cost, and fuel consumption cost. Constraints (3) ensure that each ship berths within the available space. Constraints (4) and (5) define the relative position of each pair of ships in each dimension by enabling the auxiliary variables $\sigma^{c,c'}$ and $\delta^{c,c'}$. The M value can be limited to the latest finish time of the pair of ships. Constraints (6) ensure that berthing periods do not overlap in time and space. Constraints (7) compute the arrival time to a port based on the sailing speed chosen to travel from the previous port. Constraints (8) and (9) enforce that the berthing starts strictly after arrival at port and after the time window starts, respectively. Constraints (10) compute the delay if the expected finish time is exceeded and constraints (11) define if the last finish time is respected. Constraints (12) compute the handling time for each ship and port visit while constraints (13) and (14) compute the deviation from the preferred berthing position. Finally. constraints (15) ensure that only one speed is chosen to sail between ports, and constraints (16)–(19) define the domain of the decision variables.

4. Solution method

To solve (2)–(19) we present an Adaptive Large Neighborhood Search (ALNS) algorithm. The ALNS algorithm, introduced by Ropke and Pisinger (2006), extends the large neighborhood search method by Shaw (1998). At each iteration, the method partially destroys and reconstructs a solution to generate a new solution. In our case, to destroy part of a solution, we remove the berthing time and locations of a subset of ships at a subset of ports. The combination of a scheduled berthing time and position for a ship at one of the ports in its route is denoted as a *port visit*, and we will refer to this term frequently in the remainder of the paper. Additionally, in some cases, we will refer to the scheduled port visit as a *rectangle*, in reference to how we can depict berthing position and time in a time–space diagram (e.g., see Fig. 2).

The overview of the solution method is summarized in Algorithm 1.

Algorithm 1: Adaptive large neighborhood search procedure

Data: $inst, param$: a problem instance and a parameter setting for the algorithm
Result: $bestSol$: best found solution

```
begin
    // initialize operator selection parameters
    ψ, π ← initialize(inst)
    // construct initial solution
    sol ← constructHeuristic(inst)
    bestSol ← sol
    while timelimit not reached do
        currSol ← sol
        // select operators
        removal, insertion ← selectOperator(π)
        sol ← insertion(removal(currSol))      // get new solution
        if c(sol) < c(currSol) then
            sol ← localSearch(sol)
        if isAccepted(sol) then
            if c(sol) < c(bestSol) then
                bestSol ← sol
            currSol ← sol
        π ← updateOperatorParams(ψ)
```

4.1. Construction heuristic

The ALNS requires an initial solution to start with. We present a construction heuristic process for this step that aims at finding a good initial solution. Note that the BAP can be seen as a two-dimensional packing problem. However, in the continuous berth setting, the BAP has the increased complexity that the length of the rectangles (i.e., port visits) vary depending on the berthing location in the quay. In the case of the MCBAP, we are solving multiple continuous BAP problems with the additional constraint that some of those berthing times depend on a sailing time. Moreover, the fact that ships follow different routes complicates the problem as greedy approaches become harder to apply. Our construction method prioritizes reducing the delay of ships at ports. We approach this by (i) trying to place port visits early in time and close to their ideal space, therefore reducing the handling time, and (ii) by reducing "useless" space, or, in other words, placing port visits efficiently not to create empty spots in the decision space that cannot be filled by remaining port visit. Notice that any possible solution is mathematically feasible since we allow it to exceed the latest finish time, and the time horizon is not limited. However, we aim to construct solutions where none of the ships exceed the LFT as those can be perceived as *infeasible* by the port operators and are also heavily penalized. The method acts as a greedy heuristic, where we schedule one port visit at a time. The port visit to schedule is selected as the *most constrained* one. To find it, we compute the set of feasible berthing positions and times for each ship and port visit. Efficient search techniques such as the one presented in Lee et al. (2010) allow to precompute a subset of promising positions. In our case, to balance the computational complexity and achieve a fast initial solution, we use a simplified version of the approach and consider a finite set of positions and times by dividing the quay into segments of a given length (e.g., 10 m) and the planning horizon into intervals of 1 h. For each time instant and segment, we compute if the ship can berth starting at that time and with its left-most side starting at the segment. We do not count berthing times exceeding the latest finish time to measure how constrained a ship's port visit is. From all unscheduled port visits, we define the one with the fewest possible positions as the *most constrained* one. We then schedule the port visit in one of the feasible positions. In fact, we do not consider the entire set but only the subset of feasible positions, where the port visit rectangle is directly adjacent to another scheduled port visit or to the limits of the decision space (i.e., the limit of the quay or planning horizon). From this subset of positions, we select the one resulting in the minimal change to the objective function. Besides the handling and delay cost directly computed when scheduling the port visit, we need to compute fuel consumption and waiting time costs. We consider these only if the previous port visit of the ship is scheduled.

Once a port visit is scheduled, we repeat the computation and selection of the *most constrained* unscheduled port visit and schedule it at the least costly *efficient* position. The procedure is described in Algorithm 2.

Algorithm 2: Construction heuristic

Data: $inst$: problem instance
Result: sol: a solution with all port visits scheduled

```
begin
    // initialize entire set of port visits to schedule
    unsch ← inst
    sol ← ∅
    while unsch ≠ ∅ do
        // sort unplanned port visits by increasing number of
           feasible positions
        unsch ← sort(unsch)
        // get first port visit from the list
        toSchedule ← popfirst(unsch)
        planned ← false
        // position at the earliest start time and closest to
           the ideal position
        pos ← bestStartingPosition(toSchedule)
        while not planned do
            if feasible(pos, sol) then
                planned ← true
                // schedule the port visit
                sol ← plan(toSchedule, pos)
            else
                // update to next feasible position with lowest
                   cost
                pos ← updatePosition(pos, sol)
```

4.2. Removal and insertion operators

The goal of a removal operator is to select a set of scheduled port visits to be removed from the current solution. All the operators presented in this paper select K number of assignments to be removed, computed as a percentage ρ of the total number of port visits to be scheduled. It should be noted that removing port visits that are totally unrelated does not provide any potential gain. Therefore, a removal operator should aim at removing assignments that are related.

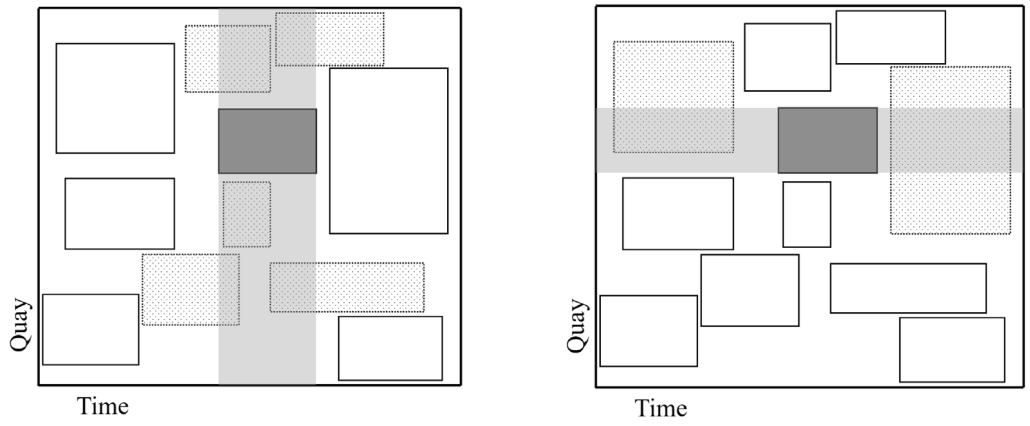
After applying a removal operator, the partial solution has K missing port visits that need to be scheduled. They need to be assigned efficiently while respecting the other assignments and ensuring that the solution remains feasible. This is the goal of the insertion operators.

4.2.1. Shaw removal

This operator, first introduced by Shaw (1998), selects the most related pairs of assignments. To select them, we define a measure of relatedness $M_{i,j}$ between assignments i and j in Eq. (20), similar to the one presented in Iris et al. (2017).

$$M_{i,j} = A|x_i - x_j| + B|y_i - y_j| + C|(y_i + h_i) - (y_j + h_j)|, \tag{20}$$

where x_i, y_i and $y_i + h_i$ are the berthing positions, berthing start time, and berthing end time of assignment i, respectively. A, B, and



(a) Time-related port visits (b) Space-related port visits

Fig. 3. Neighbor port visits in time and space for a given port visit in dark gray.

C are custom parameters that define the importance of each of the aspects. Observe that a lower value of $M_{i,j}$ translates into a higher level of relatedness. To select a total of K assignments, we select them following a greedy randomized criterion. To introduce randomness in the selection of the assignments, we define a parameter α. We sort all the port visit pairs in increasing order of $M_{i,j}$ and store them in the list Ω. We then select the ith element of the list applying Eq. (21):

$$i = \lceil |\Omega| \cdot p^{\alpha} \rceil, \tag{21}$$

where p is a random number $[0, 1)$. Note that if $\alpha = 1$, the selection is completely random, but as the value of α increases, the resulting value has a more deterministic behavior. The element selected will consist of two port visits to be removed. The selection process continues until K port visits are removed. Note that this method differs from the original method from Shaw (1998) in that the subsequent pairs do not necessarily need to be related with the first pair selected.

4.2.2. Time and space-relatedness removal

This removal uses a different relatedness measure. We first sort all port visits by cost. The cost B_i^c of port visit $c \in C_i$ for ship $i \in N^*$ is defined in Eq. (22). It is measured by the ship's waiting, handling, and delay time at the port visit, plus half of the fueling costs from sailing from the previous port (if any) and to the next port (if any).

$$B_i^c = Hh_i^c + Dd_i^c + I(y_i^c - a_i^c) + \frac{F_i^c}{2} \tag{22}$$

F_i^c is the fuel costs associated with the previous and next port visits if any. For example, if the ship sails from a previous port visit c_p to port visit c, and then continues to the next port visit c_n, then the fuel costs are computed as in Eq. (23).

$$F_i^c = F(v_i^{c_p,c} \Gamma_s^i \Delta^{\rho(c_p),\rho(c)}) + F(v_i^{c,c_n} \Gamma_s^i \Delta^{\rho(c),\rho(c_n)}) \tag{23}$$

In the case that port visit c is the first or last port visit in the route for the ship, the corresponding missing sailing leg is removed from the fuel cost computation.

We then select the i^{th} most expensive assignment applying Eq. (21) and remove all *neighbor* assignments. We define as *neighbors* all the assignments that are within a *distance* of the assignment. We consider the *distance* in both time and space. If an assignment is depicted as a rectangle in a time–space diagram of the port, the *neighbor* area represents the one that overlaps in time or space with it. All other assignments that overlap partially or completely with the neighbor area are considered neighbors and removed. We then select the most expensive assignment and remove all neighbor assignments. We repeat the process until K assignments are removed.

Fig. 3 shows an example of neighbor port visits in time and space. Depending on the dimension considered we define the two removal operators as *cost-time removal* and *cost-space removal*.

4.2.3. Random removal

We also consider a fully randomized destroy operator. It randomly selects K assignments to be removed. The goal of this operator is not to select relevant port visits to remove but rather to help diversify the search.

4.2.4. Randomized greedy insertion

This method follows the same procedure as the construction heuristic with the addition of a randomized component when selecting the port visit to schedule at each step.

All unplanned port visits are sorted based on the number of available insertion positions. An available insertion position is one that maintains a feasible solution. For instance, the port visit needs to ensure that the previous, or following port visits, are connected through a feasible sailing speed if any of these are already scheduled. We select the port visit using a randomization parameter γ in the same way that α is used in Eq. (21). This prioritizes the port visits with fewer available insertion positions. The selected port visit is scheduled in the position that increases the objective function the least (i.e., lowest cost). The process iterates by recalculating the new number of insertion positions for the remaining port visits.

4.2.5. κ-Regret insertion

This insertion method is based on the *regret-k* heuristic presented in Potvin and Rousseau (1993). This method has an additional *look-ahead* component compared to a basic greedy heuristic. For each of the port visits, we compute the κ best scheduling positions, and we then measure the *regret* cost for each of them as the difference between the best and κ-best positions. The one with the highest regret cost becomes the next port visit to plan. The process is described in Algorithm 3.

4.2.6. Packing greedy insertion

This insertion method is similar to the randomized greedy insertion described in Section 4.2.4. The main difference is the position where the port visits are planned. Scheduling the port visits in a position with lower objective value can lead to the creation of empty spaces and, therefore, to inefficient use of the decision space. This method restricts the set of possible insertion positions to the ones *strictly adjacent* to other scheduled ships, or to the limits of the quay or planning horizon. By *strictly adjacent*, we mean that the port visit to schedule needs to berth strictly next to another ship during at least one interval of time (e.g., one hour) or berth strictly before (or after) another ship with

Algorithm 3: κ-regret insertion

Data: $sol, unsch, \kappa$: partially destroyed solution, set of port visits to schedule, and the parameter κ
Result: sol: repaired solution with all port visits scheduled.

```
begin
    while unsch ≠ ∅ do
        order ← ∅                                   // initialize empty list
        for portVisit ∈ unsch do
            // compute κ best insert positions
            [pos] ← findBestPositions(κ)
            // compute regret cost
            regretCost ← c(pos[κ] − pos[1])
            // update list by regret cost
            order ← sortList(portVisit, regretCost)
        sol ← plan(order[1])                        // plan selected port visit
        // update set of unplanned port visits
        unsch ← pop(order[1])
```

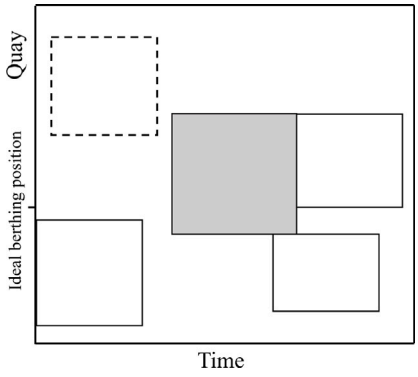


Fig. 4. Graphical representation of example positions (continuous line) *strictly adjacent* to the gray ship or the quay space. The position represented with a dashed line is not part of the set of positions as it is not adjacent to another planned ship or the boundaries of the decision space.

at least one quay segment in common. Also, we consider berthing positions where one of the sides is at one end of the quay, or if the berthing period starts or ends at the earliest and latest possible berthing time, respectively. Fig. 4 shows some example positions considered.

4.2.7. Arrival greedy insertion

This method is identical to the one presented in Section 4.2.4 with the only difference that instead of sorting the unplanned port visits by increasing the number of feasible insertion positions, we sort the unplanned port visits by the earliest possible arrival time. One of the main goals of this method is to schedule port visits earlier, at the expense of a potentially higher cost, in order to increase the number of possible insertion positions for the remaining unplanned port visits.

4.3. Acceptance criterion

Once a new solution is reconstructed, we either accept it as the new current solution or reject it and reuse the previous one. We use a simulated annealing (SA) based criterion to take this decision. Such an acceptance criterion has been widely used for ALNS studies (see e.g., Ropke and Pisinger (2006) and Iris et al. (2017)). We accept the new solution x' over the current one x if it is better ($f(x') < f(x)$), or if it is worse with a probability $e^{\frac{-(f(x')-f(x))}{T}}$, where T is the current temperature at a particular iteration, and $f(x)$ is the objective function. We define an starting an ending temperature, T_{start} and T_{end} respectively, and the cooling time t_{cool} that defines the duration of going from T_{start} to T_{end}. Based on these parameters, we can define the cooling factor τ ($0 < \tau < 1$), by isolating it from the formula $T_{end} = T_{start}\tau^{t_{cool}}$. This cooling factor allows us to compute the temperature at any given instant. Given temperature T at iteration i, we find the temperature T' to be used at iteration $i+1$ by computing $T' = T\tau^{t_{it}}$, where t_{it} is the duration of the iteration i. Following the strategy used in Iris et al. (2017), we compute T_{start} and T_{end} based on the cost of the initial solution $f(x_0)$ described in Section 4.1, where ξ and ϕ define the percentage of the cost used to compute $T_{start} = \xi f(x_0)$ and $T_{end} = \phi f(x_0)$.

Table 1
Method reward categories.

Category	Parameter
Current best solution	ψ_1
Better than current solution	ψ_2
Not better but accepted solution	ψ_3
Rejected solution	ψ_4

4.4. Adaptive weight adjustment

One of the main differences between the ALNS method and the standard Large Neighborhood Search (LNS) is the adaptive component of the former. The performance of the employed removal and insertion operators is measured at each iteration. These measures are then used to update the weight and, therefore, the probability of choosing the respective methods. The most common way of measuring the performance of a method is to give it a different score depending on the quality of the solution. In our case, we define four reward categories as shown in Table 1.

Let R and D denote the set of insertion and removal operators. Each removal and insertion method has a probability π_i^R, π_i^D respectively of being selected at each iteration. Throughout the algorithm run, the probability of selecting these methods gets updated depending on their performance. In our study, we update the probabilities after a Δ_{update} time interval. During these iterations we accumulate the sum of ψ_i^R, ψ_i^D rewards for each method, and update the weight ω_i^R, ω_i^D of each method as indicated in Eq. (24)

$$\omega_i^R = (1-\lambda)\omega_i^R + \lambda\psi_i^R, \quad \omega_i^D = (1-\lambda)\omega_i^D + \lambda\psi_i^D \tag{24}$$

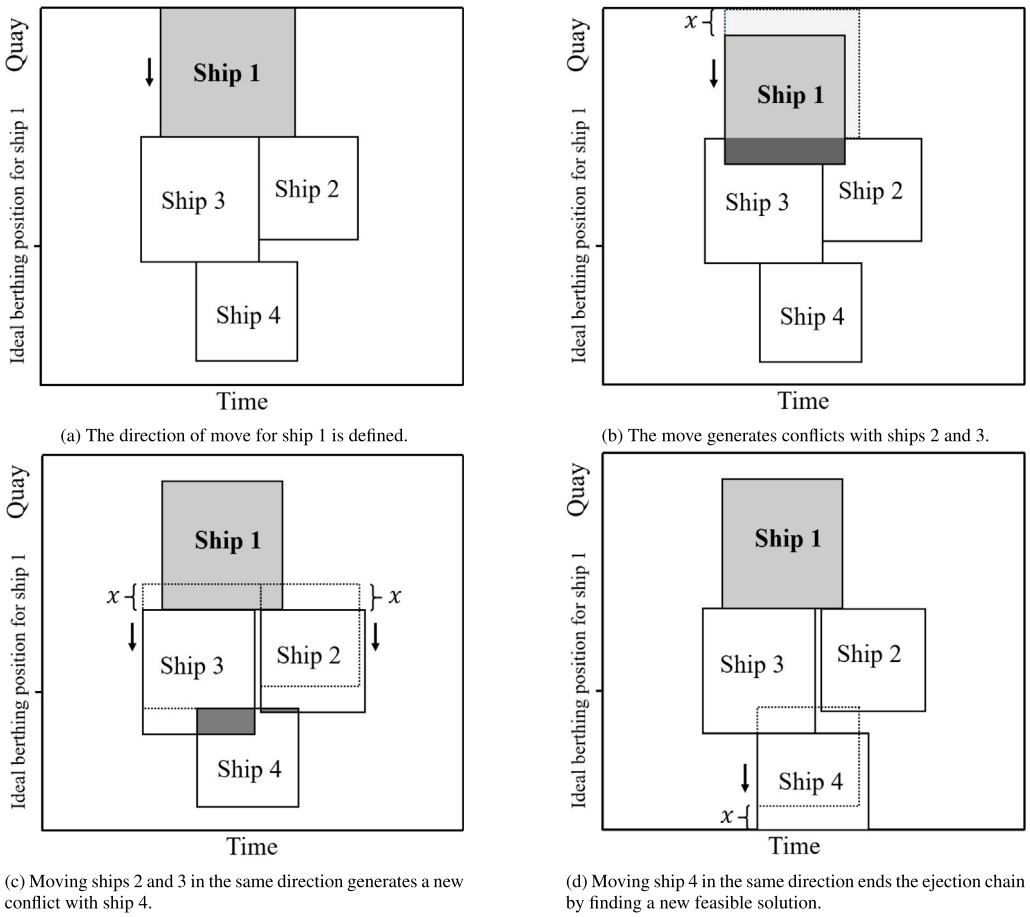
where λ is a parameter between 0 and 1 that denotes the degree of adaptability of the method. If $\lambda = 0$, the weight remains equal to the previous one. This means that each method would have the same probability throughout the entire algorithm run, behaving like an LNS with multiple neighborhoods. If $\lambda = 1$, the new operator's probability solely depends on the score achieved during the last Δ_{update} and not on previous scores. It is common to use an intermediate value for λ strictly between 0 and 1. Once the weights are updated, the probability of each repair method π_i^R and destroy method π_i^D can be computed as indicated in Eq. (25).

$$\pi_i^R = \frac{\omega_i}{\sum_{i\in R}\omega_i}, \quad \pi_i^D = \frac{\omega_i}{\sum_{i\in D}\omega_i} \tag{25}$$

4.5. Local search

An extension of the method is implemented where we perform a local search procedure after reconstructing a new solution. This idea has also been used in, for example, François et al. (2016) and Vieira et al. (2021). This step aims to incrementally improve the solution by testing small adjustments to the port visits.

The procedure is based on the *ejection chains* strategy used in many routing and network-based problems (see Bräysy (2003), Glover (1992), Rego (1998)). The idea, in our case, is to perturbate the solution by re-planning a port visit to a better position (i.e., lower



(a) The direction of move for ship 1 is defined.

(b) The move generates conflicts with ships 2 and 3.

(c) Moving ships 2 and 3 in the same direction generates a new conflict with ship 4.

(d) Moving ship 4 in the same direction ends the ejection chain by finding a new feasible solution.

Fig. 5. Example representation of a local search step. The chain of moves originates from ship 1 being moved one step x towards its ideal berthing position. The port visit in gray depicts the first ship to move, and the dark gray indicates an overlapping area. The dashed rectangles represent the original position of the ship before the move.

operational cost) and iteratively re-plan any port visits that conflict with the change. This method is also similar to the *pushing ruin-and-recreate* heuristic presented in Correcher and Alvarez-Valdes (2017). The chain of perturbations is limited to a maximum number of port visits to re-plan K_{chain}, and it terminates if this limit is reached or if a conflict-free solution is achieved. Fig. 5 shows an example of this move. Note that the handling time (i.e., the vertical dimension of the port visitsships) is reduced or increased for the ships as their position changes with respect to their ideal position.

A pseudo-code of the procedure is described in Algorithm 4. The function $movePortVisit(p, n)$ performs the perturbation for a given port visit (i.e., ship n at port p). It should be noted that the direction is given by the first perturbation made. To find the direction of the first perturbation, we compute the cost variation of moving the port visit in three directions: (i) one segment length towards the ideal position along the spatial axis, and (ii) one time instant earlier and (iii) one time instant later along the temporal axis. The direction in the spatial axis is checked if the port visit is not scheduled already at its ideal position. Once the perturbation is performed in the chosen direction, the following port visits in conflict are perturbed in the same direction.

A high value of K_{chain} increases the probability of finding a better solution and the number of operations to compute. The parameter K_{chain} should leverage both solution quality and low computational complexity. Therefore, we define the value of K_{chain} to depend on the number of instances ships and equal to $K_{chain} = 2 \cdot |N|$. The reason for $K_{chain} > |N|$ is that for some movements, a conflicting port visit may require multiple perturbations to achieve a feasible new position, and selecting a lower K_{chain} value may be too restrictive.

Due to the additional computational effort of the local search procedure, we do not execute it at each ALNS iteration. Instead, we only perform it if the reconstructed solution is better than the current one. This reduces the number of times that the local search is performed, allowing the algorithm to perform more iterations while at the same time filtering the times the local search is performed to those where we already have promising solutions.

5. Computational results

In this section, we first describe the generation process for the set of benchmark instances, and we then perform a computational study where we cover both the performance of the method and practical insights of the problem.

5.1. Instance generation

To the best of our knowledge, Martin-Iradi et al. (2022a) is the only study on the MCBAP. The instances presented in the study are rather small and limited. Therefore, we develop a more comprehensive set of benchmark instances. In the absence of real-life data, one could extend current benchmarks instances of the continuous BAP to multiple ports. Instead, we decided to use the public access to port data (Marine Traffic, 2023), and we validated it with additional data from an industrial

Algorithm 4: Local search procedure

Data: sol, K_{chain}: current solution, and the length of the ejection chain (i.e., the maximum number of port visit moves)
Result: sol: resulting solution

```
begin
    done ← false                          // initialize termination criterion
    while not done do
        nextSol ← sol                     // initialize current best solution
        Δ ← 0                             // initialize delta cost variation
        for p ∈ P do
            for n ∈ N_p do
                // track the port visits to re-plan
                toMove = [(p, n)]
                // initialize copy of current solution
                sol' ← sol
                while k ≤ K_chain and toMove ≠ ∅ do
                    // get port visit to re-plan
                    (p, n) ← pop(toMove)
                    // move port visit
                    sol' ← movePortVisit(p, n, sol')
                    // check for conflicts
                    toMove ← computeConflicts(sol')
                    // update the ejection chain length
                    k ← k + |toMove|
                if toMove = ∅ then
                    // compute cost variation
                    δ ← computeDeltaCost(sol, sol')
                    if δ < Δ then
                        Δ ← δ             // update best delta cost
                        // update current best solution
                        nextSol ← sol'
        if Δ < 0 then
            sol ← nextSol                 // update solution to return
        else
            done ← true                   // no improving neighbor solution
```

research partner to create an instance generator for the MBAP with a continuous quay.

We consider three different ship types: (i) feeders or small vessels with a length of up to 200 m, (ii) medium-size vessels with a length between 200 and 300 m, and (iii) large vessels longer than 300 m. Each ship type has a different speed-fuel consumption relation. Moreover, we consider three terminals at the three main ports in the north sea: (i) *Rotterdam APMT* with a quay length of 1600 meters (APM Terminals, 2022b), (ii) Bremerhaven NTB, with a quay length of 1800 meters (APM Terminals, 2022a), and Hamburg EGH, with a quay measuring 2100 meters (Eurogate, 2022). These three ports are relatively close to each other, and large, medium, and small vessels visit them in different sequences as part of their routes.

The duration of the vessel time window is based on the planned port call duration. The planned duration of a vessel's port call can often be updated the days previous to the arrival time. Therefore, we establish a fixed point in time for each ship two weeks before the actual arrival time and retrieve the planned port call duration as the time difference between the estimated time of arrival (ETA) and the estimated time of departure (ETD). We compute this by averaging the planned port call duration for each port and ship type berthing in a period of three months (January-March 2021). This value is also used to define the minimum handling time h_0 (see Table 2). Port service times that exceed 48, 72, and 96 h for small, medium, and large ship types, respectively, are categorized as outliers and removed from the dataset. The reason

Table 2
Minimum handling type in hours per ship type and terminal. These values define $h_0^{i,c}$.

Ship type \Terminal	DEHAM	DEBRV	NLRTM
Feeder	10.1	12.1	10.4
Medium	18.0	21.8	18.4
Large	41.0	33.7	26.7

for this is that such long service times usually involve maintenance or fueling operations that are not usually performed on a regular basis. Thus, they are not part of the problem.

We define six different ship patterns, each with a given route, type of ship, and length. All ships visit two or three ports in different orders. The N ships for a given instance are sampled from the six patterns.

For each ship, we randomize the (i) desired berthing position at each port visited and (ii) the earliest start time EST_i^c, following parameters ensuring that feasible sailing times between ports exist. The estimated finish time EFT_i^c is computed by adding the average handling time across the quay to EST_i^c ($EFT_i^c = EST_i^c + \frac{h_{max}^{i,c}+h_0^{i,c}}{2}$). Similarly, we compute the latest finish time LFT_i^c by adding the average handling time to EFT_i^c ($LFT_i^c = EFT_i^c + \frac{h_{max}^{i,c}+h_0^{i,c}}{2}$).

As input to the instance generator, we define the number of external ships at each port $Nout$ that are considered fixed. For each external ship, we randomly define: (i) the berthing position and time, (ii) the length (comprised between 180 and 330 m), and (iii) the handling time, adapted to be proportional to the length of the vessel.

We assume the entire quay is available for berthing unless an external ship is occupying it. We also consider 10 different speed levels, ranging uniformly between 17-21.5 knots. The ALNS heuristic we present can handle any continuous value of the speed but not the MIP formulation we present. To ensure a fair comparison of the methods we employ a discretized set of speed levels in both the formulation and the solution method. Furthermore, the distance between ports is computed based on the actual sea distance of the routes.

5.1.1. Handling time

It is a general practice, especially on the discrete version of the BAP, to define a different handling time depending on the berthing position. For the continuous version implemented in this paper, we follow the handling time definition presented in Meisel and Bierwirth (2009) where deviations from a preferred berthing position are penalized using a deviation factor $\beta \geq 0$ (relative increase in handling time per unit of distance, i.e., meters). Given the minimum handling time $h_0^{i,c}$ at the preferred berthing position and the actual deviation from the chosen position Δb (measured in meters), the handling time is computed as follows:

$$h_i^c = (1 + \beta \Delta b) h_0^{i,c} \tag{26}$$

In this study, the value of β is set to 0.001 as in Meisel and Bierwirth (2009), corresponding to a 0.1% increase in handling time per meter. As a reference, the handling time ranges between 20 and 60 h for medium vessels and between 30 and 110 h for large ones. This is given by berthing at the best and worst places, respectively.

5.1.2. Time windows

In the MCBAP, there are two types of time windows:

- *The time window for each ship at each visited port.* This is given by the port call duration. The time window start must be respected, but the end can be exceeded. The berthing period can, therefore, exceed the end of the time window counting the additional time as a delay.

  We also consider a time window end that defines the latest finish time (LFT). Ideally, this time window must be respected.

Table 3
Parameter settings of the benchmark instance set.

Parameter	Seed	Number of ships	Number of external ships per port	Distance between positions
Values	1–10	30, 50, 70	5, 10	10, 20, 40, 80

However, we allow violating this time window by adding a very high penalty cost.

- *The time window of the berthing positions.* This time window can be seen as the operational hours of a given berthing position. We assume that all berthing positions are available at any time. It should be noted that potential maintenance windows or partial closures of the quay can be modeled in the same way as an external ship occupying the given positions and time period.

5.1.3. Benchmark instances

Using the instance generator described in this section, we create a set of benchmark instances. The entire set comprises 240 instances. Each instance is a combination of the parameter values listed in Table 3: (i) a randomized seed, (ii) the number of ships to optimize, (iii) and the number of external fixed port visits per port, and (iv) the length of the quay segment used to define possible berthing positions. Finally, following the cost values used in Venturini et al. (2017) and Martin-Iradi et al. (2022b), we set the fuel price per ton (F) to 500 $ per tonne, the cost of waiting at port (I) and the terminal handling cost (H) to 200 $ per hour, and the delay costs for exceeding the expected finish time (D) and latest finish time (U) to 300 $ per hour and to 2000 $ per hour, respectively.

5.2. Parameter tuning

The ALNS algorithm has a total of 18 algorithm parameters. These include parameters used to calibrate the operators, the selection and acceptance criteria, and the weight of the scores for new solutions (see Table 4). To select the best value setting for the algorithm parameters we conducted a parameter tuning. We selected a subset of 12 instances that are representative of the entire set. For the tuning, we run the automatic algorithm configurator *Pydgga* (Ansótegui et al., 2021) for the 30 generations with a time limit of 5 min for each ALNS run. The configurator allows to parallelize the process, and the entire parameter tuning lasted 8 h. In Table 4, we define the domain of each algorithm parameter used for tuning and the found setting. The models and solution methods are written in *Julia* and run in a 2.90 GHz Intel Xeon Gold 6226R using one thread and 16 GB of RAM.

5.3. Method performance

We solve problem (2)–(19) with the presented method and its variants and compare the solution quality with the one obtained using baseline commercial solvers; *CPLEX v20.1* in this case.

Table 5 compares the results between CPLEX and the ALNS method. We compute the objective gap for each instance run as $\frac{z_{obj}-z_{best}}{z_{best}}$ where z_{obj} is the objective value of the best solution of the run, and z_{best} is the best-known solution across all experiments. We have grouped all instances per number of ships, external ships, and distance between berthing positions. Each group contains 10 instances and is named *X-Y-Z* according to their common characteristics. X is the number of ships, Y is the number of external ships per port, and Z is the distance between consecutive positions considered in meters (i.e., segment length). Therefore, each row in the table corresponds to the average value across instances with different seed values. The ALNS is tested by running each instance 10 times and computing the average, best and worst run.

CPLEX is not able to solve any of the instances from Table 5 to optimality within the time limit. Furthermore, we observe that CPLEX scales poorly, especially in instances with more than 30 ships. The gap is better for the smallest instances but quickly worsens. On average, the ALNS method outperforms the commercial solver by achieving tighter gaps in most instances. To further assess the performance of the method, we conduct a Wilcoxon signed-rank test (Haynes, 2013; Wilcoxon, 1945) using solution values of both the CPLEX and ALNS solutions. We normalized the values by comparing, per problem instance, the solution gap to the best-known solution in the same way as the gaps shown in Table 5. We observe the p-values to be $1.6 \cdot 10^{-29}$ for the five-minute time limit and $1.6 \cdot 10^{-24}$ for the one-hour time limit. Since the obtained p-values are well below the threshold of 0.05, this indicates that the difference between the ALNS and CPLEX objective values is statistically significant.

The solution gap is an indicator of the method performance relative to each other but does not provide an optimality guarantee. The lower bounds obtained with CPLEX indicate a high optimality gap. This could be due to a low-quality solution or a poor lower bound. Martin-Iradi et al. (2022b) indicated that the relaxation of the MIP formulation for the MBAP with a discrete quay could be worse than that of a network-flow reformulation. As indicated in Martin-Iradi et al. (2022a), network-flow formulations for the MCBAP can suffer from scalability but show that the relaxation is stronger. We compare the results from the branch-and-price method presented in Martin-Iradi et al. (2022a) with the ALNS method. The formulation presented in Martin-Iradi et al. (2022a) is slightly different from the one addressed in this paper. The formulation from Martin-Iradi et al. (2022a) defines the latest finish time for each ship berthing at a port that must be satisfied. We have adapted the method from Martin-Iradi et al. (2022a) to the formulation of this study. The branch-and-price method is based on a graph representation, and therefore, we need to establish the latest possible berthing time. This is set to 50% more than the latest finish time. This allows the method to exceed the latest finish time while maintaining the graph at a reasonable size. This is a generous bound, and our empirical studies show that this bound does not affect the optimal solution. For that, we have also generated a new set of instances of similar size to the ones presented in Martin-Iradi et al. (2022a) using the instance generator defined in Section 5.1. The input parameters for the instance set are defined in Table 6. The entire set comprises 720 instances, one for each combination of input parameters. The results are shown in Table 7, where each row shows the average values across 90 instances with the same distance between considered positions and with a similar number of ships. The optimality gap is computed as $\frac{z_{obj}-z_{lr}}{z_{lr}}$ where z_{obj} and z_{lr} are the upper and lower bound respectively obtained by each exact method. We compare the branch-and-price method with the ALNS and MIP formulation presented in this study. For both the branch-and-price and CPLEX we compute their optimality gap, where we do not observe a significant difference as both methods converge to optimality in most cases. We also compute the gap to the best-known solution (BKS) for all three methods as indicated at the beginning of Section 5.3. In this case, we observe that CPLEX provides the best performance, showing that despite its potential poorer relaxation, the upper bounds found are optimal. The branch-and-price method shows a robust performance achieving optimal solutions consistently, but fails to solve instances with a fine discretization of the quay due to excessive memory consumption. The ALNS method cannot achieve the same solution quality as the exact methods, but we recall that for larger instances, ALNS outperforms CPLEX significantly (see Table 5).

The ALNS method has two main components that differentiate it from other heuristics: (i) the adaptive procedure that guides the operator selection and (ii) the local search procedure that is performed when promising solutions are found. To quantify the impact of these two procedures, we compare the proposed method to its variants with and without each of the components. One variant is the method without its adaptive component (i.e., large neighborhood search (LNS)), meaning that each removal and insertion operator has an equal probability of

Table 4
Studied range and chosen setting of algorithm parameters after the parameter tuning.

Symbol	Description	Min val	Max val	Tuned setting
ϵ	value used to compute the cooling ratio	0.005	0.2	**0.157**
φ	pct of initial solution obj used to define start temperature (when reheated)	0.01	0.05	**0.0246**
ξ	pct of initial solution obj used to define end temperature (to be reheated)	0.00005	0.001	**0.000269**
ρ	degree of destruction, pct of total port visits to be removed by the removal methods	0.3	0.6	**0.326**
A	weight for position deviation in shaw removal	0.5	2	**0.55**
B	weight for berthing start time deviation in shaw removal	0.5	2	**1.36**
C	weight for berthing end time deviation in shaw removal	0.5	2	**0.89**
α	randomness parameter for shaw removal method	1	3	**2.66**
γ	randomness parameter for random greedy repair method	1	3	**2.85**
μ	randomness parameter for arrival greedy repair method	1	3	**2.6**
κ	k-regret parameter	2	4	**2**
Δ	number of iterations between updating the weights of each method	0.01	0.05	**0.017**
η	parameter to adjust the importance of recent scores vs. previous weight	0.3	0.7	**0.456**
ψ_1	score when finding a current best solution	10	20	**11**
ψ_2	score when finding a solution better than the current solution	4	8	**4**
ψ_3	score when the solution is accepted	1	3	**2**
ψ_4	score when the solution is rejected	–	–	**0**
Φ	parameter that defines the position bounds for ship (times its length)	2	5	**4.02**

Table 5
Gap for the MIP formulation solved by CPLEX and the ALNS method with a time limit of 5 min and 1 h. We also report the average gap between the best and worst runs of the ALNS. Each row corresponds to an instance group (i.e., the average results across 10 instances of the same size).

Instance group	5 min				1 h*			
	MIP gap (%)	ALNS gap (%)	Best ALNS gap (%)	Worst ALNS gap (%)	MIP gap (%)	ALNS gap (%)	Best ALNS gap (%)	Worst ALNS gap (%)
30-5-10	**1.3**	3.9	3.1	4.6	**0.1**	2.6	1.9	3.1
30-5-20	**1.6**	3.7	3.0	4.4	**0.1**	2.4	1.9	2.9
30-5-40	**1.8**	3.2	2.5	3.8	**0.4**	1.9	1.4	2.4
30-5-80	3.0	**2.8**	2.0	3.5	**0.5**	1.4	0.8	2.0
30-10-10	**2.0**	4.3	3.0	5.4	**0.1**	2.5	1.9	3.1
30-10-20	**3.3**	3.9	2.9	5.0	**0.4**	2.0	1.4	2.6
30-10-40	3.7	**3.4**	2.5	4.5	**0.8**	1.7	1.3	2.4
30-10-80	10.7	**2.9**	1.9	3.9	3.6	**1.0**	0.3	1.8
50-5-10	87.0	**15.5**	7.4	24.5	61.0	**4.6**	1.0	8.0
50-5-20	102.8	**15.6**	8.4	23.7	61.6	**5.3**	1.6	9.9
50-5-40	103.5	**16.0**	9.4	24.0	63.5	**4.6**	0.3	8.4
50-5-80	127.3	**17.8**	9.6	26.0	80.6	**5.1**	1.0	8.9
50-10-10	87.6	**22.4**	12.4	35.9	56.4	**7.9**	1.9	14.3
50-10-20	109.6	**21.2**	11.7	30.7	87.4	**7.7**	2.8	13.4
50-10-40	160.4	**20.5**	10.5	30.9	80.1	**5.9**	0.6	10.7
50-10-80	176.5	**21.9**	14.4	31.8	138.9	**6.7**	1.3	11.4
70-5-10	124.1	**29.6**	17.2	42.4	124.1	**9.8**	2.1	16.7
70-5-20	122.4	**26.7**	12.9	41.0	120.4	**8.7**	0.5	15.2
70-5-40	146.1	**22.1**	11.4	33.3	146.1	**8.3**	1.6	13.7
70-5-80	144.8	**19.8**	9.7	30.4	144.8	**7.6**	0.5	13.3
70-10-10	110.7	**25.8**	11.5	40.4	110.7	**9.1**	1.2	15.6
70-10-20	111.8	**21.7**	10.0	33.9	111.8	**7.4**	0.4	14.5
70-10-40	113.3	**18.9**	10.2	26.4	113.3	**7.4**	0.9	14.3
70-10-80	112.0	**16.4**	7.4	27.1	112.0	**5.4**	0.7	9.4
Average	82.0	**15.0**	8.1	22.4	67.4	**5.3**	1.2	9.1

*We do not report results for longer execution times as they have proved not to have a significant impact.

Table 6
Parameter settings of the instance set based on the ones from Martin-Iradi et al. (2022a).

Parameter	Seed	Number of ships	Number of external ships per port	Distance between positions
Values	1–5	4–15	3–5	10, 20, 40, 80

being selected throughout the algorithm run. Another variant is the ALNS method without the local search (LS). The objective gap across the methods is compared in Table 8, with a time limit of 5 min, and 1 h. In most cases, the ALNS with the local search provides the best performance. To further assess the value of both components, we conduct a pairwise statistical analysis by performing a Wilcoxon test for each pair of method variants. Table 9 shows the p-values of these comparisons and underscores the statistically significant benefits of the local search procedure and the adaptive component of the algorithm.

To measure the impact of the local search procedure, we compare different strategies that differ in the frequency of execution of the local search procedure. We test three other variants of the algorithm in which the local search is called every 1, 2, and 4 iterations. The results are summarized in Table 10 where we can observe the increased computational complexity that the local search can add to each iteration. Performing the local search procedure very frequently can lead to good solutions in fewer iterations, but it also results in longer computational times. The proposed strategy performs the local search at iterations where the reconstructed solution is better than the incumbent one, and

Table 7
Performance comparison across 720 instances between the MIP formulation solved by CPLEX, the adapted branch-and-price method from Martin-Iradi et al. (2022a), and the ALNS method, with a time limit of 5 min and 1 h. Each row shows the average gap values across all instances with same number of ships.

Number of ships	Dist. btw positions	CPLEX Opt. gap (%)		Branch-and-price Opt. gap (%)		CPLEX Gap (%)		Branch-and-price Gap (%)		ALNS+LS Gap (%)	
		5 min	1 h	5 min	1 h	5 min	1 h	5 min	1 h	5 min	1 h
4–9	80	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1.5	1.4
	40	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1.6	1.6
	20	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1.7	1.7
	10	**0.0**	**0.0**	–	–	**0.0**	**0.0**	–	–	1.6	1.5
10–15	80	0.1	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1.3	1.0
	40	0.1	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1.4	1.2
	20	**0.1**	**0.0**	0.2	**0.0**	**0.0**	**0.0**	0.1	**0.0**	1.6	1.3
	10	**0.0**	**0.0**	–	–	**0.0**	**0.0**	–	–	1.8	1.5
Average		0.04	**0.00**	0.03	0.00	**0.00**	**0.00**	0.02	0.00	1.56	1.40

"–" denotes that not all instances could be solved due to the memory limit.

Table 8
Performance comparison between variants of the proposed ALNS method.

Instance group	5 min				1 h			
	ALNS + LS gap (%)	ALNS (no LS) gap (%)	LNS + LS gap (%)	LNS (no LS) gap (%)	ALNS + LS gap (%)	ALNS (no LS) gap (%)	LNS + LS gap (%)	LNS (no LS) gap (%)
30-5-10	**3.9**	6.3	4.1	6.4	**2.6**	4.3	2.7	4.6
30-5-20	**3.7**	5.7	3.9	6.0	**2.4**	4.1	2.5	4.2
30-5-40	**3.2**	5.1	**3.2**	5.2	**1.9**	3.4	2.0	3.5
30-5-80	**2.8**	4.3	3.0	4.4	**1.4**	2.7	1.5	2.8
30-10-10	**4.3**	6.0	**4.3**	6.3	**2.5**	3.6	2.6	3.8
30-10-20	**3.9**	5.4	4.0	5.6	**2.0**	3.3	2.2	3.4
30-10-40	**3.4**	4.5	3.6	4.9	**1.7**	2.8	1.8	3.0
30-10-80	**2.9**	3.8	3.0	3.9	**1.0**	2.0	1.2	1.9
50-5-10	**15.5**	17.0	18.2	19.8	**4.6**	6.1	6.2	7.6
50-5-20	**15.6**	17.3	17.2	19.7	**5.3**	6.8	6.5	8.2
50-5-40	16.0	16.4	**15.9**	18.5	**4.6**	5.0	5.9	6.9
50-5-80	**17.8**	19.1	18.9	22.2	5.1	**5.0**	6.4	7.6
50-10-10	**22.4**	23.8	24.7	26.1	**7.9**	9.0	9.0	11.7
50-10-20	**21.2**	23.4	22.4	24.9	**7.7**	8.4	8.8	10.9
50-10-40	**20.5**	21.2	21.4	24.5	**5.9**	6.4	8.0	8.7
50-10-80	**21.9**	24.4	25.3	28.7	6.7	**6.4**	8.8	10.4
70-5-10	**29.6**	31.1	33.4	38.5	**9.8**	13.4	14.0	20.0
70-5-20	**26.7**	29.7	31.2	36.6	**8.7**	13.8	13.2	18.1
70-5-40	**22.1**	26.1	26.3	32.4	**8.3**	12.0	12.0	17.6
70-5-80	**19.8**	24.8	25.1	30.8	**7.6**	10.6	10.1	14.4
70-10-10	**25.8**	27.9	27.3	33.5	**9.1**	12.3	11.6	17.4
70-10-20	**21.7**	25.9	26.1	31.5	**7.4**	11.3	11.3	16.3
70-10-40	**18.9**	23.1	24.2	29.4	**7.4**	11.3	11.3	14.3
70-10-80	**16.4**	19.0	19.7	24.5	**5.4**	7.9	8.2	10.4
Average	**15.0**	17.1	16.9	20.2	**5.3**	7.2	7.0	9.5

Table 9
p-value of pair-wise comparison of solution method variants.

	Methods compared	ALNS (no LS)	LNS + LS	LNS (no LS)
5 min	ALNS + LS	$6.2 \cdot 10^{-24}$	$6.0 \cdot 10^{-20}$	$1.5 \cdot 10^{-38}$
	ALNS (no LS)	–	0.0231	$2.7 \cdot 10^{-33}$
	LNS + LS	–	–	$2.5 \cdot 10^{-36}$
1 h	ALNS + LS	$1.5 \cdot 10^{-32}$	$6.2 \cdot 10^{-29}$	$4.7 \cdot 10^{-41}$
	ALNS (no LS)	–	0.0023	$1.4 \cdot 10^{-33}$
	LNS + LS	–	–	$4.7 \cdot 10^{-37}$

we show that this strategy performs the best. This method allows us to perform the local search in a fewer number of iterations, but at the same time, has the potential to result in promising and better solutions.

Tables 11 and 12 summarized the performance of the different removal and insertion operators used within the ALNS method. For each removal operator, we compute three metrics: (i) the percentage of iterations in which the operator was selected *% its*, (ii) the percentage of current best solutions found using the operator *% new best*, and (iii) the percentage of times that the resulting solution was better than the current one using the operator *% better than current*. For the insertion methods, we also display a fourth column *% of time*, which indicates the percentage of time each operator has consumed from the total time spent repairing solutions. The time spent in removal methods is significantly lower than in insertion operators; therefore, we do not compute this metric for the removal operators. We observe that the random and cost-time removals are the better-performing removal methods when the number of ships is 30. However, for larger instances, the performance of the cost-time removal improves. In particular, this operator is efficient at improving the current solution, which leads to finding most of the best-known solutions. This also suggests that when the number of port visits increases, the probability of removing port visits that are not related at all also increases, making pure random methods less efficient. Furthermore, removing port visits that overlap in time is more effective than removing the ones that overlap in berthing space. While ships can berth at multiple positions along the quay without major delays and disruptions in their schedule, berthing earlier or later can negatively impact the sailing in the rest of the voyage legs. Therefore, their flexibility comes at a larger cost. We also notice that the Shaw removal becomes less useful in larger instances. This

Table 10
Algorithm comparison with different frequencies for the local search procedure with a time limit of one hour.

Instance group	Local search every iteration		Local search every 2 iterations		Local search every 4 iterations		Local search every iteration where the solution is better than the incumbent		
	Gap (%)	Iterations x1000	Gap (%)	Iterations x1000	Gap (%)	Iterations x1000	Gap (%)	Iterations x1000	% of iterations with local search
30-5-10	1.6	46.6	1.8	58.6	2.0	67.3	2.6	79.1	1.7
30-5-20	1.5	72.8	1.7	99.0	1.9	120.3	2.4	153.2	1.5
30-5-40	1.1	102.8	1.2	154.2	1.4	203.8	1.9	307.5	1.3
30-5-80	0.9	130.7	1.0	211.2	1.0	300.5	1.4	550.6	1.1
30-10-10	1.5	49.4	1.7	61.1	1.9	69.3	2.5	81.0	1.4
30-10-20	1.2	79.2	1.4	105.2	1.7	126.1	2.0	157.0	1.3
30-10-40	1.0	115.8	1.2	167.4	1.4	218.1	1.7	316.9	1.2
30-10-80	0.8	162.2	1.0	247.5	1.0	340.1	1.0	568.3	1.2
50-5-10	5.5	11.1	5.9	16.5	6.0	20.6	4.6	30.6	0.4
50-5-20	7.1	15.1	6.8	23.5	6.3	32.9	5.3	56.2	0.3
50-5-40	8.5	18.8	7.9	31.4	7.4	48.7	4.6	118.2	0.3
50-5-80	13.0	20.3	12.0	36.4	10.2	59.8	5.1	218.7	0.3
50-10-10	8.4	12.3	9.5	17.6	8.4	21.1	7.9	30.1	0.3
50-10-20	11.2	17.5	9.8	26.2	9.9	35.1	7.7	57.5	0.3
50-10-40	11.6	22.4	10.9	36.4	9.3	53.9	5.9	118.2	0.3
50-10-80	16.2	27.6	14.5	46.7	12.3	73.2	6.7	223.1	0.3
70-5-10	12.5	3.7	11.2	5.8	12.6	8.5	9.8	16.3	0.3
70-5-20	13.8	4.8	12.9	8.1	13.9	12.6	8.7	30.2	0.2
70-5-40	17.0	5.7	15.7	10.1	14.1	17.0	8.3	62.0	0.3
70-5-80	18.6	5.8	17.7	10.6	16.5	18.9	7.6	115.0	0.3
70-10-10	11.0	4.7	10.2	7.2	11.1	9.7	9.1	16.7	0.4
70-10-20	12.1	6.1	11.9	9.9	11.8	14.5	7.4	31.0	0.3
70-10-40	14.4	7.4	14.5	12.7	13.6	20.8	7.4	64.4	0.3
70-10-80	14.1	8.3	13.4	14.7	13.3	25.1	5.4	116.8	0.4
Average	**8.5**	**39.6**	**8.2**	**59.1**	**7.9**	**79.9**	**5.3**	**146.6**	**0.7**

Table 11
Performance summary of the four removal operators. The instances are grouped per number of ships.

Number of ships	Cost-berth removal			Cost-time removal			Shaw removal			Random removal		
	% its	% new best	% better than current	% its	% new best	% better than current	% its	% new best	% better than current	% its	% new best	% better than current
30	13	9	8	30	38	42	23	16	16	34	37	35
50	8	9	4	46	49	58	15	12	8	32	29	30
70	10	12	6	68	67	81	8	7	3	15	14	10

Table 12
Performance summary of the four insertion operators. The instances are grouped per number of ships.

Number of ships	Efficient packing insertion				Random greedy insertion				Arrival greedy insertion				κ-regret insertion			
	% its	% of time	% new best	% better than current	% its	% of time	% new best	% better than current	% its	% of time	% new best	% better than current	% its	% of time	% new best	% better than current
30	21	16	24	18	28	19	24	31	24	17	23	22	27	38	29	28
50	12	9	12	7	30	23	29	29	28	22	28	29	31	40	30	34
70	19	14	19	16	21	17	20	18	40	30	42	48	20	33	19	18

operator removes pairs of port visits at a time. These pairs can be highly unrelated between them, worsening the effects of the overall operator. Similarly to the random removal, this inter-relatedness issue increases with the number of port visits.

Regarding the repair methods, the performance is more homogeneous and all contribute similarly. The κ-regret insertion method is computationally more intensive, requiring more time. All methods help achieve the best-found solutions in a similar with fewer ships, and the arrival greedy insertion becomes more effective in larger instances.

To better understand the algorithm's behavior and when the operators are used, we tracked the use of each operator during the algorithm run. Figs. 6 and 7 show an example run of one hour for an instance with 30 ships, 10 external ships per port, and a distance between consecutive positions of 10 m. We observe that the cost-space removal is only used at the first half of the algorithm run when each operator has a more balanced probability of being chosen. The packing greedy heuristic shows a similar behavior and correlates with the low usage of these two operators, as shown in Table 11 and 12. Nonetheless, the rest of the operators are used for most algorithm runs. Some operators show an oscillating behavior, such as the cost-time removal operator. This behavior correlates with the temperature of the acceptance criterion, which is reheated periodically and suggests that the operator is more likely to be selected when the temperature is higher.

Additionally, for both removal and insertion operators, we tested the algorithm removing the worst performing operators, one at a time, but the performance of the method worsened in all cases, indicating that all operators are to some extent useful and combine well together.

5.4. Practical impact

This problem involves two main stakeholders, namely, the terminal operators and the shipping carriers. The objective function covers the operational costs of both of them. This section disaggregates and analyzes the different operational costs by performing various sensitivity analyses.

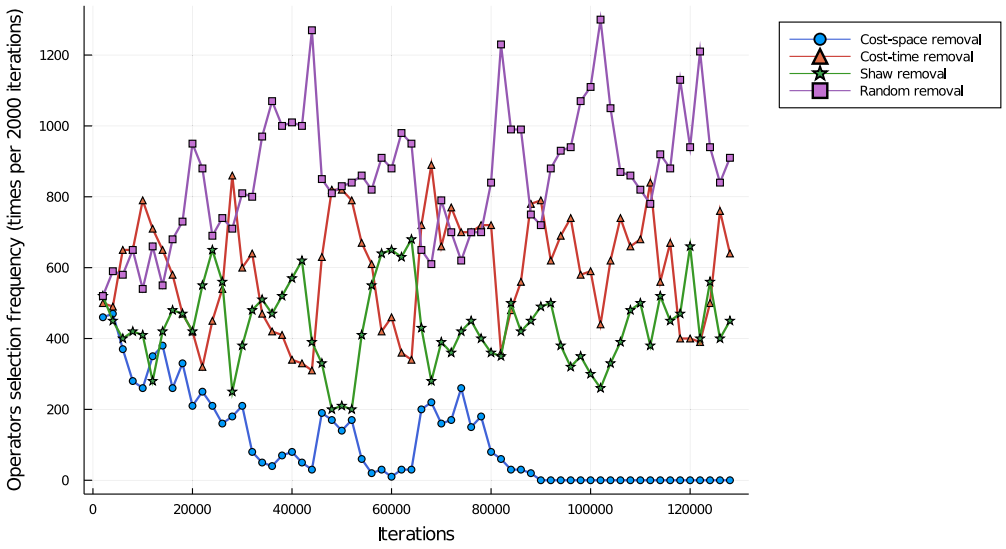


Fig. 6. Usage of each removal operator during an algorithm run of 1 h for an instance from group $30-10-10$.

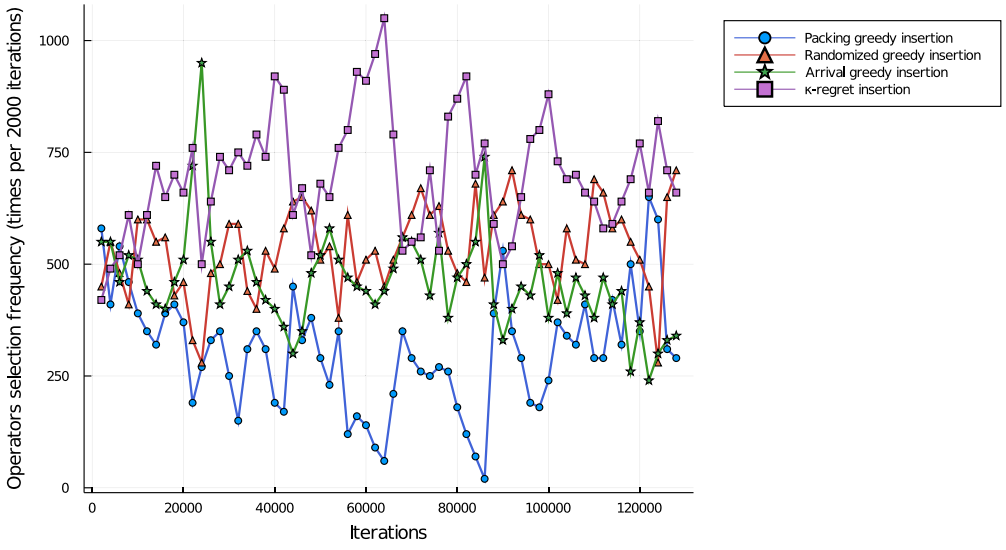


Fig. 7. Usage of each insertion operator during an algorithm run of 1 h for an instance from group $30-10-10$.

Table 13
Average operational costs and cost variation across instances with different distance between consecutive positions. All instances are run for one hour. The costs are in thousands of USD.

Distance btw. positions	Waiting	Handling	Delay	Fuel	Penalty	Total	% of penalized port visits
10	519	534	930	1277	1750	5010	12
20	518	535	933	1278	1753	5018	12
40	540	535	1019	1277	2066	5437	13
80	586	538	1243	1280	2893	6540	16

In Table 13, we group the instances per distance between consecutive positions. We observe a natural trade-off here: A shorter distance between positions allows a more granular set of berthing positions and, therefore, a potentially better solution quality. However, this increases the complexity of the problem, and in the case of our method, it results in fewer iterations per hour. Despite performing less than a fifth of the iterations of the instances with 80 meters distance, the method finds better solutions for the shorter-distance instances. The improvement in objective value mainly translates into shorter delays, and waiting time. The handling and fuel costs remain similar. The vessel time windows or port calls are already pre-planned, considering a low sailing speed. This, together with the fact that fuel costs account for a large part of the total costs, results in that ships already sailing at the slowest speed in most of the solutions (see Table 15). We observe that considering more berthing positions improves the number of penalized ships that exceed their time window, reducing their average exceeded time in up to 50%.

Another operational aspect we inspect is the impact of the external ships in the planning process. We solve the problem instances with 30 ships with a varying number of external ships per port, from none to twenty ships per port. The results are summarized in Table 14. The results support the rationale that an increased number of external ships

Table 14
Operational costs for instances grouped by different number of external ships per port.

External ships per port	Waiting	Handling	Delay	Fuel	Penalty	Total	% increase of penalized port visits
0	82	308	116	755	16	1278	–
5	111	315	222	756	170	1573	+6%
10	148	322	466	761	837	2534	+21%
20	288	323	1316	762	5186	7874	+54%

Table 15
Average fuel consumption per ship and sailing speed based on different fuel prices.

Number of ships	Fuel price (USD/metric tonne)					
	200		500		1100	
	Average fuel consumption (metric tonne per ship)	Average speed (knots)	Average fuel consumption (metric tonne per ship)	Average speed (knots)	Average fuel consumption (metric tonne per ship)	Average speed (knots)
30	50.3	17.50	50.3	17.50	50.3	17.50
50	51.2	17.54	51.0	17.51	50.9	17.50
70	52.6	17.72	51.6	17.60	51.1	17.55
Average	51.4	17.59	51.0	17.54	50.8	17.52

per port results in a more congested berth allocation and, as a result, higher operational costs. In this case, the port congestion is reflected in the number of penalized port visits. Since the external ships are considered fixed, a higher number of these can lead to excessive delays due to port visits exceeding their latest finish time and propagating such delays to following ports in their routes. These results also indicate that the level of impact of this type of collaborative problem can increase significantly when more ships are involved. When more ships collaborate, their potential joint savings increase and the terminal has more planning flexibility.

As mentioned previously, fuel consumption is the main cost driver for carriers. Fuel prices have fluctuated significantly in the last two years due to the global socio-economical and political situation. We consider that ships use a very low sulfur fuel oil (VLSFO) with an estimated price of 500 USD per metric ton. However, the prices of this fuel have ranged between 200 and 1100 USD per metric ton in the last two years. Therefore, we have also tested our method using a fuel price of 200 and 1100 USD per metric ton (Ship & Bunker, 2022)

Table 15 shows the average fuel consumption per ship and sailing speed, grouped by instances with the same number of ships. We observe that the average consumption can increase by almost half a metric tonne when the fuel price decreases from 500 USD per tonne to 200 USD per tonne. This difference is more prominent in instances with a large number of ships, where more ships sail marginally faster to arrive earlier at the next port to get a better service. However, when the fuel price increases above 500 USD per metric tonne, the reductions in fuel consumption are relatively small. The main explanation for this is due to the low sailing speeds in general. The fuel costs already account for a large part of the operational costs, and the solutions indicate that ships sail close to the slowest speed of 17 knots in most cases. We observe a slight increase in average sailing speed when the fuel price is low and the number of ships increases. At the highest fuel price, the average speed never reaches the minimum of 17 knots, underscoring the need for continued speed optimization, as imposing all ships to travel at the lowest speed could lead to higher overall operational costs. A similar sensitivity analysis performed by Venturini et al. (2017) indicated a similar behavior.

6. Conclusions

In this work, we address an emerging problem in maritime collaborative logistics that integrates the operations of both shipping carriers and terminal operators. We present both a new MIP formulation for the multi-port continuous berth allocation problem with speed optimization, and an ALNS algorithm to solve it. The ALNS algorithm takes advantage of a diverse set of tailored insertion and removal methods. It guides the algorithm by prioritizing the better-performing methods. The modular characteristic of the algorithm could be exploited to develop a decision support tool for terminal operators, where the operators' experience can lead to new tailored operators. Furthermore, in terms of computational performance, the heuristic method is able to find high-quality solutions to larger instances than the ones studied in the literature and outperforms commercial solvers such as CPLEX. We also study the practical impact of the problem in terms of operational costs for the carriers and terminal operators and analyze the resulting quality of the berth plans and sailing speeds. We conclude that engaging in this type of collaboration can result in overall cost reductions for the stakeholders and also benefits to the environment due to the potential lower fuel consumption.

Some aspects of this study remain as future work or research direction. Regarding the solution method, the insertion operators are the main bottleneck in terms of computational complexity. One could explore simpler insertion operators or other heuristic variants. Studying the scalability of the method in more detail could be relevant. There is no doubt that the heuristic method scales better than CPLEX, and results in small instances indicate that the ALNS achieves near-optimal solutions. For larger instances, the optimality gap of CPLEX increases significantly, and the lower bound becomes impractical. Finally, incorporating practical aspects such as transhipments or disruptions management is an attractive research direction. We envision the use of frameworks such as stochastic programming to tackle this type of problem. All in all, this type of study highlights the potential impact of collaborative logistics and the value of integration in the transportation sector.

Acknowledgments

The authors thank the Danish Maritime Fund for supporting this work and three anonymous reviewers for their valuable comments.

References

Ansótegui, C., Pon, J., & Sellmann, M. (2021). Boosting evolutionary algorithm configuration. *Annals of Mathematics and Artificial Intelligence*, http://dx.doi.org/10.1007/s10472-020-09726-y.

APM Terminals (2022a). APM terminals bremerhaven NTB. https://www.apmterminals.com/en/bremerhaven/practical-information/practical-information, (Accessed: 2022-11-4).

APM Terminals (2022b). APM terminals maasvlakte II. https://www.apmterminals.com/en/maasvlakte/about/our-terminal, (Accessed: 2022-09-15).

Bierwirth, C., & Meisel, F. (2015). A follow-up survey of berth allocation and quay crane scheduling problems in container terminals. *European Journal of Operational Research*, *244*(3), 12689, 675–689. http://dx.doi.org/10.1016/j.ejor.2014.12.030.

Bräysy, O. (2003). A reactive variable neighborhood search for the vehicle-routing problem with time windows. *Informs Journal on Computing*, *15*(4), 347–368. http://dx.doi.org/10.1287/ijoc.15.4.347.24896.

Carlo, H. J., Vis, I. F., & Roodbergen, K. J. (2014). Transport operations in container terminals: Literature overview, trends, research directions and classification scheme. *European Journal of Operational Research*, *236*(1), 1–13. http://dx.doi.org/10.1016/j.ejor.2013.11.023.

Cheimanoff, N., Fontane, F., Kitri, M. N., & Tchernev, N. (2022). Exact and heuristic methods for the integrated berth allocation and specific time-invariant quay crane assignment problems. *Computers & Operations Research*, *141*, Article 105695. http://dx.doi.org/10.1016/j.cor.2022.105695.

Cordeau, J. F., Laporte, G., Legato, P., & Moccia, L. (2005). Models and tabu search heuristics for the berth-allocation problem. *Transportation Science*, *39*(4), 526–538. http://dx.doi.org/10.1287/trsc.1050.0120.

Correcher, J. F., & Alvarez-Valdes, R. (2017). A biased random-Key genetic algorithm for the time-invariant berth allocation and quay crane assignment problem. *Expert Systems with Applications*, *89*, 112–128. http://dx.doi.org/10.1016/j.eswa.2017.07.028.

De Oliveira, R. M., Mauri, G. R., & Lorena, L. A. N. (2012). Clustering search heuristic for solving a continuous berth allocation problem. In M. Jin-Kao Hao (Ed.), *Vol. 7245*, *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)* (pp. 49–62). Springer-Verlag, http://dx.doi.org/10.1007/978-3-642-29124-1_5.

Du, Y., Chen, Q., Quan, X., Long, L., & Fung, R. Y. (2011). Berth allocation considering fuel consumption and vessel emissions. *Transportation Research Part E: Logistics and Transportation Review*, *47*(6), 1021–1037. http://dx.doi.org/10.1016/j.tre.2011.05.011.

Dulebenets, M. A. (2018). A comprehensive multi-objective optimization model for the vessel scheduling problem in liner shipping. *International Journal of Production Economics*, *196*, 293–318. http://dx.doi.org/10.1016/j.ijpe.2017.10.027.

Dulebenets, M. A. (2019). Minimizing the total liner shipping route service costs via application of an efficient collaborative agreement. *Ieee Transactions on Intelligent Transportation Systems*, *20*(1), Article 8315131. http://dx.doi.org/10.1109/TITS.2018.2801823.

Dulebenets, M. A., Golias, M. M., & Mishra, S. (2018). A collaborative agreement for berth allocation under excessive demand. *Engineering Applications of Artificial Intelligence*, *69*, 76–92. http://dx.doi.org/10.1016/j.engappai.2017.11.009.

Dulebenets, M. A., Pasha, J., Abioye, O. F., & Kavoosi, M. (2019). Vessel scheduling in liner shipping: a critical literature review and future research needs. *Flexible Services and Manufacturing Journal*, *33*(1), 43–106. http://dx.doi.org/10.1007/s10696-019-09367-2.

Eurogate (2022). Eurogate hamburg. http://www1.eurogate.de/en/EUROGATE/Terminals/Hamburg, (Accessed: 2022-11-4).

Fagerholt, K. (2001). Ship scheduling with soft time windows: An optimisation based approach. *European Journal of Operational Research*, *131*(3), 559–571. http://dx.doi.org/10.1016/S0377-2217(00)00098-9.

Fagerholt, K., Laporte, G., & Norstad, I. (2010). Reducing fuel emissions by optimizing speed on shipping routes. *Journal of the Operational Research Society*, *61*(3), 523–529. http://dx.doi.org/10.1057/jors.2009.77.

François, V., Arda, Y., Crama, Y., & Laporte, G. (2016). Large neighborhood search for multi-trip vehicle routing. *European Journal of Operational Research*, *255*(2), 422–441. http://dx.doi.org/10.1016/j.ejor.2016.04.065.

Glover, F. (1992). New ejection chain and alternating path methods for traveling salesman problems. In O. Balci, R. Sharda, & S. A. Zenios (Eds.), *Computer Science and Operations Research. New Developments in their Interfaces*, 491–508.

Guan, Y., & Cheung, R. K. (2005). The berth allocation problem: Models and solution methods. *Container Terminals and Automated Transport Systems: Logistics Control Issues and Quantitative Decision Support*, 141–158. http://dx.doi.org/10.1007/3-540-26686-0_6.

Guo, L., Zheng, J., Du, H., Du, J., & Zhu, Z. (2022). The berth assignment and allocation problem considering cooperative liner carriers. *Transportation Research Part E: Logistics and Transportation Review*, *164*, Article 102793. http://dx.doi.org/10.1016/j.tre.2022.102793.

Haynes, W. (2013). Wilcoxon rank sum test. In W. Dubitzky, O. Wolkenhauer, K.-H. Cho, & H. Yokota (Eds.), *Encyclopedia of systems biology* (pp. 2354–2355). New York, NY: Springer New York, http://dx.doi.org/10.1007/978-1-4419-9863-7_1185.

Hellsten, E. O., Sacramento Lechado, D., & Pisinger, D. (2020). An adaptive large neighbourhood search heuristic for routing and scheduling feeder vessels in multi-terminal ports. *European Journal of Operational Research*, *287*(2), 682–698. http://dx.doi.org/10.1016/j.ejor.2020.04.050.

Imai, A., Sun, X., Nishimura, E., & Papadimitriou, S. (2005). Berth allocation in a container port: using a continuous location space approach. *Transportation Research Part B-methodological*, *39*(3), 199–221. http://dx.doi.org/10.1016/j.trb.2004.04.004.

IMO (2018). Initial IMO strategy on reduction of GHG emissions from ships. *Tech. Rep. MEPC.304(72)*, (p. 3). International Maritime Organization, URL http://www.imo.org/en/OurWork/Environment/PollutionPrevention/AirPollution/Pages/GHG-Emissions.aspx, (Accessed on 04.05.2020).

Iris, C., & Lam, J. S. L. (2018). Models for continuous berth allocation and quay crane assignment: Computational comparison. *Ieee International Conference on Industrial Engineering and Engineering Management*, *2017-*, 374–378. http://dx.doi.org/10.1109/IEEM.2017.8289915.

Iris, C., Pacino, D., & Røpke, S. (2017). Improved formulations and an adaptive large neighborhood search heuristic for the integrated berth allocation and quay crane assignment problem. *Transportation Research. Part E: Logistics and Transportation Review*, *105*, 123–147. http://dx.doi.org/10.1016/j.tre.2017.06.013.

Kim, K. H., & Moon, K. C. (2003). Berth scheduling by simulated annealing. *Transportation Research, Part B (Methodological)*, *37*(6), 541–560. http://dx.doi.org/10.1016/S0191-2615(02)00027-9.

Kordić, S., Davidović, T., Kovač, N., & Dragović, B. (2016). Combinatorial approach to exactly solving discrete and hybrid berth allocation problem. *Applied Mathematical Modelling*, *40*(21–22), 8952–8973. http://dx.doi.org/10.1016/j.apm.2016.05.004.

Lalla-Ruiz, E., Melián-Batista, B., & Moreno-Vega, J. M. (2016). A cooperative search for berth scheduling. *Knowledge Engineering Review*, *31*(05), 498–507. http://dx.doi.org/10.1017/s0269888916000266.

Lee, D. H., Chen, J. H., & Cao, J. X. (2010). The continuous berth allocation problem: A greedy randomized adaptive search solution. *Transportation Research Part E: Logistics and Transportation Review*, *46*(6), 1017–1029. http://dx.doi.org/10.1016/j.tre.2010.01.009.

Lim, A. (1998). The berth planning problem. *Operations Research Letters*, *22*(2–3), 105–110. http://dx.doi.org/10.1016/S0167-6377(98)00010-8.

Lyu, X., Negenborn, R. R., Shi, X., & Schulte, F. (2022). A collaborative berth planning approach for disruption recovery. *Ieee Open Journal of Intelligent Transportation Systems*, *3*, 153–164. http://dx.doi.org/10.1109/OJITS.2022.3150585.

Marine Traffic (2023). Port calls data. https://www.marinetraffic.com/en/data/, (Accessed: 2023-01-14).

Martin-Iradi, B., Pacino, D., & Ropke, S. (2022a). The multi-port continuous berth allocation problem with speed optimization. In J. de Armas, H. Ramalhinho, & S. Voß (Eds.), *Computational logistics* (pp. 31–43). Cham: Springer International Publishing.

Martin-Iradi, B., Pacino, D., & Ropke, S. (2022b). The multiport berth allocation problem with speed optimization: Exact methods and a cooperative game analysis. *Transportation Science*, *56*(4), 972–999. http://dx.doi.org/10.1287/trsc.2021.1112.

Mauri, G. R., Ribeiro, G. M., Lorena, L. A. N., & Laporte, G. (2016). An adaptive large neighborhood search for the discrete and continuous Berth allocation problem. *Computers & Operations Research*, *70*, 140–154. http://dx.doi.org/10.1016/j.cor.2016.01.002.

Meisel, F., & Bierwirth, C. (2009). Heuristics for the integration of crane productivity in the berth allocation problem. *Transportation Research Part E: Logistics and Transportation Review*, *45*(1), 196–209. http://dx.doi.org/10.1016/j.tre.2008.03.001.

Potvin, J., & Rousseau, J. (1993). A parallel route building algorithm for the vehicle-routing and scheduling problem with time windows. *European Journal of Operational Research*, *66*(3), 331–340. http://dx.doi.org/10.1016/0377-2217(93)90221-8.

Rego, C. (1998). A subpath ejection method for the vehicle routing problem. *Management Science*, *44*(10), 1447–1459.

Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, *40*(4), 455–472. http://dx.doi.org/10.1287/trsc.1050.0135.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. *Lecture Notes in Computer Science*, *1520*, 417–431.

Ship & Bunker (2022). Global average bunker price - VLSFO. https://shipandbunker.com/prices/av/global/av-g20-global-20-ports-average, (Accessed: 2022-12-18).

Steenken, D., Voß, S., & Stahlbock, R. (2004). Container terminal operation and operations research - a classification and literature review. *Or Spectrum*, *26*(1), 3–49. http://dx.doi.org/10.1007/s00291-003-0157-z.

Sun, B., Niu, B., Xu, H., & Ying, W. (2018). Cooperative optimization for port and shipping line with unpredictable disturbance consideration. In M. Li, X. Ning, Z. Xiao, G. Xiao, K. Li, & L. Wang (Eds.), *Icnc-fskd 2018 - 14th international conference on natural computation, fuzzy systems and knowledge discovery*. Institute of Electrical and Electronics Engineers Inc., http://dx.doi.org/10.1109/FSKD.2018.8686901, 8686901, 113–118.

UNCTAD (2020). Review of maritime transport 2020. *Tech. rep.*, UNCTAD.

Venturini, G., Iris, C., Kontovas, C. A., & Larsen, A. (2017). The multi-port berth allocation problem with speed optimization and emission considerations. *Transportation Research. Part D: Transport and Environment*, *54*, 142–159. http://dx.doi.org/10.1016/j.trd.2017.05.002.

Vieira, B. S., Ribeiro, G. M., Bahiense, L., Cruz, R., Mendes, A. B., & Laporte, G. (2021). Exact and heuristic algorithms for the fleet composition and periodic routing problem of offshore supply vessels with berth allocation decisions. *European Journal of Operational Research*, *295*(3), 908–923. http://dx.doi.org/10.1016/j.ejor.2021.03.022.

Wang, S., Liu, Z., & Qu, X. (2015). Collaborative mechanisms for berth allocation. *Advanced Engineering Informatics*, *29*(3), 572, 332–338. http://dx.doi.org/10.1016/j.aei.2014.12.003.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, *1*(6), 80–83, URL http://www.jstor.org/stable/3001968.

Yu, J., Tang, G., & Song, X. (2022). Collaboration of vessel speed optimization with berth allocation and quay crane assignment considering vessel service differentiation. *Transportation Research Part E: Logistics and Transportation Review*, *160*, Article 102651. http://dx.doi.org/10.1016/j.tre.2022.102651.